Next Generation Automotive Embedded Systems-on-Chip and Their Applications

Executive Summary

Mohamed Anas

A portfolio submitted to

The Universities of

Edinburgh

Glasgow

Heriot Watt

Strathclyde

for the Degree of

Doctor of Engineering in System Level Integration

March 2005

Abstract

It is a well known fact in the automotive industry that critical and costly delays in the development cycle of powertrain[1] controllers are unavoidable due to the complex nature of the systems-on-chip used in them. The primary goal of this portfolio is to show the development of new methodologies for the fast and efficient implementation of next generation powertrain applications and the associated automotive qualified systems-on-chip. A general guideline for rapid automotive applications development, promoting the integration of state-of-the-art tools and techniques necessary, is presented. The methods developed in this portfolio demonstrate a new and better approach to co-design of automotive systems that also raises the level of design abstraction.

An integrated business plan for the development of a camless engine controller platform is presented. The plan provides details for the marketing plan, management and financial data.

A comprehensive real-time system level development methodology for the implementation of an electromagnetic actuator based camless internal combustion engine is developed. The proposed development platform enables developers to complete complex software and hardware development before moving to silicon, significantly shortening the development cycle and improving confidence in the design.

A novel high performance internal combustion engine knock processing strategy using the next generation automotive system-on-chip, particularly highlighting the capabilities of the first-of-its-kind single-instruction-multiple-data micro-architecture is presented. A patent application has been filed for the methodology and the details of the invention are also presented.

Enhancements required for the performance optimisation of several resource properties such as memory accesses, energy consumption and execution time of embedded powertrain applications running on the developed system-on-chip and its next generation of devices is proposed. The approach used allows the replacement of various software segments by hardware units to speed up processing.

---

[1] *Powertrain:* A name applied to the group of components used to transmit engine power to the driving wheels. It can consist of engine, clutch, transmission, universal joints, drive shaft, differential gear, and axle shafts.

Acknowledgements

I would like to thank my supervisors Professor David R. S. Cumming and Dr. Bill Nailon for their guidance and advice. I am very grateful to David and Bill for providing the intellectual stimulation for the projects carried out in this area and their consistently well focused, yet nonetheless visionary, critique of my work.

I would also like to thank Dr. Franz Fink, vice president and general manager, Freescale Semiconductors, formerly Motorola Semiconductor Products Sector, for many discussions at the outset of my EngD to strategically define various aspects of my industrial projects. Various cross functional team members, particularly my industrial supervisor Robin Paling, 32-bit Embedded Controller Division, has provided much encouragement and support together with an unerring desire to meet all my goals, create real-time demonstrator platform to prove the usefulness of next generation automotive advanced single instruction multiple data systems-on-a-chip – one of the greatest strengths of this department. I would also like to give special thanks to the systems engineering team members with whom I have worked closely throughout my EngD and who provided an excellent intellectual "sparing partner".

A very large number of people within the universities of Edinburgh, Glasgow, Heriot-Watt and Strathclyde, and particularly from the Institute for System Level Integration, have contributed to the development of various demonstration platforms. I would like to thank the 32-bit Embedded Controller Division of Freescale Semiconductors for their technical and financial support.

And last but not least, for their support and understanding over the years, I thank my family and friends.

Preface

This portfolio presents work undertaken by the author while working in the 32-bit Embedded Controller Division of Motorola SPS, presently known as Freescale Semiconductors. A writing style has been chosen that is fairly explanatory and, it is hoped, easy to read. The aim throughout has been to take a reader, who may not be an expert in the field of automotive systems-on-a-chip and their applications, through the work carried out in such a manner that at the end he or she may have gained a good understanding of what has been done and glimpsed the future possibilities of this exciting area. The author is *convinced* of both the usefulness and practicality of the basic idea of advanced single instruction multiple data system-on-chip and their applications with tightly coupled digital signal processing functionality: it is hoped that some of this enthusiasm for the work has been captured in the contents of this portfolio and imparted to its readers.

Journal Publications and Conference Proceedings

1. Mohamed Anas, D. R. S. Cumming, W. H. Nailon and R. J. Paling, "Fast Internal Combustion Knock Processing Algorithm Using an Automotive PowerPC System-on-Chip," IEEE *Transaction on Vehicular Technology*, accepted for publication, manuscript ID VT-2004-00099, (November 2004).
2. Mohamed Anas, D. R. S. Cumming, W. H. Nailon and R. J. Paling, "Fast Internal Combustion Knock Processing Algorithm Using an Automotive PowerPC System-on-Chip," proceedings of the 5th Australasian Control Conference, Automotive Applications, Melbourne, Australia, vol. 1, pp. 30 – 38, (July 19-23 2005).
3. Mohamed Anas, D. R. S. Cumming and W. H. Nailon, "System Level Design of a Controller Platform for a Camless, Electromagnetic Actuator Driven Next Generation Car Engine," IEEE *Transaction on Vehicular Technology*, submitted for review.

System-on-Chip Book Chapter

1. Next Generation Automotive PowerPC Von-Neumann Cache Memory: An Architectural Overview and Efficient Use in Fast Time Critical Applications Development, November 22, 2002.

Key Awards and Achievements

1. International IEE Research and Best Presentation Medal and Award for the value of £5000 – May 2003
2. Best conference presentation award, 5th Australasian Control Conference, Melbourne, Australia, July 2005.

Industrial and Academic Conference Presentations

1. Mohamed Anas, Next Generation Automotive SoCs and their Application, IEE International Hudswell Research Conference, *Overall Best Presentation Award and Winner of £5000 Scholarship*, Savoy Place, London, (May 2002).
2. Mohamed Anas, Distributed Camless Engine Controller Platform, SAE TOPTEC Congress 2002, Ann Arbour, USA, (September 23-26, 2002).
3. Mohamed Anas, Next Generation Automotive Powertrain Systems-on-a-Chip and Their Applications, Motorola TechTalk, East Kilbride, Scotland, (February 21, 2003).
4. Mohamed Anas, Next Generation Automotive Powertrain Systems-on-a-Chip and Their Applications, Motorola Smart Networks Developer Forum, Hyatt Regency Hotel, Dallas, Texas, USA, (March 20-23, 2003).
5. Mohamed Anas, Fast Internal Combustion Engine Knock Processing using an Automotive Qualified SoC, Automotive TechTalk, East Kilbride, Scotland, (August 21, 2003).
6. Mohamed Anas, Fast Internal Combustion Engine Knock Processing, 5th Australasian Control Conference, Automotive Applications, Melbourne, Australia, (July 19-23 2004).
4. Mohamed Anas, Various Major Customer Presentations and Seminars

Patents Pending

1. Mohamed Anas, "Distributed Electromagnetic Actuator Driven Camless Engine Controller Platform," US Patent Pending
2. Mohamed Anas, "Fast Internal Combustion Engine Knock Processing Methodology," US Patent Pending

Glossary of Products Used

1. Metrowerks CodeWarrior Development Suite -Compiler v1.5 (MW v1.5)
2. Greenhills Compiler, Functional Simulator and Debugger v4.0.1 (GHS v4.0.1)
3. WindRiver Diab Compiler and Functional Simulator v5.0
4. MATLAB, Simulink and Stateflow Release 12 and Release 13
5. Microsoft Office Professional Suite – 2000 and XP
6. LAUTERBACH - TRACE32 BDM Debugger for PowerPC
7. SIMD SoC Core Cycle Accurate Architectural Modelling Environment
8. Various MPC5xx automotive qualified evaluation boards
9. MPC5554 automotive qualified evaluation board
10. HiWare Functional Simulator v2.0
11. Microsoft Project 2000

Table of Contents

List of Figures and Tables

1 Introduction

1.1 Systems-on-Chip in Modern Cars

The automobile has proven to be the ultimate "vehicle" for integrating new electronic technologies. Its average consumption level, variety of semiconductors and growth rate, from 10% in 1998 and projected to be 40% in 2009 according to leading global automotive manufacturers continues to be impressive [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]. Today's premier vehicles use over 70 microcontrollers [1, 2, 6, 11]. In fact, the term "system-on-chip" (SoC) is often used to describe this class of powerful microcontrollers. In the automotive world, the transition from 4, 8 and 16-bit micro controller units (MCUs) to powerful 32 and 64-bit SoCs is not without some important considerations. It often means learning a new central processing unit (CPU) architecture, investing in new development tools, and importantly porting existing software. While there is a significant investment, there is also significant payback in the long term, if correct decisions are made.

Traditionally, advancements in microprocessor speed have been gained by increasing clock speed, which is a measurement of how fast a microprocessor can execute instructions [12]. However, recent design efforts have focused upon increasing the number of instructions that can be executed simultaneously [12, 13, 14, 15, 16]. When more instructions can be executed at the same time, an SoC's speed increases without requiring an increase in clock speed [12]. To increase speed without increasing the clock speed, various single instruction multiple data (SIMD) technologies are being designed to allow SoCs apply a single computation to many pieces of data at the same time [13, 16]. Additionally, modern embedded SoC based automotive systems are used in complex powertrain control, antilock braking systems and airbag systems. These applications require the manipulation of large pieces of data and particularly perform complex calculations.

More limited applications such as near obstacle detection, integral cellular phones, advanced body computers, telematics and vehicle mobile Internet access products taking vehicular SoCs to a new level. High performance SoCs is providing a growing list of feature differentiators for many vehicles - cars, trucks and all the emerging classifications in between.

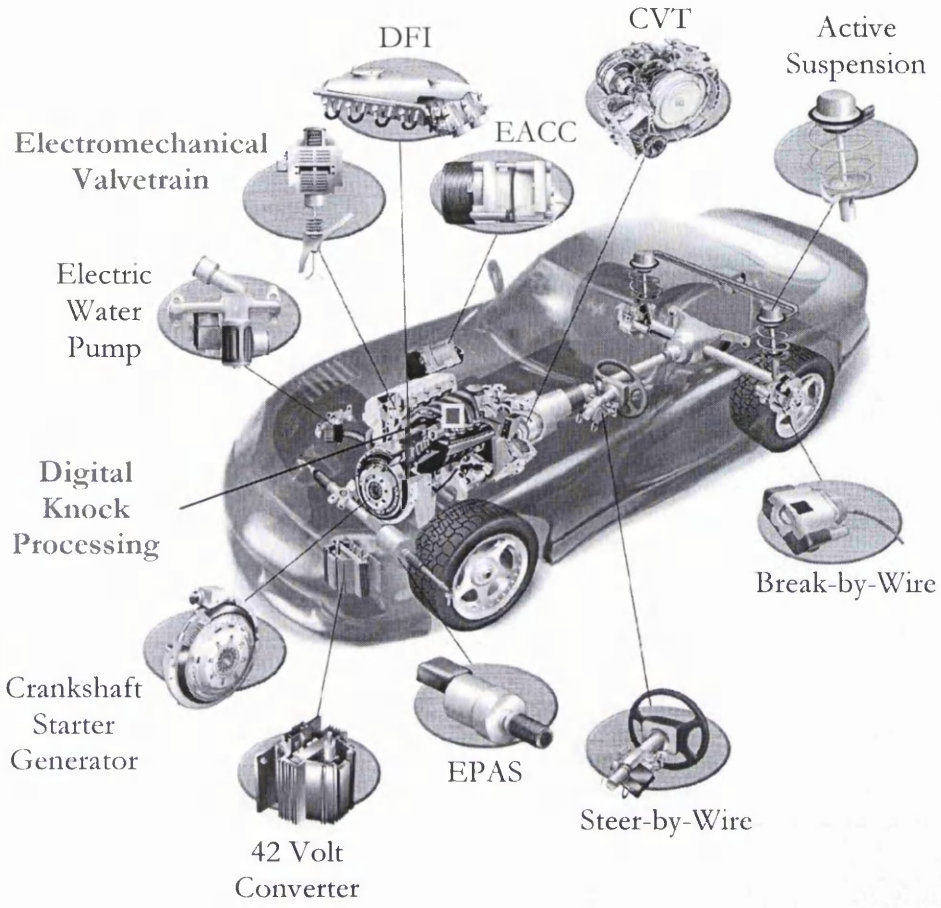

Figure 1.1: Performance Demanding SoC Controlled Modules in a Modern Car

Where,

CVT - Continuous Variable Transmission

DFI - Digital Fuel Injection

EACC - Electric Air Conditioning Control

EAPS - Electric Power Assisted Steering

Many vehicle applications use several semiconductor technologies to meet the performance and cost targets of the system. Figure 1.1 illustrates some of the existing and next generation processor bandwidth intensive embedded automotive applications.

Increasingly sophisticated automotive and transportation infrastructure requirements are defining navigation, collision avoidance, automatic toll collection, night vision and even auto-pilot type systems for future automobiles. The proof-of-concept-technologies in many of these systems continue to trace their roots to military or aerospace [17, 18, 19, 20].

New technologies will most likely be required to attract broad consumer acceptance. Computing technology adapted for automotive applications is providing the initial Internet connectivity for vehicles.

With the SoC technology poised to radically change the automotive electronic industry landscape, it is essential to understand the future SoC trends and the performance requirements of the applications to be run on them. As a result, research was conducted on various architectural elements of the next generation embedded automotive PowerPC SoC and its applications incorporating a mix of hardware and software. The proposed approach responds to the requirements for new design and development technology, driven by the automotive market demand for high performance and particularly cost-effective SoCs.

1.2 Portfolio Structure and Original Contributions

The key aspects of the research undertaken are as follows:

- To set-up a viable business plan for the next generation automotive qualified advanced single-instruction-multiple-data (SIMD) PowerPC SoCs with tightly coupled digital signal processing (DSP) functionality
- To assist the rapid development of processor bandwidth intensive embedded automotive applications and their SoCs
- To evaluate and solidify various core architectural enhancements as per performance requirements of such applications
- Introduction and implementation of first-of-its-kind single-instruction-multiple-data (SIMD) based real-time embedded automotive algorithms
- To facilitate the simulation and prototyping of heterogeneous automotive systems, by supporting specification, the co-existence and interaction of different models of computation, mixed-mode system simulation, design and generation of automatic code from a block diagram description of the applications and its algorithms.
- To promote the reuse of intellectual property (IP)

These aspects are encapsulated in the themed portfolio. In particular, embedded automotive applications are developed; demonstrating the capabilities of the next generation embedded SIMD PowerPC SoC. This section of the executive summary insets the historical background to the current slow transformation from legacy microcontrollers to advanced SoCs and the motivating forces behind it.

Portfolio I, *Integrated Business Plan for the SoC Development of Camless Engine Platform*, discusses the business plan put forward to raise necessary funds from the Motorola Semiconductor Products Sector (SPS) funding body for the design and development of the distributed SoC controller platform for the control of electromagnetic actuator driven camless engines. This portfolio lays emphasis on the overall industrial and commercial relevance of the industrial EngD research undertaken.

Portfolio II, *System Level Design of a Controller Platform for a Camless, Electromagnetic Actuator Driven Car Engine*, discusses the design of a patent-pending rapid development methodology of a controller platform and the associated algorithm, for an electromagnetic actuator driven camless engine, based on the developed PowerPC SoC. This portfolio also reviews the current state of camless engines. This work was presented at a number of international automotive conferences and permission has been granted for the publication of the attached journal paper.

Portfolio III, *Fast Internal Combustion Engine Knock Processing Using an Automotive PowerPC System-on-Chip*, introduces a patent-pending, journal published, novel high performance knock detection strategy using the next generation automotive PowerPC SoC, particularly highlighting the capabilities of the SIMD micro architecture implemented. The developed SIMD knock processing algorithm is based on autonomous on-chip modules and a custom designed auxiliary signal processing extension tightly coupled to the main SoC core. The author has also presented various software development techniques with an advanced software circular buffer implementation for processing the streaming knock sensor data.

The SoC Book Chapter, *Embedded Von-Neumann On-Chip PowerPC SoC Cache*, introduces the design and efficient use of the on-chip first-of-its-kind Von-Neumann cache of the automotive SoC. This document also introduces various cache control hardware instructions developed, particularly, addressing the optimal handling of time-critical segments of applications.

Next Generation Powertrain SoC Performance Requirements, is a discussion of author evaluated key SoC architectural enhancements and additions required in the natural successor to the abovementioned single-issue SoC, based on a number of next generation powertrain applications. This investigation was done by profiling author and automotive customer developed applications. This document also discusses the model based control approach taken with the development of some of the profiled applications.

The document, *Architectural Modelling Environment for the PowerPC SoC*, discusses the various fully-functional and cycle-accurate SoC core specific simulation modules developed by the author for pre-silicon evaluation. This simulation platform is primarily used for performance analysis of applications which in turn solidifies the pre-silicon SoC core requirements to be implemented in real-time. Particularly architectural trade-offs at the processor, cache, and memory interface level of system design, also known as core-complex for embedded SoC designs is investigated.

Figure 1.2 overleaf shows a synopsis of the above themed portfolio plan.

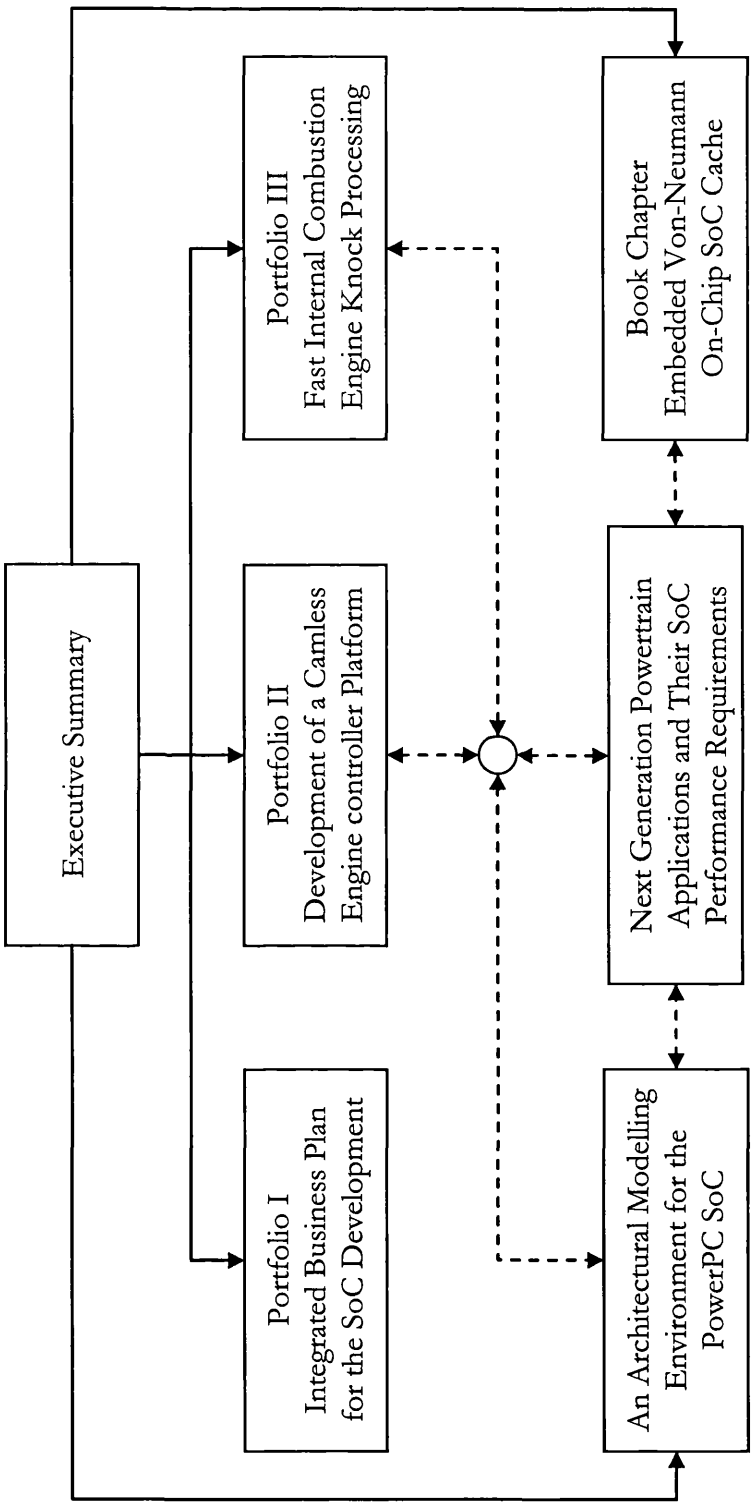

Figure 1.2: Synopsis of Themed Portfolio

2 Industrial Context

2.1 SoCs in Embedded Automotive Applications

The automobile as a self-contained microcosm is undergoing radical changes due to the advancements in automotive SoC technology. The relative part of electronics in the automobile is increasing dramatically whilst the future of new electronic systems based around SoCs is extremely promising. Beginning with electronic controlled ignition in the early sixties [23], past two decades have witnessed a tremendous change in vehicle electronics [21, 22, 23, 24, 25, 26]. New applications such as mobile multimedia, control-by-wire systems, advanced safety interiors, and collision avoidance highlights a portion of the significantly more electronic content in the automobile of the future. SoCs are now essential to control the movements of the automobile, of the chemical and electrical processes taking place in it, to entertain the passengers, to establish connectivity with the rest of the world, and importantly to ensure safety. Of particular importance is the development of automotive SoC technologies necessary to offer timely, reliable, and cost effective products to the consumer.

2.1.1 Automotive Powertrain Controllers

Automotive powertrain controllers generally satisfy diverse and often conflicting requirements [27, 28, 29, 30, 31]. Such robust controllers in and around the engine room require high reliability, advanced performance, high voltage (>35V) and high temperature (>125 °C) characteristics [32, 33, 34, 35, 36, 37, 38]. Dense integration of complex sub-controllers is also key to providing expanded functions without increasing the size of the overall controller or the number of devices on it. By adopting high performance SoCs, such expanded functions can be provided in smaller and more lightweight packages with high reliability [39, 40].

In high performance engines, the number of application to be controlled and the associated control actions have been increasing in order to achieve better driveability, higher power, lower fuel consumption and particularly lower emissions. Therefore, in addition to being adaptive and self-diagnostic, the powertrain control systems have to control many applications such as,

- Fuel Injection
- Ignition Timing
- Idle Speed
- Exhaust Gas Recirculation (EGR)
- Knock Control
- Electronic Controlled Transmission (ECT)
- Camless Engine Control (CEC)

Additionally, such application need to interact with safety critical applications such as antilock braking and traction control, necessitating the need to for the integration of time and event driven high performance fault-tolerant communications protocols such as CAN [23], the time triggered protocol (TTP) [41] and FlexRay [11].

2.1.2 Development of Powertrain Controllers

Development of powertrain controllers describes techniques, tools, roles, deliverables, standards and activities [21, 22, 23, 24, 25, 26, 28, 29, 30, 32, 33, 34, 35, 36]. With the advent of the new low-cost complex automotive qualified high performance SoCs, there are unique challenges and trade-offs for development engineers and associated tool vendors [42, 43, 44, 45, 46, 81, 89, 90, 88].

Ever increasing performance needs of advanced powertrain applications have led the development of sophisticated embedded development tool chains, SoCs with complex cores, various types of on-chip memories and autonomous peripherals, particularly supporting single cycle on-chip instruction and data buses [81, 86, 87, 88, 89, 90]. Designers of such SoCs strive to provide the visibilities needed by logic analysers, processor emulation and calibration systems, while not compromising performance and cost saving.

It is evident that there is less support for the development of powertrain controllers based on advanced SoCs introducing critical and costly delays in the development cycle [22, 23, 24, 25, 26, 47]. Additionally, based on industrial trend, it can simply be stated that powertrain SoCs would soon be based on superscalar[2] or multiple-issue cores, magnifying the complexities already present with the application development process [81, 86, 87, 88, 89, 90]. Thus it is vital to establish a strategy early enough in the development process to avoid costly business disruptions.

As with other industries, a new powertrain controller concept originates in research and development and once this new concept is proven, it makes its way into the appropriate automotive platform.

A systematic approach, outlined in Figure 2.1 is generally used to translate the concept into an embedded powertrain controller [22, 23, 24, 25, 26, 47]. A number of steps in the illustrated development process require a model of the powertrain system for which the controller is being developed.

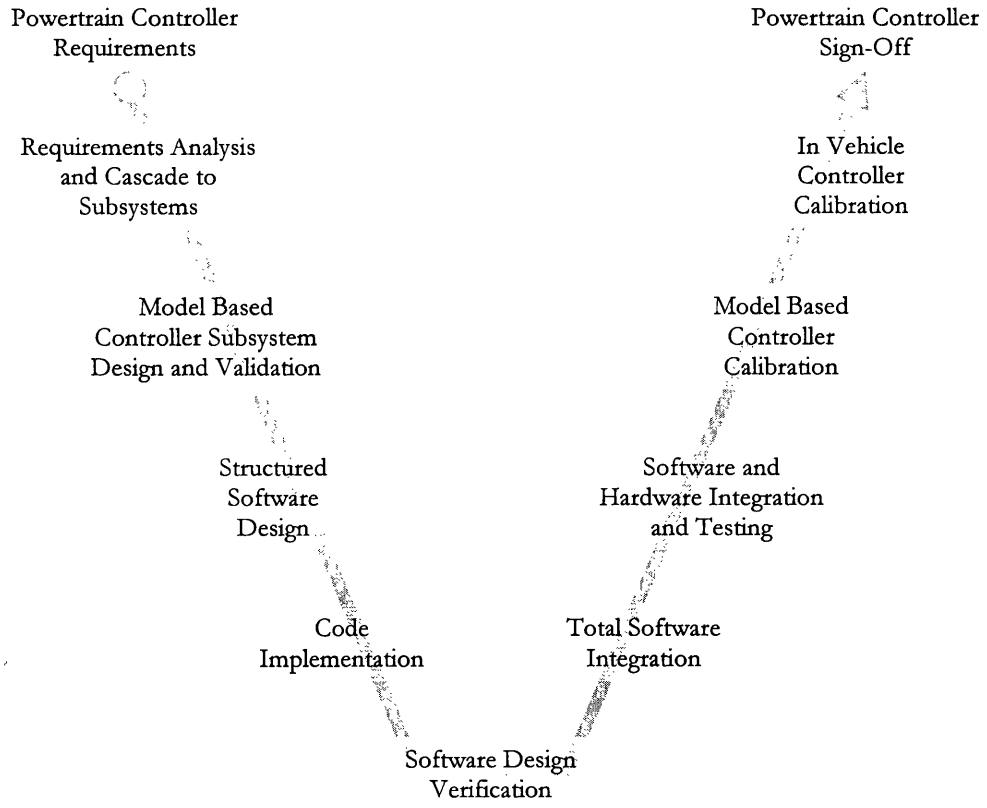

Figure 2.1: Powertrain Controller Software V Development Cycle

[2] A microprocessor architecture that contains more than one execution unit or pipeline, allowing the processor to execute more than one instruction per clock cycle.

Figure 2.1 also demonstrates the complexity and expansiveness of the overall powertrain controller development process, and how different skills and organisations are involved. As shown above, once the sub-systems are solidified, they are then qualified and calibrated to verify conformance with the initial requirements.

3 EngD Research Contribution

3.1 Integrated Business Plan

This portfolio encapsulates a comprehensive business plan developed by the author for the camless engine controller platform business based on the PowerPC SoC. The overall plan enclosed is divided into four distinct sections:

1. The offer, summarising the funds required and the benefits
2. Marketing plan
3. Financial plan
4. Organisational plan

This plan was used for the allocation of required resources, handle unforeseen complications, and make good business decisions. Specifically, organised information about the company and proposal as to how the repayment of the invested money is to be made is highlighted. Additionally, it informs sales personnel, suppliers, and others about all the project operations and goals.

Of particular importance to the overall plan is the principle agreement with BMW Group and various other OEMs and Tier Is to supply 138 million valve control SoCs over a 10 year period starting in 2006. This provides the venture with a reliable income stream through highly reputed premier automotive engine technologist and manufacturers.

During the process, the biggest challenge faced was the understanding of the technology and business risks, in order to establish methodologies to manage those risks and rewards, and to develop the business models to fairly allocate those risks and rewards to the appropriate business players.

Figure 3.1 overleaf shows the overall financial summary of the camless engine controller business.

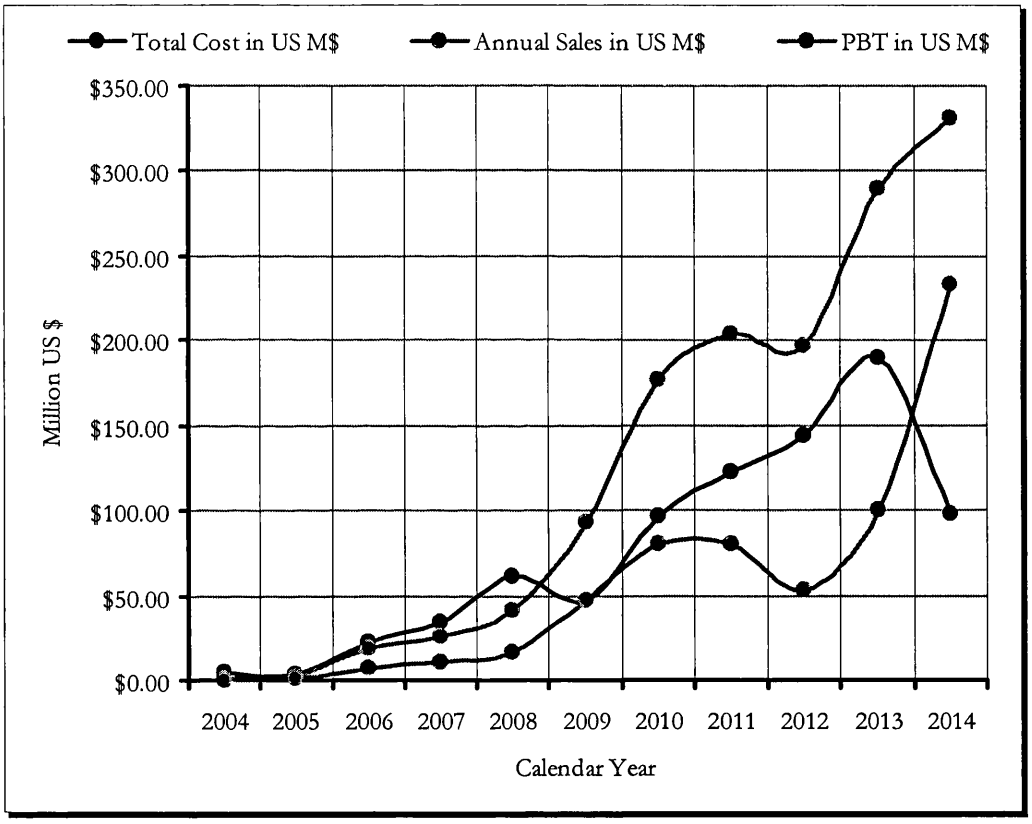

Figure 3.1: Camless Project – Financial Summary

3.2 An Overview of the SoC Developed

The SoC developed, as per the outcome of the business plan highlighted in section 3.1, is the first member of a family of next generation powertrain SoCs based on the PowerPC Book E architecture, containing many new features coupled with high performance complementary metal oxide semiconductor (CMOS) technology to provide substantial reduction of cost per feature and significant performance improvement over the legacy devices, particularly the presently used MPC565 [81]. This SoC is targeted toward middle to high-end powertrain applications, such as camless engine controller platforms and digital knock processing.

The host processor core of the SoC complies with the PowerPC Book E architecture. It is 100% user mode compatible with the classic PowerPC instruction set including the floating point library. The Book E architecture has enhancements that improve the PowerPC architecture's fit in embedded applications. This core also has additional instructions, including DSP instructions, supported by the signal processing extension (SPE), beyond the classic PowerPC instruction set.

The SoC has two levels of memory hierarchy. The fastest accesses are to the 32kB unified, aka von Neumann cache. The next level in the hierarchy contains the 64kB on-chip static random access memory (SRAM) and the 2 MB internal flash memory. Both the SRAM and the flash memory can hold instructions and data. The External Bus Interface has been designed to support most of the standard embedded memories widely available.

The complex I/O timer functions of SoC are performed by two Enhanced Time Processor Unit engines (eTPU). Each eTPU micro-engine controls 32 hardware channels. The eTPU consists of 24-bit timers, double action hardware channels, variable number of parameters per channel, angle clock hardware, and additional control and arithmetic instructions. The eTPU can be programmed using a high-level programming language.

The less complex timer functions required are performed by the Modular Timer System (eMIOS/MTS). The eMIOS' 24 hardware channels are capable of single action, double action, Pulse Width Modulation (PWM) and modulus counter operation. Off-chip communication is performed by a suite of serial protocols including controller area networks (CAN), enhanced serial peripheral interface (SPI) and serial communication interfaces (SCI).

Additionally the SoC has an on-chip 40-channel Enhanced Queued dual Analogue-to-Digital Converter (eQADC). The System Integration Unit (SIU) performs several chip-wide configuration functions. Pad configuration and General-Purpose Input and Output (GPIO) are controlled from the SIU. External interrupts and reset control are also found in the SIU. The Internal Multiplexer sub-block (IMUX) provides multiplexing of eQADC trigger sources, daisy chaining the DSPIs and external interrupt signal multiplexing.

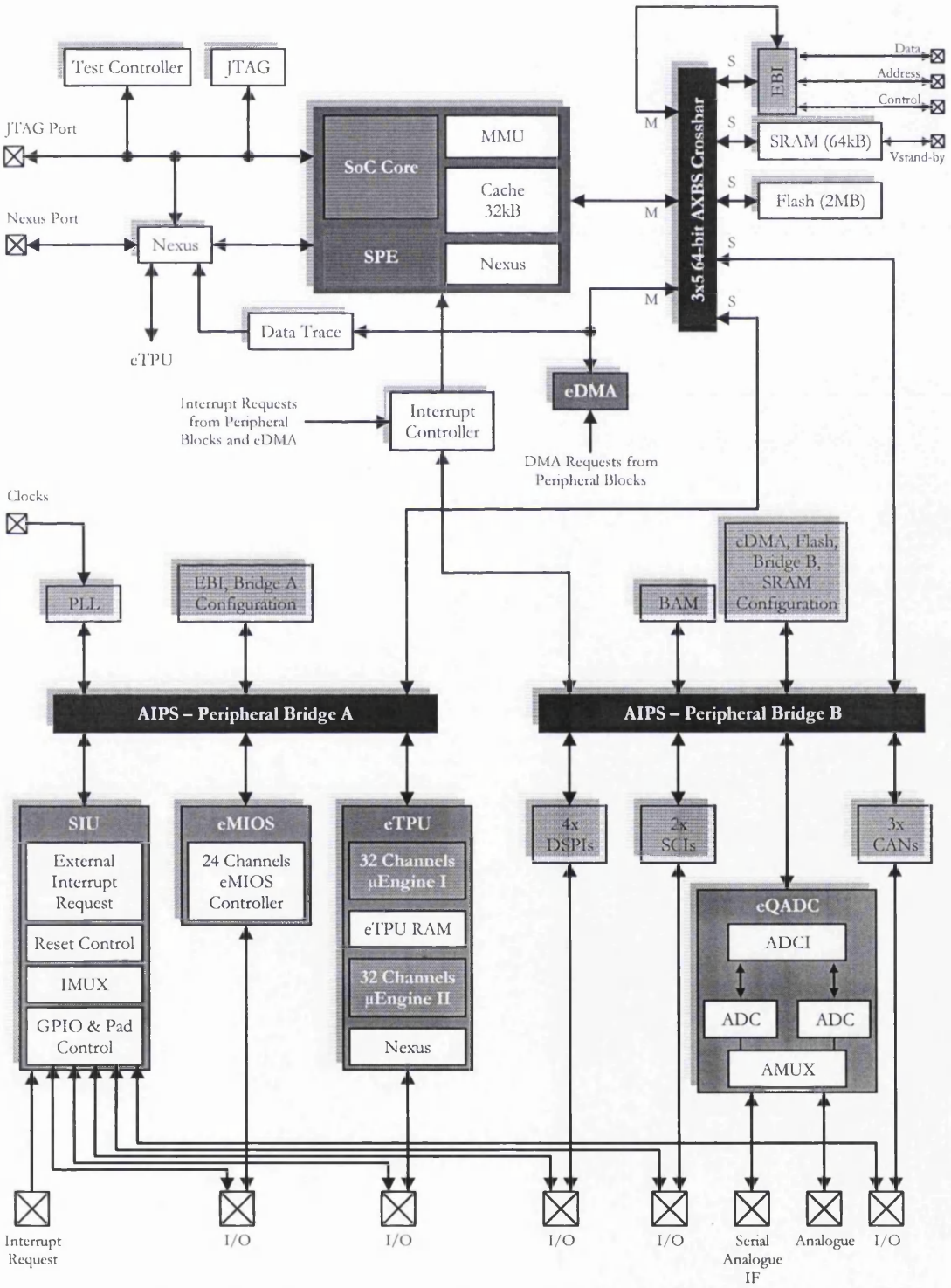

Figure 3.2: An Overview of the SIMD SoC Developed

3.3 An Architectural Modelling Environment for the PowerPC SoC

Over the years, core architectural simulation models have migrated to either fully functional models or to cycle accurate models [42, 43, 44, 45, 48]. Fully functional models are characterised by modelling system features without regard to hardware timing. Functional models are typically preferred by software developers since they tend to be the highest performance models available for developing application code. The drawback is that they offer very little information for performance evaluation. On the other hand, cycle accurate models provide very precise behaviour in regards to timing but tend to be relatively slow in performance due to the extreme detailed nature of the model [48, 81, 88]. The area of performance evaluation is in a middle ground; requiring the simulation performance of a functional model but needing the information provided by a cycle accurate model. Ideally, performance evaluation studies would like to simulate proposed system architectures over a wide range of benchmarks to determine system level performance, then adjust the system architecture and re-evaluate the performance.

The Architectural Modelling Environment (AME) targets the middle ground simulation requirements of performance evaluation by recognising that typical benchmark code does not require a fully functional model to run and that only a vital few key architectural elements determine a systems throughput [45, 48, 88, 12]. Typical benchmark code does not target a specific architecture which allows the benchmark to run across many different system implementations. Therefore, every *nook and cranny* of a fully functional simulation model is never used due to the generalised nature of benchmarks. Development time devoted to functional features which are never used by a benchmark is thus wasted effort in regards to performance evaluation. Similarly, most of the development time associated with creating a very precise, cycle accurate model is lost since a system's throughput is generally controlled by a very few architectural elements or bottlenecks.

While AME allows both fully functional and cycle accurate models to be developed, the intent of AME is to target the specific requirements of performance evaluation. The current area of study that AME is being used for is in architectural trade-off studies at the processor, cache, and memory interface level of system design or what is being called the core-complex for embedded SoC designs.

As shown in Figure 3.3 overleaf, in order to efficiently performance profile an application using the AME, the modules around it require the designer to work in two fields; on the one hand, the development of the software part including compiler, assembler, linker, and simulator and, on the other hand, the development of the target architecture itself. The AME produces the characteristics of the core architecture specific application binary performance and, thus, may answer questions concerning the instruction set, the performance of an algorithm, and the required size of memory and registers. The required silicon area or power consumption can only be determined in conjunction with a synthesisable hardware description language (HDL) model.

Core architectural analysis using the AME provides a unique opportunity to develop system metric estimation methodologies for power, area and performance in conjunction with low-level RT-synthesisable HDL for soft cores, netlist for firm cores and layout for hard cores. As an example, consider a UART (universal asynchronous receiver/transmitter) core implemented in synthesisable HDL, having its buffer size as a parameter. By performing gate-level simulation of UARTs with different buffer sizes, one can obtain area and toggle switch information for different parameter settings; likewise, after simulation of a core-based design at system-level, one can use low-level toggle data to accumulate total toggle counts and estimate power consumption of the design for a given technology.

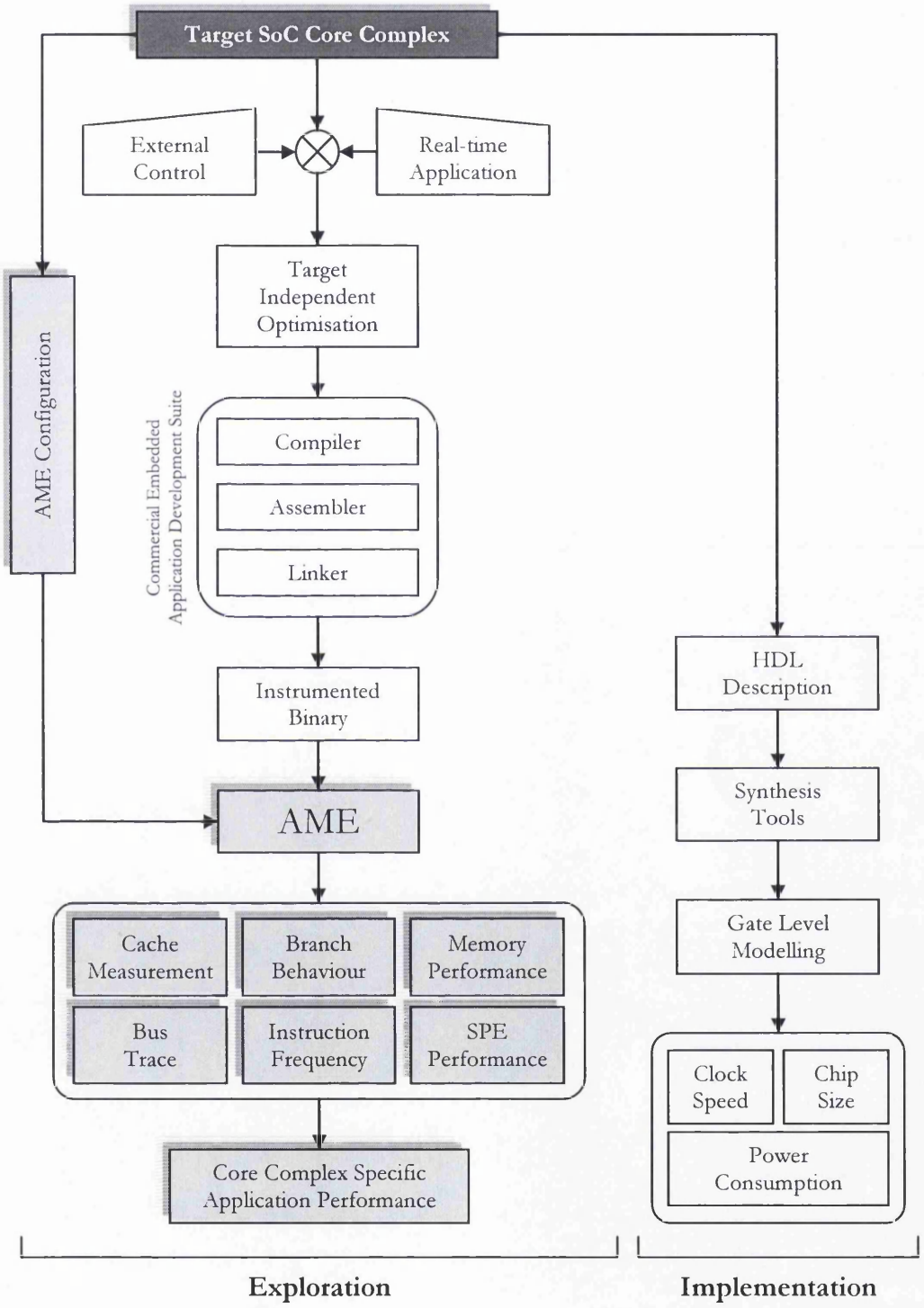

Figure 3.3: Application Binary Characterisation Process Using the AME

3.4 Electromagnetic Actuator Driven Camless Engines

3.4.1 Introduction

Conventional internal combustion engines use mechanically driven camshafts to actuate intake and exhaust valves. Like a very simple software program that contains only one set of instructions, such mechanical cams always open and close the valves at the same precise moment in each cylinder's constantly repeated cycle of fuel-air intake, compression, combustion, and exhaust [49]. They do so regardless of whether the engine is idling or spinning at maximum rpm. While this system is convenient and reliable, the fixed timing of the valve events with respect to the piston motion is typically selected as a compromise among fuel economy, emissions, maximum torque output, valvetrain noise, vibration and harshness [49, 50].

The growing need to improve fuel economy and particularly reduce emissions led to the introduction of an alternative valvetrain technology, namely a camless valvetrain [51]. Camless engines, employing electrohydraulic, electromagnetic or hydromechanical control of the valves, offer the next step in engine flexibility. Such engines allow independent control of valve timing and lift without mechanical linkage to the crankshaft [50, 51]. Various studies have shown that a camless valvetrain can alleviate many otherwise necessary engine design tradeoffs by supplying extra degrees of freedom to the overall powertrain system [50].

Automotive engines equipped with electromagnetic camless valvetrains have been studied for over thirty years but production worthy vehicles with engines of this type are still not available due to difficulties in ensuring adequate and reliable electromagnetic valve performance [1, 7, 50, 52, 53]. For an electromagnetic camless valvetrain (EMCV), the actuator noise caused by high contact velocities of the moving parts has been identified as a key problem [50, 52, 53].

Critical to the successful implementation of such a camless engine is the electromagnetic valve control strategy, its real-time software and hardware, satisfying performance and cost targets of the overall platform in a timely manner.

3.4.2 Construction and Operation of the Electromechanical Actuator

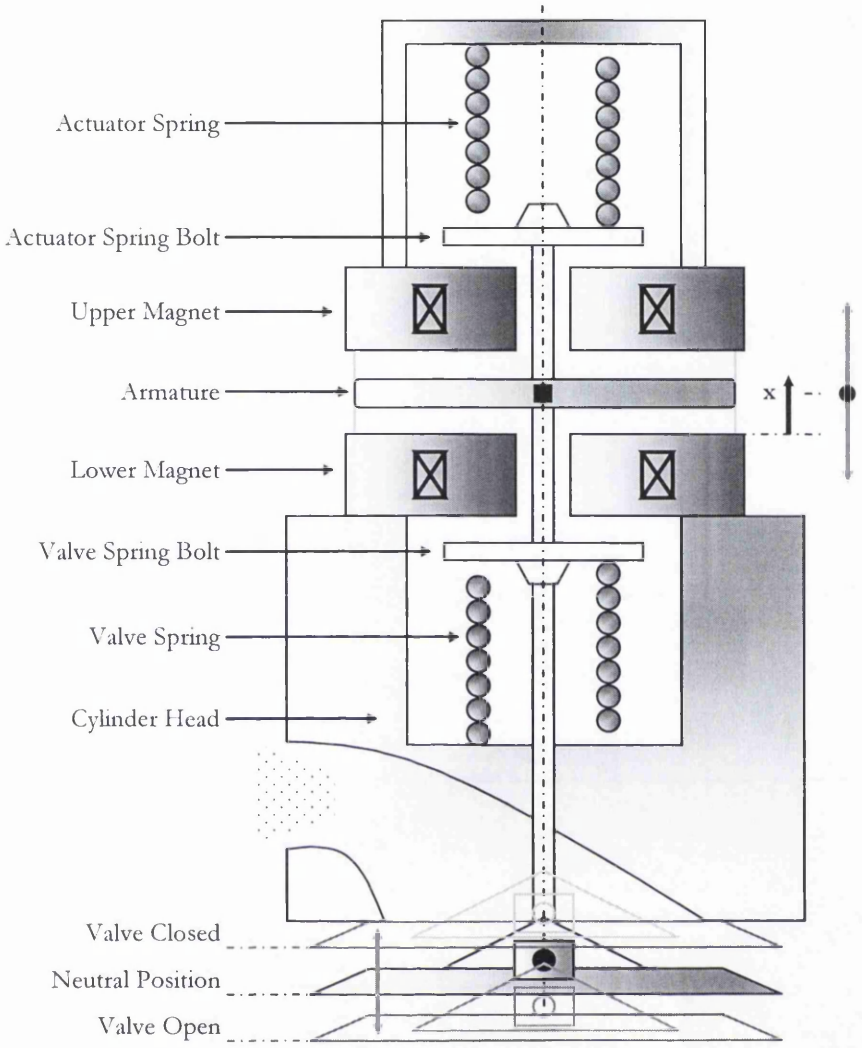

Figure 3.4: The Electromechanical Actuator

Figure 3.4 illustrates the actuator at open, neutral and closed positions. There are two magnets, two springs and an armature in the actuator. The two magnets are coil wound on ferromagnetic material. The coils are driven by currents generated by an electronic system, driven by a pulse-width modulated voltage. The activated coil generates a magnetic field applying a force on the armature. The two springs are adjusted such that both are always compressed for any position of the armature. The actuator uses the spring force to accelerate the masses, then uses the electromagnetic force to attract and dwell the valve. When there is no current on coils, the spring-mass system stays at the neutral position. Voltage applied on the upper coil closes the valve generating a holding current, which depends on the spring force and the pressure difference between the cylinder and the exhaust/intake manifold. Because of the symmetry, analysis is done only on the valve-opening event.

3.4.3 Design and Implementation of the Actuator Controller Platform

The PowerPC SoC based actuator control system is required to ensure accurate valve closing and opening events (timing). The engine management system deploys the timing strategy on the drivers torque demand and other vehicle variables [51, 52, 53]. One of the key objectives of the controller platform is to reduce the armature-coil and valve-cylinder contact velocities, the so called *soft-landing*, which in turn reduce noise and component wear [50, 51, 52, 53, 54]. Modern engine manufacturers design camshafts to achieve a low 0.04 m/s contact velocity at low engine speeds and in conventionally driven engines, this velocity increase linearly with engine speed [1, 3, 6, 9, 54, 55].

The electromechanical actuator operating in the harsh engine room is a highly nonlinear drive system but it is greatly welcome in high-speed and high accuracy applications such as camless engine control [50, 51, 52]. One of the most commonly used controller techniques for such actuator position control is the linear PI/PID[3] controller. For set point regulation, PID controller provides satisfactory performance. However under time-varying trajectory reference such as the one in the camless engine, the performance of the PID controller degrades due to its linearity characteristic [56].

In order to achieve better tracking of the desired valve position, an iterative learning controller (ILC) is implemented as it is a proven technique for improving the transient response of systems following the same trajectory motion or operation over and over [57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67].

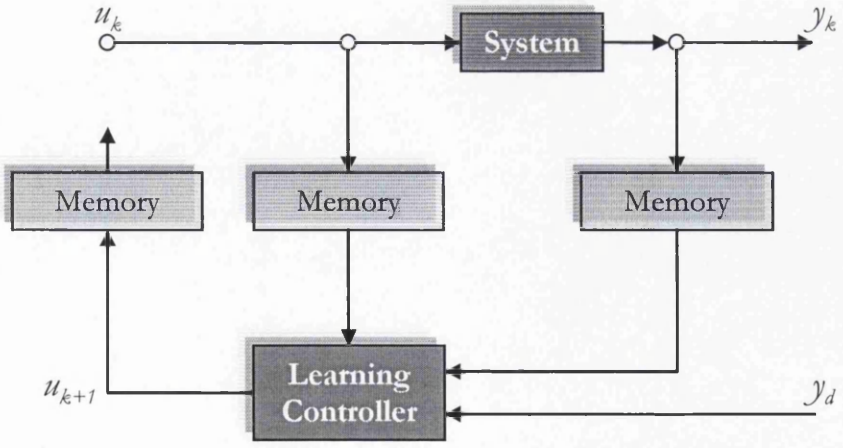

Figure 3.5: Configuration of the Actuator Iterative Learning Controller

[3] PID - Proportional, Integral, Derivative - A three mode control action where the controller has time proportioning, integral (auto reset) and derivative rate action.

The basic idea of the ILC implemented is illustrated in Figure 3.5. All the signals shown are assumed to be defined on a finite interval. The subscript k indicates the number of full armature travel cycles. During the k^{th} cycle, the input armature position $u_k(t)$ is applied to the system, producing the output $y_k(t)$. These signals are stored in the memory units until the trial is over, at which time they are processed by the ILC algorithm. Based on the error $(e_k(t) = y_d(t) - y_k(t))$ that is observed between the actual output and the desired output, the ILC algorithm computes a modified input signal $u_{k+1}(t)$ upon full armature travel that will be stored in memory until the next time the system operates, at which time this new input signal is applied to the system. This new input produces smaller error than the previous input.

The overall controller is modelled and implemented using Matlab®, Simulink® and Stateflow®. MathWorks' Real-Time Workshop is then used to generate executable stand-alone C code of the algorithm modelled in Simulink [68]. The resulting code is then hand optimised before generating the binary executable for the PowerPC SoC.

Primarily, the approach taken enables the use of the actuator controller:

1. As an embedded model within a control algorithm or observer
2. As a real-time engine model for hardware-in-the-loop testing
3. As a system model for evaluating engine sensor and actuator models
4. As a subsystem in a powertrain or vehicle dynamics model

3.4.4 The Camless Engine Actuator Controller Platform

Figure 3.6 illustrates the real-time system level development platform implemented for the evaluation of the electromechanical valvetrain. First of all, the maximum supply voltage is limited to 42-Volts to simulate the available voltage on a future vehicle. Two power amplifiers driven by the power supply drive the magnetic coils of the actuator. The SoC designed controls the voltage across the coil through a custom designed I/O board. A laser sensor is used to measure the actual valve position with a 5μm resolution. Coil current is also monitored for diagnostic purposes.

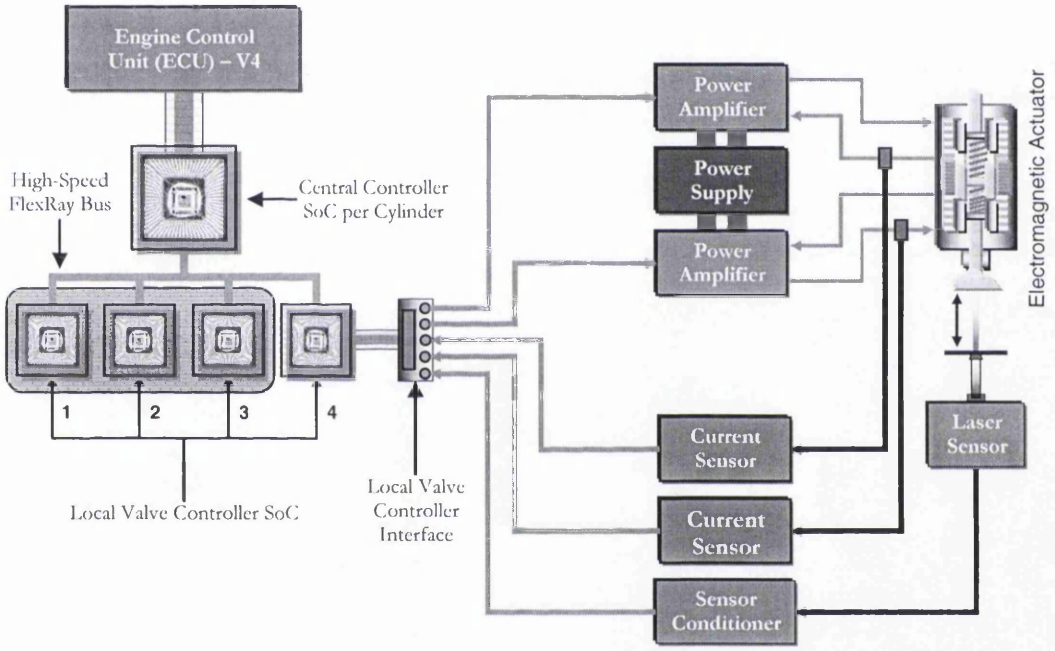

Figure 3.6: Experimental Set-up of the Actuator Controller Platform

This central controller SoC shown above receives its drive cycle information from the existing engine control unit (ECU). The local valve controller SoC is used for the execution of control algorithms and to drive the power stages. Other inputs to this controller include position of the actuators. Data transfer to actuator occurs at 25μs (40kHz) intervals.

3.4.5 Results and Concluding Remarks

An electromechanical camless valvetrain controller platform is realised based on the PowerPC SoC developed. Simulations and real-time measurements confirm the functional ability of the electromechanical actuator to vary valve timing, lift, velocity and event duration, as well as to perform cylinder deactivation in a four-valve multi-cylinder engine.

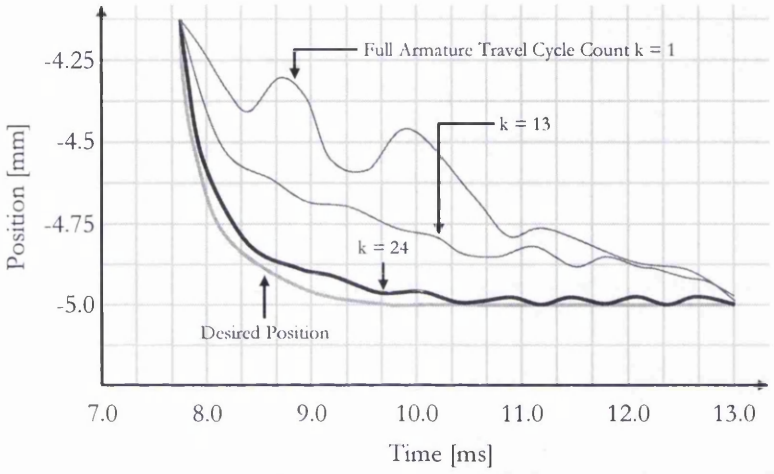

Figure 3.7: Valve Position Control Results based on the ILC (k ϵ [1, 13, 24])

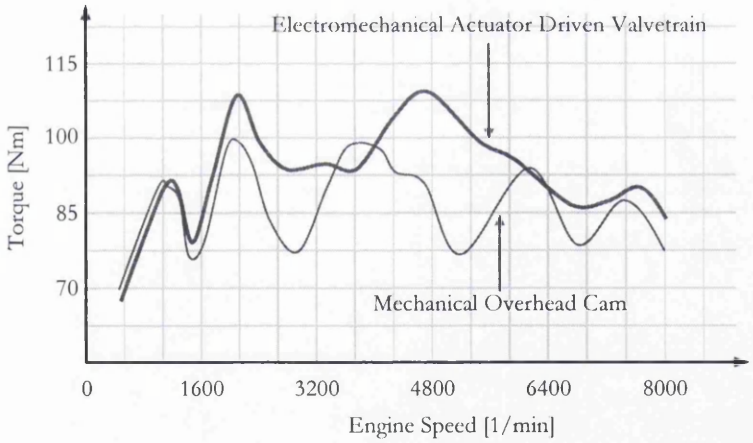

Figure 3.8: Torque Production with Mechanical Cam and Camless Engine

As shown in Figure 3.7, employing an ILC reduces the valve contact velocity as the armature cycles are increased. Measured results from [69] in Figure 3.8 confirms that the torque produced by the four cylinder engine, driven by an electromechanically driven valvetrain is 13% better than that of a classic dual overhead camshaft (DOHC) arrangement.

3.5 Embedded Von-Neumann On-Chip SoC Cache

A high performance, 32-kilobyte, 8-way set-associative, unified, aka von Neumann (instruction and data) cache with a 32-byte line size has been developed for the SoC. The enclosed SoC cache book chapter examines the new cache control instructions and provides the user with sample routines to invalidate the cache and relocate object code in memory using the cache. Most applications only use these cache control instructions during power-on initialisation and when necessary to flush cache contents to system memory. However, in time-critical code segments these instructions can often improve throughput via preloading of required cache contents and by reducing unnecessary transfers between external memory and the unified cache.

Measured results using author and customer developed powertrain applications confirm the developed cache improves system performance by providing low-latency data to the SoC core's instruction and data pipelines, which decouples processor performance from system memory performance. The cache is virtually indexed and physically tagged.

Due to areal overhead, the cache developed does not consist of dedicated hardware for enforcing coherency, and a software technique to accomplish this task is also proposed in the document.

This book chapter also discusses a technique as to how the cache can be used as local memory with minimal reconfiguration overhead using software. Particularly, inability to reuse data and under utilisation of cache capacity are responsible for poor cache performance on various commonly used automotive applications is highlighted.

3.6 Fast Internal Combustion Engine Knock Processing

3.6.1 Introduction

Knock in internal combustion engines (ICE) refer to the premature *self or auto* ignition of the air-fuel mixture in the engine when the unburnt mixture's temperature and pressure have exceeded a critical point. Frequent occurrence of this knock phenomenon causes permanent damage to the ICE and should be avoided. However, in order to obtain maximum power, modern engines are run at their borderline limit of incipient knock using closed-loop control of spark timing based on knock sensor feedback [49, 70, 71, 72].

The developed knock processing strategy is based on autonomous on-chip modules and an auxiliary signal processing extension to the main SoC core. Real-time software development techniques with an advanced software circular buffer implementation for processing the streaming knock sensor data have been developed. Various SIMD software optimisation techniques are employed to reduce the real-time knock algorithmic execution time. Real-time and simulation results are presented for the detection of knock on a four cylinder internal combustion engine, but, the approach is widely applicable.

3.6.2 Impact of Knock and its Real-Time Processing

Impact of knocking in an engine depends on its intensity and duration [72]. Trace knock has no significant effect on engine performance or durability. Heavy knock can lead to extensive engine damage [49, 70, 71]. The engine can be damaged by knock in different ways: piston ring sticking; breakage of the piston rings and lands; failure of the cylinder head gasket; cylinder head erosion; piston crown and top land erosion; piston melting and holing. Knocking is one important factor limiting the efficiency of an engine and is therefore of great importance to the engine manufacturers [49, 70, 71, 72, 73].

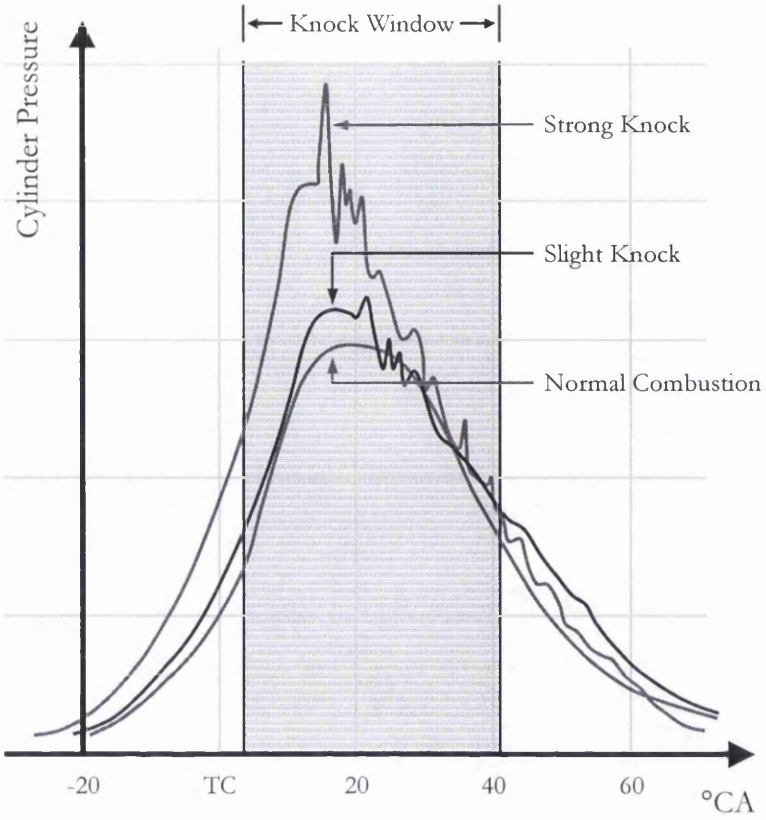

Figure 3.9: Cylinder Pressure versus Crank Angle (CA)

In severe cases, the piston may still be moving upwards to compress the air-fuel mixture. As such, it cannot move away to ease the build up of pressure wave. This results in severe stress on the engine and should be prevented as permanent engine damage can occur [49]. Figure 3.9 shows three plots of cylinder pressure against crank angle of a single cylinder engine with ignition timing three degrees apart between each trace.

There are several different approaches to detect the presence of knock in engines, see e.g. [73, 74, 75, 76, 77, 78, 79, 80]. One of the classic techniques presently used in production engines is based on application specific integrated circuits (ASIC) with limited programmability, such as the ProSAK™ knock control ASIC [81] and The HIP9011 ASIC [82].

Due to the high cost of direct knock sensors, most of the current knock detection systems are based on structural vibration signals obtained using an accelerometer [78].

3.6.3 Modelling the Pre-Silicon Knock Processing Platform

Functional and behavioural modelling of the overall knock processing strategy was carried out using the AME, Matlab, Simulink and Stateflow simulation platforms [68, 83], which was initially used as the primary demonstrator to customers before the development of the SoC based real-time environment. A top-down, modular design approach was taken with the overall implementation. The knock simulation platform is divided into hierarchical subsystems, making it more generic by separating engine and knock control system specific parameters.

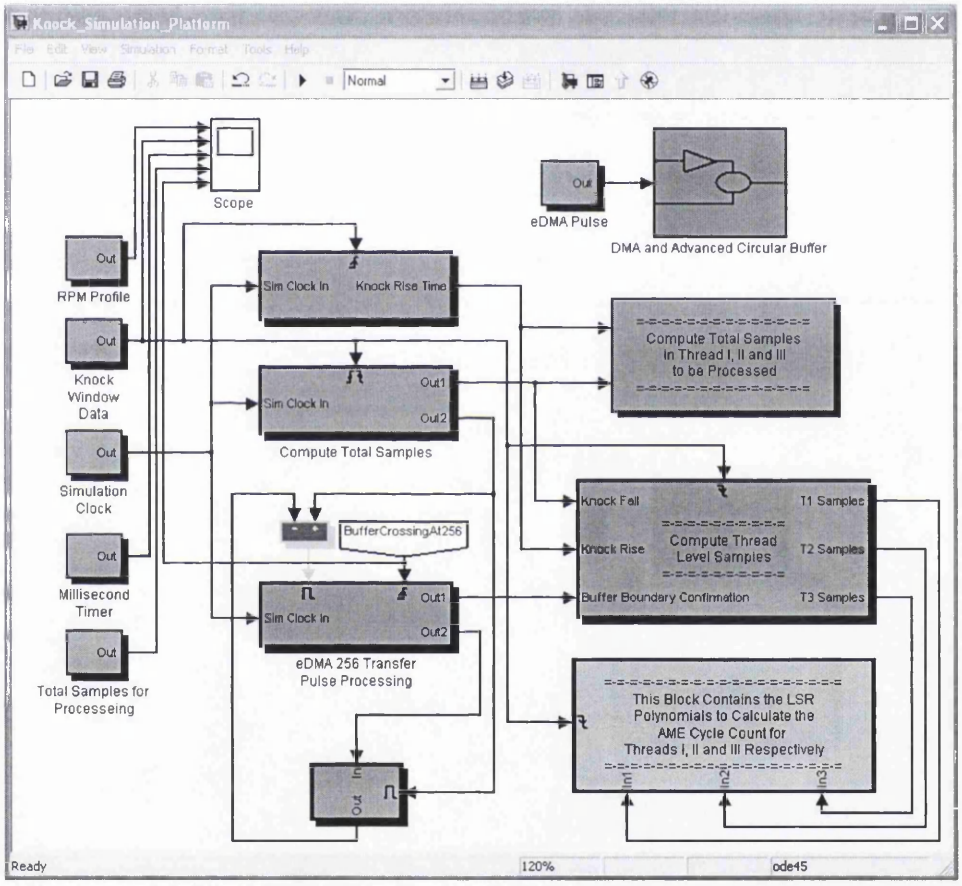

Figure 3.10: Pre-Silicon Knock Functional and Behavioural Simulation Platform

AME based linear and non-linear regressed performance analysis tables of the developed SIMD knock signal processing kernels are also incorporated in order to simulate different knock signal acquisition windows as shown in Figure 3.9. As a result, performance optimisation of the entire system, eliminating laborious programming and delivering substantial time was achieved.

3.6.4 SoC Based Hardware Knock Evaluation Platform

The following figure shows the fully populated printed circuit board (PCB), partly designed by the author, used for the real-time evaluation of the overall knock processing strategy developed. It employs an MC33394 Power Supply IC [81] and connectivity to other basic optional communication protocols available on the SoC as shown in Figure 3.2.

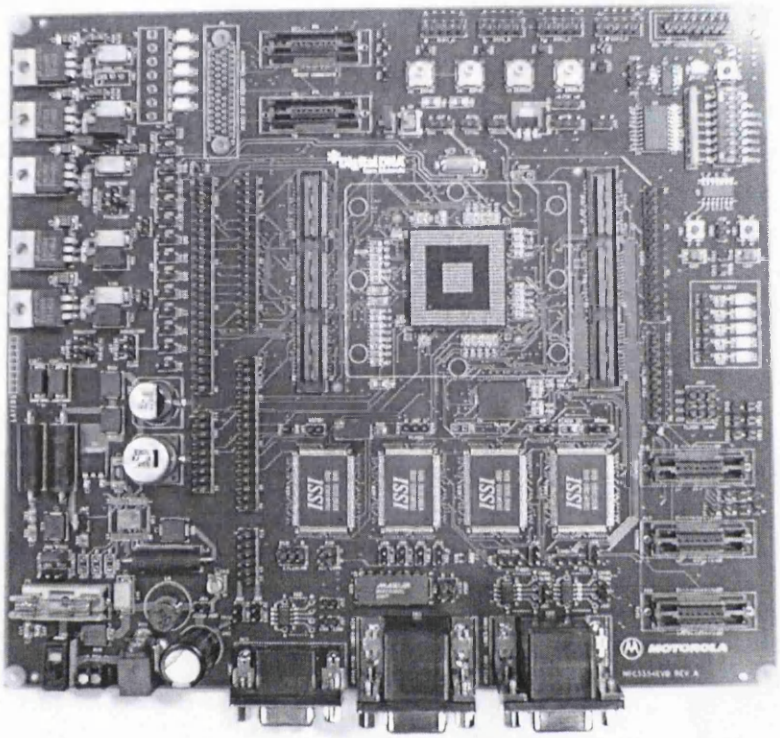

Figure 3.11: Photograph of fully populated knock hardware evaluation board

Where possible, components mounted on the PCB are automotive qualified to allow system evaluation over the full automotive temperature range (-40 °C to 125 °C).

3.6.5 Knock Processing Methodology – Data and Algorithmic Flow

Figure 3.12 illustrates the developed knock sensor data and the algorithmic flow. User programmed eQADC commands are contained in the on-chip memory in a user defined data structure. The eQADC command data is moved from the command queue to the command FIFO buffer by either the host CPU or by the enhanced Direct Memory Access (eDMA) controller. Once the command FIFO is triggered and is transferred into the ADCs on chip, the ADC executes the command, and the result, i.e. a pair of time stamp and data is moved through the result FIFO by the eDMA or the host CPU in to the circular buffer in the on chip memory.

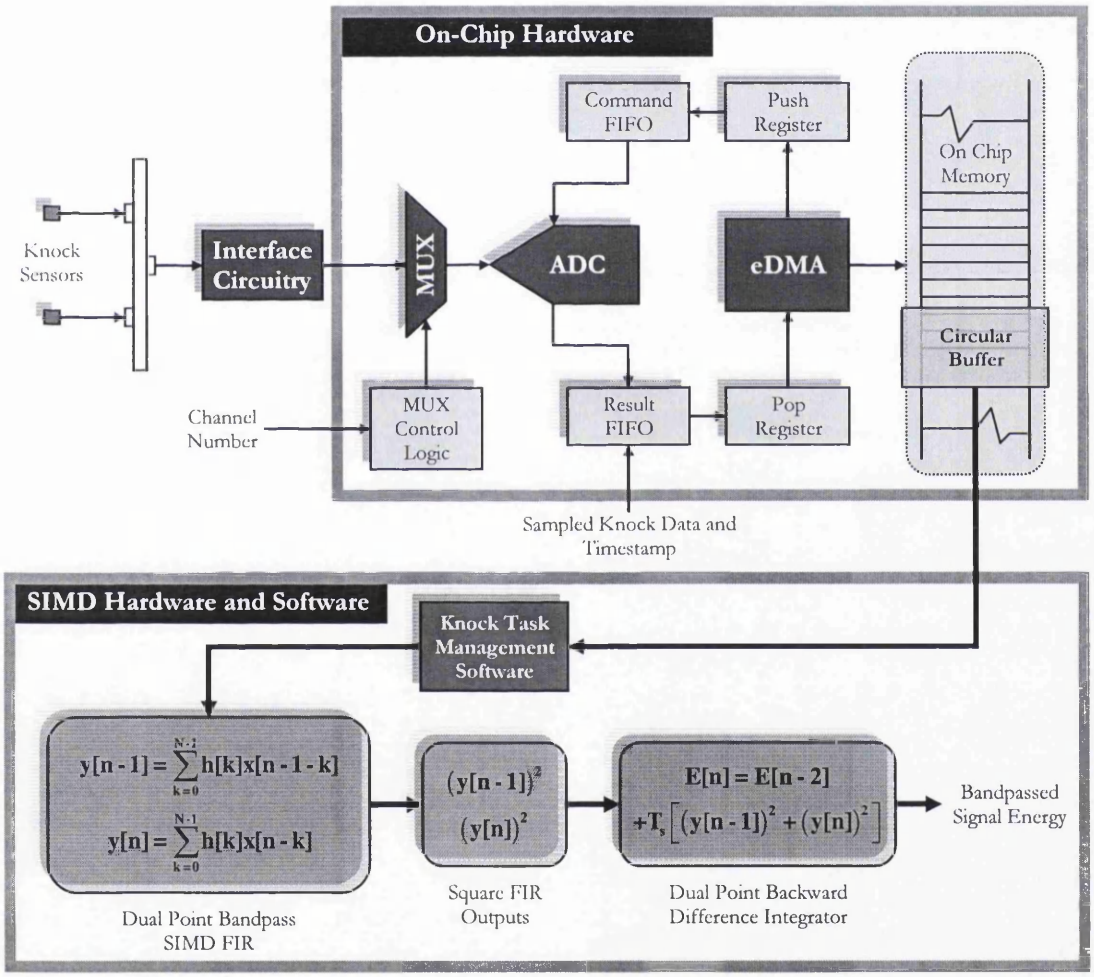

Figure 3.12: Data and Algorithmic Flow of the Developed SIMD Knock Kernel

The data in the circular buffer is then processed and presented by the knock task management software threads to the key SIMD knock signal energy extraction elements. The streaming data is then subjected to various SIMD signal processing elements in order to extract the signal of interest [84, 85].

3.6.6 Measured Results and conclusions

Streaming knock processing constitutes a significant part of current day average microprocessor workloads. To address this, a SoC combining SIMD DSP functionality with a classic microprocessor core has been developed. Peripherals on this SoC designed use fast register and memory based communication and synchronisation mechanisms to deliver high performance. Memory based communication and synchronization is realised using the eDMA module. Parallelism in this application is exploited using a combination of orthogonal parallel processing techniques, namely instruction and data level parallelism (ILP and DLP).

It has been shown that using a common architecture for both RISC and DSP instructions, in combination with autonomous on-chip peripherals, allows complex systems to be built around a single SoC platform, where previously two or more different processors would have been used together [78, 79, 80]. Based on the overall development strategy, it is also evident that real SIMD computers need to have a mixture of single instruction single data (SISD) and SIMD instructions. Importance of SISD elements in the micro-core to perform operations such as branches and address calculations that do not need parallel operation is also highlighted. It is also worth nothing that for efficient dynamic power management and flexibility, unused individual execution units of the SoC are disabled during algorithmic execution.

Thorough experimental analysis of the developed knock control platform confirms that SIMD works best in dealing with arrays of streaming data. Additionally, in the proposed architecture, sustained MAC instructions are executed in a single CPU cycle. In contrast, in a typical fixed-point microprocessor used in an engine control environment, a multiply and an add typically executes in 15 to 20 CPU cycles [81, 86, 87, 88, 89, 90].

The SIMD unit implemented also significantly increases execution speed by performing multiple operations in parallel. For instance, in the same instruction cycle that a MAC operation is performed, a parallel data move is carried out. SIMD enhancements in the SoC supplement the computational speed of present generation real-time processors used and make them ideal for high-performance real-time applications.

As shown in Figure 3.13, computational bandwidth is what separates the SIMD based core from the classic CPU – the ability to process an abundance of data, consistently, in an uninterrupted stream.

Measured performance results shown in Figure 3.13 confirms that the efficient coding and optimisation techniques used for the SIMD implementation of the knock kernel have improved performance by a minimum of x1.8.

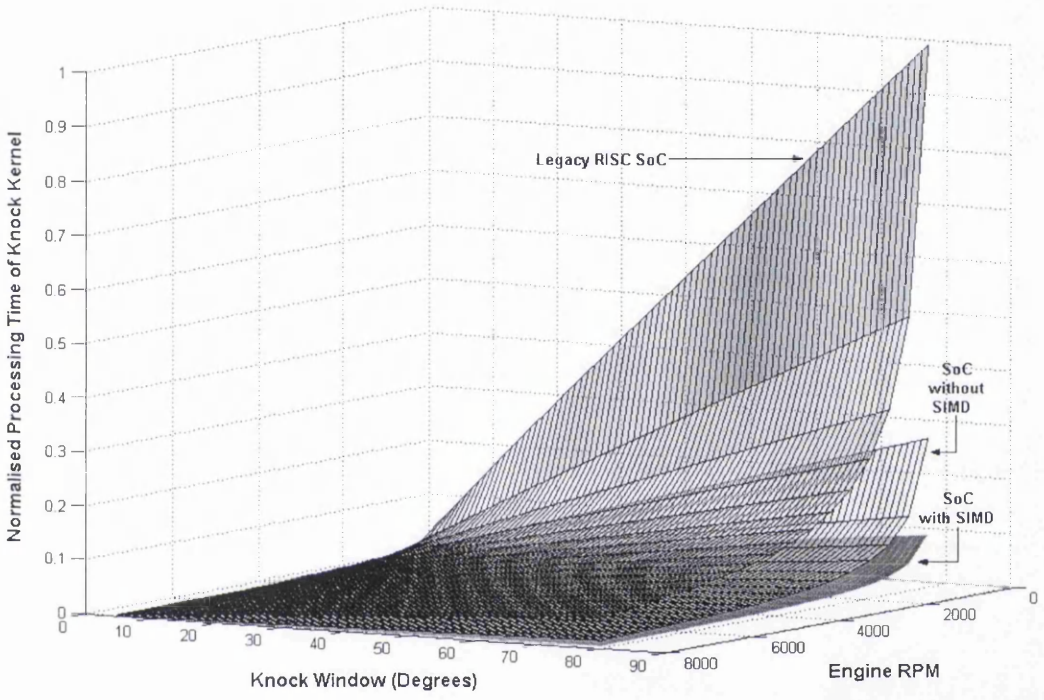

Figure 3.13: Knock Kernel Processing Time with and without SIMD

3.7 Next Generation Powertrain SoCs and Their Applications

3.7.1 Introduction

Ever increasing performance and increased on-chip memory requirements from various automotive powertrain applications radically increase the demand on powertrain controller overhead and the bandwidth required to run such applications [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 21, 22, 23, 24, 25, 26].

Powertrain applications not only take cost, performance and issues of real-time deterministic operation into account when choosing an SoC, but also lay emphasis on devices with sufficient flexibility and scalability to cope with complex, and modern development methodologies being employed. This document summarises the studies and benchmarking evaluations of customer and author developed application requirements for the natural successor to the developed SoC core.

In general, powertrain applications are driven by two main goals:

- Reduced fuel consumption
- Reduced emissions

These goals are to be achieved at the same or even higher engine power and, of course, at the same system cost level [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 21, 22, 23, 24, 25, 26].

Analysis done on the present customer applications and general trends confirm that new software development techniques presently being employed in powertrain applications increases the code size and requires higher processor bandwidth. Rapidly growing performance requirements are being primarily driven by applications developed with auto code generation using embedded real-time targets [1, 4, 6, 24, 25, 26], enhanced DSP functionality, model fitting to automatically generate calibration tables [24, 25], frequent use of specialist math functions, especially in areas like model based development [1, 4, 6, 22, 24, 26], along with ever increasing quality, reliability and safety.

3.7.2 Methodology and Challenges Faced

It is a very well known fact that implementing various number crunching and signal processing algorithms on a general-purpose powertrain CPU core can be very challenging [81, 87, 90]. Issues such as numeric formats and precision, type conversion, cache behaviour, dynamic instruction scheduling, and data-dependent instruction execution times pose hazards for high throughput powertrain applications, especially those running with tight real-time constraints [50, 51, 54, 55, 58, 70, 74, 81, 84, 85, 90,]. An appraisal of such application requirements is done by profiling the critical computational blocks within them using the real-time hardware evaluation board and the AME.

This report summarises the performance requirements of the natural successor core to the one already developed and appraises on the benchmarked and analysed wide range of author and customer developed real-time powertrain applications. Both automotive OEM and Tier I [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 22, 23, 24, 25, 26] application signatures profiled include:

- Table manipulation mechanisms that allows powertrain calibrations system to its most robust operating conditions
- Use of torque-spark curve to model port fuel injection (PFI)
- Fixed and floating point FIR filtering
- Fixed and floating point IIR filtering
- Various adaptive filters
- Fixed and floating point FFT including fast vector magnitude calculation
- Vector dot product
- Advanced motor control for x-by-wire
- Linear regression and radial basis function techniques

3.7.3 Suggestions and Conclusions

Based on the analysis done, it is evident that the applications evaluated require various enhancements to present SoC core in order to meet performance targets. Granular profiling of the applications signatures confirm that the features listed in the report are required to performance accelerate *fixed* and *single precision floating* point number crunching algorithms including DSP functionality without sacrificing a clean, intuitive programming model. The following summarises the minimum "standard" features required by the natural successor powertrain SoC platform:

- < ~70% of code density of present SoC to meet customer requirements and to particularly beat competition
- Zero-overhead looping mechanism supporting a minimum of 4K iterations
- Hardware cache coherency mechanism
- An enhanced branch target address and instruction cache (BTAC/BTIC) unit supporting speculative execution and branch prediction.
- Reasonably low capacity, on chip tightly coupled SRAM memory, aka scratch pad memory is required to place and manipulate frequently used data. This memory will be used to create circular buffers as developed in the SIMD based streaming knock processing methodology.
- Multiple hardware single precision floating point multiply accumulate (SPFPMAC) operations per cycle
- Modulo addressing mode supporting circular buffer management
- Misaligned load store support in SIMD
- Fast hardware math functions, particularly single precision squareroot
- Enhanced memory management unit (MMU) granularity
- 100% binary compatibility with the existing core
- Faster bit reversing hardware for FFT
- Wider issue mechanism
- Most of all it would be ideal to have a super compiler to vectorise and exploit full potential of SIMD
- Multithreading and Multiprocessing – If the required performance can be met with wider issue, this would obviously be the successor to present core

3.8 Portfolio Conclusions and Future Direction

The themed portfolio has examined and proposed strategies for the rapid and efficient development of high-performance automotive powertrain SoCs and their applications. It has been shown how to employ the first-of-its-kind SIMD based SoC micro-architecture to design and develop applications consisting of complex algorithmic control structures and strategies depending on the desired performance, flexibility and particularly cost. The portfolio developed two main novel implementation strategies to support camless engine control and advanced knock processing based on the SIMD SoC. Various next generation SoC core elements required have also been proposed based on thorough profiling of advanced powertrain applications, allowing SoC designers to make hardware and software trade-offs without having to enter the detail design process.

3.8.1 Summary of Industrial Research Undertaken

Business Plan for the SoC Development of Camless Engine Control

Automotive powertrain controllers are undergoing a drastic transition, from the traditional RISC based microcontrollers to technologically advanced SoCs with increased DSP functionality tightly coupled to the micro-architecture in order to support advanced applications requirements. As the SoCs being introduced are challenged to continue this evolution, the business model established by the author implements the necessary changes to remain a competitive, and to execute the proposed strategies in a manner that ensures continued and improved support and profitability. All efforts required by the proposed business are closely managed to ensure technological advances are not sacrificed and responsibilities are not compromised.

The SoC developed uses heterogeneous computational fabrics to meet conflicting requirements with respect to cost, performance and flexibility related to the camless engine controller market. Two variants of the SoC, targeting two types of camless engine controller platform have already been introduced to the market. Seven patents are initially incorporated. In year three, $26 million in sales with approximately $11 million gross profit is projected. It is expected that the gross profit after year six would be approximately $81 million.

The camless market segments are clearly defined and all are subject to a high growth trend. Close collaboration with engine technologist in the field has improved upon the overall marketing strategy significantly. A new, particularly low-cost and innovative design has been created to answer the needs of the ubiquitous low-end camless market. This market should begin at $61 million but could expand to several hundred million as soon as the technology is fully accepted. The SoC business becomes mature in year four with gross margins of approximately 50% producing $47 million in gross earnings.

The keys to success for the camless engine controller SoC business are as follows:

- Initial capitalisation obtained
- SoC for the high-end camless engine controllers have been introduced
- All patent applications filed
- The ability to generate early revenue from the low-end market is established
- Top-notch team is in place for successful market roll-out
- Successful implementation of sales and marketing plan to the leading automotive markets in Asia, Europe and the U.S
- Increased product development with technological breakthroughs and continued market share gains to produce a $47 million revenue by year five

An Architectural Modelling Environment for the PowerPC SoC

The diversity of powertrain applications coupled with the radically decreasing time to market is creating the need for new tools to support rapid SoC design at a level of abstraction above the register-transfer level. The architectural modelling environment developed consists of a library of author developed reusable performance models that correspond to the core architectural elements of the powertrain SoC roadmap plus a facility to rapidly assemble these elements into a complete performance model for a new SoC design. The environment helps the successful introduction of advanced powertrain SoCs facilitating the trade-offs between hardware and software elements of all applications targeted.

All application binaries are profiled and their performance metrics on the SoC are collected during simulation by the author. On the AME, the user can always trade simulation performance against depth of profiling by selecting only the relevant parameters, for example the wait states and branching mechanisms. Tight integration of bus and software analysis with the core complex lets the tracing of the segment of application currently executed and what bus transactions are initiated.

These powerful profiling capabilities introduced in the AME enables the determination of exact SoC core complex requirements. The complex UNIX based APIs encapsulated allows complete control over the SIMD micro-architecture efficiently.

Development of a Camless Engine Controller Platform

Based on the extensive research and development undertaken by the author and engine technology trends, it is evident that camless technology is the future for internal combustion engines. It has been shown by the author that electronic control of engine valves independently using electromagnetic actuation yields benefits such as improving torque production and driveability, increasing fuel economy, reducing weight and decreasing emissions. For example, first-of-its-kind, Honda's mechanical cam based *limited* "variable valve timing and electronic lift control" (VTEC) module first came out in 1991 and was used in the NSX model [3]. This VTEC module is still being incorporated in various Honda engines due to its successful demonstration of the increased engine torque production with lower emissions.

A low cost, automotive qualified SoC based distributed camless engine controller platform, meeting all performance requirements has been developed. The complexity problem of inter SoC communications has also been addressed using FlexRay [11] in order to cope with high data traffic that cannot be supported by existing communications protocols such as CAN [23]. An electromechanical valve actuator model was developed and experiments were used to identify unknown model parameters and functions and to validate the model predictions.

Actuators are programmed to do the same task repeatedly which enables the observation of the control error in successive iterations. An ILC has been developed by the author in order to compensate for and, hence, remove this repetitive error. It has been shown that using such an ILC algorithm, allows the valve motion to approximately converge to a desired trajectory achieving a landing velocity of 0.065m/s in twenty four cycles, potentially eliminating undesired valve landing transients, the so called *soft-landing.*

Overall, the principal emphasis was laid on the modelling, analysis, and SoC hardware linkable real-time simulation for the camless engine controller platform. Such a hardware-in-the-loop environment deployed enables the cost-effective evaluation of new SoC technologies and control strategies. Use of such a camless engine controller development platform promises shorter development time and concomitant reduction of system cost.

Fast Internal Combustion Engine Knock Processing

It has been shown that frequent occurrence of the knock phenomenon in an internal combustion engine causes permanent damage. However, in order to obtain maximum power, modern engines are run at their borderline limit of incipient knock using closed-loop control of spark timing based on knock sensor feedback.

Streaming knock processing constitutes a significant part of current day average powertrain microprocessor workloads. To address this, an automotive SoC combining tightly coupled DSP functionality with SIMD has been developed. Peripherals on this SoC designed use fast register and memory based communication and synchronisation mechanisms to deliver high performance. Memory based communication and synchronisation is realised using the eDMA module.

A novel high performance knock detection strategy based on this SoC has been developed by the author encapsulating the optimal use of relevant autonomous on-chip modules.

In the developed SoC micro-architecture, it has been experimentally shown by the author that SIMD works best in dealing with arrays of streaming data. Hence, to have the opportunity for massive parallelism in SIMD there must be massive amounts of data, or data parallelism. It was shown that the performance of the SIMD based knock processing kernel is higher than 80% compared to that of hand optimised classic RISC/non-SIMD based approach.

The C/C++ interface introduce by the author to access the SIMD elements of the SoC micro-architecture eliminates the issues associated with writing code at the assembly level: register allocation, scheduling, stack management and conformance to the underlying application binary interface (ABI). It has been shown that the intrinsic interface developed enables the embedded compiler to optimise the instruction scheduling.

Overall, for the automotive powertrain environment, the change from standard scalar 32-bit to SIMD vector 64-bit register based cores has been established, allowing the implementation of advanced algorithms with minimal processing time.

Embedded Von-Neumann On-Chip PowerPC SoC Cache

It has been shown by the author that the performance of the unified cache module incorporated on the SoC is critical to alleviate the increasing gap between the SoC core and on-chip memory speed. The first-of-its-kind cache implemented on the SoC improves overall system performance by providing low-latency data to the core instruction and data pipelines. However, it was confirmed that poor reuse of data, conflicts between various references and underutilisation of cache capacity lead to poor cache performance for various commonly used applications.

Software techniques have been introduced by the author to explicitly accomplish cache coherency as the cache does not have dedicated hardware enforcing coherency. In conjunction with the compiler, with no changes to existing instruction set, it has been shown by the author that cache can be used as local memory with minimal overhead. Data pre-fetching, blocking, and data copying techniques have been also been introduced to improve overall cache performance.

It was established by the author that the best general approach to making good use of the cache module is to keep the performance critical elements of applications as small as possible so that it can completely be loaded onto the cache module. Particularly, linked lists intensive applications are bad for the cache due to the widely spread out memory accesses.

Hardware configurable, load and lock mechanism has also been implemented on the cache in order to guarantee the availability of instruction and data. This feature is particularly useful for critical interrupt handler routines in order to provide fast and guaranteed interrupt performance.

Next Generation Powertrain SoC Performance Requirements

It has been established by the author that performance evaluation of the developed powertrain SIMD SoC core is more complex than the classic RISC machines. The overall performance of powertrain applications profiled by the author depends on a number of performance metrics such as the average instruction length, cache configuration, branching mechanisms and the number of memory references per instruction. The analysis conducted by the author is used for the solidification of future powertrain SoC core architecture and their embedded compilers.

Future Work

1. The first step is to establish a stronger and particularly a realistic vision for the SoC business targeting the *low-end* engine market for both camless engine controllers and knock processing platforms.

2. Though it may be impossible, it would first and foremost be useful to identify all disturbances generated by the engine as a whole. In theory, the ILC algorithm implemented fulfils the stability conditions finding the optimal input that gives zero error for the actuator, but, it is not clear as to what would happen due to disturbances from the engine as a whole. Ideally a controller needs to be modelled compensating for gas forces acting on the valve due to the combustion in the cylinder and any valve lash that may be present to compensate for the thermal expansion between the valve stem and the armature. Clearly more research needs to be done to resolve this. It is also necessary to investigate the possibilities of implementing a controller to converge with lower number of iterations. Additionally, further analysis is required to determine the wear levels of the electromagnetic actuator driven engines as with conventional camshaft driven engines.

3. All SIMD code has been implemented manually giving direct control over the performance of the SIMD execution. Although this is the best way to obtain highest performance, from the user's point of view and as an ultimate research goal, it would be ideal to have an advanced embedded compiler that could vectorise legacy scalar 32-bit code efficiently in order to realise the full benefits of the underlying SIMD architecture. It is evident that all leading embedded automotive application compilers used in all development work done is far from this goal. Clearly this needs to be addressed in order to substantially reduce development time.

4. Although design automation tools for system development used were very helpful for developing complex and time critical embedded systems and applications, there is still plenty of room to augment these tools with more specific support for subsystem interfacing, improved synthesis capabilities, inclusion of more generic libraries preferably including fault models and support for additional analysis capabilities, especially with regard to inherent robustness. I personally consider an open tool architecture to be critical for achieving these goals. It should either provide a mechanism for linking control code with ease, as in Stateflow, or tool boxes as in Matlab.

5. The magnetisation curves, in principle, contain all the information that is needed to operate the electromagnetic actuator. With constant advancements in automotive qualified SoCs with increasing DSP functionality such as the one developed, investigations on suitable control strategies need to be developed for sensorless control providing opportunities for greater flexibility, higher productivity and particularly lower total life cycle costs.

6. Development of cost effective technology to integrate the robust power circuits needed to power actuators and intelligent control circuits on the same SoC.

7. Most of the pre-silicon performance analysis was done using AME, primarily supporting functional and behavioural simulation of the core complex. However, in order to further reduce development time, a complete SoC simulation platform needs to be developed supporting the entire target hardware including external I/O and on-chip peripherals.

8. My personal experience confirms that there is room for improvement with early SoC integration. Generally, such SoCs are either over-designed or fail to fulfil the performance specification initially guaranteed. Especially the performance related to the software elements, for example, CPU load, impact of the RTOS or software response time - can hardly be analysed without the availability of a robust cycle accurate simulation model. Furthermore, the software performance is heavily affected by the communication and memory architecture, so an isolated consideration of a single SoC may hide potential bottlenecks due to bus use and memory access latency. Therefore, such issues have to be addressed as early in the design flow as possible to prevent late and costly changes of the architecture specification.

9. In the powertrain application domain, further real-time evaluations particularly run-time on-chip analysis of memory access patterns needs to be done. Such heuristics could facilitate the determination of application domain specific optimal caching strategy including the attainment of a faster coherent cache.

10. And last but not least, great emphasis needs to be laid on the transitioning to 42-volts from the current 14-volt automotive bus which is clearly insufficient to run future applications like camless engine controllers. It is a well known fact that increased power, greater comfort and greater safety features can easily be realised with a 42-volt bus, necessitating the immediate standardisation of the 42-volt bus in unison with research work. It is evident that much work has been done and much remains to do.

3.9 References

[1] *BMW (Bayerische Motoren Werke)*, Web home-page: http://www.bmw.com

[2] DaimlerChrysler, Web home-page: http://www.daimlerchrysler.com/

[3] Honda Automobiles, Web home-page: http://automobiles.honda.com/

[4] Toyota, Web home-page: http://www.toyota.com/

[5] *AUDI (Auto Union Deutschland Ingolstadt)*, Web home-page: http://www.audi.com/

[6] Volvo Car Corporation, Web home-page: http://www.volvocars.com/

[7] General Motors Corporation, Web home-page: http://www.gm.com/

[8] Ford Motor Company, Web home-page: http://www.ford.com/

[9] Renault, Web home-page: http://www.renault.com/

[10] Peugeot, Web home-page: http://www.peugeot.com/

[11] FlexRay, next generation fault-tolerant automotive communications protocol, Web home-page: http://www.flexray.com/

[12] J. L. Hennessy and D. A. Patterson, "Computer Architecture: A Quantitative Approach", Morgan Kaufmann, Second Edition, 1996.

[13] J. Tyler, J. Lent, A. Mather, and Huy Nguyen, "AltiVec™: bringing vector technology to the PowerPC™ processor family Performance," proceedings of *IEEE International Computing and Communications Conference, (IPCCC '99)*, pp. 437– 444, (February 10-12, 1999).

[14] R. Jaenicke, "An open systems approach to real-time signal processing using AltiVec technology," proceedings of the *Digital Avionics Systems Conference,* Vol. 1, No. 18, pp. 2.B.4-1–2.B.4-8, (October 24-29, 1999).

[15] S. Fuller, "Motorola PowerPC™ microprocessor with AltiVec™ technology – the system engineer's answer to revolutionizing life cycle COTS design paradigms with software-based high performance computing," proceedings of the *Digital Avionics Systems Conference*, Vol. 1, No. 18, pp. C.4-1–1.C.4-8, (October 24-29, 999).

[16] P. Biswas, A. Hasegawa, S. Mandaville, M. Debbage, A. Sturges, F. Arakawa, Y. Saito, and K. Uchiyama, SH-5: the 64 bit superH architecture; *IEEE* Micro, Vol. 14, Issue. 4, pp. 28–39, (July-August 2000).

[17] *NASA (National Aeronautics and Space Administration)*, Web home-page: http://www.nasa.gov/

[18] *DTIC (Defense Technical Information Center)*, Web home-page: http://www.dtic.mil/

[19] *The Boeing Company*, Web home-page: http://www.boeing.com/

[20] *Airbus*, Web home-page: http://www.airbus.com/

[21] *Continental Automotive Systems*, producer of systems, components and engineering services for vehicle safety, comfort and powertrain performance, Web home-page: http://www.conti-online.com/

[22] *Siemens VDO Automotive*, one of the world's leading suppliers of electronics, electrical products and mechatronics to the automotive industry, Web home-page: http://www.siemensvdo.com/

[23] *Bosch Automotive Technology*, producer of automotive technology products for gasoline, diesel and chassis systems and car electronics, Web home-page: http://www.bosch.com/

[24] *CRF (Centro Ricerche Fiat)*, providers of architectures, technologies and materials for internal combustion engines using gasoline, diesel and natural gases, Web home-page: http://www.crf.it/

[25] *Denso Corporation*, global supplier of automotive technology, systems and components, Web home-page: http://www.globaldenso.com/en/

[26] *Visteon Corporation*, global supplier of automotive technology, systems and components, Web home-page: http://www.visteon.com/

[27] M. Deacon, C. J. Brace, M. Guebeli, N. D. Vaughan, C. R. Burrows and R. E. Dorey, "A modular approach to the computer simulation of a passenger car powertrain incorporating a diesel engine and continuously variable transmission," proceedings of the *International Control Conference*, Vol. 1, pp. 320 – 325, (March 21-24, 1994).

[28] S. Liu and T. R. Bewley, "Adjoint-based system identification and feedforward control optimization in automotive powertrain subsystems," proceedings of the *American Control Conference*, Vol. 3, pp. 2566 – 2571, (June 4-6, 2003).

[29] N. Sivashankar and K. Butts, "A modeling environment for production powertrain controller development," proceedings of the IEEE *International Symposium on Computer Aided Control System Design*, pp. 563 – 568, (August 22-27, 1999).

[30] L. Mianzo, "A transmission model for hardware-in-the-loop powertrain control system software development," proceedings of the IEEE *International Conference on Control Applications*, pp. 1 – 8, (September 25-27, 2000).

[31] P. R. Crossley and J. A. Cook, "A nonlinear engine model for drivetrain system development," proceedings of the *International Control Conference*, Vol. 2, pp. 921 – 925, (March 25-28, 1991).

[32] R. K. Stobart, A. May, B. Challen and T. Morel, "Engine and control system modelling to reduce powertrain development risk," proceeding of the IEE *Colloquium on System Control Integration and Rapid Prototyping in the Automotive Industry*, Digest No. 1997/388, pp. 2/1 – 2/4, (December 09, 1997).

[33] K. R. Butts, "An application of integrated CASE/CACSD to automotive powertrain systems," proceedings of the IEEE *International Symposium on Computer-Aided Control System Design*, pp. 339 – 345, (September 15-18, 1996).

[34] J. A. Cook, Sun Jing and J. W. Grizzle, "Opportunities in automotive powertrain control applications," proceedings of the *International Conference on Control Applications*, Vol. 1, pp. xlii – xlli, (September 20, 2002).

[35] S. Liu and A. G. Stefanopoulou, "Effects of control structure on performance for an automotive powertrain with a continuously variable transmission," IEEE *Transactions on Control Systems Technology*, Vol. 10 , Issue. 5, pp. 701 – 708, (September 2002).

[36] Hu Xiaobo, J. G. D'Ambrosio, B. T. Murray and Tang Dah-Lain, "Co-design of architectures for automotive powertrain modules," IEEE *Micro*, Vol. 14, Issue. 4, pp. 17 – 25, (August 1994).

[37] I. V. Kolmanovsky and A. G. Stefanopoulou, "Optimal control techniques for assessing feasibility and defining subsystem level requirements: an automotive case study," IEEE *Transactions on Control Systems Technology*, Vol. 9, Issue. 3, pp. 524 – 534, (May 2001).

[38] W. J. Fleming, "Overview of automotive sensors," IEEE *Sensors Journal*, Vol. 1, Issue. 4, pp. 296 – 308, (December 2001).

[39] J. Nash and P. Smith, "An analysis of the design processes required for the technology conversion of SoC intellectual property," proceedings of IEEE *Custom Integrated Circuits Conference*, pp. 525 – 527, (May 2000).

[40] M. Lajolo, A. Raghunathan, S. Dey and L. Lavagno, "Efficient power co-estimation techniques for system-on-chip design," proceedings of the *Design, Automation and Test in Europe Conference and Exhibition*, pp. 27 – 34, (March 27 – 30, 2000).

[41] The Time Triggered Protocol (TTP), developers and producers of aerospace, automotive, and industrial control and communications equipment, Web home-page: http://www.tttech.com/

[42] WindRiver, providers of high performance compilers, software development tools and real-time operating systems for developers of embedded systems, Web home-page: http://www.widriver.com/

[43] Green Hills Software Inc., providers of high performance compilers, software development tools and real-time operating systems for developers of embedded systems, Web home-page: http://www.ghs.com/

[44] Metrowerks, providers of high performance compilers, software development tools and real-time operating systems for developers of embedded systems, Web home-page: http://www.metrowerks.com/

[45] *ARM (Advanced RISC Machines)*, designers of digital products for wireless, networking, consumer entertainment, imaging, automotive, security and storage devices, Web home-page: http://www.arm.com/

[46] IAR Systems, suppliers of development tools and services for embedded systems, Web home-page: http://www.iar.com/

[47] Ricardo plc, engineering technology provider, undertaking research, design, development and strategic services to automotive manufacturers, Web home-page: http://www.ricardo.com/

[48] *IBM (International Business Machines)*, Web home-page: http://www.ibm.com/

[49] J. B. Heywood, "Internal Combustion Engine Fundamentals," New York: McGraw-Hill, (1988).

[50] T. Ahmad and M. A. Theobald, "A Survey of Variable Valve Actuation Technology," *SAE Transactions*, Paper No. 891674.

[51] R. Flierl and M. Kluting, "The Third Generation of Valvetrains for Throttle-Free Load Control," *SAE 2000 World Congress*, SAE Technical Paper Series, Paper No. 2000-01-1227.

[52] C. Gray, "A Review of Variable Engine Valve Timing," SAE *Transactions*, Paper No. 880386.

[53] Yan Wang, Anna Stefanopoulou, "Modelling of an Electromechanical Valve Actuator for a Camless Engine," proceedings of the *5th International Symposium on Advanced Vehicle Control*, pp. 601–608, (August 22–24, 2000).

[54] S. Butzmann, J. Melbert, and A. Koch, "Sensorless Control of Electromagnetic Actuators for Variable Valve Train," *SAE Transactions*, Paper No. 2000-01-1225.

[55] Dr. Martin Pischinger, "A New Opening," *Engine Technology International*, pp. 20–22, (July 2001).

[56] C. H. Lim, Y. K. Tan, S. K. Panda and J. X. Xu, "Position control of linear permanent magnet BLDC servo using iterative learning control," *International Conference on Power System Technology 2004*, PowerCon 2004, Volume 2, 21-24 Nov. 2004, pp. 1936 - 1941

[57] S. R. Oh, Z. Bien, and I. H. Suh, "An iterative learning control method with application to robot manipulators," IEEE *Journal of Robotics and Automation*, Vol. 4, Issue. 5, pp. 508 – 514, (October 1988).

[58] Tae-Yong Kuc, Kwanghee Nam and J. S. Lee, "An iterative learning control of robot manipulators," IEEE *Transactions on Robotics and Automation*, Vol. 7, Issue. 6, pp. 835 – 842, (December 1991).

[59] Han-Fu Chen, Hai-Tao Fang, "Output tracking for nonlinear stochastic systems by iterative learning control," IEEE *Transactions on Automatic Control*, Vol. 49, Issue. 4, pp. 583 – 588, (April 2004).

[60] M. Pandit and K. H. Buchheit, "Optimizing iterative learning control of cyclic production processes with application to extruders," IEEE *Transactions on Control Systems Technology*, Vol. 7, Issue. 3, pp. 382 – 390, (May 1999).

[61] Kim Dong and Kim Sungkwun, "An iterative learning control method with application for CNC machine tools," IEEE *Transactions on Industry Applications*, Vol. 32, Issue. 1, pp. 66 – 72, (January – February 1996).

[62] S. S. Garimella and K. C. Srinivasan, "Application of iterative learning control to coil-to-coil control in rolling," IEEE *Transactions on Control Systems Technology*, Vol. 6, Issue. 2, pp. 281 – 293, (March 1998).

[63] Y. Chen, C. Wen, Z. Gong and M. Sun, "An iterative learning controller with initial state learning," IEEE *Transactions on Automatic Control*, Vol. 44, Issue. 2, pp. 371 – 376, (February 1999).

[64] S. K. Sahoo, S. K. Panda and J. X. Xu, "Iterative learning-based high-performance current controller for switched reluctance motors," IEEE *Transaction on Energy Conversion*, Vol. 19, Issue. 3, pp. 491 – 498, (September 2004).

[65] M. Norrlof and S. Gunnarsson, "Experimental comparison of some classical iterative learning control algorithms," IEEE *Transactions on Robotics and Automation*, Vol. 18, Issue. 4, pp. 636 – 641, (August 2002).

[66] P. Bondi, G. Casalino and L. Gambardella, "On the iterative learning control theory for robotic manipulators," IEEE *Transactions on Robotics and Automation*, Vol. 4, Issue. 1, pp. 14 – 22, (February 1988).

[67] M. Norrlof, "An adaptive iterative learning control algorithm with experiments on an industrial robot," IEEE *Transactions on Robotics and Automation*, Vol. 18, Issue. 2, pp. 245 – 251, (April 2002).

[68] The Mathworks, leading global provider of technical computing and model-based design software, Web home-page: http://www.mathworks.com/

[69] Compact Dynamics GmbH, Web home-page: http://www.compact-dynamics.de

[70] C. F. Taylor, "The Internal Combustion Engine in Theory and Practice," Volume II: Combustion, Fuels, Materials, Design, M.I.T. Press, MA, (1985).

[71] C. R. Ferguson, "Internal Combustion Engines", Applied Thermosciences, Wiley and Sons, NY, 1986.

[72] C. S. Draper, "Pressure waves accompanying detonation in engines," *Journal of Aeronautical Science*, vol. 5, pp. 219-226, (1938).

[73] S. Ortmann, M. Rychetsky, M. Glesner, R. Groppo, P. Tubetti, G. Morra, "Engine knock estimation using neural networks based on a real-world database," *SAE Technical Paper* 980513.

[74] M. D. Checkel, J. D. Dale, "Computerised knock detection from engine pressure records," *SAE Technical Paper* 860028.

[75] M. Kaneyasu et al., "Engine knock detection using multi-spectrum method," *SAE Technical Paper* 920702.

[76] B. Samimy, G. Rizzoni, "Engine knock analysis and detection using time frequency analysis," *SAE Technical Paper* 960618.

[77] R. Hickling, D. A. Feldmaier, F. H. K. Chen and J. S. Morel, "Cavity Resonances in Engine Combustion Chambers and Some Applications," *Journal of Acoustical Society of America*, Vol. 73, no. 4, pp. 1170 – 1178, (1983).

[78] J. Wagner, J. Keane, R. Koseluk, and W. Whitlock, "Engine Knock Detection: Products, Tools and Emerging Research," *SAE Technical Paper* 980522.

[79] S. Carstens-Behrens and J.F. Bohme, "Fast Knock Detection using Pattern Signals," proceeding of IEEE *International Conference on Acoustics, Speech and Signal Processing*, Vol. 5, pp. 3145 – 3148, (2001).

[80] A. Kirschbaum, S. Ortmann, and M. Glesner, "Rapid Prototyping of a Co-processor based Engine Knock Detection System," proceedings of the *International Workshop on Rapid System Prototyping*, pp. 124 – 129, (1998).

[81] Motorola Semiconductor Products Sector, [Online], Available: http://www.motorola.com/automotive/products.html

[82] Intersil Corporation, High Performance Analog Division [Online], Available: http://www.intersil.com/product_tree/product_tree.asp?x=9

[83] Mohamed Anas, "An Architectural Modelling Environment for the PowerPC SoC," EngD research portfolio, (February 2005).

[84] Mohamed Anas, R. J. Paling, W. H. Nailon and D. R. S. Cumming, "Fast Internal Combustion Knock Processing Algorithm Using an Automotive PowerPC System-on-Chip," IEEE *Transaction on Vehicular Technology*, Accepted for publication, Manuscript ID VT-2004-00099, (November 2004).

[85] Mohamed Anas, R. J. Paling, W. H. Nailon and D. R. S. Cumming, "Fast Internal Combustion Knock Processing Algorithm Using an Automotive PowerPC System-on-Chip," proceedings of the *5th Australasian Control Conference*, Automotive Applications, Melbourne, Australia, vol. 1, pp. 30 – 38, (July 19-23 2005).

[86] Infineon Technologies, Web home-page: http://www.infineon.com/

[87] Texas Instruments, Web home-page: http://www.ti.com/

[88] Intel Corporation, Web home-page: http://www.intel.com/

[89] Renesas Technology Limited, Web home-page: http://www.renesas.com/

[90] NEC Semiconductors, Web home-page: http://www.ee.nec.de/

Integrated Business Plan for the SoC Development of Camless Engine Platform

32-bit Embedded Controller Division
Motorola Semiconductor Products Sector
FreeScale Semiconductors

Created and Submitted by
Mohamed Anas
32-bit Embedded Controller Division
Motorola SPS

Submitted in fulfilment of the requirements for the EngD Doctoral MBA Business Modules

ACKNOWLEDGEMENTS

This business plan would not have come to form were it not for my interaction with several other individuals and I am happily in their debt. I would like to acknowledge my advisors, Dr. George Burns and Dr. Ron Campbell for their guidance and helpful suggestions.

I would also like to thank Mr. Robin Paling, 32-bit Embedded Controller Division, Motorola SPS, for providing me with many helpful comments and suggestions.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AIDC - Austin and India Design Centres
ASIC - Application Specific Integrated Chip
ASSP - Application Specific Standard Product
AVSD - Advanced Vehicular Systems Division
BMW - Bayerischen Motoren Werke AG
DIS - Driver Infotainment Systems
DSP - Digital Signal Processor
EMA - Electro-Magnetic Actuator
EMVT - Electro-Magnetic Valvetrain Controller
EVVC - Electro-Magnetic Variable Valvetrain Controller
FAE - Field Application Engineer
FDT - Flash Development Tool
FPGA - Field Programmable Gate Array
IP - Intellectual Property
LLD - Low Level Driver
MPU - Micro Processor Unit
NDA - Non Disclosure Agreement
OEM - Original Equipment Manufacturer
PAT - Profit After Tax
PBT - Profit Before Tax
PSG - Project Steering Group
SIMD - Single Instruction Multiple Data
SoC - System-on-a-Chip
SPS - Semiconductor Products Sector
TI - Texas Instruments, Inc.

1. EXECUTIVE SUMMARY

The purpose of this business plan is to raise US $6 million from the Motorola Semiconductor Products Sector (SPS) funding body with the approval of Motorola SPS Project Development and Steering Group. The 32-bit Embedded Controller Division in the Advanced Vehicular System Dynamics (AVSD) division of Motorola SPS is dedicated to developing an innovative Electronic Variable Valvetrain Controller (EVVC) System-on-a-Chip (SoC) platform for the advance Electromechanical Actuator (EMA) controlled next generation camless engines.

1.1. THE CAMLESS ENGINE MARKET ANALYSIS DESCRIPTION

Present valvetrain lift and timing control market is highly competitive and is driven by many major car engine manufacturers. One of the key players, The BMW Group, who is not only the main licensee of existing various camless valvetrain control techniques, but also the patent holder of the most recent electric motor driven Valvetronic IP, which dispenses the need for employing a throttle butterfly. It is envisaged that by 2008, 46% of the cars would have EVVC based control methodologies. With an extremely positive market outlook, research and development technology, EVVC customers are focusing on employing electromagnetic actuators as the primary valve driver. As emission regulations are being tightened with the emphasis of enhancing performance, Motorola SPS is required to develop more advanced system level solutions in order to meet these requirements.

1.2. THE PRODUCT

Electronic Variable Valvetrain Control System-on-a-Chip (EVVC-SoC) – A Technically Patented Infinitely Flexible Camless Engine Controller SoC based on the next generation PowerPC™ eSys Single Instruction Multiple Data (SIMD) Platform. This SoC along with the electromagnetic actuator replaces the classic mechanically fixed camshaft with breakthrough technology facilitating a sleek and attractive profile.

1.3. MAJOR BENEFITS OF THE EVVC SOC

Unlike the traditional fixed profile mechanical cam, with EVVC, any engine valve can be opened at any time to any lift position and held for any duration, optimising engine performance. Motorola is in the process of developing a new SoC family, namely MPC5500, which will ideally suit the controller requirements of an electromagnetic valvetrain. To facilitate this, SoC team has developed strong working relationships with various actuator technologists including OEMs and Tier I automotive clients and plan to further this area by continuing to vertically-integrate the business.

Primary business and technical benefits of the EVVC to Motorola and automotive customers are as follows:

- Deeper market penetration of the new PowerPC SIMD core devices
- Enhanced SIMD based PowerPC SoC permits the control of the EMA to infinitely variable valve lift and timing
- Development of a true systems solution not only for EVVC but also the general market
- Reduction in emission levels
- Cylinder deactivatation for dynamic power delivery control
- Supports hard and soft IP for generic high precision actuator market
- Dynamic valvetrain timing and lift control
- Dynamic variation of air-fuel mixture

- Low pumping losses
- Effective braking and thrust operation
- Independent valve and cylinder deactivation
- Improved cold-start and warm-up operation
- Adaptive cylinder range
- Multi stroke operation
- Reduction of idle speed
- Simplified cranking procedure
- Optimised internal residual gas recirculation

1.4. OVERVIEW OF THE EVVC SOC PLATFORM

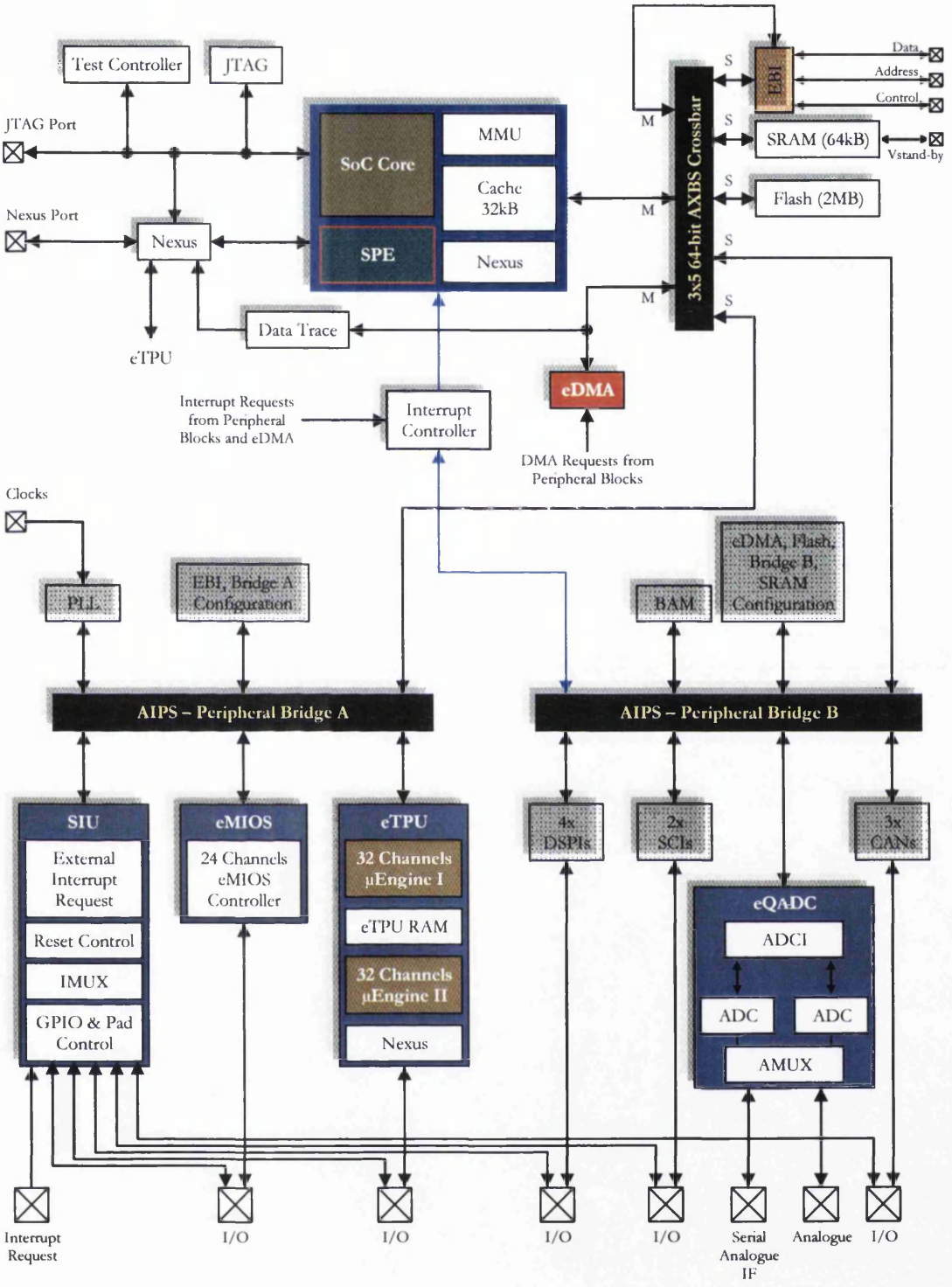

Figure 1: EVVC SoC Overview

1.5. The BMW Group, Other OEMs and Tier I Contracts

Motorola has an in principle agreement and NDA with The BMW Group and various other OEMs and Tier Is to supply 138 million EVVC SoCs over a 10 year period starting in 2005. This provides the venture with a reliable income stream through highly reputed premier automotive manufacturers through whom the establishment of this new SoC platform will be solidified for the next generation of camless engines.

1.6. The Offer

The 32-bit EVVC SoC Embedded Controller Team seeks PSG approval and US $6 million from the internal Motorola project funding body. For this investment, initially forecast returns are as follows:

- 10 times the original investment by year 6
- An Internal Rate of Return (IRR) of 57% over 10 years.

The funds are required in two slots:

- Initially US $1.7 million to finance the fully functional prototype platform including production worthy software
- Subsequently US $4.3 million to fund various SoC design and system level software development groups and to purchase factory equipment and materials supplies for production of the EVVC SoC

1.7. Financial Summary

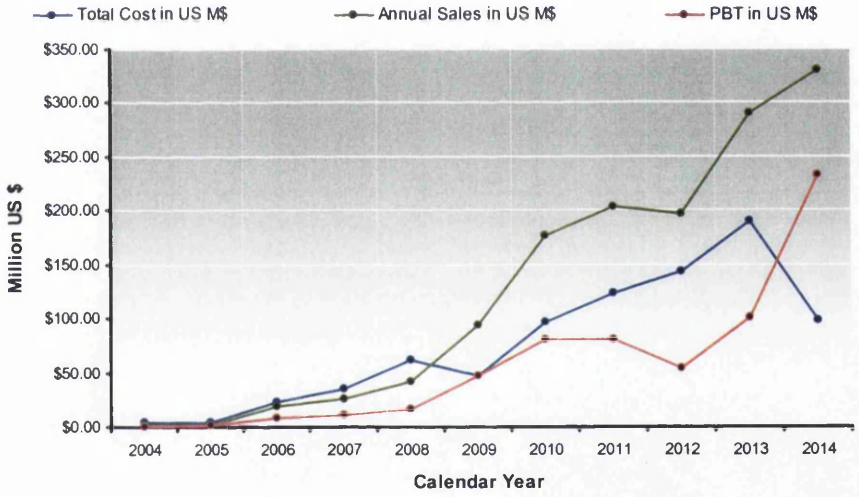

Figure 2: EVVC SoC Project Financial Summary

1.8. EVVC SoC - A Lucrative SoC Business Model for Motorola SPS

Motorola Semiconductor Products Sector (SPS) stands at the threshold of its next great technological and market leap: the advent of the EVVC SoC which will allow Motorola SPS to integrate the functions of multiple chips into a single chip, ushering in a new generation of chips with superior system-level integration, device performance and cost efficiency. SoC chips will comprise both newly designed functionality and pre-designed blocks of intellectual property (IP). The re-use and re-integration of existing designs will reduce design effort, leverage chip capacity and performance.

From the automotive OEMs perspective, the EVVC SoC will deliver what Motorola SPS has always delivered: better, faster and cheaper products. From Motorola SPS' perspective, the EVVC SoC represents the continued need to manage complexity in design, integration, verification and test. To achieve this level of cooperation, The 32-bit Embedded Controller Division in Motorola SPS has developed a business and legal model to support the re-use and integration of designs originating from multiple sources into a single SoC addressing the next generation automotive market.

The biggest challenge is to understand the technology and business risks, to establish methodologies to manage those risks and rewards, and to develop the business models to fairly allocate those risks and rewards to the appropriate business players. Motorola SPS' legal system is capable of providing answers to all of the questions posed by the EVVC SoC business model. Legal and contractual models exist for IP protection, licensing, indemnification, value sharing and dispute resolution sufficient to address this new market.

The hallmark of this unique EVVC SoC market will be the shift in roles between the key 32-bit Embedded Controller Division's players and their effort to establish successful business, legal and technological inter-relationships. Each player comes to the SoC market with a unique set of value-added propositions, needs and objectives.

1.9. The Future

The 32-bit Embedded Controller team's EVVC SoC product offering will be positioned as cost-effective and specifically reliable in a hostile automotive environment so that current and future controller functional needs of the radical electromagnetic camless internal combustion engines can be fulfilled. Along with the presently proposed SoC architecture, suitability of two other enhanced PowerPC architectures are being investigated to address the changing SoC market during the course of the project.

2. THE OFFER

2.1. FUNDS REQUIRED

A total investment of US $6 million is sought from the internal project funding body of Motorola that has experience in the SoC design, development and marketing industries. This funding required will be split and invested in two parts:

The first of US $1.7 million will be made once the BMW Group contract is finalised. This is due to be completed by January 2002. This money will be used to finance the initial prototype development based on a single cylinder 7 series engine.

The second portion of the US $6 million will be required by January 2003. This money will be used to finance the SoC module level integration, SoC testing, assembly, critical raw material supplies as well as to provide a working capital.

2.2. BENEFITS TO MOTOROLA'S FUNDING BODY

For the US $6 million investment, the financial projections forecast an Internal Rate of Return of 57% over ten years, providing Motorola SPS with a PBT of 12 times the original investment at the end of calendar year 2010. If the Board unanimously decides, dividends may be distributed; however, this business plan does not contemplate any dividend payments, only capital gains.

3. THE PRODUCT – EVVC SOC PLATFORM

The 32-bit Embedded EVVC SoC has two innovative initial product strategies offered to the SoC customers. The two strategies are; Single-Issue (executes one instruction per system clock), and the Dual-Issue (executes two instructions per system clock cycle) SoC. Research and development is currently being undertaken on subsequent SoC products. Book E based PowerPC is a breakthrough technology and is the only patented SoC architecture in the world that makes distributed electromagnetic valvetrain based camless engines feasible.

At present the Single-Issue SoC is being offered to the customers. The state-of-the-art Single Instruction Multiple Data (SIMD) core on this SoC variant has a combination of many innovative, user friendly features not included on parts presently offered by the competitors or the legacy Motorola's PowerPC products.

3.1. THE EVVC SOC

The EVVC SoC is the first member of a family of next generation powertrain microcontrollers based on the PowerPC Book-E architecture. This family of parts contains many new features coupled with high performance CMOS technology to provide substantial reduction of cost per feature and significant performance improvement over the legacy PowerPC platforms. In addition to the new features introduced, the SoC core complies with the previous PowerPC standards Book E architecture. It is 100% user mode compatible (with floating point library) with the classic PowerPC instruction set. The enhanced version of the Book E architecture incorporated in the SoC has enhancements that improve the PowerPC architecture's fit in a multitude of embedded applications. The new set of instructions, including the SIMD DSP, offers the generation of optimal time and code optimised instruction sequence for a given embedded application. The SoC has two levels of memory hierarchy. The fastest accesses are to the 32-kilobyte unified (aka Von Neumann) cache. The next level in the hierarchy contains the 64-kilobyte on-chip Level 2 (L2) Static Random Access Memory (SRAM) and the 2-Mbytes of on chip flash memory. Both the L2 SRAM and the flash memory can hold instructions and data. The External Bus Interface has been designed to support most of the standard memories used with the legacy PowerPC product family.

The complex Input/Output (I/O) timer functions of the SoC are performed by two enhanced Time Processor Unit (eTPU) engines. Each eTPU engine controls 32 hardware channels. The eTPU can be programmed using a high-level programming language unlike the traditional Time Processor Units (TPU). The less complex timer functions of the SoC are performed by the enhanced Modular Timer System (eMTS). Off-chip communication is performed by a suite of serial protocols including Controller Area Networks (CAN), enhanced Serial Peripheral Interface (eSPI) and Serial Communications Interface (SCI). The SoC has an on-chip 40-channel Enhanced Queued dual Analogue-to-Digital Converter (eQADC). The System Integration Unit (SIU) performs several chip-wide configuration functions. Pad configuration and General-Purpose Input and Output (GPIO) are controlled from the SIU. External interrupts and reset control are also found in the SIU.

3.2. EVVC SOC OVERVIEW

See Figure 1

3.3. COMPETING SOCS

Our primary competitors in the EVVC SoC program are Infineon Technologies AG and Texas Instruments (TI). Overall, our solution has better performance and benefits with the SIMD unit. However, since our wafer technology is slightly behind TI, our die size is somewhat higher. Consequently we have tremendous price pressure to remain competitive and still be profitable.

With TI's strong background in the DSP derivatives it is successfully leveraging these in mid to long-term automotive applications. TI's flash memory technology is recognized as similar to that of Motorola's in quality and reliability.

However, Motorola will differentiate itself from its competitors through offering lower priced, higher performance valvetrain controller SoCs. By gaining wider knowledge of the camless engine technology, the EVVC SoC team exposes itself to a larger segment of the growing market. This tightly focussed approach also makes it easier for Motorola to establish and retain a position as an industry leader on the EVVC SoC market.

Therefore, in summary, the financial strength that will be achieved with the EVVC expansion will give Motorola SPS the capacity to establish a larger, more diversified customer base, which will generate increased sales revenue. This is an exponential growth opportunity for the company.

4. OVERVIEW OF THE ORGANISATION

4.1. REGISTERED NAME

32-bit Embedded Controller Division. Motorola Semiconductor Products Sector Limited.

4.2. COMMENCEMENT OF OPERATIONS

The 32-bit EVVC SoC team will start executing the project in January 2003 (upon PSG approval and final sign-off of NDA).

4.3. HISTORY OF THE ESTABLISHMENT OF THE SOC PLATFORM FOR THE CAMLESS ENGINE

This venture initially grew out of a traditionally strong relationship with The BMW Group. The engineering management and next generation car strategy team of BMWs plan to introduce the camless engine into its 5 and 7 series models drove the need for the invention of such an advanced SoC platform and subsequent arrangements with the complete development plan was established. Since the introduction of the legacy 32-bit Embedded microprocessors and SoCs, Motorola have been successfully marketing the PowerPC based platforms to BMW powertrain in Germany. This venture has been extremely successful. As a result, BMW approached Motorola to assist them with the development strategy to take their radical camless engine into their well established global automotive market. BMW senior management saw in the 32-bit Embedded Controller Division, the necessary skills and drive that would ensure the success of execution and on-time delivery of the EVVC project. The BMW Group are in the business of marketing premier, cost-effective and reliable cars, so there is a perfect fit between this globally renowned company and the skills and products of Motorola SPS Ltd. Currently, negotiations are being done with various other high-end premier car OEMs and Tier I automotive module manufacturers worldwide to enhance the target market of this innovative SoC.

4.4. MISSION STATEMENT

The EVVC SoC team designs, develops and markets advanced engine controller platforms. They are sold to small, medium and large-sized automotive companies for a range of specialist applications. These SoC platforms are distinguished from competition by their sophisticated interfaces, scalability, reliability and ease of programmability and are extensively patented. Sales are made directly and through major distributors and OEMs in the home and overseas market segments. The team challenges benchmarks in the SoC sector. We find ways to "do what the others don't".

4.5. VISION STATEMENT

The SoC team will be operating from various Motorola branches around the world. It will have annualised PBT of US$ 62.6 million and be profitable in four calendar years. It will employ 41 people mainly engaged in management, R and D, design, marketing, support and administration. The team will offer 6 variants of the core SoC and provide added-value services to a large customer base throughout the automotive market segments worldwide. Motorola SPS' product offerings will be technically advanced and offer many clear-cut advantages and improvements over competitors' possible offerings. Motorola SPS will continue to expand through acquisitions in related technology and market segments.

4.6. ORGANISATIONAL OBJECTIVES

- Supply the fully qualified EVVC SoC platform to customers by October 2005.
- Achieve recurring profits of US$ 40 million or greater by year five.
- Research and establish enhanced Book E based SoC platforms by year two, ready to market by year seven.

4.7. ORGANISATIONAL VALUES

- Full commitment to The BMW Group and other future customers with appropriate levels of NRE and exclusivity
- Collaborative approach to new powertrain SoC platforms for the future

4.8. EVVC SOC MANAGEMENT TEAM

The Management team is comprised of Ross McOuat, Robin Paling and Stephen Lynas. The Management team is complemented by two advisors: Franz Fink (SPS Automotive General Manager) and Bill Pfaff (DIS Technical Officer). The Management team is highly motivated, experienced and well qualified. The team is strongly positioned to take advantage of this opportunity and a brief description of the team profile is as follows:

- Proven business start-up skills, with bottom line responsibility
- Experience in business start up finance, marketing, operations and legal aspects
- Personality profiles that reflect the synergies of cohesive group dynamics
- Stephen Lynas has significant skills and experience in marketing and strategy.
- Franz Fink successfully established the revolutionising 32-bit Embedded Controller SoC Platform group in East Kilbride.

4.9. MAJOR MILESTONES ACHIEVED TO DATE

- Identification of unsatisfied market need for EVVC SoC platforms
- Generation of NDA, accurate pricing matrices with NRE for BMW with exclusive rights
- Submission of various patents
- In principle agreement with BMW and other OEMs to supply 138 million SoC platforms over 9 years
- Successful customer trials of SoC based behavioural and functional software suites using automotive applications
- Competitor analysis undertaken to establish uniqueness of the EVVC SoC
- Development of detailed Integrated Business Plan (IBP)

4.10. BOARD STRUCTURE

The EVVC SoC Team, in addition to the capital introduced to the venture by the internal Motorola SPS funding body, also has a wealth of experience in dealing with and/or contacts in large organisations that are active in the SoC application market.

An independent non-executive chairperson will be appointed. This person will be able to introduce the Management team to major car OEMs and will have developed a reputation for integrity in all business dealings. The EVVC SoC Team will be represented on the Board by two members. Initially, these will be Ross McOuat and Robin Paling.

5. STRATEGIC ANALYSIS

5.1. EXTERNAL ENVIRONMENT - MACRO ENVIRONMENT ANALYSIS

5.1.1. TECHNOLOGICAL DEVELOPMENTS

Technological developments in the SoC industry have focused on single core, legacy Microprocessor Units (MPUs) ignoring the back-to-back requirement of embedding Digital Signal Processing (DSP) elements on the same die to implement advanced vehicle powertrain applications. The EMVT SoC proposed eliminates the need for requiring two independent platforms (MPU + DSP) to implement such algorithms. The SoC also addresses this issue by offering a single development unit on the same Silicon so that the performance, cost and space benefits can be optimised.

5.1.2. ASIC BASED PLATFORMS

ASICs are ideal for applications that are well-defined and not expected to change during the customer's duration of ownership of the product. The recent history of automotive applications, however, is characterised by rapid improvement in the state-of-the-art, creating significant risk of early obsolescence for any ASIC based accelerator, even if microcoded. Moreover, it is a well known fact that the most efficient implementation of the heterogeneous elements required to meet modern application demands is usually an ASIC; however, developing an ASIC is expensive, time-consuming, and inherently risky.

Even though ASIC vendors seem to be under attack from all sides, they will survive the current assault. A customer simply can't acquire a silicon solution with the lowest possible cost and still have the highest possible performance without pursuing an ASIC design. What has changed is that the volumes needed to justify an ASIC have grown to where low- and now medium-volume applications are almost forced to use FPGAs or application-specific standard product (ASSP) solutions.

5.1.3. FPGA BASED PLATFORMS

Although, FPGAs are the ultra-flexible alternative to the above-mentioned ASICs, but FPGA efficiency is severely compromised by the very thing that makes FPGAs flexible – configurable logic elements and interconnections that are generic.

5.1.4. WHY SOC BASED DESIGN METHODOLOGY?

SoC based design approach combines the flexibility of FPGA with the efficiency of an ASIC. This approach is not unlike toggling a set of different options when buying a new PC – working within a general system architecture framework, one can choose various processor grades, speeds, peripherals, etc. Thus, using an SoC based design approach, a systems architect could create a platform that incorporates reconfigurable, fixed-function, and programmable elements; each chip could be configured for different functionalities.

5.1.5. ECONOMIC TRENDS

The ASIC and the FPGA markets face several economic and technological factors that call into question its ability to survive and prosper. Even after recovering from the dramatic decline in market revenues that started at the end of 1999, this market must come to terms with many issues that confront it in the near term.

On the economic front, the rising cost of mask sets and very large nonrecurring engineering (NRE) charges may force smaller customers and start-up companies to rethink their product strategies and possibly delay their designs from entering the production stage. This may also push these companies toward using field-programmable gate arrays (FPGAs) as a production technology. In addition, the well-identified flow of ideas and innovative solutions derived from smaller companies in the semiconductor industry can be impeded by these high start-up costs.

However, in the SoC domain, the availability of optimal and specifically cost-effective Electronic Design Automation (EDA) tools substantially reduces the design cycles and therefore reduces the time-to-market of SoC based solutions. Designers find themselves in the enviable position of seeing product life cycles and windows of market opportunity expand while design-cycle times decrease. At present, EDA tool vendors are addressing the design-cycle problem with even better and more comprehensive tools that provide more control to designers over their work. However, it is understood that the rapid increase in potential gate counts due to smaller process geometries offsets any boost in designer productivity that comes from using better tools.

Moreover, emphasis is laid on the greater use and reuse of legacy semiconductor intellectual property (SIP) blocks in SoC platforms. The result is that designers can pick the SIP best suited for their design and capitalise on the expertise of others. The SIP industry has its own set of issues, such as the interoperability between SIP blocks in a design, quality control, and the portability between silicon vendors.

5.2. MARKET ANALYSIS DESCRIPTION OF KEY CUSTOMER – THE BMW GROUP

The BMW Group continued its steady route to growth in 2001: the number of BMW, Rover, Land Rover, MG and Mini brand vehicles delivered - total: 795,700 units - was the highest annual unit sale within the last four years. This includes the best figures for both the BMW Group and the BMW brand. (Note: After the handover of the Rover brand, the sales of Rover vehicles are also included). The most recent growth in 2001 marks a nine percent increase over 2000. This success is primarily driven by the successes enjoyed by the BMW 3 series Valvetronic models, which is also a variable valve timing mechanism with limited dynamic valvetrain control flexibility. Chairman of the Board of BMW AG, commented: "This development means that we can forecast a level of sales of over 1,000,000 BMW cars for the whole of 2002 - assuming that there are no major distortions of any of the important individual markets".

The generally positive situation is the result of the marketing success of the entire model range. In detail: The EMVT intended 5 and the 7 series are continuing to develop very successfully; the most recent 5 and 7 series models are showing high rates of growth.

The above figures show that high-end BMW car market is an exponentially growing market with continued expansion forecast; the growth is primarily based around the emergence of advanced engine technology with better fuel efficiency, greater power delivery with low emission levels. Therefore, the EMVT technology plays a major role in achieving this objective.

In addition to the BMW group's future advanced engine roadmap, the trend towards the inclusion of EVVC systems in mid-to-high-end cars is extremely significant and promising. Such opportunities have enables us to develop a complete systems solution including flash MCU, SmartMOS and simulation platforms. SoC performance required by these engine technologists also facilitates the acceleration of our technology development in order to remain competitive. This not only gives us the opportunity to have greater market share, but also, a strong position in advanced, high performance systems level integration. Moreover, these techniques will help the customer to reduce development time and allow fast prototyping with ease. This market opportunity is estimated to be worth approximately $1.2 Billion over the next 10 years.

5.3. COMPETITIVE ENVIRONMENT ANALYSIS

Please see Figure 3, §5.4 for more details on the key competitor analysis.

Intensity of Competitor Rivalry	
▪ Targeting an under-serviced technology sector of the market ▪ Resistance from established competitors ▪ Unique combination of products catering to such niche automotive applications needs ▪ Not price competitive in engine applications segment	**Low Risk**

Risk of Potential Entrants	
▪ Three of the largest premier car manufacturers in the world secured as a strategic partner ▪ Intellectual Property (IP) Protection	**Low Risk**

Power of Buyers	
▪ Existing relationship with BMW Groups' rivals ▪ Low-end automotive buyers	**Medium Risk**

Third Party and Supplier Bargaining Power	
▪ Extensive industry networks via key board member ▪ Less critical software and IP cores are generic and can be outsourced ▪ Identification of multiple suppliers for third party module ▪ Expertise in-house to design and develop complex modules	**Low Risk**

Threat of Substitute Products	
▪ No alternatives offering combination of SoC features ▪ Early mover ▪ Low cost of production ▪ Higher reliability ▪ East of use ▪ Established product support	**Low-Medium Risk**

Table 1: Competitive Environment Analysis

5.3.1. IMPLICATIONS OF ANALYSIS

Due to the market segment targeted and the nature of the industry, it is possible to earn above normal profits in this part of the automotive industry.

5.4. Competitor Analysis

In the current SoC marketplace it is sometimes said that understanding competitors is more important than understanding customers. The following Radar chart summarises the overall market position of two key competitors, Infineon Technologies AG and Texas Instruments with respect to Motorola SPS, in terms of technology, product, financial backing, customer exclusivity, distribution channels, after sales service, position in life cycle, cost structure and selling force. This analysis is used for the evaluation and monitoring of the competitors to establish a strong understanding of the competitor SoC market. Please see Appendix J - for more details. This analysis confirms that Motorola SPS is in a stronger position in the EVVC SoC market segment.

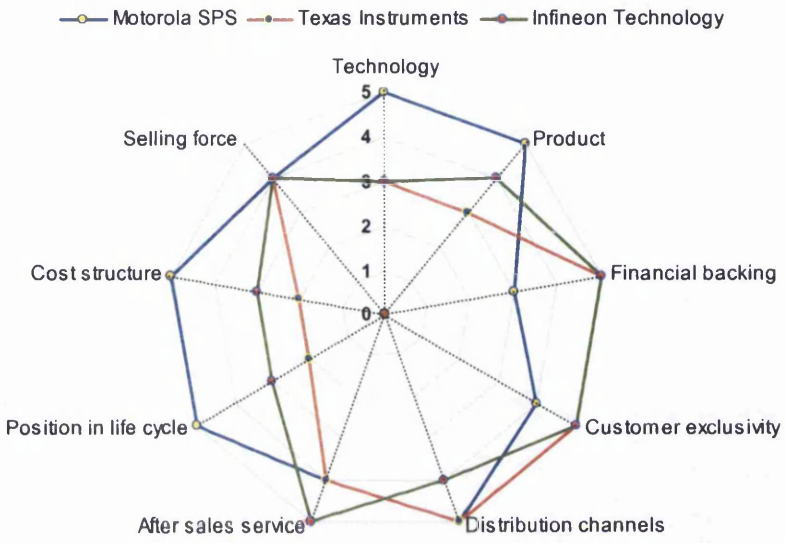

Figure 3: Dimensional Analysis of Key Competitors

5.5. Customer Profile

5.5.1. The BMW Group

The BMW Group concentrates on selected premium segments in the automobile market. This means that it is the only multi-brand automobile manufacturer in the world that is not active in the mass market, i.e. the volume segments of the automobile market. The aim of the premium brand strategy is to achieve higher revenues per vehicle, on the basis of a high-value product substance and an unmistakeable brand profile. Presently, the BMW Group pursues this premium brand strategy with the BMW and MINI brands and since January 2003, with Rolls-Royce. This means that it will cover the premium segments from the small car to the absolute luxury category.

The premium brand is thus creating the preconditions for further profitable growth for the BMW Group. The Company expects that in the next ten years, the premium segments of the automobile market will grow worldwide by around 50%. In contrast, the mass volume segments will increase by "just" 25% or so during the same period. The BMW Group is aiming to achieve profitable growth through its advanced technological improvements and in that order of priority - "profitable" followed by "growth".

5.5.2. VOLVO CARS

Volvo is a premium automotive brand with unique appeal and a world-class reputation for safety, quality, durability, and environmental responsibility. In an agreement reached in 1999, Volvo sold its worldwide passenger vehicle business, Volvo Cars, to the Ford Motor Company of Dearborn, Michigan.

Invoiced sales of Volvo cars in 2000 amounted to 686,400 units, an increase by 5% compared to 1999. Sales trends were positive, particularly in Europe and North America. Demand for the EMVT intended Volvo S40 and V40 exceeded expectations and production was stepped up in order to meet the inflow of orders. Particularly, these models were met with swift and positive reception in North America.

According to one of Volvo Cars' senior marketing executives, strong positive sales trends in Europe, which began in late 1997, have now secured a strong foothold on a number of key markets.

In 1999, In the UK, Volvo's third largest market, the number of Volvo cars registered increased by 21% to 40,700 units in 1999, whereof more than half consisted of the Volvo S40 and V40. In Germany, where the total market was largely unchanged, 37,000 Volvo cars were registered. The 15% improvement in Italy, to 25,300 units, was wholly attributable to the Volvo S40 and V40. Registrations also increased in Spain to 10,400 (9,300), in Switzerland to 7,000 (6,000), and in Finland to 6,800 (6,400). Such trends are also true in the United States. Therefore, in conclusion, these figures confirm the promising EMVT target platforms' saleability and global acceptance.

5.5.3. COMPACT DYNAMICS

Compact Dynamics is an internationally renowned Tier II development company specialising in Mechatronic development for vehicle manufacturers and suppliers in the field of innovative electric drives, including prototype construction. This company not only solves technical system problems, but also develops custom components such as motors, power electronics and control systems for the automotive powertrain market. Most of Compact Dynamics' products are designed with optimised power density, increased power to weight ratio, lower energy consumption and reliability. Control concepts, topologies and structural engineering offered by Compact Dynamics integrates well with the EMVT platform offered to the Tier I automotive industry.

The profile of the above customer group includes the following criteria:

- Significant market share
- Able to offer corporate support and distribution
- Solutions for emission reduction

In addition to the above customers, the SoC team will target various other well-established automotive segments to capitalise on the current euphoria of SoC based development platforms.

5.6. INTERNAL ENVIRONMENT ANALYSIS

See Appendix A for summary of Bell Mason Diagnostic. Major strengths and weaknesses that were identified during the Diagnostic process appear in the SWOT analysis in Appendix C.

5.7. PROJECT STRATEGY

The following illustration explains the strategy in place for the project. Each strategic element is internally linked although the below chart emphasises on the individuality of key strategic elements for clarity.

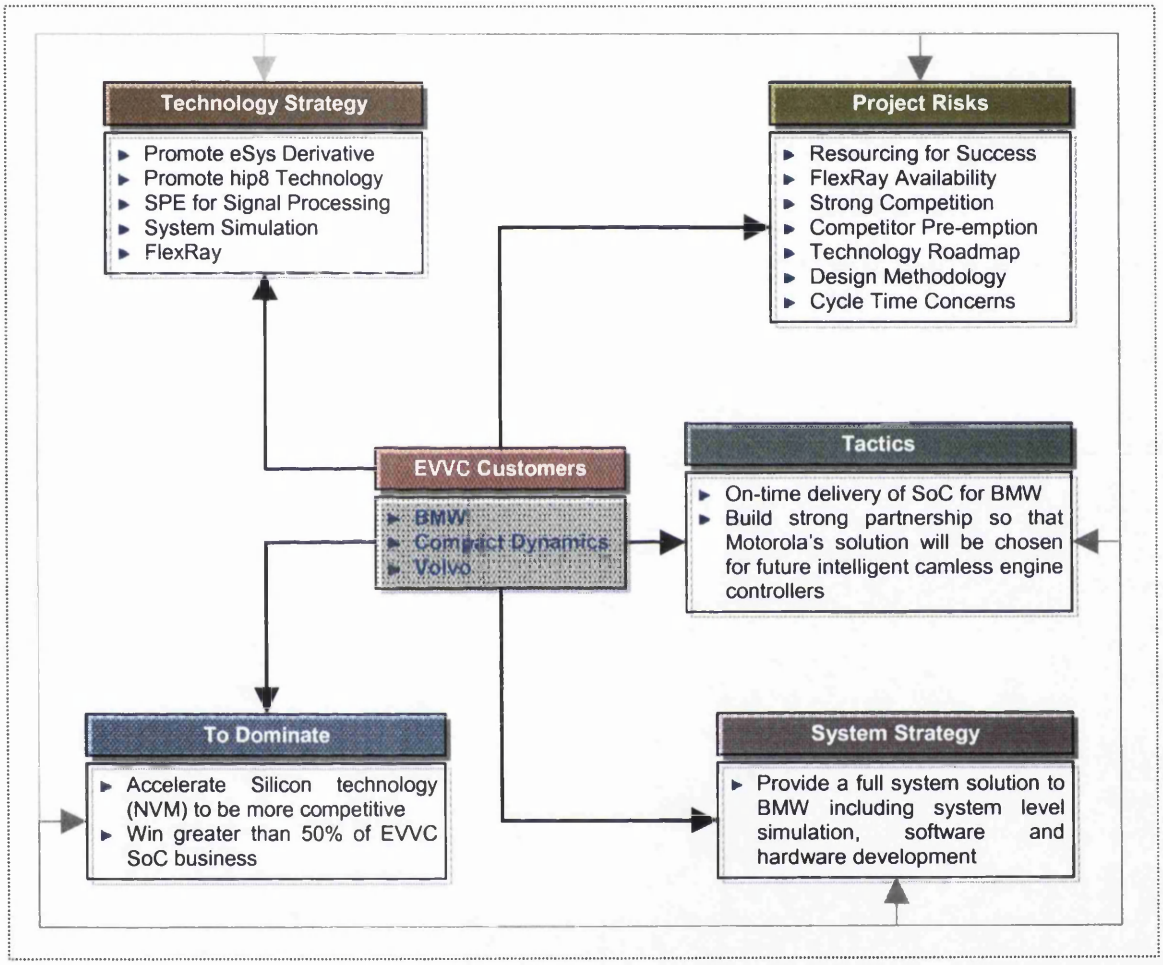

Figure 4: Project Strategy

6. KEY STRATEGIC ISSUES

6.1. SUSTAINABLE COMPETITIVE ADVANTAGE

The EMVT SoC will "succeed" because of the following:

- A strategic alliance with two of the world's largest premier automotive OEMs providing it with credibility and recognition to establish the business in its formative stages.
- Unique, reliable and innovative development platforms
- Strong team of committed people

6.2. BASIS FOR GROWTH

The basis for growing the venture is reflected in the following two strategies:

Priority 1: continue research and development of new and innovative SoC architectures to meet the current and future needs of advanced automotive manufacturers.

Priority 2: Enter new geographic markets (Japan, China, Canada, South America and Mexico).

7. THE MARKETING PLAN

7.1. MARKETING OBJECTIVES

- Establish a strong presence in the European and US market
- Use the BMW and Volvo association as a conduit for entry into the US market, market penetration, worldwide distribution and an opportunity to gauge market acceptance of the EMVT platform at a reduced business and financial risk.
- Utilise acquired market knowledge and presence to establish customers out with initial collaborators through other automotive project affiliations and through the efforts of Motorola SPS' own Field Applications Engineers (FAE) and sales force.
- Establish significant high-margin sales.

7.2. SALES FORECASTS

Based on the BMW and various other OEM inputs, market research undertaken and strategies developed, the following sales forecasts were developed:

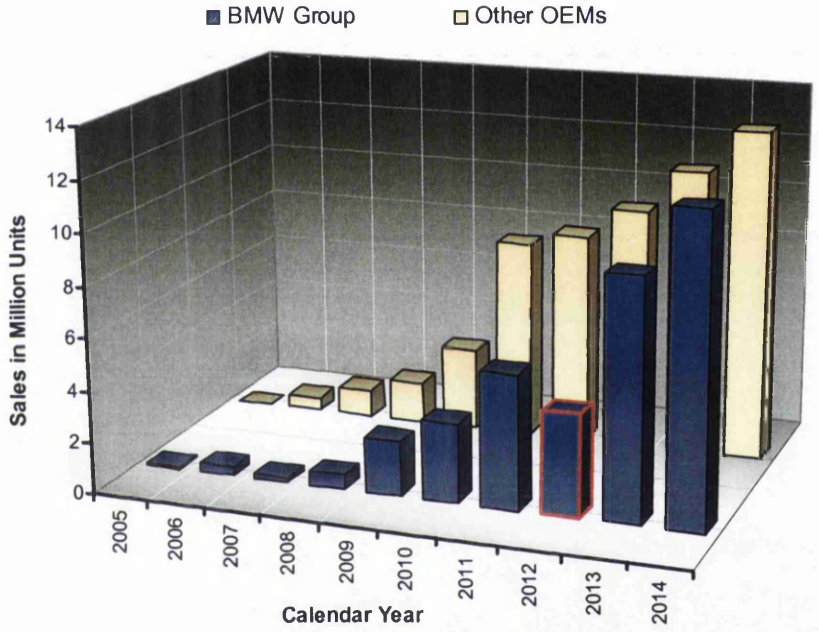

Figure 5: SoC Sales Forecast

The highlighted drop in BMW Group's sales in 2012 is due to the increased production of hybrid and fuel cell powered hydrogen cars.

7.3. SALES ASSUMPTIONS

Assumptions underlying the development of the sales forecasts are as follows:

7.3.1. YEAR 2005

- 100% of EMVT SoC production is sold to the BMW Group in accordance with the amount specified in the contract. This is a cautious market strategy to establish market acceptance and increase market knowledge.

- $120,000 will be spent on developing the relationship with the BMW Group, training the BMW Groups' in-house application development team, producing application notes, quick start-up guides and seeking out the key systems architects and decision-makers in the camless engine segment.

7.3.2. YEAR 2006

- BMW Group and other OEM contracts continue with increased supply to 900,000 SoC platforms.

- 500,000 SoC platforms will be sold to new customers some of whom may be reached through BMW Group affiliations and some as a result of our extensive marketing effort.

- $600,000 will be spent on engine expositions, trade shows, advertising, presence and developing trade show materials, a worldwide selling campaign, webinars and direct customer meetings.

7.3.3. YEAR 2007

- BMW Group and other OEMs contract continues

- Based on current knowledge, further contracts with engine technologists will be formed

- 200,000 SoCs will be sold to customers out with BMW, Volvo and Compact Dynamics. This will require the strengthening of further strategic alliances.

- $300,000 will be spent continuing the marketing effort commenced in 2006.

7.3.4. YEARS 2008 TO 2014

- BMW Group and other OEMs contract continues

- Customers out with BMW Group now represent the majority of our business with demand reaching the pinnacle in Year 2014 as the benefits of our technology are recognised in the growing market.

- $250,000 will be spent in each year continuing the search for further customers and developing the relationships with current customers.

7.4. Marketing Strategies

The key to the marketing strategy is to identify influential individuals and groups in the automotive engine development segments of the OEMs, Tier I and potentially Tier II organisations with decision-making authority. These individuals can be approached through existing contacts formed in the past, trade shows, FAE visits and business meetings. The marketing approach will demonstrate the benefits of the EMVT SoC platform.

Emphasis will be laid on the use of the powerful system level software development platform eliminate the bugs and bottlenecks found late in the design cycle. Benefits of the completely new development methodology will also be highlighted to avoid the product missing a critical market window. In addition the following will also be promoted:

- Ease of transition to hardware using our full system-level modelling tool
- Application code optimisation early on to achieve maximum system performance
- Deeper visibility both in software and hardware to catch and fix costly bugs in the early stages of the development cycle, eliminating wasted weeks or months
- Demonstration of already deployed applications using the development tools which significantly improve the design process

7.4.1. Products

The EMVT SoC will be positioned as a cost-effective, reliable and user-friendly solution to the current and future needs of advanced engine manufacturing industry. EMVT SoC and its successors will be the pinnacle in camless engine controller technology, able to rival the most established competitor products through its superior performance and technology benefits, supported by a committed training and customer support force. This will enable the previously cautious engine technologists to introduce the EMVT based camless engine at no risk.

7.4.2. EVVT SoC Pricing

The EMVT SoC will be priced competitively as follows. Note that the following pricing is based on The BMW Group sales and is dependent on the sales volumes and various internal pricing strategies.

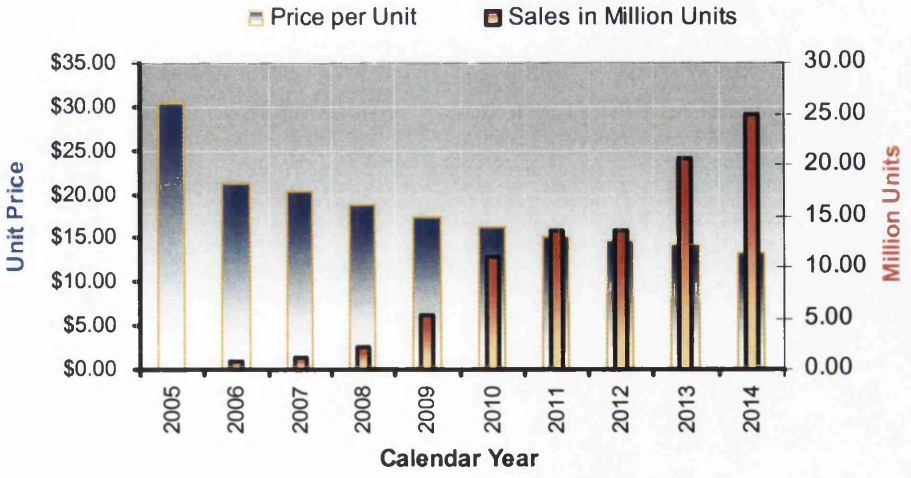

Figure 6: EVVT SoC Pricing

This pricing strategy will be high enough to convince customers that they are purchasing an effective high-quality product. Primary research indicates that the target market is not price sensitive. The sales price will return an estimated 40% gross margin on sales.

7.4.3. Distribution

Product distribution will be facilitated by utilising the internal and external established automotive distribution channels. Such distribution channels will typically have worldwide coverage. This strategy will ensure a presence in the market that increases awareness and builds demand.

7.4.4. Promotion

EMVT SoC's entry into the US market will be supported by a campaign to establish its profile in the camless engine industry. This will include the following:

- Seek out the decision-makers in the engine development segments of Tier I automotive manufacturers
- Business meetings
- Undertake a campaign of corporate level selling
- Advertise in industry journals, on the Internet, and on our website
- Attend/sponsor exhibitions and trade shows
- Continual PR: press write-ups; personal interviews; testimonials; product trials

8. Production Plan

The initial production plant for the SoC will be located at the Phoenix semiconductor manufacturing unit in the US.

8.1. Production Policy

- Motorola SPS will outsource production aspects where possible
- Third party components and ready-to-use IPs delivered on a "just-in-time" basis.
- Quality checks are in place throughout the manufacturing process

8.2. Plant Location and Layout

- Based on previous manufacturing experience, the five primary divisions (R and D, Design, Systems, Product Testing and Verification) will be housed together.
- Systems approach to assembly
- Scaleable factory design

8.3. Production Capacity

Year	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014
Production Capacity in Million Units	0.50	3.0	5.0	5.0	15.0	20.0	25.0	30.0	35.0	35.0

Table 2: EVVC SoC Production Capacity

8.4. Production Scheduling

The SoC manufacturing process is supported by sophisticated production management software. It consists of a fully relational database which keeps track of inventory and invoicing and it produces detailed job cards for each day's operations.

8.5. Supplies and Inventory

To ensure a consistently high quality product, the Motorola SPS supplies and inventory system employed enables management to instantly access:

- All parts and materials on hand
- Raw materials required to meet orders placed
- Suppliers database with lead times, credit terms, price and contact details
- Under and over stock warnings
- Payment due date and credit position
- Quality control is a standard procedure at the completion of every task.

In addition, checkpoints within the production process have been put in place to guarantee standards

9. Organisational Plan

9.1. Organisation Structure Chart – The Core Team and Product Delivery

The design activity is planned to start in Q2-2002 with specification definition completed by Q4-2002. Both Austin and India Design Centres (AIDC) will be used to design and develop the SoC platform with Build 1 due to be completed by March 2004. Tape out for the first derivative is scheduled for August 2004 with first samples by January 2005 and fully qualified by August 2005.

Software design and development will commence in Q3-2002 with first deliverables being the initial Low Level Drivers (LLD) and Flash Download Tool (FDT). The Organisational Structure charts appearing below show how the organisation's staffing needs change over the next eight years.

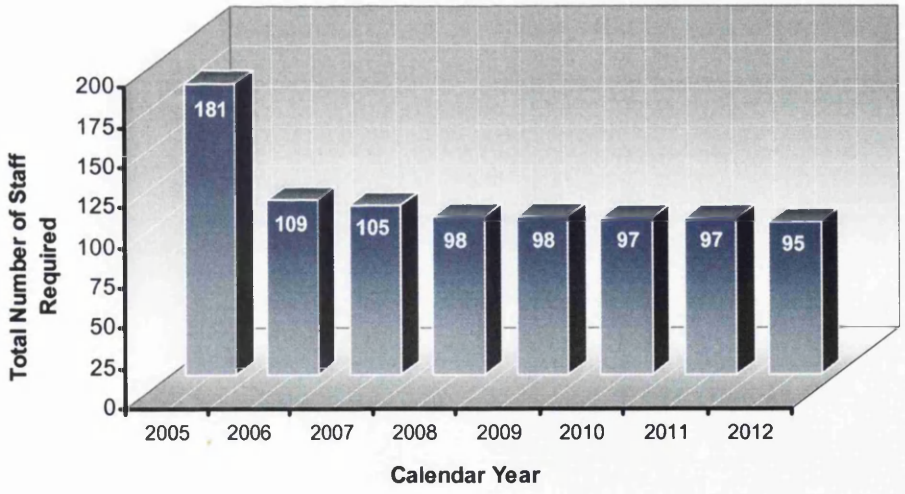

Figure 7: EVVC SoC – Staffing Needs

9.2. ORGANISATIONAL BUDGET

Total project development costs are split between the SoC Dual Core (MCU+DSP), Third Party IP and System Level Software although the higher costs reside with the SoC Core as it requires significant design, software development, system level simulation, integration, test and verification.

Due to the critical nature of the SoC development program and based on customer meetings conducted so far, regular design reviews will be required during the lifetime of the program. Initially, in 2003, these will be every 2½ months alternating between customer and Motorola SPS in East Kilbride. Occasional Reviews will also be needed with the internal Motorola SPS' SoC Creation, Product Fulfilment and Program Management teams in Phoenix, US. It is envisaged that in 2003 and 2004 design reviews every quarter will suffice. Specific line volume independent budget items appear below:

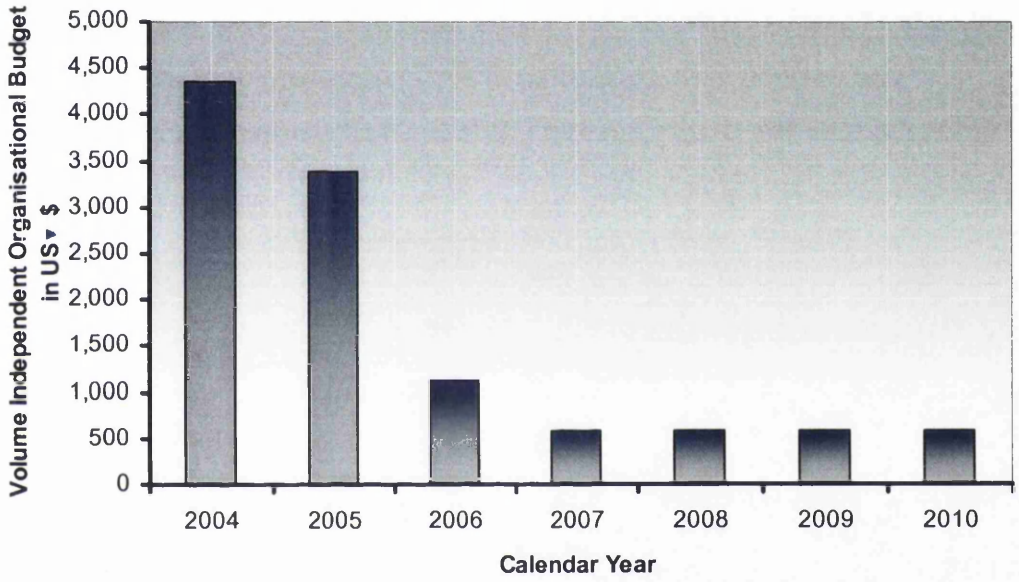

Figure 8: EVVC SoC – Organisational Budget

9.3. PROJECT ORGANISATION AND KEY FUNCTIONAL AREAS

The EVVC SoC program core team consists of representatives from the key functional areas within the 32-bit Embedded Controller Division, Field Sales and Design. Extended team members will be brought in to support the program as needed. The table below summarises the inter-departmental organisation with functional areas.

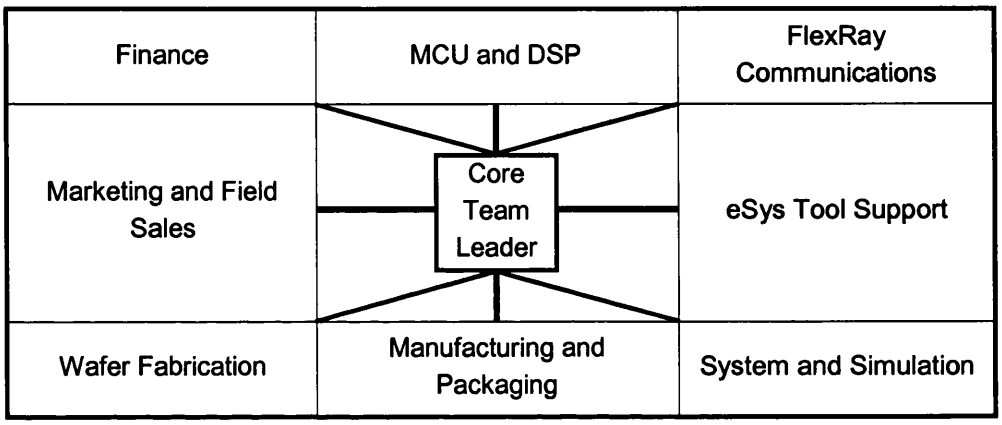

Table 3: Project Organisation and Key Functional Areas

10. Financial Plan

10.1. Underlying Assumptions

- BMW Group will purchase all SoCs in year 2005 under a contract allowing price fluctuations in-line with materials and labour costs
- Sales invoices to be paid when units dispatched from factory
- Creditors paid 7 days end of month
- Each month's production is sold in the following month
- Factory operations will be set up in Phoenix
- Pay as you go has been assumed for income taxes

10.2. Financial Highlights (Best Case Scenario)

- Cash positive in each year of operation from the year of positive PAT
- $3.1 million committed to R&D
- EMVT SoC cash surplus reinvested into next generation SoC

10.3. Financial Analysis

A summary of most likely annual PBT, Net Cash Flow and Total Cost are shown below:

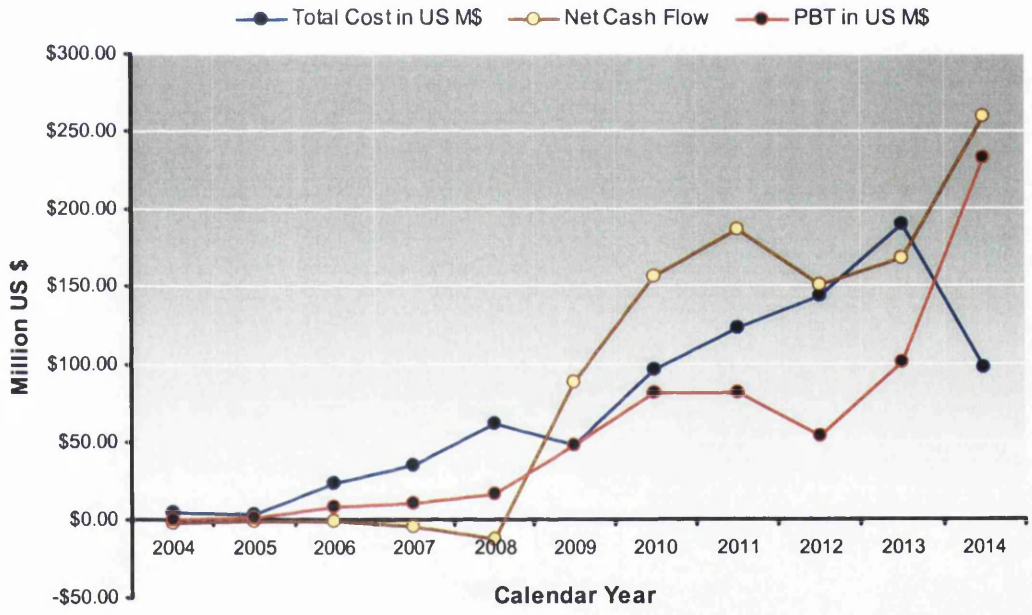

Figure 9: EVVC SoC – Financial Analysis

10.4. SENSITIVITY ANALYSIS

The first scenario to consider is that the initial EMVT SoC contract with the BMW group does not materialise, the Motorola Project Financing Team, the investor, will not be required to provide any funds. Should BMW Groups' requirements not meet expectations then production of the EMVT SoC could be scaled back. Of course this would impede the EMVT SoC controller development team's ability to look into the subsequent SoC platform. For example, if the SoC production was reduced to 100,000 units per month and there was no other external funding than that contemplated in the offer, The 32-bit Embedded Controller Division would not be able to fund the second production facility until 2007. The majority of cash expenditure is related to production and sales volumes and allowances have been made in the customer contract for raw material and labour cost increases to be reflected in the selling price of the SoC platform. It is envisaged that similar type arrangements will be put in place for the subsequent SoC platforms.

The following sensitivity analysis compares best case, most likely case and worst cases of the business. All cases assume no additional degradation of ASP's over time other than the price/volume breaks proposed in the RFQ. In addition a 3% annual increase in labour costs has been included.

Total Revenue ($M)	M2 ($M)	NPV ($M)	IRR (%)	Break-Even Point (Yr)
$350	$140	$4.44	57%	2010

Table 4: Best Case Financial Sensitivity Analysis

Total Revenue ($M)	M2 ($M)	NPV ($M)	IRR (%)	Break-Even Point (Yr)
$231	$92	$1.73	36%	2010

Table 5: Most Likely Case Financial Sensitivity Analysis

Total Revenue ($M)	M2 ($M)	NPV ($M)	IRR (%)	Break-Even Point (Yr)
$221	$88	-$5.21	5%	2011

Table 6: Worst Case Financial Sensitivity Analysis

10.5. CRITICAL RISKS AND PROBLEMS

Please see Appendix B - for more details

Risk Dimension	Perceived Risk
Development	Zero
Marketing	Low to Moderate
Financial	Low to Moderate
Quality	Low
Valuation	Low
Financing	Low
Exit	Low

Table 7: Critical Risks and Problems

Appendix A - Internal Environment Analysis

Bell Mason Dimension	Analysis
Technology, Engineering and R and D	The venture has secured its IP. There is a commitment to innovation and technology in all areas. Production is driven by technology and R and D initiatives. In addition, budgets are also in place.
Product	The EMVT SoC has well-defined, reliable and unique features and functions. Future products (Dual Issue EMVT SoC, ECU SoC, and other advanced development platforms) are in early stages of development.
Manufacturing	Motorola SPS has well defined divisions and processes to produce its products at the cost, quality, and schedules required by customers. Raw materials and finished goods are managed in an optimal fashion according to just-in-time principles.
Business plan and vision	The venture's 7-year business plan is workable, realistic and particularly spells out the first year of operation in detail. The plan identifies the corporate vision and mission, product strategy, market segmentation and competitive market position.
Marketing	A strategic and tactical marketing plan, together with a competent team and organisation to implement it, is in place. The inclusion of an Operations Manager and venture partners who have a wealth of experience in, and affiliation with, the European and American automotive industry will strengthen this plan.
Sales	A fully experienced and committed sales group is in place.
Operations, Systems and Marketing Managers	These managers have proven experience in the semiconductor segment of the automotive industry. In addition, they also have experience with leading automotive specific semiconductor organisations with annual turnovers in excess of US$1 billion. The 32-bit Embedded Controller Division within Motorola SPS under the these managers has experienced fast growth over the last eight years due to strong leadership, intelligence, energy and ethical business practices.
Team	The top-level team is composed of high-quality individuals who have measurable experience and expertise in a variety of areas. They are capable of filling several positions within their teams and adopt a `can-do' attitude. The management team has not worked together for very long as a unit, but have considerable experience in working within a team environment.
Board of Directors	Still to be finalised.
Cash	The venture is dependent on an injection of funds to establish.
Financeability	The venture is attractive to multiple investors as it is `real business' opportunity.
Control	Corporate governance issues have been addressed at all levels of the organisation.

Table 8: Internal Environment Analysis

Appendix B - Critical Risks and Problems

Development Risk- **Zero**
Currently assessed as zero due to the positive customer feedback on existing working prototypes based on the legacy PowerPC platforms.

Manufacturing Risk- **Low/Moderate**
The two main issues that need to be addressed in manufacturing are the price of the major commodity, Silicon Wafer, and the quality and supply of third party IPs. As this venture will be primarily located in the heart of "Semiconductor" country in the US, costs will be kept to a minimum due to the lower transport costs. Strong relationships will be developed with all suppliers to ensure favourable trading terms. Quality control checks have been introduced at all stages of semiconductor production. To ensure only the best supplies are sourced, two or three suppliers will be sought in the early stages of the venture. This is a `fast growth' venture and the supply of skilled labour to meet demand is paramount.

Financing Risk- **Low/Moderate**
As no traditional funds are required, this venture is not susceptible to fluctuating interest rates. However, the internal project financing body requires confidence in the expected (or promised) returns and such is guaranteed at all stages of the venture. In addition, the venture is self-funding and it is not envisaged that further injections of venture capital will be required in the future.

Marketing Risk- **Low/Moderate**
The initial marketing risk is minimised because of the BMW alliance. However, as the SoC is new to the EMVT camless engine market, the broader market needs to be educated in the features and benefits of the SoC. This will involve time and effort but will be assisted greatly with BMWs involvement.

Management Risk- **Low/Moderate**
This project falls under "a new and unique product or products" category and BMW Group have committed to assist in sponsoring the team and providing their corporate clout to arranging the required documents. Although there is a strong team in place, there is always a risk of human relationships souring over time. The team are familiar with all facets of the project and are confident that, should one member be replaced, the skills required to fill that void can be found within the team. This would be a short term solution and a professional person would be recruited to permanently replace the team member who may decide to take up his/her position. In addition, all management team members have had bottom line responsibility and have successful track records in developing profitable business ventures.

Valuation Risk- **Low**
The risk that the investor pays too much for the venture is offset as

- Investor funds are in segments, and
- The BMW Group contract will be in place and provides a base from which to work.

Exit Risk- **Low/Moderate**
Given the technological advancements, forecast sales, the solid returns and specifically the sale strategy in place, the exit risk for the investor is assessed to be very low.

Appendix C - SWOT Analysis

Strengths	Capitalise on Strengths
▪ Alliance with The BMW Group and other premier car OEMs ▪ Quality products ▪ Successfully tested prototype ▪ Well established reputation with customers ▪ Skilled and committed team ▪ Global recognition based on legacy products ▪ High margins on the SoC platform ▪ Outsourcing ▪ Low set up costs	▪ Market entry and to gain market knowledge ▪ Favourable consumer perception ▪ Europe seen as test market ground for US ▪ Second to none product quality ▪ Offers investor opportunity to balance portfolio ▪ Management has skills and experience in a variety of legacy product application areas ▪ Ability to negotiate on bulk purchases ▪ Reduces capital costs ▪ Reduces investor's risk and exposure
Weaknesses	**Address Weaknesses**
▪ Reliance on one client initially ▪ Exposure to fluctuating wafer costs ▪ Management team has not worked together for long period ▪ Board not yet finalised ▪ Sales team's lack of knowledge on SoC platforms ▪ Limited European experience	▪ Committed to expansion ▪ High margins provide flexibility ▪ Independent consultant appointed and Corporate Governance structures in place ▪ Positions to be offered to internal venture partners ▪ Professional internal sales team recruited with assistance of venture partners who will have affinity with, and experience in the next generation engine manufacturing industry ▪ Addressed by introductions through venture partners and independent Chairman
Opportunities	**Maximise Opportunities**
▪ Expanding automotive market ▪ Capitalise on low end automotive manufacturers ▪ Increasing awareness of camless engines ▪ Scope for innovation in existing market	▪ Build consumer preference for camless engine based cars ▪ Target diverse range of automotive companies ▪ Education of EMVT based camless engines as option to specialist more expensive electro hydraulic camless options ▪ Commitment to relentless innovation ensures market benchmarks challenged

Threats	Minimise Threats
▪ Imitation products ▪ Window of opportunity may be limited ▪ High number of competitors ▪ Adverse reaction to more electronics	▪ IP protection ▪ Venture turn-key and ready to be actioned ▪ Guaranteed demand through BMW Group contract ▪ Backing of one of the well-established premier and largest automotive OEM, The BMW Group

Table 9: SWOT Analysis

Appendix D - SoC Development Cycle

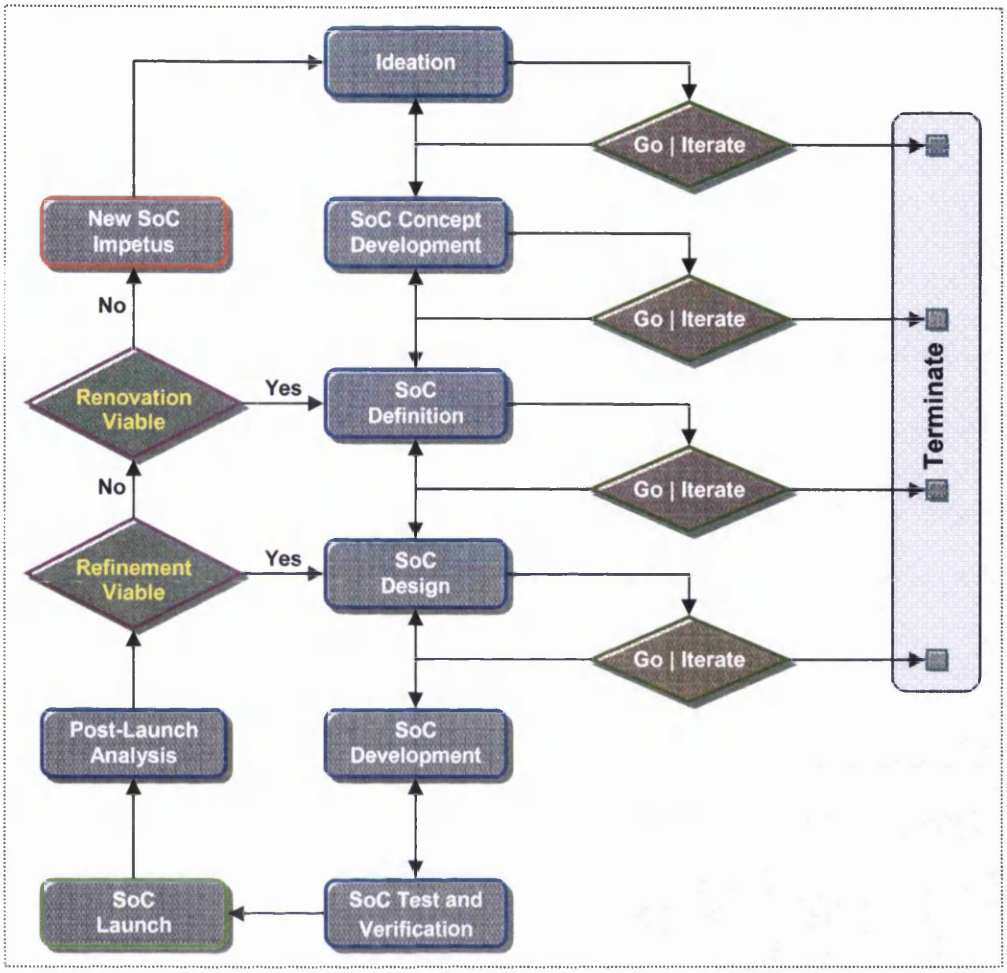

Figure 10: EVVC SoC Development Cycle

The graphic above illustrates the SoC development cycle executed by the team. This version encapsulates the core elements that are widely recognised in the semiconductor industry.

The cycle begins with an impetus: the discovery of a new technology, with or without clarity in advance as to how it might be commercialised – a great market opportunity identified through some form of market research, an obvious need to retire an existing product that has been eclipsed by the competition, etc.

Appendix E - A Non-Technical Overview of Engine Valvetrain History

After multi-valve technology became standard in engine design, varying the valve timing and the lift becomes the next step to enhance engine output, no matter power or torque. As mentioned previously, valves activate the breathing of engine. The timing of breathing, that is, the timing of air intake and exhaust is controlled by the shape and phase angle of cams. To optimise the breathing, engine requires different valve timing at different speeds.

When the rev increases, the duration of intake and exhaust stroke decreases so that fresh air becomes not fast enough to enter the combustion chamber, while the exhaust becomes not fast enough to leave the combustion chamber. Therefore, the best solution is to open the inlet valves earlier and close the exhaust valves later. In other words, the *Overlapping* between intake period and exhaust period should be increased as rev increases.
Variable Valvetrain Timing (VVT)

Without VV technology, engineers used to choose the best compromise timing. For example, a van may adopt less overlapping for the benefits of low speed output. A racing engine may adopt considerable overlapping for high-speed power. An ordinary sedan may adopt valve timing optimise for mid-rev so that both the low speed drivability and high-speed output will not be sacrificed too much. No matter which one, the result is just optimised for a particular speed. With Variable Valve Timing, power and torque can be optimised across a wide rpm band.

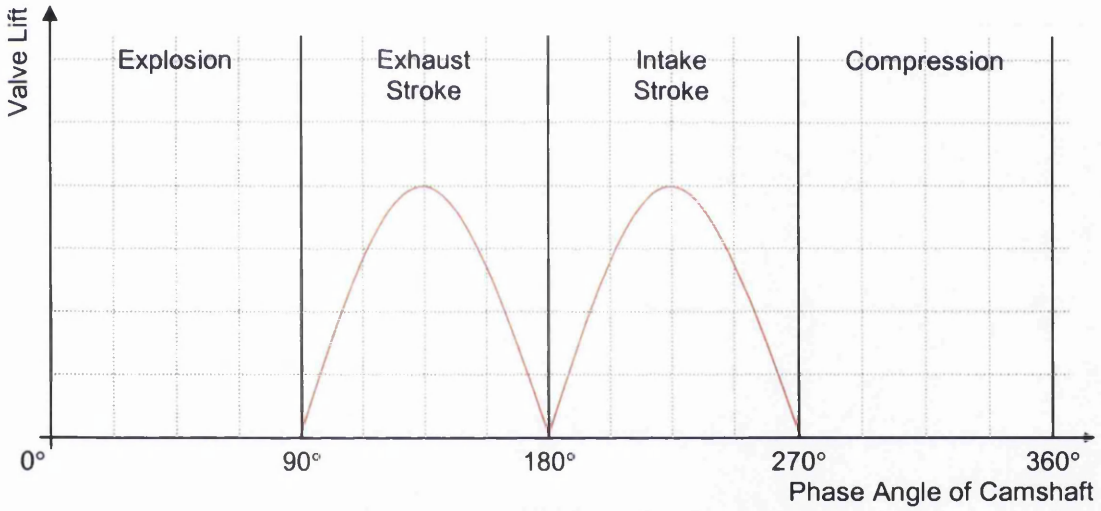

Figure 11: Graphical Illustration of Normal Valve Timing

Therefore, the most noticeable differences achieved using classic variable valvetrain timing mechanisms are; the engine can rev higher, thus raises peak power. For example, Nissan's 2-litre Neo VVL engine outputs 25% more peak power than its non-VVT version. Low-speed torque increases, thus improves drivability. For example, Fiat Barchetta's 1.8 VVT engine provides 90% peak torque between 2,000 and 6,000 rpm. Moreover, it is important to note that all these benefits come without any drawback.

Variable Valvetrain Lift (VVL)

In some designs, valve lift can also be varied according to engine speed. At high speed, higher lift quickens air intake and exhaust, thus further optimise the breathing. Of course, at lower speed such lift will generate counter effects like deteriorating the mixing process of fuel and air, thus decrease output or even leads to misfire. Therefore the lift should be variable according to engine speed.

Types of VVT and VVL Mechanisms

Cam-Changing VVT

Honda pioneered VVT in the late 80s by launching its famous VTEC [22] system (Valve Timing Electronic Control). First appeared in Civic, CRX and NS-X, then became standard in most models.

This employs two sets of cams having different shapes to enable different timing and lift. One set operates during normal speed, say, below 4,500 rpm. Another substitutes at higher speed. Obviously, such layout does not allow continuous change of timing, therefore the engine performs modestly below 4,500 rpm but above that it will suddenly transform into an enhanced version.

As low-speed torque gains too little (remembering, the cams of a normal engine usually serves across 0-6,000 rpm, while the "slow cams" of VTEC engine still need to serve across 0-4,500 rpm), drivability will not be too impressive. In short, cam-changing system is best suited to sports cars.

Honda has already improved its 2-stage VTEC into 3 stages for some models. Of course, the more stages it has, the more refined it becomes. It still offers less broad spread of torque as other continuously variable systems. However, cam-changing system remains to be the most powerful VVT, since no other system can vary the Lift of valve as it does.

Advantages and Disadvantages of Using Cam Changing VVT

Advantage: Powerful at top end.
Disadvantage: 2 or 3 stages only, non-continuous; no much improvement to torque; complex; power-to-weight ration compromise.

OEMs Employing Cam Changing VVT

- Honda VTEC
- Mitsubishi MIVEC
- Nissan Neo VVL.

Cam-Phasing VVT

Cam-phasing VVT is the simplest, cheapest and most commonly used mechanism at this moment. However, its performance gain is also the least, very fair indeed. Basically, it varies the valve timing by shifting the phase angle of camshafts. For example, at high speed, the inlet camshaft will be rotated in advance by 30° so to enable earlier intake. This movement is controlled by engine management system according to need, and actuated by hydraulic valve gears.

Note that cam-phasing VVT cannot vary the duration of valve opening. It only allows earlier or later valve opening. Earlier open results in earlier close, of course. It also cannot vary the valve

lift, unlike cam-changing VVT. However, cam-phasing VVT is the simplest and cheapest form of VVT because each camshaft needs only one hydraulic phasing actuator, unlike other systems that employ individual mechanism for every cylinder.

Continuous or Discrete Cam Phasing

Simpler cam-phasing VVT systems have just 2 or 3 fixed shift angle settings to choose from, such as either 0° or 30°. Better systems have continuous variable shifting, say, any arbitrary value between 0° and 30°, depends on rpm. Obviously this provide the most suitable valve timing at any speed, thus greatly enhance engine flexibility. Moreover, the transition is so smooth and is hardly noticeable.

Cam Phasing at Intake and Exhaust Camshafts

Some designs, such as BMW's Double Vanos system, has cam-phasing VVT at both intake and exhaust camshafts which enables more overlapping, hence higher efficiency. This explain why BMW M3 3.2 (100hp/litre) is more efficient than its predecessor, M3 3.0 (95hp/litre) whose VVT is bounded at the inlet valves. In the E46 3-series, the Double Vanos shifts the intake and exhaust camshafts within a maximum range of 40° and 25° respectively.

BMW's Vanos: A Cam Phasing VVT Mechanism

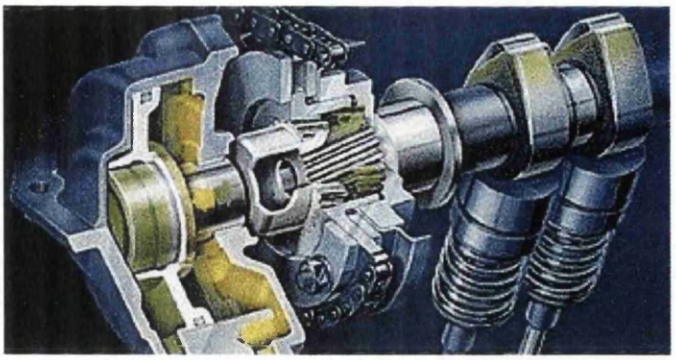

Figure 12: BMW's VANOS

As shown in the figure, the end of camshaft incorporates a gear thread. The thread is coupled by a cap, which can move towards and away from the camshaft. Because the gear thread is not in parallel to the axis of camshaft, phase angle will shift forward if the cap is pushed towards the camshaft. Similarly, pulling the cap away from the camshaft results in shifting the phase angle backward. The hydraulic pressure determines this push or pull.

There are 2 chambers right beside the cap and they are filled with liquid (these chambers are coloured green and yellow respectively in the picture). A thin piston separates these 2 chambers, the former attaches rigidly to the cap. Liquid enter the chambers via electromagnetic valves, which controls the hydraulic pressure acting on which chambers. For instance, if the engine management system signals the valve at the green chamber open, then hydraulic pressure acts on the thin piston and push the latter, accompany with the cap, towards the camshaft, thus shift the phase angle forward. Continuous variation in timing is easily implemented by positioning the cap at a suitable distance according to engine speed.

Advantages and Disadvantages of Using Cam Phasing VVT

Advantages: Cheap and simple, continuous VVT improves torque delivery across the whole rev range.
Disadvantages: Lack of variable lift and variable valve opening duration, thus less top end power than cam-changing VVT.

OEMs Employing Cam Phasing VVT

- Audi 2.0-litre - continuous inlet, 2-stage exhaust and 2-stage discrete
- BMW Double Vanos – continuous inlet and exhaust
- Ferrari 360 Modena - 2-stage discrete exhaust
- Fiat (Alfa) SUPER FIRE - 2-stage discrete inlet
- Ford Puma 1.7 Zetec SE - 2-stage discrete inlet
- Ford Falcon XR6's VCT - 2-stage discrete inlet
- Jaguar AJ-V6 and updated AJ-V8 – continuous inlet
- Lamborghini Diablo V12 since SV - 2-stage discrete inlet
- Mazda MX-5's S-VT - continuous inlet
- Mercedes V6 and V8 - 2-stage inlet (?)
- Nissan QR four-pot and V8 - continuous inlet, electromagnetic continuous inlet
- Porsche Variocam - 3-stage discrete inlet
- PSA / Renault 3.0 V6 - 2-stage inlet
- Subaru AVCS - 2-stage inlet (?)

- Toyota VVT-i - mostly continuous inlet but some also exhaust
- Volvo 4 / 5 / 6-cylinder modular engines - continuous inlet
- Volkswagen VR6 – continuous inlet

Cam-Changing and Cam-Phasing VVT

Combining cam-changing and cam-phasing VVT mechanisms could satisfy the requirement of both top-end power and flexibility throughout the whole rev range, but, as the name suggests, it is inevitably more complex. At present, only Toyota and Porsche have such designs. However, It is hoped that in the future more and more sports cars will adopt this kind of VVT.

Toyota's VVTL-i: A VVT and VVL Mechanism

Toyota's VVTL-i is, by far, the most sophisticated VVT design yet. Its powerful functions include:

- Continuous cam-phasing variable valve timing
- 2-stage variable valve lift plus valve-opening duration
- Applied to both intake and exhaust valves

The system could be seen as a combination of the existing VVT-i and Honda's VTEC, although the mechanism for the variable lift is different from Honda. Like VVT-i, the variable valve timing is implemented by shifting the phase angle of the whole camshaft forward or reverse by means of a hydraulic actuator attached to the end of the camshaft.
The timing is calculated by the engine management system with engine speed, acceleration, going up hill or down hill etc. Moreover, the variation is continuous across a wide range of up to 60°, therefore the variable timing alone is perhaps the most perfect design up to now. What makes the VVTL-i superior to the ordinary VVT-i is the "L", which stands for Lift (valve lift) as everybody knows.

In general, with VVTL-I, the variable valve-opening duration is a 2-stage design, unlike Rover VVC's continuous design, which is explained in the following section. However, VVTL-i offers variable lift, which lifts its high-speed power output substantially. Compare with Honda VTEC and similar designs for Mitsubishi and Nissan, Toyota's system has continuously variable valve timing, which helps it to achieve, far better low to medium speed flexibility. Therefore it is undoubtedly the best dynamic valvetrain control mechanism today. However, it is also more complex and more expensive to build.

Porsche's Variocam Plus: A Cam-Phasing and Cam-Changing Mechanism

Figure 13: Porsche's Variocam Plus

The Variocam was first introduced to the 968 in 1991. It used timing chain to vary the phase angle of camshaft, thus provided 3-stage variable valve timing. 996 Carrera and Boxster also use the same system. This design is unique and patented, but it is actually inferior to the hydraulic actuator favoured by other carmakers, especially it doesn't allow as much variation to phase angle. Therefore, the Variocam Plus used in the new 911 Turbo finally uses the popular hydraulic actuator instead of chain. However, the most influential changes of the *Plus* is the addition of variable valve lift. It is implemented by using variable hydraulic tappets. As shown in the figure, three cam lobes serve each valve - the centre one has obviously less lift (3 mm only) and shorter duration for valve opening. In other words, it is the "slow" cam. The outer two cam lobes are exactly the same, with fast timing and high lift (10 mm). Selection of cam lobes is made by the variable tappet, which actually consists of an inner tappet and an outer (ring-shape) tappet. They could by locked together by a hydraulic-operated pin passing through them. In this way, the "fast" cam lobes actuate the valve, providing high lift and long duration opening. If the tappets are not locked together, the "slow" cam lobe via the inner tappet will actuate the valve. The outer tappet will move independent of the valve lifter. As seen, the variable lift mechanism is unusually simple and space saving. The variable tappets are just marginally heavier than ordinary tappets and engage nearly no more space. Nevertheless, at the moment the Variocam Plus is just offered for the intake valves.

Advantages and Disadvantages of Using VVT and VVL Mechanisms Together

Advantage: Continuous VVT improves torque delivery across the whole rev range, Variable lift and duration lift high rev power.

Disadvantage: More complex and expensive

OEMs Employing Cam Changing VVT

- Porsche 911 Turbo
- Toyota 1.8-litre 190hp for Celica GT-S and hot Corolla
- 911 Carrera 3.6

Rover's unique VVC System

Rover introduced its own system called VVC (Variable Valve Control) in MGF in 1995. Many experts regard it as the best VVT considering its all-round ability - unlike cam-changing VVT, it provides continuously variable timing, thus improving low to medium rev torque delivery; and unlike cam-phasing VVT, it can lengthen the duration of valves opening (and continuously), thus boost power.

Basically, VVC employs an eccentric rotating disc to drive the inlet valves of every two cylinder. Since eccentric shape creates non-linear rotation, valves opening period can be varied.

VVC has one draw back: since every individual mechanism serves 2 adjacent cylinders, a V6 engine needs 4 such mechanisms, and that's not cheap. V8 also needs 4 such mechanism. V12 is impossible to be fitted, since there is insufficient space to fit the eccentric disc and drive gears between cylinders.

Advantages and Disadvantages of Using the VVC Mechanism

Advantage: Continuously variable timing and duration of opening achieve both drivability and high-speed power.

Disadvantage: Not ultimately as powerful as cam-changing VVT, because of the lack of variable lift; Expensive for V6 and V8; impossible for V12.

OEMs Employing Rover's VVT

- Rover 1.8 VVC engine serving MGF
- Caterham
- Lotus Elise 111S

Graphical Explanation of Mechanical Valvetrain Control Mechanisms

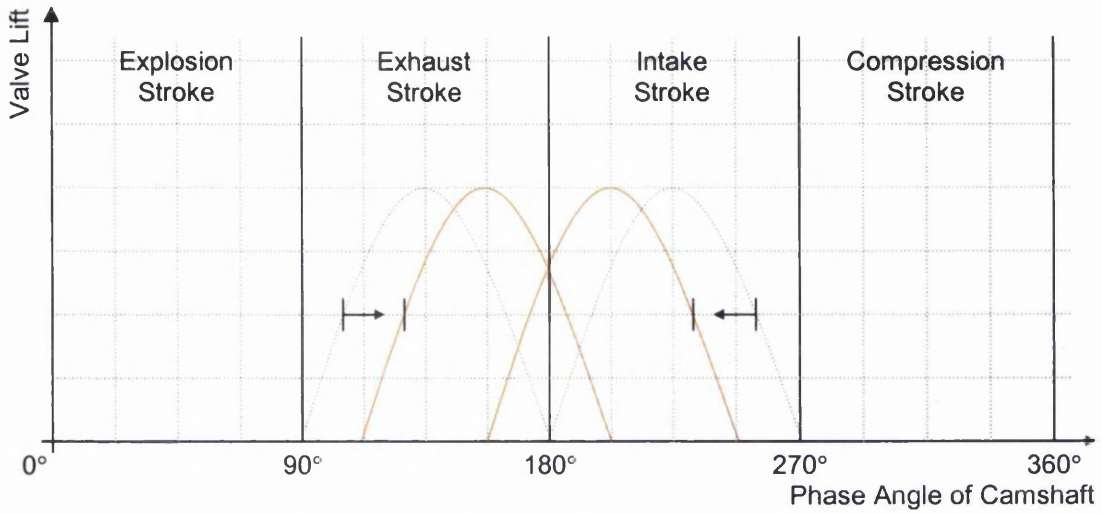

Figure 14: BMW's High Speed Double VANOS

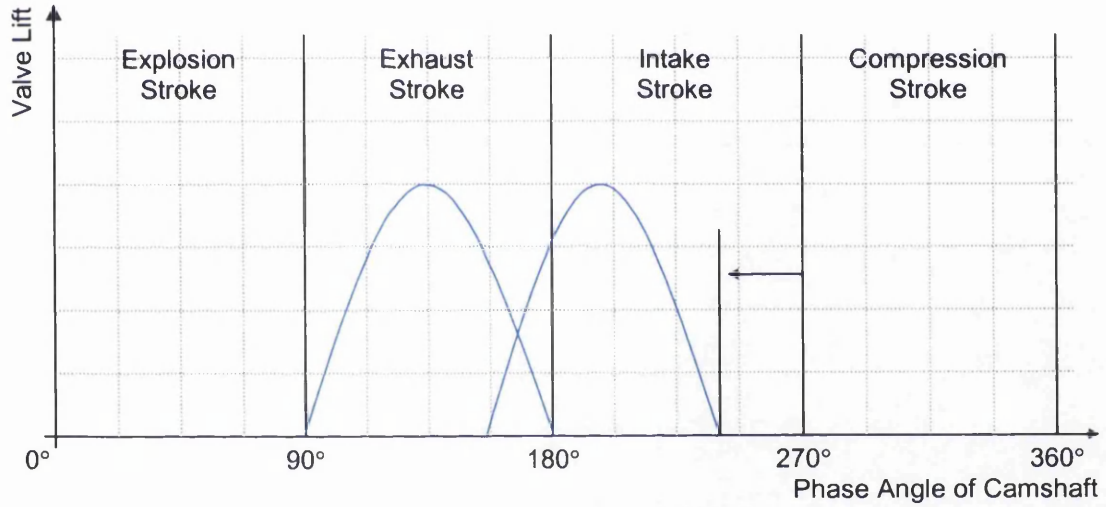

Figure 15: Toyota's VVT-i

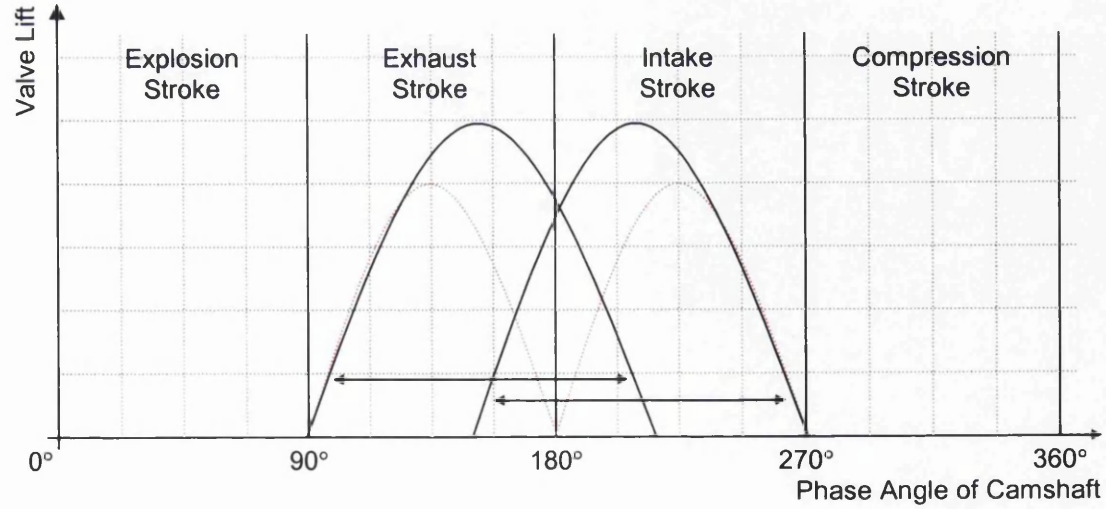

Figure 16: Rover's VVC

Appendix F - Generic EVVC Distributed System Design – The Motorola Approach

In general, the following device architecture proves to be the most efficient and cost effective solution for a V4 engine. However, the module specifications within the device need to be finalised based on the electromagnetic control algorithm of the engine OEM.

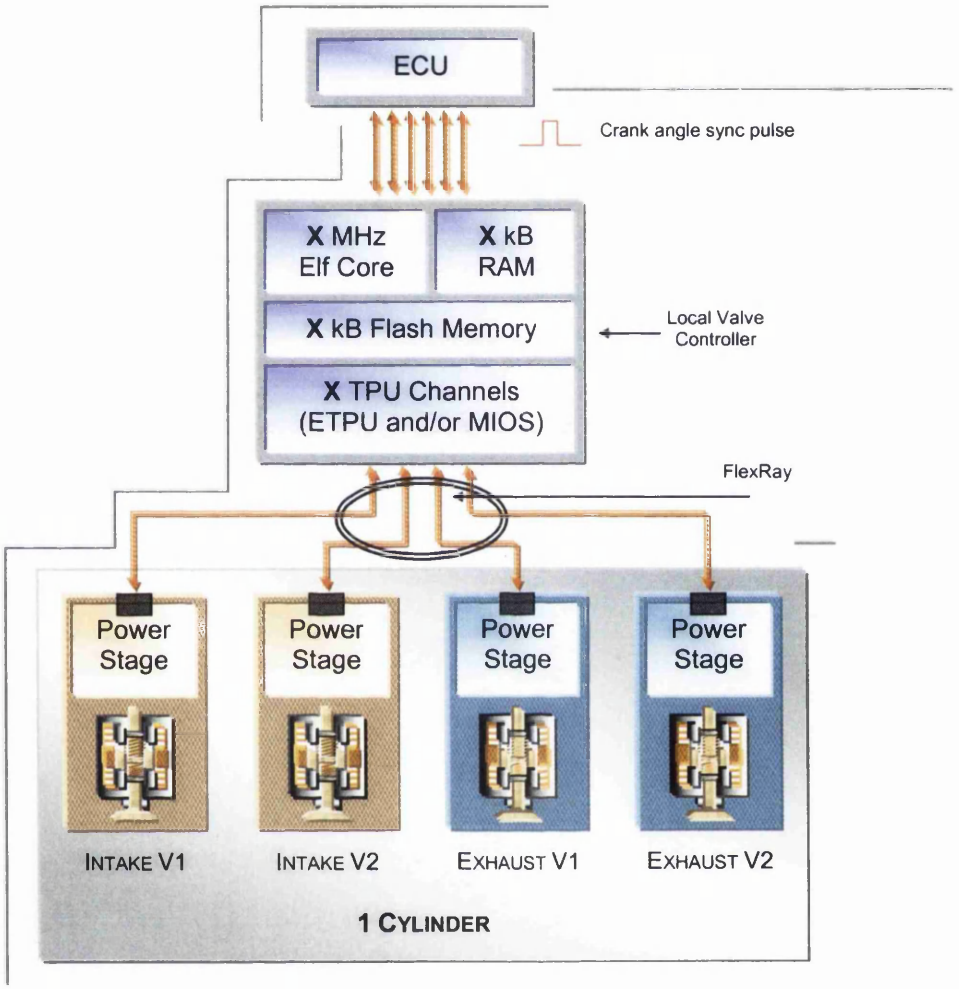

Figure 17: Distributed Valvetrain Control Using the EVVC SoC

EVVC with FlexRay

In general, as the required data rates between the local valve controller (LVC) and the power driver stage on the actuator is greater than existing industrial protocol (e.g. SPI) rates, it is hoped that the EVVC industry will vote in favour of the FlexRay protocol which can support up to 10Mbaud gross. Initial system proposal to one of our EVVC customers, namely BMW, is implemented using FlexRay and the proposed system for a V4 engine consists of 29 FlexRay controllers. Therefore, it can be concluded that implementing EVVC clusters with FlexRay will most certainly introduce an exponential growth opportunity for the FlexRay IP.

Technical Challenges and Inhibiting Factors in EVVC Introduction

EVVC mechanism, one of the most attractive panaceas to many valvetrain control issues has some hurdles to overcome. As an overview of some of the concepts mentioned above, the following are perceived:

- Fail-safe mechanism without the need of mechanical backup
- Soft landing of valves over complete engine RPM range
- Analysis of thermal stresses of actuators, sensors and control electronics at the cylinder
- Extra costs incurred with employing a 42V system. Such as the additional 42V battery, DC/DC converter bridge between the networks and added high voltage protection. DC/DC converter cost in the range of $80 to $100, while a 42V battery is expected to cost in the orders of $20 to $40 depending on kW capability
- Change in the Powertrain production technique. Assembly of valve, actuator, sensor and electronics in one module and then the attachment to the engine block accurately and securely
- Potential increase in height and thus more demanding placement options under the bonnet
- Electronics cost of the MOSFETS for the individual valve actuator
- Processing power at high temperature to keep full control of each valve
- Integration with existing powertrain control modules
- Increased wiring loads
- Meeting power budget of around 100W per control of valve

Appendix G - Design and Systems Approach Summary

Support Systems Development Summary

Development Tools

In order to fully support the EVVC program we are required to provide a range of development tools comprising the following:

Software and Hardware Tool Description			
Elf (E500)	Compiler	ETPU	Compiler
	Simulator		Simulator
	Debugger		Debugger
ACE	Compiler	Hardware	Evaluation Board
	Simulator		Nexus HTI
	Debugger	System	IDE
Software	SysGen		Debugger
	OSEK		Profiler
	LLDs		Functional Simulator

Table 10: Development Tools – Software and Hardware

Module Description	Supplier
Elf (E500)	Metrowerks and Motorola ASP
ETPU	Bytecraft, Ashware and Metrowerks
ACE	TBD
System	Metrowerks and Motorola GSG
Software	Unis and Metrowerks
Hardware	Metrowerks

Table 11: Module Tool Suppliers

The Elf compiler and debugger is supplied by Metrowerks and the Elf simulator is supplied by Motorola ASP. Incorporated into the compiler suite would be a number of modifications requested by the customer and documented separately by the tool developer. These will be evaluated by both the systems groups within Motorola and the tool supplier and will be incorporated within their next standard software revision so that BMW will not require a proprietary version of the compiler suite.

The most significant of these will be the addition of the SIMD unit. An Alpha version of the new RTA tools suite will be available by Q202.

Motorola will supply all the licenses. Early notification of updates/new releases, with a possibility to upgrade will be as per appropriate modular tool supplier.

Use of SteamRoller for the Evaluation of SIMD Capabilities

High-level meeting with NCSG in November 2001 confirmed that there would possibly be a delay with the tapeout of the SteamRoller IP. In addition, it was also decided that BMW and Volvo would be provide with an Alpha version of the eSys development platform, based around the SteamRoller architecture, by early Q302.

Third Party Integration Assessment Summary

All software development tools are supplied from third party vendors. The customer expects Motorola to assume full responsibility for the provision of all development tools required for the duration of the project. Motorola will work with the selected third parties in order to ensure that customer requirements are met and will assume responsibility for addressing and resolving any development tools issues that arise during the customer's project development phase.

In addition to the low and high-level software development platforms, it is envisaged that FlexRay IP will be used to support BMW's LVC cluster with data transfer rate of up to 7.5Mbaud. This will primarily be done in the Munich and Shaumberg offices. The FlexRay group within Motorola has provided all the high-end tools necessary to develop a complete multi-processor platform based around the elf core (please refer to elf core diagrams for further explanations).

Role of External Provider in Solution Development

All supplied tools will be, as far as possible, the vendor's standard products. The supply, installation and technical support will be provided by the responsible third party, although Motorola will, if required, act as the first point of contact for any development or simulation tools issues. Where BMW have requirements not accommodated by the vendors' standard products, attempts will be made, where possible, to include modifications to the standard products to meet those requirements. The provision of proprietary development tools is to be avoided where possible.

Recommended Partner(s):

The following third parties will be used to provide development and simulation tools:

- Metrowerks – Various module compilers, simulators and debuggers
- Bytecraft – ETPU compiler
- Unis – SysGen software
- Ashware – ETPU simulator

Software Deliverables

- Description of the Test Environment
- User description for each function and functional specification
- Traceability Matrix between the Requirements and Tests
- Software Release Guide (SRG), including installation instructions
- Quality Assurance Plan
- LLD Object files
- Tool executable files
- Project Schedule
- Delivery of LLDs and Tools to Teves: Alpha version – Q302

Appendix H - Legal and Standards Assessment

A development contract framework agreement needs to be finalised by both parties, i.e. Motorola and BMW. The agreement will cover development objectives, scope of development activities, delivery schedule and pricing. Exclusivity, Intellectual Property and Liability will also be included in the document.

Appendix I - Technology Transfer and Intellectual Property (TT and IP)

Please see attached (available at request) Memorandum of Understanding (MoU) and Non-Disclosure Agreement (NDA) for information on TT and IP.

Appendix J - Competitor Analysis

List of Companies (→) and Competitive Elements (↓)	Motorola SPS	Infineon Technology	Texas Instruments	STM Inc.	NEC
Technology	5	3	3	5	3
Product	5	4	3	4	3
Financial backing	3	5	5	4	4
Customer exclusivity	4	5	5	5	4
Distribution channels	5	4	5	4	5
After sales service	4	5	4	4	4
Position in life cycle	5	3	2	1	3
Cost structure	5	3	2	2	3
Selling force	4	4	4	2	4
Totals	**40**	**36**	**33**	**31**	**33**
Legend: 5 = excellent; 1 = poor					

Table 12: Competitor Analysis

Appendix K - EVVC – SoC Project Staffing Needs

Organisational Member	Calendar Year							
	2005	2006	2007	2008	2009	2010	2011	2012
Operations Manger	1	1	1	1	1	1	1	1
Systems Manager	1	1	1	1	1	1	1	1
Admin Staff	2	2	2	2	2	2	2	2
R and D Staff	12	9	7	7	7	8	9	12
Design Staff	85	20	20	20	20	20	20	15
Technical Marketing	5	5	5	3	3	3	3	3
Sales Staff	6	7	7	7	7	7	6	6
Finance Staff	10	7	7	2	2	2	2	2
Quality	5	5	5	5	5	5	5	5
Packaging	6	6	6	6	6	6	6	6
Wafer Fab	12	12	10	10	10	8	8	8
FAEs	14	12	12	12	12	12	12	12
Third Party Tools Support	5	5	5	5	5	5	5	5
Software Support	8	8	8	8	8	8	8	8
Hardware Support	9	9	9	9	9	9	9	9
Total	**181**	**109**	**105**	**98**	**98**	**97**	**97**	**95**

Table 13: EVVC SoC – Staffing Needs

Appendix L - EVVC – SoC Project Volume independent Organisational Budget

Organisational Element	Budget In k$ (Base Cost) – Volume Independent							Total
	2004	2005	2006	2007	2008	2009	2010	
Systems Engineering	80	60	18	2.7	2.7	2.7	2.7	168.8
Design Engineering	500	375	112.5	16.9	16.9	16.9	16.9	1,055
R and D Staff	300	300	300	300	300	300	300	2,100
Administration	20	15	4.5	0.7	0.7	0.7	0.7	42.2
Product Engineering	50	37.5	11.3	1.7	1.7	1.7	1.7	105.5
Software Engineering	200	150	45	6.8	6.8	6.8	6.8	422
Process Engineers	30	22.5	6.8	1	1	1	1	63.3
Device Technicians	35	26.3	7.9	1.2	1.2	1.2	1.2	73.9
Process Technicians	30	22.5	6.8	1	1	1	1	63.3
Modelling and Simulation	150	112.5	33.8	5.1	5.1	5.1	5.1	316.5
Characterisation	75	56.3	16.9	2.5	2.5	2.5	2.5	158.3
Test Engineer	110	82.5	24.8	3.7	3.7	3.7	3.7	232.1
Marketing	50	50	50	50	50	50	50	350
Sales	100	100	100	100	100	100	100	700
Materials and Planning	30	22.5	6.8	1	1	1	1	63.3
Documentation	25	25	25	25	25	25	25	175
3rd Party / Contractor	100	75	22.5	3.4	3.4	3.4	3.4	211
Reliability	40	30	9	1.4	1.4	1.4	1.4	84.4
Manufacturing Base Cost	750	562.5	168.8	25.3	25.3	25.3	25.3	1,582.5
Packaging	200	150	45	6.8	6.8	6.8	6.8	422
Mask Preparation	1,000	750	0	0	0	0	0	1,750
Layout	80	60	18	2.7	2.7	2.7	2.7	168.8
Program / Project Leader	70	52.5	15.8	2.4	2.4	2.4	2.4	147.7
Travel	125	93.8	28.1	4.2	4.2	4.2	4.2	263.8
Other	200	150	45	6.8	6.8	6.8	6.8	422
Total	4,350	3,381.3	1,121.9	572	572	572	572	11,141.3

Table 14: EVVC SoC – Organisational Budget

Appendix M - BMW SoC Variants and Sample Proposal (Submitted to BMW – Highly Confidential)

The proposed solution for BMW's SoC has five variants. Please see attached BMW proposal (available at request) for more details. Summary of the five variants are mentioned below:

SoC Variants:

- eSys derivative @200 MHz with MIOS, no EBI
- eSys derivative @200 MHz with ETPU, no EBI
- eSys derivative @200 MHz with MIOS and EBI
- eSys derivative @200 MHz with ETPU and EBI
- Full eSys (e.g. Cobra or Coral) @ 200MHz, 7-stage pipeline with ETPU (48 channels) and MIOS

Along with the device offers, BMW expects Motorola to include the following tasks/items:

- Components/System Cycle accurate Simulation Support.
- Low-level driver software & tools support (as official products)
- Full FlexRay support
- Flash download
- Parity
- CRC
- Parameter exchange
- Stack depth analyser
- Documentation, verification & qualification of all software related items
- Compiler, assembler, linker & debugger requirements
- Initial elf based device, namely SteamRoller for silicon evaluation
- Design Support for proprietary custom blocks

The schedules for the SoC variants are shown in the table below: Please note that only one of the options mentioned in table will be pursed.

Description	Tape Out	Samples	Qualification	Production
Option I	Q2-2004	Q3-2004	Q4-2004	Q1-2005
Option II	Q2-2004	Q3-2004	Q4-2004	Q1-2005
Option III	Q2-2004	Q3-2004	Q4-2004	Q1-2005
Option IV	Q2-2004	Q3-2004	Q4-2004	Q1-2005
Option V	Q2-2004	Q3-2004	Q4-2004	Q1-2005

Table 15: BMW Schedules for the EVVC SoC

Technology Assessment and Approach for BMW

In order to achieve the eSys derivative price target set by BMW and therefore have the opportunity to participate in the EVVC project, die size is absolutely essential to minimise die size. Our existing wafer fabrication technology does not allow us to achieve a small enough die size to be sufficiently competitive. However, initial size estimations show that the preferred die sizes are achievable using the hip8 technology.

On the other hand, our FlexRay IP is chosen as the communications protocol between the LVC and local controller clusters. In addition, the new embedded NVM roadmap is critical to the success of the EVVC program.

REFERENCES AND BIBLIOGRAPHY

1. www.bized.ac.uk
2. World Semiconductor Trade Statistics, SIA, http://www.semichips.org/pre_statistics.cfm
3. Various Motorola resources in the internal online repository
4. Competitive Strategy: Techniques for analysing industries and competitors, Michael Porter – Freepress - New York circa 1980
5. The McLean Report 2002 Edition: An In-Depth Analysis and Forecast of the Integrated Circuit Industry, IC Insights (2002)
6. The Marketing Plan, M. McDonald and P Morris 1990 – Heinemann Professional publishing
7. Dees, J. Gregory. 2001. "Social Entrepreneurship" in Dees, J. Gregory, et al., eds. *Enterprising Nonprofits: A Toolkit for Social Entrepreneurs.* New York: Wiley. pp. 1-18
8. Doyle, Michael and David Straus. 1976. "How to Find Win/Win Solutions" in *How to Make Meetings Work!* New York: Berkley Books. pp. 55-59
9. Kelly, Tom. 2001. "The Makings of a Hot Group" in *The Art of Innovation.* New York: Random House. pp. 70-71
10. Drucker, Peter F. 1985. "The Practice of Innovation" in *Innovation and Entrepreneurship.* New York: HarperCollins. pp. 21-36, 133-140
11. Collins, James C. and Jerry I. Porras. 1996. "Building Your Company's Vision." *Harvard Business Review,* September-October
12. Porter, Michael E. 1996. "What is Strategy?" *Harvard Business Review,* November-December
13. Sahlman, William A. 1997. "How to Write a Great Business Plan." *Harvard Business Review,* July-August
14. Timmons, Jeffry A. 1999. "The Business Plan" in *New Venture Creation,* pp. 367-407
15. Weston, Anthony. 2001. "Families of Moral Values," "Some Tradtional Ethical Theories," and "When Values Clash" in *A 21st Century Ethical Toolbox.* New York: Oxford University Press, pp. 68-116
16. Hart, Stuart L. and Clayton M. Christensen 2002. "The Great Leap: Driving Innovation From the Base of the Pyramid. *MIT Sloan Management Review,* Fall
17. Open Innovation: The New Imperative for Creating and Profiting from Technology, Henry Chesbrough, Harvard Business School Press, April, 2003
18. Managing Innovation 2nd ed, Joe Tidd, et al, John Wiley and Sons Ltd. - July, 2001
19. Leading for Innovation and Organising for Results, Francis Hesselbein, et al Jossey Bass Wiley, October, 2001
20. At Work with Thomas Edison: 10 Business Lessons from America's Greatest Innovator, John P. Keegan (Foreword), Blaine McCormick, National Book Network, September, 2001
21. Creative Technological Change: The Shaping of Technology and Organisations (The Management of Technology and Innovation) Ian McLoughlin Routledge, an imprint of Taylor & Francis Books Ltd - February, 1999
22. How Stuff Works, http://auto.howstuffworks.com/question229.htm

System Level Definition, Design and Algorithmic Evaluation of an SoC Platform for the Control of Camless, Electromagnetic Actuator Driven Next Generation Car Engines

Mohamed Anas

A portfolio submitted to

The Universities of

Edinburgh

Glasgow

Heriot Watt

Strathclyde

for the Degree of

Doctor of Engineering in System Level Integration

March 2005

Abstract

This portfolio presents a comprehensive real-time system level rapid development methodology and a controller platform to evaluate the operation of high performance electromagnetic actuator driven camless internal combustion engines (ICE). This development platform is based on the next generation automotive qualified single-instruction-multiple-data (SIMD) PowerPC[1] system-on-chip (SoC) with tightly coupled digital signal processing (DSP) functionality.

Camless engine configurations are multivariable and nonlinear, thus imposing challenging control problems associated with control authority, long sensor delays, and strongly coupled subsystems [1]. A complete study of the impact of modular controller architecture on the camless engine dynamic response is also investigated.

The implemented methodology demonstrates the advantages of a systematic approach to developing advanced technology powertrain control systems. The proposed development platform enables developers to complete complex software and hardware development before moving to silicon, significantly shortening the development cycle and improving confidence in the design.

[1] PowerPC is a Family of microprocessors produced by an industry group including Motorola, IBM, and Apple.

Table of Contents

List of Figures and Tables

1 Introduction

This portfolio describes the development, implementation and testing of advanced real-time control strategies using the next generation PowerPC based SoC for the control of electromagnetic actuator (EMA) driven valvetrains of camless internal combustion engines. This section examines the present status and identifies some of the shortcomings of such actuators and their controller structures. Through reference to previous work the need for standardised rapid development methodologies to control the actuator platform is made apparent.

Conventional internal combustion engines use mechanically driven camshafts to actuate intake and exhaust valves. Like a very simple software program that contains only one set of instructions, such mechanical cams always open and close the valves at the same precise moment in each cylinder's constantly repeated cycle of fuel-air intake, compression, combustion, and exhaust [2]. They do so regardless of whether the engine is idling or spinning at maximum rpm. While this system is convenient and reliable, the fixed timing of the valve events with respect to the piston motion is typically selected as a compromise among fuel economy, emissions, maximum torque output, valvetrain noise, vibration and harshness [2, 3].

The growing need to improve fuel economy and particularly reduce emissions led to the introduction of an alternative valvetrain technology, namely a camless valvetrain [1]. Camless engines, employing electrohydraulic, electromagnetic or hydromechanical valvetrains, offer the next step in engine flexibility. Such engines allow independent control of valve timing and lift without mechanical linkage to the crankshaft [1, 3]. Various studies have shown that a camless valvetrain can alleviate many otherwise necessary engine design tradeoffs by supplying extra degrees of freedom to the overall powertrain system [1].

Automotive engines equipped with electromagnetic camless valvetrains have been studied for over thirty years but production worthy vehicles with engines of this type are still not available due to difficulties in ensuring adequate and reliable electromagnetic valve performance [3, 4, 5, 6, 7]. For an electromagnetic camless valvetrain (EMCV), the actuator noise caused by high contact velocities of the moving parts has been identified as a key problem [3, 6, 7].

1.1 Background

Various studies have shown that optimisation of the inlet and exhaust valve timing of an automotive internal combustion engine results in high fuel efficiency, low emissions and improved torque performance [1, 2, 3, 6, 7, 17, 18]. Because of the potential benefits many automotive engine manufacturers and research laboratories are developing mechanisms that can provide the valve event variability [4, 5]. A promising mechanism is the EMA as shown in Figure 1.1. This EMA relies on two electromagnets that catch and hold an armature that moves with a damped oscillation between two extreme positions under the forcing of two springs. The control signal to the electromechanical actuator is the voltage applied to either one of the coils of the two electromagnets. Therefore, for reliable operation, the primary objective is of the controller platform is to ensure accurate valve opening and closing with small contact velocity of all the moving parts.

Conventional internal combustion engines use mechanically driven camshafts to actuate intake and exhaust valves. While this system is convenient and reliable, the fixed timing of the valve events with respect to the piston motion is typically selected as a compromise among fuel economy, emissions, maximum torque output, valvetrain noise, vibration and harshness (NVH) [3].

Specifically it has been shown that controlling the intake valve events can eliminate the need for throttled operation in gasoline engines [8], thereby improving fuel economy [3]. Other benefits of camless engines include, higher maximum torque output, which is optimised for different driving conditions, cylinder de-activation, and elimination of external exhaust gas recirculation (EGR), etc. [6].

While variable valve timing (VVT) can be obtained using a wide spectrum of different technologies [1, 6], the highest degree of flexibility and the fastest VVT capability is achieved in truly camless engines with either electro-hydraulic [6] or electromechanical actuators [1]. These camless actuator technologies are under intensive development by several engine manufacturers with the electromechanical technology currently considered by many to be in a relatively more developed stage [4, 5, 9, 10, 11, 12, 13, 14, 15, 16]. The issues that have to be addressed in the actuator design include cost, reliability, packaging, power consumption, noise and vibrations. The valve landing noise has been identified as the main problem with the electromechanical actuator technology. It may, in fact, preclude the usage of such systems, if satisfactory solutions are not found [6].

1.1.1 Construction and Operation of the Electromechanical Actuator

Figure 1.1 illustrates the actuator at open, neutral and closed positions. There are two magnets, two springs and an armature in the actuator. The two magnets are coil wound on ferromagnetic material. The coils are driven by currents generated by an electronic system, driven by a pulse-width modulated voltage. The activated coil generates a magnetic field applying a force on the armature.

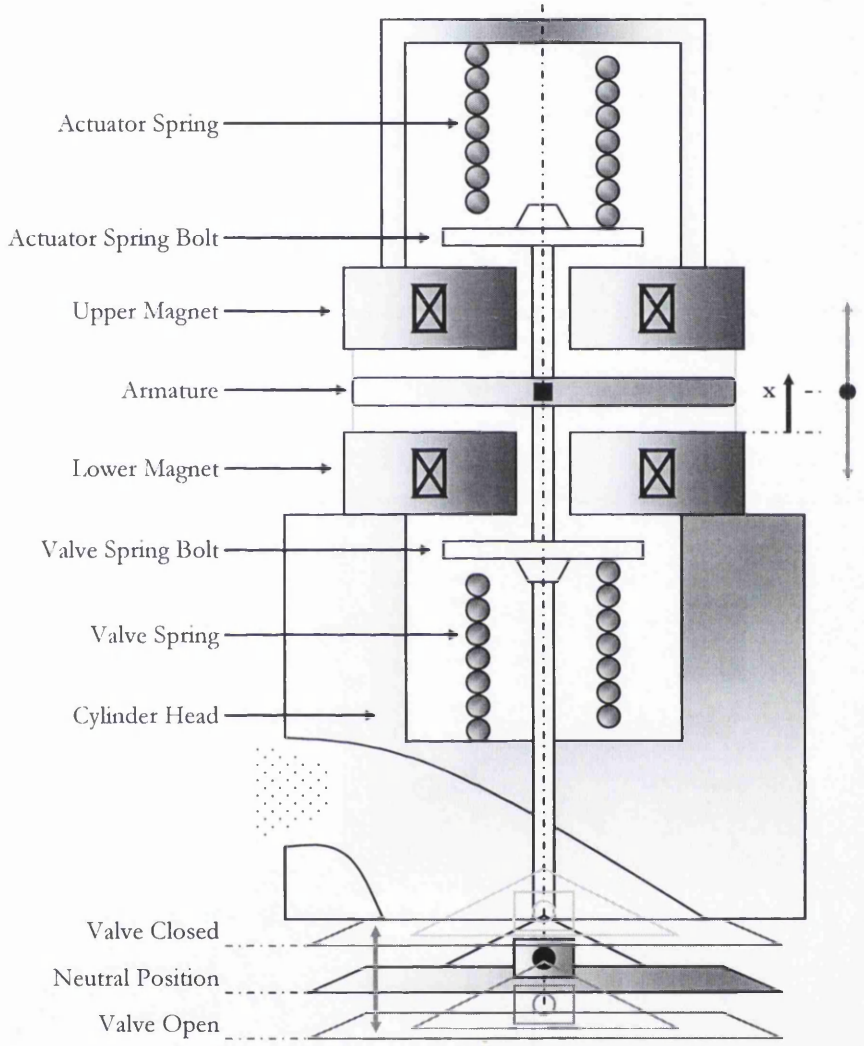

Figure 1.1: The Elctromechanical Actuator

The two springs are adjusted such that both are always compressed for any position of the armature. The actuator uses the spring force to accelerate the masses, then uses the electromagnetic force to attract and dwell the valve. When there is no current on coils, the spring-mass system stays at the neutral position. Voltage applied on the upper coil closes the valve generating a holding current, which depends on the spring force and the pressure difference between the cylinder and the exhaust/intake manifold. Because of the symmetry, analysis is done only on the valve-opening event.

The driving current of the EMA and the corresponding valve profile is shown in Figure 1.2 [7]. If the valve is scheduled to open at t_0, the voltage applied to the upper coil should be regulated to zero at, $t_0 - \tau_m$; the time τ_m is necessary for the discharge of the magnetic field. Then a voltage with high duty cycle, d_c, is applied to the lower coil at t_1. A magnetic field is generated that attracts the armature to contact the lower coil and maintains the maximum valve lift. The current generated by this high voltage, d_c, is denoted as the catching current. Once the contact is ensured and quasi-static conditions of the mechanical subsystem are reached, the voltage applied to the lower coil can be reduced. The time that the lower duty cycle voltage, d_h, is applied to the lower coil is denoted by the t_2. Controlling t_2 varies the power consumption of the electrical subsystem. When the valve is closing, the operation is similar with the voltage applied on the lower coil regulated to zero at t_3. In Figure 1.2, the time intervals are defined as follows: $\tau_1 = t_1 - t_0$ and $\tau_2 = t_2 - t_0$.

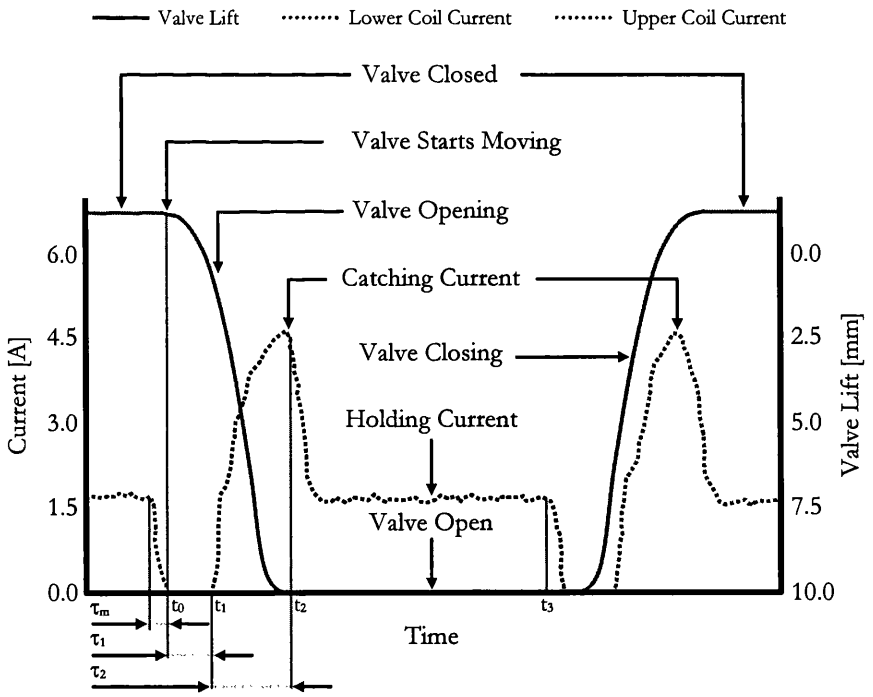

Figure 1.2: The Driving Current of the EMA and the Corresponding Valve Profile

To summarise, the actuator consists of electrical, magnetic and mechanical subsystems, which are interconnected with each other as shown in Figure 1.1. Because of the symmetry, analysis is done only on the valve-opening event.

1.1.2 Electromechanical Actuator – Soft Landing

Prior work confirms that high contact velocities of the moving parts of the actuator cause the noise in electromechanical actuator [17]. The noise can be reduced if the contact velocities are reduced, i.e. the so-called *soft-landing* is achieved. In a conventional valvetrain the fixed valve profiles are carefully optimised to reduce the noise and the optimal solution is mechanically embedded into the precision valvetrain design during manufacturing of the camshaft lobes. In a camless valvetrain, it is the responsibility of the electronic controller platform to ensure that adequate actuator performance at varying engine operating conditions is achieved.

2 Actuator Controller SoC Platform

2.1 Overview of System Level Development Methodology

In order to considerably reduce the development time, decision was made to use the Real-Time Workshop Embedded Coder (RTWEC) [30] to generate, test, and deploy production worthy C code for use in the PowerPC based VVT embedded platform particularly enabling the integration of various legacy code signatures. Therefore, it was necessary to establish thorough understanding of the RTWEC to provide the setting up of the real time actuator electromagnetic controller framework as shown in Figure 2.1.

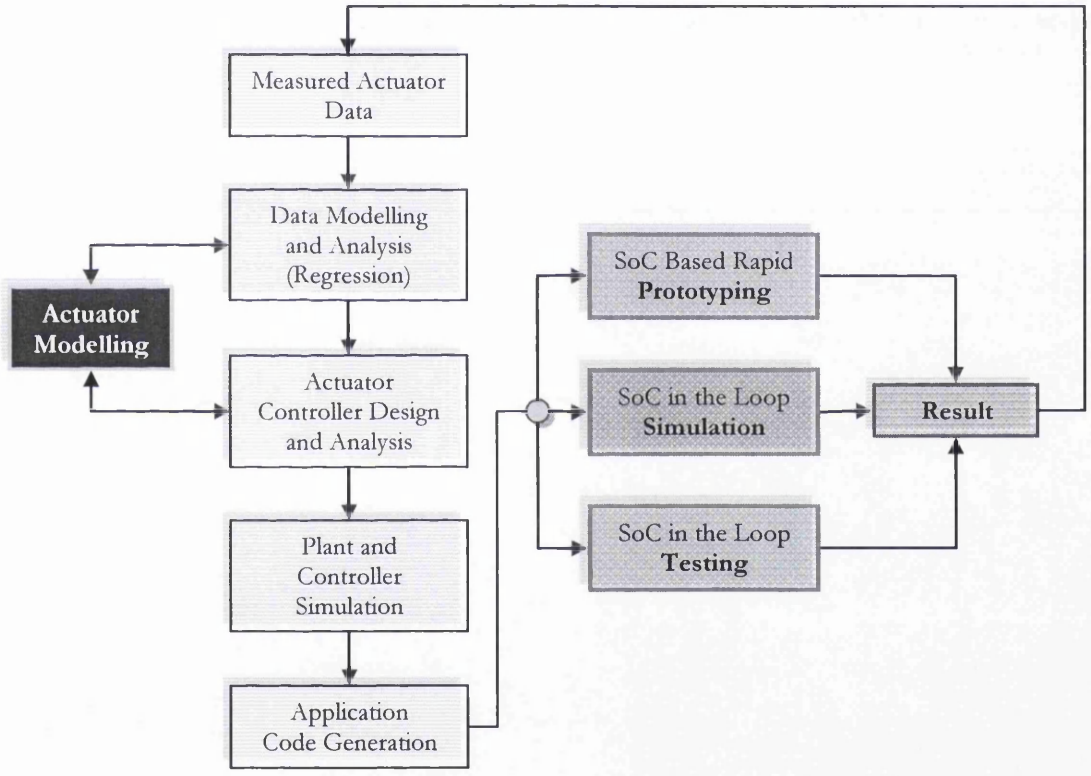

Figure 2.1: Camless Engine Controller Development Platform

2.2 Actuator Controller Design, Development and Analysis

The PowerPC SoC based actuator control system is required to ensure accurate valve closing and opening events (timing). The engine management system deploys the timing strategy on the drivers torque demand and other vehicle variables [1, 6, 7]. One of the key objectives of the controller platform is to reduce the armature-coil and valve-cylinder contact velocities, the so called *soft-landing*, which in turn reduce noise and component wear [1, 3, 6, 7, 17]. Modern engine manufacturers design camshafts to achieve a low 0.04 m/s contact velocity at low engine speeds and in conventionally driven engines, this velocity increase linearly with engine speed [4, 10, 13, 15, 17, 18].

In order to achieve better tracking of the desired valve position, an iterative learning controller (ILC) is implemented as it is a proven technique for improving the transient response of systems following the same trajectory motion or operation over and over [19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29].

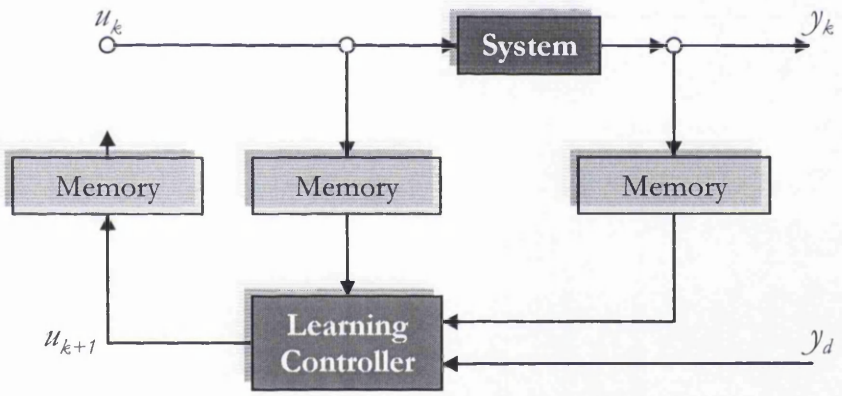

Figure 2.2: Configuration of the Actuator Iterative Learning Controller

The basic idea of the ILC implemented is illustrated in Figure 2.2. All the signals shown are assumed to be defined on a finite interval. The subscript k indicates the number of full armature travel cycles. During the k^{th} cycle, the input armature position $u_k(t)$ is applied to the system, producing the output $y_k(t)$. These signals are stored in the memory units until the trial is over, at which time they are processed by the ILC algorithm. Based on the error $(e_k(t) = y_d(t) - y_k(t))$ that is observed between the actual output and the desired output, the ILC algorithm computes a modified input signal $u_{k+1}(t)$ upon full armature travel that will be stored in memory until the next time the system operates, at which time this new input signal is applied to the system. This new input produces smaller error than the previous input.

The overall controller is modelled and implemented using Matlab®, Simulink® and Stateflow®. MathWorks' Real-Time Workshop is then used to generate executable stand-alone C code of the algorithm modelled in Simulink [30]. The resulting code is then hand optimised before generating the binary executable for the PowerPC SoC.

3 Simulation and Modelling for Control

3.1 Modelling the Actuator

The model of the actuator consists of a mechanical, electrical and a magnetic subsystem. Refer Table 1 for an explanation of symbols and parameters used for the development of the model.

Symbol	Unit	Explanation
(Y_d), Y	m	desired armature position
V	ms^{-1}	armature velocity
$V_{u/l}$	V	voltage upper / lower coil
$I_{u/l}$	A	current upper / lower coil
$\Phi_{u/l}$	NmA	flux upper / lower coil
$F_{u/l}$	N	magnetic force
F_{flow}	N	force due to gas airflow
m	kg	mass of moving part of actuator
G	kgs^{-1}	friction coefficient
D	kgs^{-2}	Spring constant
2h	m	thickness of armature disc
R	Ω	resistance or a coil
	Notations	
S(t)		continuous time signal
S[n]		discrete time signal
S^0		signal at equilibrium point
$s[n] = S[n] - S^0$		deviation between signal and equilibrium point
s[n, k]		discrete signal of the k^{th} cycle
v		Vector
M		Matrix

Table 1: Signals, Parameters and Notations

The mechanical subsystem is modelled as a spring-mass-damper system including the external magnetic forces F_u of the upper and F_l of the lower electromagnet. A force balance yields [7];

$$m\frac{dV(t)}{dt} = -DY(t) - GV(t) + F_l(t) + F_u(t) \tag{1}$$

The two coils are modelled by an electrical subsystem, consisting of a resistance/reluctance-circuit. The coil reluctances are inversely proportional to the armature gaps Y-Y_l-h or Y_u-Y-h, respectively. The coil currents $I_{l/u}(t)$ are modelled with a non-linear function f_I of the armature gap and the flux, yielding;

$$V_l(t) = Rf_I(\Phi_l, Y - Y_l - h) + \frac{d\Phi_l(t)}{dt} \tag{2}$$

and

$$V_u(t) = Rf_I(\Phi_u, Y_u - Y - h) + \frac{d\Phi_u(t)}{dt} \tag{3}$$

as the two equations for the lower and upper coil, respectively. The mechanical and electrical subsystems are linked by the magnetic force equations of the two electromagnets,

$$F_l = -f_{mag}(\Phi_l, Y - Y_l - h) \tag{4}$$

and

$$F_u = f_{mag}(\Phi_u, Y_u - Y - h) \tag{5}$$

To summarise, the actuator has two inputs, upper voltage $V_u(t)$ and lower voltage $V_l(t)$, respectively. The actuator output is the armature position $Y(t)$. The four elements of the state vector are position $Y(t)$, velocity $V(t)$, lower flux $\Phi_l(t)$ and the upper flux $\Phi_u(t)$. Thus, the state space description of the model is given by;

$$\frac{dY}{dt} = V \tag{6}$$

$$\frac{dV}{dt} = -\frac{D}{m}Y - \frac{G}{m}V + \frac{F_u}{m} + \frac{F_l}{m} \tag{7}$$

$$\frac{d\Phi_u}{dt} = -Rf_I(\Phi_u, Y_u - Y - h) + V_u \tag{8}$$

$$\frac{d\Phi_l}{dt} = -Rf_I(\Phi_l, Y_l - Y - h) + V_l \tag{9}$$

3.2 Design of the Iterative Learning Controller

The input and the output sequences of the closed loop system are y_d[n] and $y[n]$, respectively. To formulate an ILC in a compact way, it makes sense to describe this mapping by defining the operator;

$$\Gamma:\Re^N \mapsto \Re^N \; as \; y=\Gamma(y_d) \tag{10}$$

$$y_d=\begin{bmatrix} y_d[n_{fb}] \\ . \\ . \\ y_d[n_{fb}+N-1] \end{bmatrix} \tag{11}$$

and

$$y=\begin{bmatrix} y[n_{fb}+1] \\ . \\ . \\ y[n_{fb}+N] \end{bmatrix} \tag{12}$$

Where n_{fb} is the indice of the first sample after switching on the feedback controller. N is the number of values later involved in the ILC. As mentioned above, the lower case notation of a signal stands for its deviation from the equilibrium point.

The purpose of the ILC is to find some vector y_d^* with the property $y_d=\Gamma\left(y_d^*\right)$. In order to solve this problem, the cyclic opening and closing of the valve is exploited. Let the cycles be numbered with k. In the first cycle, the input vector $y_d[1] \equiv y_d$. is applied to the system. This vector and the corresponding output vector $y[1]$ are used to generate an improved input vector $y_d[2]$ for the next cycle, and so forth. Thus, a linear formulation of the ILC algorithm reads as [31].

$$y_d[k+1]=Sy_d[k]+E\left(y_d[1]-y[k]\right) \tag{13}$$

Where the matrices S and E weight the previous input $y_d[k]$ and the previous error $e[k] = y_d[1] - y[k]$, respectively. They have to be chosen in a way that the sequence $\{y_d[k]\}$ converges

$$y_d^*=\lim_{k\to\infty} y_d[k] \tag{14}$$

3.2.1 Design of the Learning Controller

The learning controller is designed using the linearised model of the plant previously developed. The convolution matrix,

$$P=\begin{bmatrix} h[1] & 0 & 0 & 0 & . & . & 0 \\ h[2] & h[1] & 0 & 0 & . & . & 0 \\ h[3] & h[2] & h[1] & 0 & . & . & 0 \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ h[N-1] & h[N-2] & . & . & . & h[1] & 0 \\ h[N] & h[N-1] & h[N-2] & . & . & h[2] & h[1] \end{bmatrix} \tag{15}$$

The entries of matrix P are the elements of the impulse response sequence $\{h[n]\}$ of the linearised, discretised closed loop system, therefore $\Gamma(y_d) \approx Py_d$ holds true close to the equilibrium point [31]. To derive the ILC used in this paper, the singular value decomposition (SVD) is applied to the convolution-matrix P;

$$P = L\Lambda R^T \tag{16}$$

Where R and L are orthonormal matrices, Λ is a diagonal matrix with the elements,

$$\sigma_0 > \sigma_i > 0 \;\forall i \in [1, N-1] \tag{17}$$

The largest singular value σ_0 is the L_2-norm of P. The learning algorithm is determined by setting [31],

$$S = I \tag{18}$$

and

$$E = \frac{1}{\sigma_0} RL^T \tag{19}$$

This learning algorithm is now analysed with the help of the discrete, linearised model. Using in equation (13) the linear model equation $y = Py_d$ yields,

$$y_d[k+1] = Sy_d[k] + E\left(y_d[1] - Py[k]\right) \tag{20}$$

With equations (16), (18) and (19), equation (20) can be written in the form,

$$R^T y_d[k+1] = \left(I - \frac{1}{\sigma_0}\Lambda\right) R^T y_d[k] + \frac{1}{\sigma_0} L^T y_d[1] \tag{21}$$

Let $v[k]=R^T y_d[k]$ and $\mu = L^T y_d[1]$, therefore, equation (21) reads as,

$$v[k+1]=\left(I-\frac{1}{\sigma_0}\Lambda\right)v[k]+\frac{1}{\sigma_0}\mu \tag{22}$$

or rewritten as N separate equations,

$$v[k+1]=\left(I-\frac{\sigma_i}{\sigma_0}\Lambda\right)v_i[k]+\frac{1}{\sigma_0}\mu_i \;\; \forall i\in[0,N-1] \tag{23}$$

The above choice of E and S yields a decoupled learning algorithm. Thus, to determine the convergence of the learning controller the convergence properties of N scalar equations can be studied instead of a matrix equation. Solving the recursive equation (23) yields,

$$v_i[k]=\left(1-\tilde{\sigma}_i\right)^k v_i[0]+\frac{1-(1-\tilde{\sigma}_i)^k}{\sigma_i}\mu_i \;\; \forall i\in[0,N-1] \tag{24}$$

With $\tilde{\sigma}_i=\frac{\sigma_i}{\sigma_0}$. The equation converges to $v_i^{\infty}=\frac{\mu_i}{\sigma_i}$ for $\left|1-\tilde{\sigma}_i\right|<1$.

This is always true due to the SVD property $\sigma_0 > \sigma_i > 0$. The convergence speed is determined by $1-\tilde{\sigma}_i$..

3.3 Modelling the Camless Engine Dynamics

The use of graphical dynamic system simulation software is becoming more popular as engineers strive to reduce the time to develop new control systems [32]. Dynamic system simulation software is an important tool for developing advanced reliable and high quality products. Thus, the skill of real-time complex systems modelling is the inherent goal of many modern mechatronic systems.

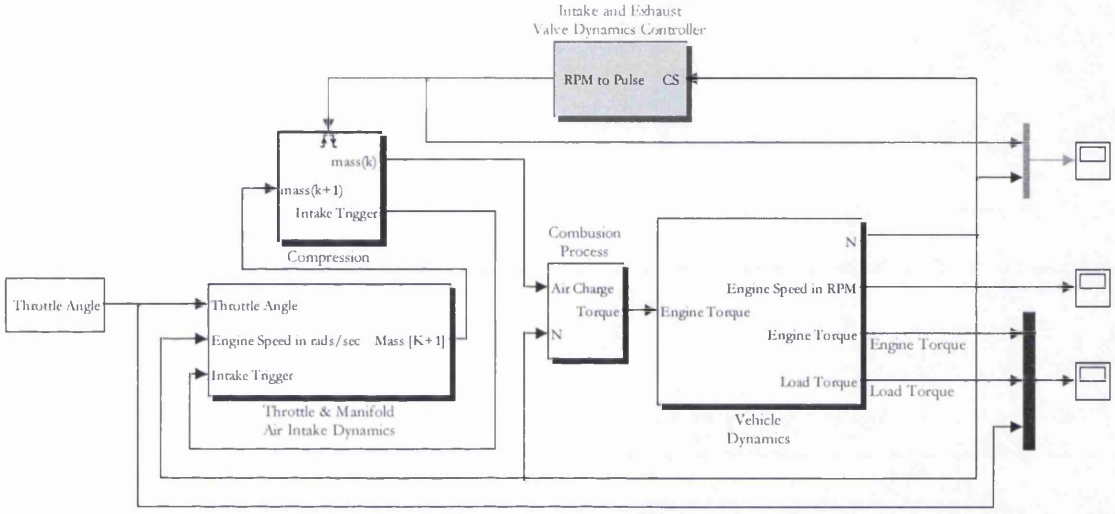

Figure 3.1: Snapshot of the Internal Combustion Engine Modelled

MathWorks' tools were chosen for model development because of the wide exposure of this software platform and support available to both academia and industry [30]. This model was initially used to evaluate the benefits of employing the electromechanical actuator controller algorithm developed. In addition, the model is used to simulate a sequential-fuel-injected, spark ignition engine and includes air and fuel dynamics in the intake manifold as well as the process delays inherent in a four-stroke cycle engine [32]. Primarily, this engine model is used as follows:

1. As an embedded model within a control algorithm or observer
2. As a real-time engine model for hardware-in-the-loop (HIL) testing
3. As a system model for evaluating engine sensor and actuator models
4. As a subsystem in a powertrain or vehicle dynamics model

Modular programming techniques [30, 32] were introduced to reduce the model complexity by dividing the engine and the actuator into hierarchical subsystems as shown in Figure 3.2 and Figure 3.3.

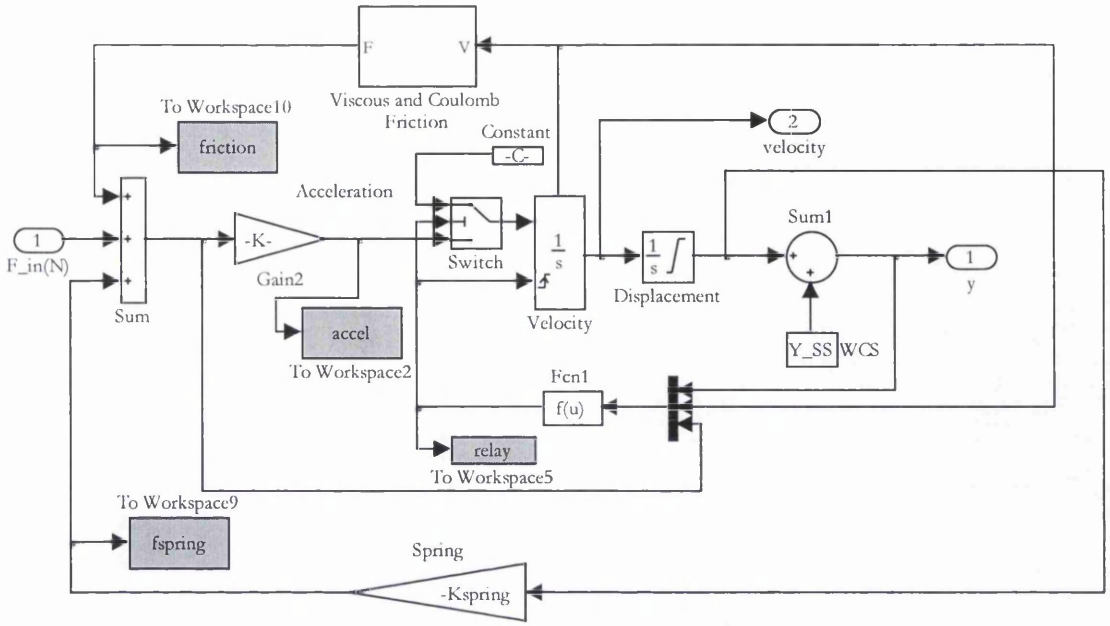

Figure 3.2: Mechanical Subsystem of Actuator

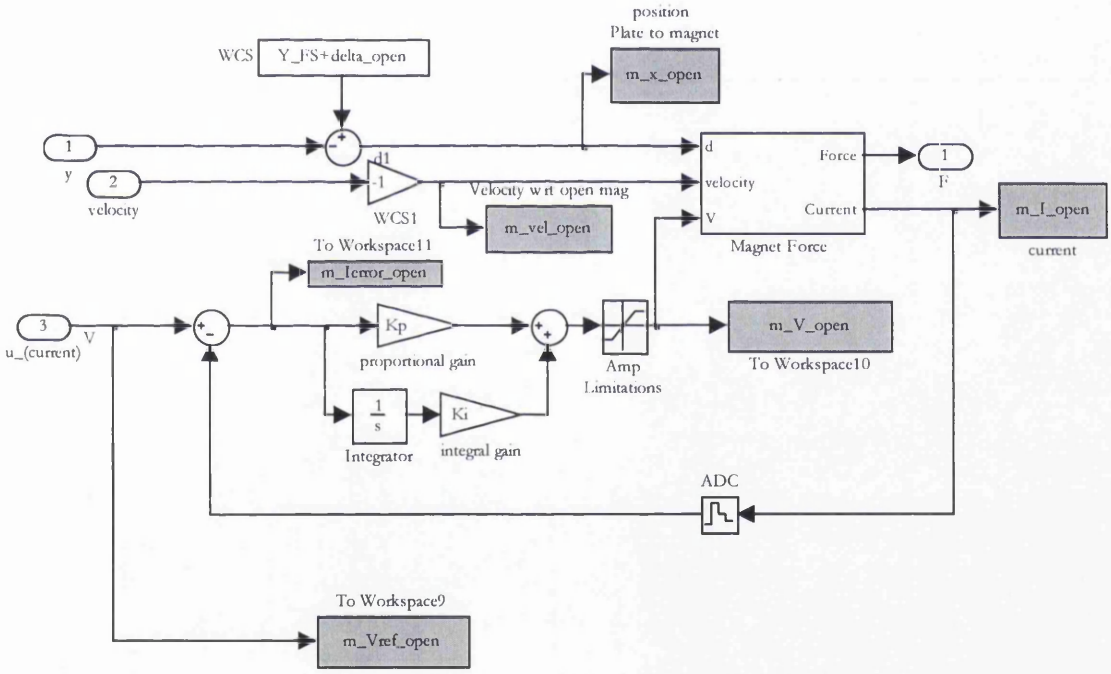

Figure 3.3: Non-Linear Events in Actuator Opening

Figure 3.2 and Figure 3.3 illustrate Simulink models of the mechanical elements and the opening events associated with the electromagnetic engine valve through the use of the first principle equations of the system used in [7, 33] and section 3.1. The system consisting of the armature, engine valve, actuator spring, and valve spring are modelled with a linear time-invariant differential equation, where the coefficients are found through direct measurement and fitted to an initial condition response of the actual electromagnetic system. There is also a coulomb friction present, which is accounted for in the models [33].

These models take into account the saturation of the flux of the magnets in the actuator using a curve-fitting algorithm. Model parameters are experimentally found via a series of static and dynamic tests and the validation of the model is done by comparing the simulation of the valve with actual dynamic tests of the valve opening and closing as explained in [33].

The mechanical model of the actuator is identified as two separate parts. The parameters of the moving parts of the system are measured, when applicable or identified using the properties of a second order prototype. The parameters, which are used to model the force generated by the magnets, are measured or derived from measurements done [33].

4 Real-Time Camless Engine Actuator Controller Platform

Figure 4.1 illustrates the real-time system level development platform implemented for the evaluation of the electromechanical valvetrain. First of all, the maximum supply voltage is limited to 42-Volts to simulate the available voltage on a future vehicle. Two power amplifiers driven by the power supply drive the magnetic coils of the actuator. The SoC designed controls the voltage across the coil through a custom designed I/O board. A laser sensor is used to measure the actual valve position with a 5μm resolution. Coil current is also monitored for diagnostic purposes.

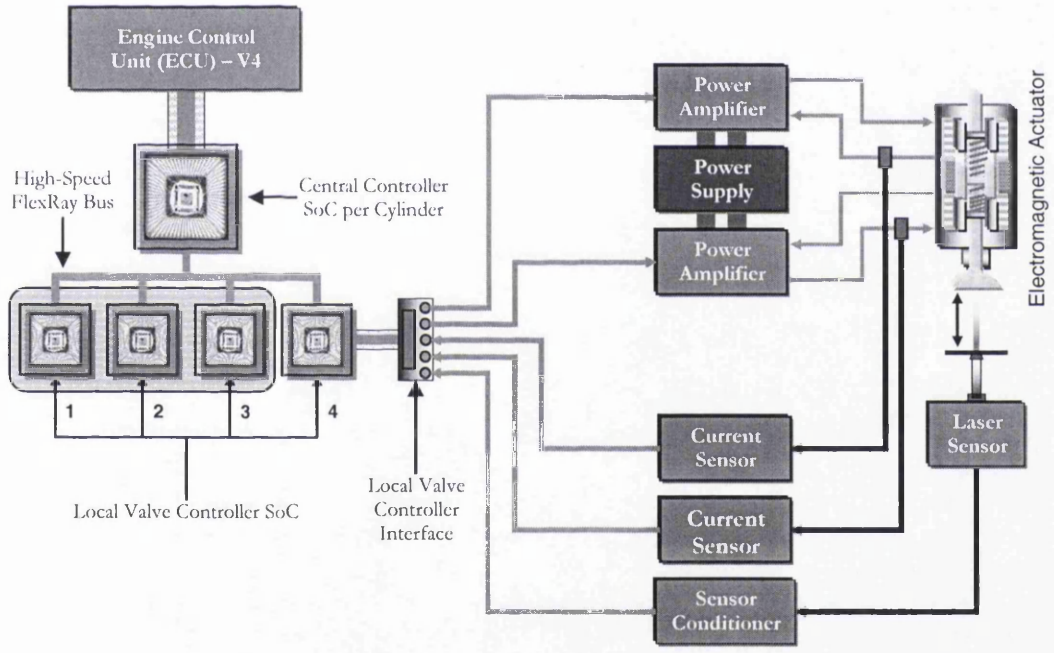

Figure 4.1: Experimental Set-up of the Actuator Controller Platform

This central controller SoC shown above receives its drive cycle information from the existing engine control unit (ECU). The local valve controller SoC is used for the execution of control algorithms and to drive the power stages. Other inputs to this controller include position of the actuators. Data transfer to actuator occurs at 25μs (40kHz) intervals.

5 Findings and Conclusions

An electromechanical camless valvetrain controller platform is realised based on a PowerPC SoC. Simulations and real-time measurements confirm the functional ability of the electromechanical actuator to vary valve timing, lift, velocity and event duration, as well as to perform cylinder deactivation in a four-valve multi-cylinder engine.

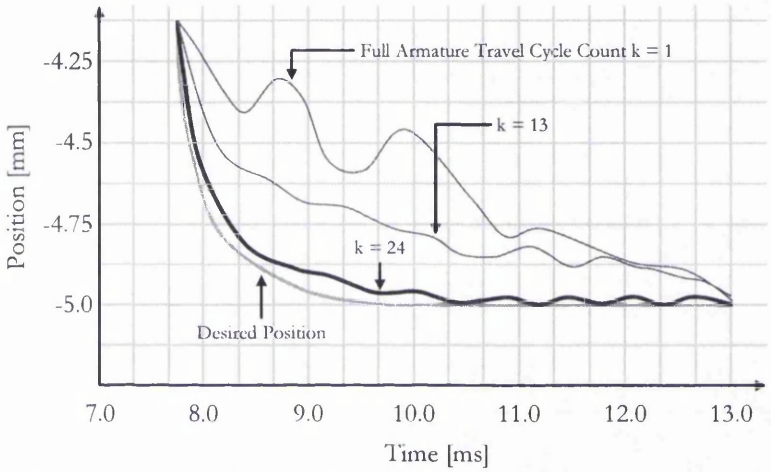

Figure 5.1: Valve Position Control Results based on the ILC (k ϵ [1, 13, 24])

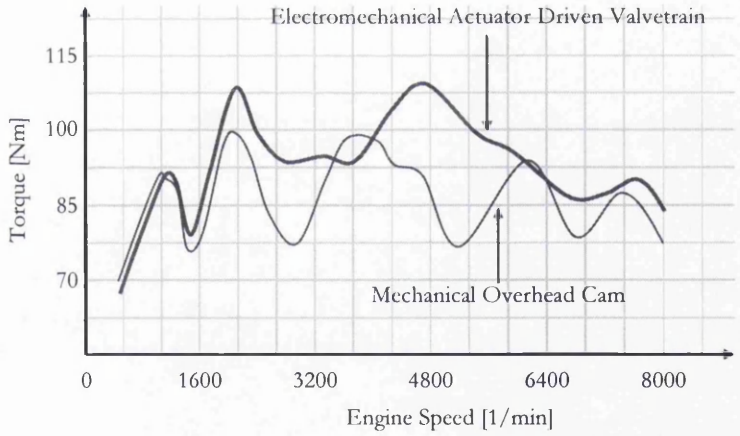

Figure 5.2: Torque Production with Mechanical Cam and Camless Engine

Figure 5.1 and Figure 5.2 show experimentally measured results obtained using the actuator controller with the learning algorithm and a comparison between engine torques obtained with a dual overhead camshaft (DOHC) engine and an electromechanical camless valvetrain (EMCV) arrangement using the setup from [34].

As shown in Figure 5.1, employing an ILC reduces the valve contact velocity as the armature cycles are increased. Figure 5.2 confirms that the torque produced by the four cylinder engine, driven by an electromechanically driven valvetrain is 13% better than that of a classic dual overhead camshaft (DOHC) arrangement.

Throttle-free load control; whether with electromechanical, electro-hydraulic or hydro-mechanical valvetrains, offers modern internal combustion engines a fuel consumption benefit of 10 per cent or greater [1, 5, 6, 8, 9, 10, 11, 12, 13, 14, 15, 16]. Based on the present trends toward the employment of the variable valve timing methodology, it is evident that BMW's Valvetronic system is the next step in the foreseeable future [4].

Modular development tools used for application development in conjunction with the PowerPC SoC provide exact emulation of complex systems such as the camless engine controller platform, leading to substantially reduced development and validation time and lower costs. The design parameters, control variables in system terminology, provide additional degrees of freedom to optimise the performance of the engine over its wide range of operation.

On the valve motion control level, the seating control has been identified to be crucial for the application of electromagnetic camless valve (EMCV) actuators. Initial open-loop analysis reveals that an EMCV actuator becomes unstable as the engine valve gets closed to its seating position. Therefore, closed-loop control is required to achieve EMA *soft-landing.* A linear plant model was constructed. In conjunction with the engine and actuator manufacturers, stem identification tests were conducted to find the model parameters. A repetitive learning controller was designed to enhance the control performance through cycle-to-cycle iterations.

6 References

[1] R. Flierl and M. Kluting, BMW Group, "The Third Generation of Valvetrains for Throttle-Free Load Control", SAE 2000 World Congress, *SAE Technical Paper Series*, Paper No. 2000-01-1227.

[2] J. B. Heywood, "Internal Combustion Engine Fundamentals," New York: McGraw-Hill, (1988).

[3] T. Ahmad and M. A. Theobald, "A Survey of Variable Valve Actuation Technology," *SAE Transactions*, Paper No. 891674.

[4] *BMW (Bayerische Motoren Werke)*, Web home-page: http://www.bmw.com/

[5] General Motors Corporation, Web home-page: http://www.gm.com/

[6] C. Gray, "A Review of Variable Engine Valve Timing," SAE *Transactions*, Paper No. 880386.

[7] Yan Wang, Anna Stefanopoulou, "Modelling of an Electromechanical Valve Actuator for a Camless Engine," proceedings of the *5th International Symposium on Advanced Vehicle Control*, pp. 601–608, (August 22–24, 2000).

[8] J. H. Tuttle, "Controlling Engine Load by Means of Early Intake-Valve Closing" SAE Paper 820408, *SAE Transactions*, Vol. 91, 1982.

[9] DaimlerChrysler, Web home-page: http://www.daimlerchrysler.com/

[10] Honda Automobiles, Web home-page: http://automobiles.honda.com/

[11] Toyota, Web home-page: http://www.toyota.com/

[12] *AUDI (Auto Union Deutschland Ingolstadt)*, Web home-page: http://www.audi.com/

[13] Volvo Car Corporation, Web home-page: http://www.volvocars.com/

[14] Ford Motor Company, Web home-page: http://www.ford.com/

[15] Renault, Web home-page: http://www.renault.com/

[16] Peugeot, Web home-page: http://www.peugeot.com/

[17] S. Butzmann, J. Melbert, and A. Koch, "Sensorless Control of Electromagnetic Actuators for Variable Valve Train," *SAE Transactions*, Paper No. 2000-01-1225.

[18] Dr. Martin Pischinger, "A New Opening," *Engine Technology International*, pp. 20–22, (July 2001).

[19] S. R. Oh, Z. Bien, and I. H. Suh, "An iterative learning control method with application to robot manipulators," IEEE *Journal of Robotics and Automation*, Vol. 4, Issue. 5, pp. 508 – 514, (October 1988).

[20] Tae-Yong Kuc, Kwanghee Nam and J. S. Lee, "An iterative learning control of robot manipulators," IEEE *Transactions on Robotics and Automation*, Vol. 7, Issue. 6, pp. 835 – 842, (December 1991).

[21] Han-Fu Chen, Hai-Tao Fang, "Output tracking for nonlinear stochastic systems by iterative learning control," IEEE *Transactions on Automatic Control*, Vol. 49, Issue. 4, pp. 583 – 588, (April 2004).

[22] M. Pandit and K. H. Buchheit, "Optimizing iterative learning control of cyclic production processes with application to extruders," IEEE *Transactions on Control Systems Technology*, Vol. 7, Issue. 3, pp. 382 – 390, (May 1999).

[23] Kim Dong and Kim Sungkwun, "An iterative learning control method with application for CNC machine tools," IEEE *Transactions on Industry Applications*, Vol. 32, Issue. 1, pp. 66 – 72, (January – February 1996).

[24] S. S. Garimella and K. C. Srinivasan, "Application of iterative learning control to coil-to-coil control in rolling," IEEE *Transactions on Control Systems Technology*, Vol. 6, Issue. 2, pp. 281 – 293, (March 1998).

[25] Y. Chen, C. Wen, Z. Gong and M. Sun, "An iterative learning controller with initial state learning," IEEE *Transactions on Automatic Control*, Vol. 44, Issue. 2, pp. 371 – 376, (February 1999).

[26] S. K. Sahoo, S. K. Panda and J. X. Xu, "Iterative learning-based high-performance current controller for switched reluctance motors," IEEE *Transaction on Energy Conversion*, Vol. 19, Issue. 3, pp. 491 – 498, (September 2004).

[27] M. Norrlof and S. Gunnarsson, "Experimental comparison of some classical iterative learning control algorithms," IEEE *Transactions on Robotics and Automation*, Vol. 18, Issue. 4, pp. 636 – 641, (August 2002).

[28] P. Bondi, G. Casalino and L. Gambardella, "On the iterative learning control theory for robotic manipulators," IEEE *Transactions on Robotics and Automation*, Vol. 4, Issue. 1, pp. 14 – 22, (February 1988).

[29] M. Norrlof, "An adaptive iterative learning control algorithm with experiments on an industrial robot," IEEE *Transactions on Robotics and Automation*, Vol. 18, Issue. 2, pp. 245 – 251, (April 2002).

[30] The Mathworks, leading global provider of technical computing and model-based design software, Web home-page: http://www.mathworks.com/

[31] Xu, J.-X. and Z. Bien (1998). The frontiers of iterative learning control. In: Iterative Learning Control (Z. Bien and J.-X. Xu, Eds.). pp. 9-35. Kluwer Academic Press, Boston.

[32] R. W. Weeks and J. J. Moskwa, "Automotive Engine Modelling for Real-time Control Using Matlab/Simulink", In: The SAE International Congress and Exposition, *SAE Transactions*, Paper No. 950417, 1995.

[33] Mohamed Anas, D. R. S. Cumming and W. H. Nailon, "System Level Design of a Controller Platform for a Camless, Electromagnetic Actuator Driven Next Generation Car Engine," *IEEE Transaction on Vehicular Technology*, submitted for review.

[34] Compact Dynamics GmbH, Web home-page: http://www.compact-dynamics.de

Fast Internal Combustion Engine Knock Processing Using an Automotive PowerPC System-on-Chip

Mohamed Anas

A portfolio submitted to

The Universities of

Edinburgh

Glasgow

Heriot Watt

Strathclyde

for the Degree of

Doctor of Engineering in System Level Integration

March 2005

Abstract

This portfolio discusses a novel high performance internal combustion engine *knock*[1] processing and detection methodology using a next generation automotive single-instruction-multiple-date System-on-Chip. The approach taken is based on autonomous on-chip modules and an auxiliary signal processing extension to the main System-on--Chip core. Real-time software development techniques with an advanced software circular buffer implementation for processing the streaming knock sensor data have been developed.

Various single instruction multiple data software optimisation techniques are employed to reduce the real-time knock algorithmic execution time. Real-time and simulation results are presented for the detection of knock on a four cylinder internal combustion engine, but, the approach is widely applicable.

The efficient single-instruction-multiple-data and classic high level language C based coding and optimisation techniques used for the algorithmic implementation have been shown to improve computational performance and as a result utilises minimum combustion event timing.

---

[1] *Knock* is the name given to the noise which is transmitted through the engine structure when essentially spontaneous ignition of a portion of the *end-gas – the fuel, air and residual gas mixture*, ahead of the propagating flame occurs.

Table of Contents

List of Figures and Tables

1 Introduction

This portfolio describes the development; implementation and testing of an advanced internal combustion engine knock processing methodology based on an automotive qualified single-instruction-multiple-data (SIMD) system-on-chip (SoC).

This section of the portfolio introduces the knock phenomenon, its existing control methodologies and examines and identifies some of its shortcomings. Through reference to previous and existing real-time knock processing strategies, the need for fast and particularly flexible method to process knock is made apparent. Finally the overall structure of the proposed SoC based knock processing is and original contributions of the portfolio is outlined.

1.1 Background

It is difficult to overstate the importance of internal combustion engine knock as it is a direct constraint on engine performance [1]. Knock in internal combustion engines (ICE) refers to the premature self or auto ignition of the air-fuel mixture in the engine when the unburnt mixture's temperature and pressure have exceeded a critical point.

Knock constraints engine efficiency, since by effectively limiting the temperature and pressure of the end-gas, it limits the engine compression ratio. The occurrence and severity of knock depend on the knock resistance of the fuel and on the antiknock characteristics of the engine. Frequent occurrence of this knock phenomenon causes permanent damage to the ICE and should be avoided. However, in order to obtain maximum power, modern engines are run at their borderline limit of incipient knock using closed-loop control of spark timing based on knock sensor feedback [1], [2], [3].

1.2 Internal Combustion Engine Knock

In a four stroke engine, during normal drive cycles, the air-fuel mixture in the cylinder is compressed by the engine's piston to a high pressure. A timely spark generated by the mounted sparkplug then ignites the fuel near sparking point, which then creates a flame front. As shown in Figure 1.1, this flame front propagates throughout the air-fuel mixture and this mixture is burnt in a highly controlled manner, which gradually increases the cylinder pressure to push the piston downwards. As the piston moves away, the pressure eases. Engine knock occurs when the air-fuel mixture ignites before the flame front can reach it [1]. This uncontrolled ignition of the air-fuel mixture causes it to burn in an irregular and explosive manner. As shown in Figure 1.1, this rapidly expanding mixture exerts a sudden pressure wave, which produces a sizeable force on the surroundings of the combustion chamber [1], [2], [3].

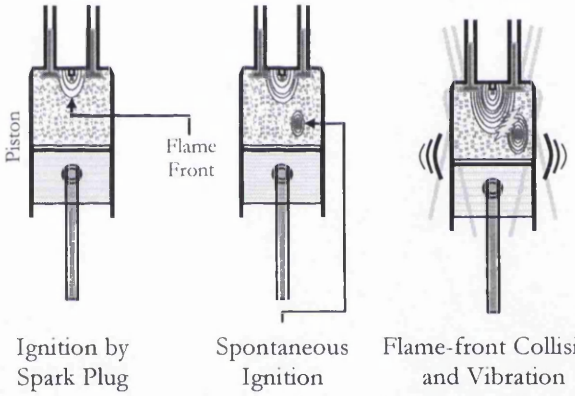

Figure 1.1: Engine Knocking and its Causes

In severe cases, the piston may still be moving upwards to compress the air-fuel mixture. As such, it cannot move away to ease the build up of pressure wave. This results in severe stress on the engine and should be prevented as permanent engine damage can occur. Figure 1.2 overleaf shows three plots of cylinder pressure against crank angle of a single cylinder engine with ignition timing three degrees apart between each trace [1]. ICE knock usually occurs under wide-open-throttle (WOT) operating conditions.

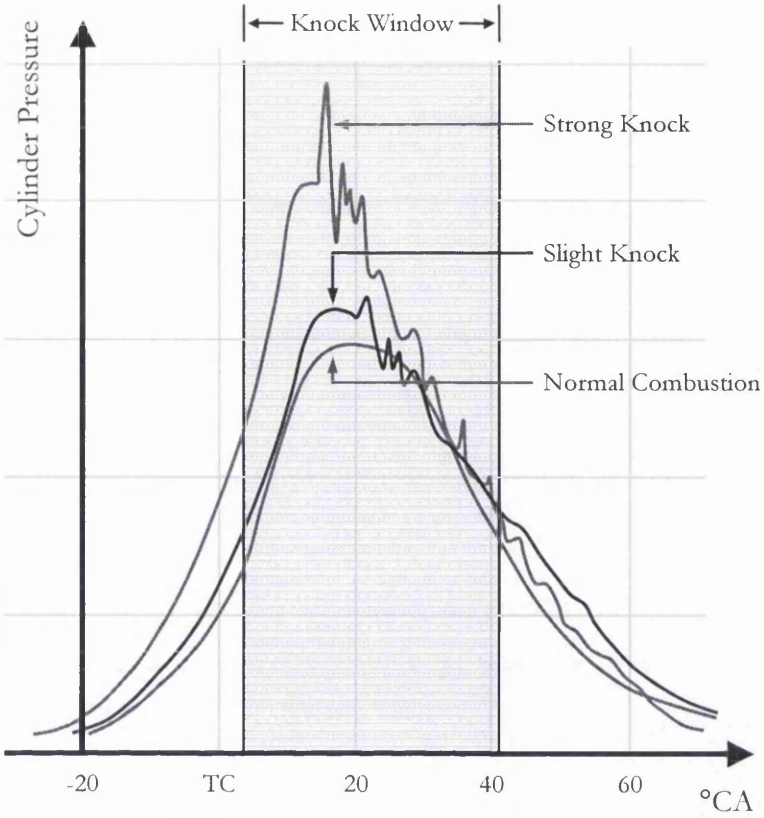

Figure 1.2: Cylinder Pressure versus Crank Angle (CA)

As shown in Figure 1.2, if the normal combustion cycle is advanced by some three degrees, a slight knock occurs which indicates the engine has reached its limit. Further timing increase produces severe knock. A common reason for engine knock is using poor quality gasoline with too low an octane rating, which has the tendency to ignite prematurely [1].

Normally, all internal combustion engines are designed to run with a minimum octane rating gasoline. Another source of combustion knock is insufficient cooling of the engine. When the temperature in the engine gets too high, it can trigger fuel to suddenly self-ignite. Even with good cooling of the engine, a poorly designed engine may have "hot spots" that do not get cooled properly. This could be due to recirculation zones, crevices or failure to properly exhaust burnt gases. Yet another possibility is the use of a turbocharger on an engine that is not designed and recommended by the engine manufacturer. Since turbochargers compress air at the engine inlet, pressures in the engine get much higher than the engine was originally designed for. Like high temperatures, an overly high pressure (above 14:1 for gasoline) also triggers the fuel to prematurely self-ignite.

1.3 History of Knock Processing Methodologies

There are several different approaches to detect the presence of knock in engines, see e.g. [5], [9] – [15]. One of the classic techniques presently used in production engines is based on application specific integrated circuits (ASIC) with limited programmability, such as the ProSAK™ knock control ASIC [16] and The HIP9011 ASIC [17].

Due to the high cost of direct knock sensors, most of the current knock detection systems are based on structural vibration signals obtained using an accelerometer [13].

This signal is then processed by aforementioned ASICs that include the functionalities, analogue filtering, rectification and integration. The final integrated value obtained is then compared to a heuristically determined threshold to determine the presence of engine knock [5]. Several laboratory based methods[5], [9], [11], [13] – [15] have also been proposed for the extraction of the energy in the resonance frequencies generated by combustion knock, however such methods are not useful for engine manufacturers as they are not cost effective and particularly computationally demanding in production engines. Additionally, there is very little information available on the real-time implementation of such detection methods.

1.4 Impact of Knocking

Impact of knocking in an engine depends on its intensity and duration. Trace knock has no significant effect on engine performance or durability. Heavy knock can lead to extensive engine damage. The engine can be damaged by knock in different ways: piston ring sticking; breakage of the piston rings and lands; failure of the cylinder head gasket; cylinder head erosion; piston crown and top land erosion; piston melting and holing. Knocking is one important factor limiting the efficiency of an engine and is therefore of great importance to the engine manufacturers [1], [2], [3], [4].

Figure 1.3: Piston Damage due to Long-term Engine Knock

Generally knocking exerts a great deal of downward force on the pistons as they are being forced upward by the mechanical action of the connecting rods. When this occurs, the resulting concussion, shock waves and heat can be severe resulting with broken or melted spark plug tips and as shown in Figure 1.3, other internal engine components, particularly the pistons can be damaged. Left unresolved, engine damage is almost certain to occur, with the spark plug usually suffering the first signs of damage.

2 Developed Knock Processing Methodology

The fundamental role of the SoC based knock processing platform is to extract a feature characteristic of knock that allows discrimination between normal engine noise and spontaneously ignited knocking noise. Knock detection is complicated by the variation in the amplitude and frequency of the knock sensor's output as functions of time or crankshaft angle, level of non-knock related engine noise, knock sensor mounting and importantly its intensity.

2.1 An Overview of the Knock Processing SoC

The SoC developed is the first member of a family of next generation powertrain SoCs based on the PowerPC Book E architecture, containing many new features coupled with high performance complementary metal oxide semiconductor (CMOS) technology to provide substantial reduction of cost per feature and significant performance improvement over the legacy devices, particularly the presently used MPC565 [79]. This SoC is targeted toward middle to high-end powertrain applications, such as camless engine controller platforms and digital knock processing.

The host processor core of the SoC complies with the PowerPC Book E architecture. It is 100% user mode compatible with the classic PowerPC instruction set including the floating point library. The Book E architecture has enhancements that improve the PowerPC architecture's fit in embedded applications. This core also has additional instructions, including DSP instructions, supported by the signal processing extension (SPE), beyond the classic PowerPC instruction set.

The SoC has two levels of memory hierarchy. The fastest accesses are to the 32kB unified, aka von Neumann cache. The next level in the hierarchy contains the 64kB on-chip static random access memory (SRAM) and the 2 MB internal flash memory. Both the SRAM and the flash memory can hold instructions and data. The External Bus Interface has been designed to support most of the standard embedded memories widely available.

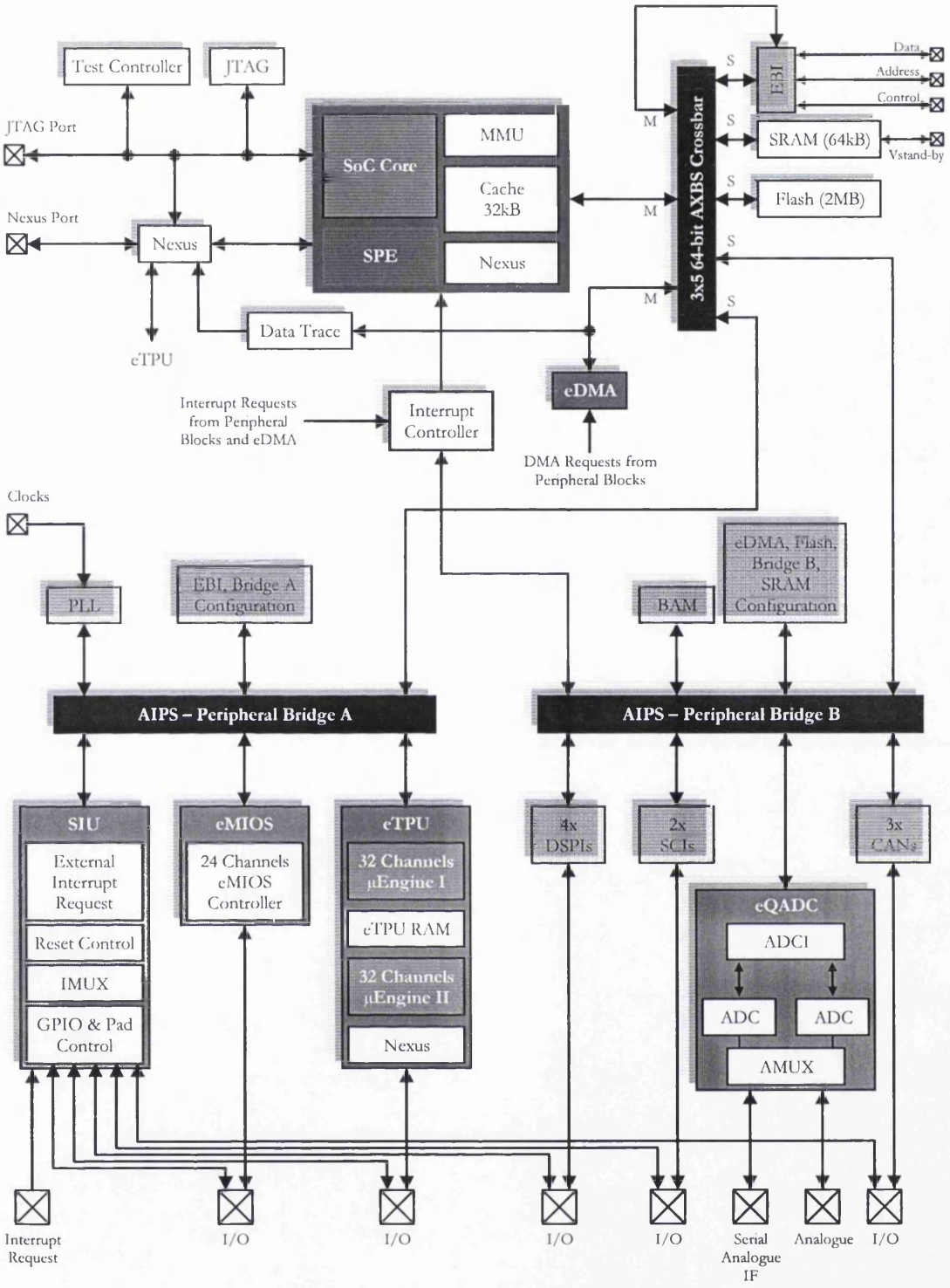

Figure 2.1: Overview of the Knock Processing SoC

The complex I/O timer functions of SoC are performed by two Enhanced Time Processor Unit engines (eTPU). Each eTPU micro-engine controls 32 hardware channels. The eTPU consists of 24-bit timers, double action hardware channels, variable number of parameters per channel, angle clock hardware, and additional control and arithmetic instructions. The eTPU can be programmed using a high-level programming language.

The less complex timer functions required are performed by the Modular Timer System (eMIOS/MTS). The eMIOS' 24 hardware channels are capable of single action, double action, Pulse Width Modulation (PWM) and modulus counter operation. Off-chip communication is performed by a suite of serial protocols including controller area networks (CAN), enhanced serial peripheral interface (SPI) and serial communication interfaces (SCI).

Additionally the SoC has an on-chip 40-channel Enhanced Queued dual Analogue-to-Digital Converter (eQADC). The System Integration Unit (SIU) performs several chip-wide configuration functions. Pad configuration and General-Purpose Input and Output (GPIO) are controlled from the SIU. External interrupts and reset control are also found in the SIU. The Internal Multiplexer sub-block (IMUX) provides multiplexing of eQADC trigger sources, daisy chaining the DSPIs and external interrupt signal multiplexing.

2.2 Knock Sensors

Knock sensors can be partitioned into the categories of direct and remote measurement devices. Direct sensors measure the pressure inside the combustion chamber necessitating each cylinder to encompass a dedicated sensor and such sensors operating in harsh conditions tend to be expensive, whereas the remote tuned or broadband measurement sensors use the vibrations transmitted through the structure of the ICE [13]. Typically, the resonant point of the tuned sensor is centred on the fundamental knocking frequency, which is between 5 kHz and 7 kHz, whereas the bandwidth of broadband sensors tend to be between 1 kHz and 25 kHz and this tends to be the most cost effective since a single sensor could cover a range of resonance frequencies [13].

The basis for the implemented knock processing methodology is based on broadband sensors. However, with minor modifications, the proposed methodology can be used with data obtained using pressure and acoustic information.

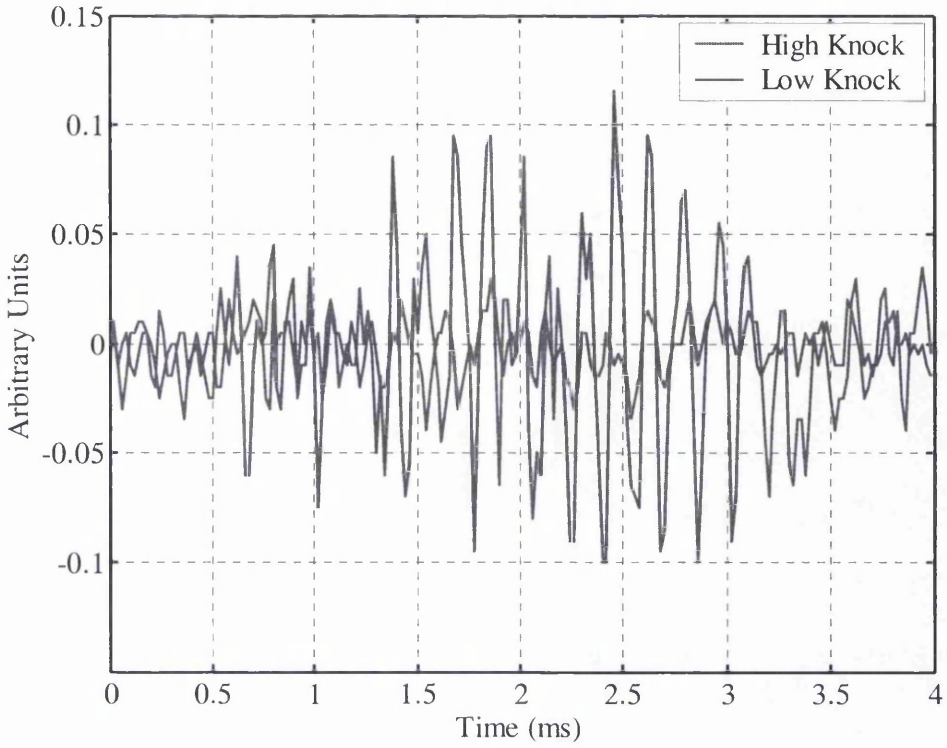

Figure 2.2: Knock Sensor Signal from Engine Block

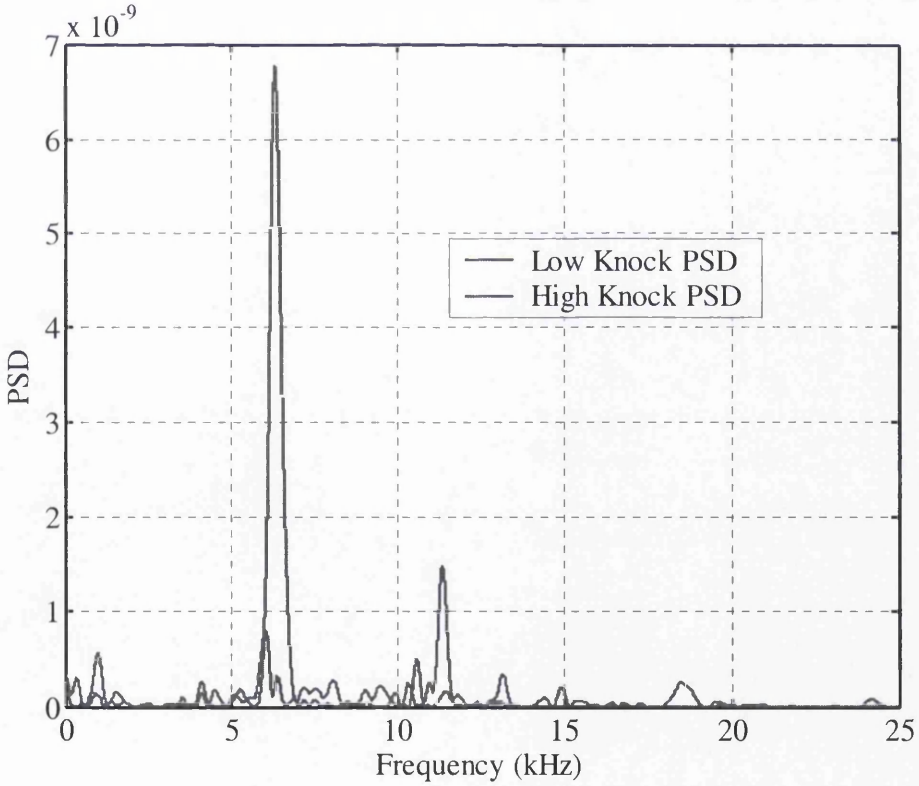

Figure 2.3: Power Spectral Densities (PSD) of Figure 1.1's Knock Signals

Optimal location of accelerometers is heuristically chosen so that the sensor is not in a "dead" area and the transmitted quality of the signal has the highest SNR. The interface circuitry shown in Fig. 5 provides impedance matching for the knock sensors and a highpass filter attenuating frequencies below 1 kHz. An on-chip digital multiplexer, selects the appropriate knock input of the active cylinder. In our evaluation, the sensor was placed in the threaded screw hole on the frame rib intersection of the upper part of the engine as prescribed by the manufacturer.

The developed knock kernel was validated using the structural vibration signal obtained using a piezoelectric accelerometer generated by a four cylinder engine (V4). This signal was acquired at a sampling rate of 50 kHz using a broadband knock sensor with a 25 kHz flat frequency response. In order to avoid engine damage, only 20% of knocking cycles were introduced at 4000 RPM in the engine in a strictly controlled manner. Figure 2.2 and Figure 2.3 show an example set of the knock sensor signals, highlighting the fundamental knock frequency of the engine used and their power spectral densities respectively.

2.3 Resonance Frequencies in a Knocking Engine

The resonant frequencies excited by the presence of knock depend on the geometry of the combustion chamber and the speed of sound in the cylinder charge [1], [2], [3], [4]. These resonant frequencies are typically estimated by assuming an acoustic model for the combustion chamber. For a homogeneous gas filled, acoustically hard walled ideal cylinder, the resonance frequencies are given by the following Draper's equation [4].

$$f_{m,n} = \frac{c_o \sqrt{T} \eta_{m,n}}{\pi B} \tag{1}$$

Where $f_{m,n}$ is resonant frequency, $\eta_{m,n}$ is a non-dimensional mode number, c_o is the phase velocity constant, T is the combustion mixture temperature, B is the cylinder bore diameter and the integer subscriptions m and n denote radial and circumferential mode numbers. Generally, the axial mode is neglected as knock generally occurs when the piston is just past the top dead centre (TDC) position and at this instance; axial dimension is negligible compared to the radial dimension.

2.4 Knock Sensor Data and Algorithmic Flow

Figure 2.4 illustrates the knock sensor data and the algorithmic flow. User programmed eQADC commands are contained in the on-chip memory in a user defined data structure. The eQADC command data is moved from the command queue to the command FIFO buffer by either the host CPU or by the enhanced Direct Memory Access (eDMA) controller. Once the command FIFO is triggered and is transferred into the ADCs on chip, the ADC executes the command, and the result, i.e. a pair of time stamp t[n] and data x[n] is moved through the result FIFO by the eDMA or the host CPU in to the on chip circular buffer.

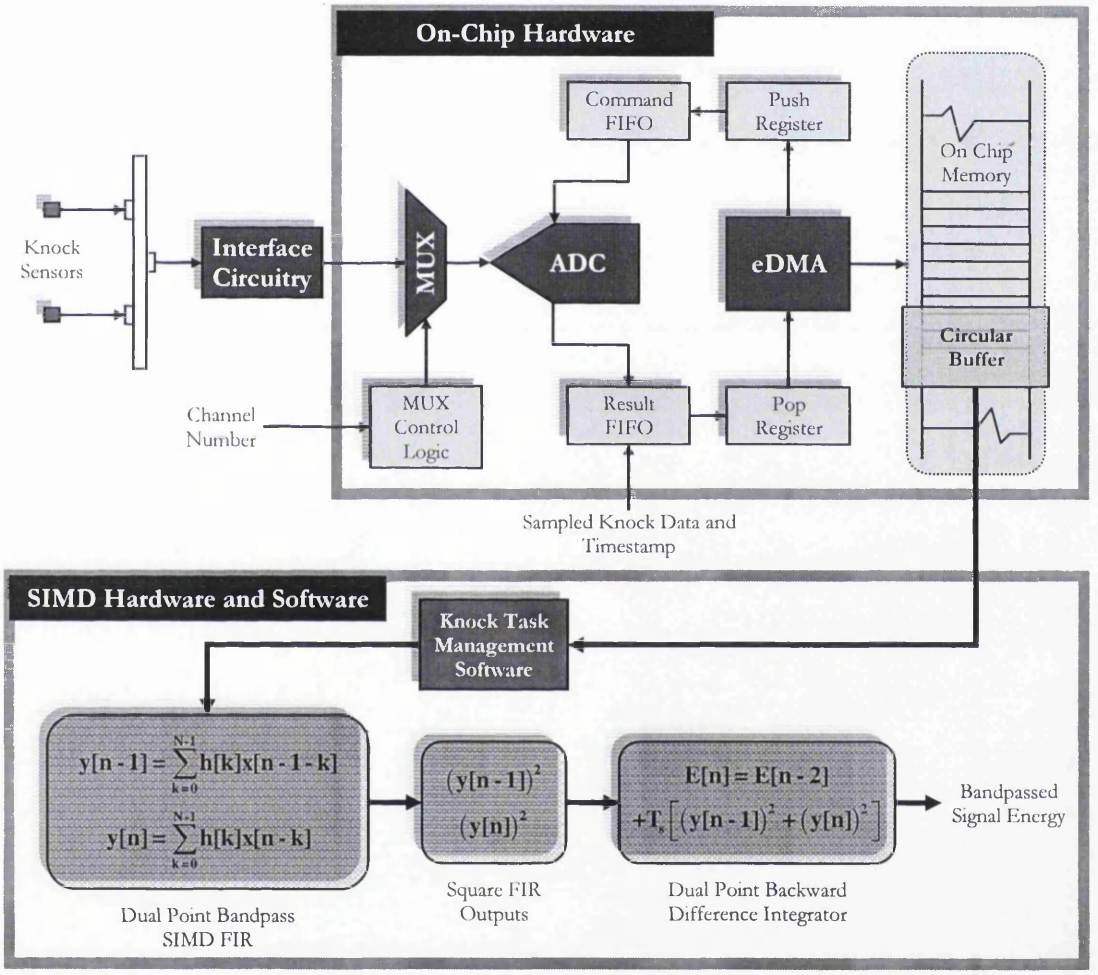

Figure 2.4: Data and Algorithmic Flow of the Developed SIMD Knock Kernel

The data in the circular buffer is then processed and presented by the knock task management software threads to the key SIMD knock signal energy extraction elements. The streaming data is then bandpass SIMD FIR filtered to extract the signal of interest. This bandpassed signal is then squared, integrated and compared to determine the presence of knock.

2.5 Minimum Length Bandpass Filter Design and Analysis

Computationally efficient and minimum length banks of FIR filters were designed to extract the frequencies of interest from the specified range of frequencies. The input, x(n) and the output y(n) signals to the FIR filter are related by the convolution sum [6].

$$y[n] = \sum_{k=0}^{N-1} h[k]x[n-k] \tag{2}$$

Where $h[k]$, k = 0, 1, 2... $N-1$, are the impulse response coefficients of the filter and N is the filter length, that is the number of filter coefficients.

To achieve highest computational efficiency and to reduce design and evaluation time, accurate estimation of the minimal FIR bandpass filter length required for the algorithm was estimated using the following empirical relationship [7], [8]. The parameters used to specify the bandpass filter are, δ_p, δ_s – passband and stopband ripples, ΔF – transition bandwidth normalised to sampling frequency, b_1 = 0.01201, b_2 = 0.09664, b_3 = -0.51325, b_4 = 0.00203, b_5 = -0.57054 and b_6 = -0.44314.

$$N = \frac{C_\infty(\delta_p, \delta_s)}{\Delta F} + g(\delta_p, \delta_s)\Delta F + 1 \tag{3}$$

Where

$$\begin{aligned} C_\infty(\delta_p, \delta_s) = \log_{10}\delta_s \left[b_1 \left(\log_{10}\delta_p\right)^2 + b_2 \log_{10}\delta_p + b_3 \right] \\ + \left[b_4 \left(\log_{10}\delta_p\right)^2 + b_5 \log_{10}\delta_p + b_6 \right] \end{aligned} \tag{4}$$

$$g(\delta_p, \delta_s) = -14.6 \log_{10}\frac{\delta_p}{\delta_s} - 16.9 \tag{5}$$

FIR coefficients were calculated using windowing, equi-ripple and least square error minimization techniques. The least-squares error minimization scheme produced the desired frequency response and the coefficients obtained were used for the real-time implementation of the bandpass filter.

2.6 Dual-Point Backward Difference Integrator

The square of the bandpassed FIR output is subjected to a backward difference integrator. For a single point integrator, the output $y[k]$ is given by,

$$y[k] = y[k-1] + T_s x[k] \tag{6}$$

Where $y[k\text{-}1]$ is the previous integrator output, T_s is the sampling interval and $x[k]$ is the square of the filter output. Computation of present integrator output requires a single or scalar data point and the previous integrator output. However, the SIMD bandpass FIR block developed computes two back-to-back filtered output points simultaneously, which are then squared in parallel. In order to exploit the concurrent availability of two such data points and to reduce the computational burden involved with conversion of vector data points into scalar format for the single point integrator, a two point backward difference integrator was developed and this is given by,

$$y[k] = y[k-2] + T_s\left(x[k] + x[k-1]\right) \tag{7}$$

This integrator saves ~15% of the computational bandwidth compared to that of a single point integrator.

3 Pre-Silicon Knock Simulation Platform

Functional and behavioural modelling of the overall knock processing strategy was carried out using the architectural modelling environment (AME), Matlab, Simulink and Stateflow simulation platforms, which was initially used as the primary demonstrator to customers before the development of the SoC based real-time environment. Figure 3.1 illustrates the overall pre-silicon Simulink® and Stateflow® based knock processing simulation platform developed by the author for both functional and behavioural verification. This simulation platform encapsulates both algorithmic and supervisory logic models of various data path elements as show in Figure 2.4 and the real-time knock processing threads as illustrated in Figure 4.4. The knock simulation platform is divided into hierarchical subsystems, making it more generic by separating engine and knock control system specific parameters.

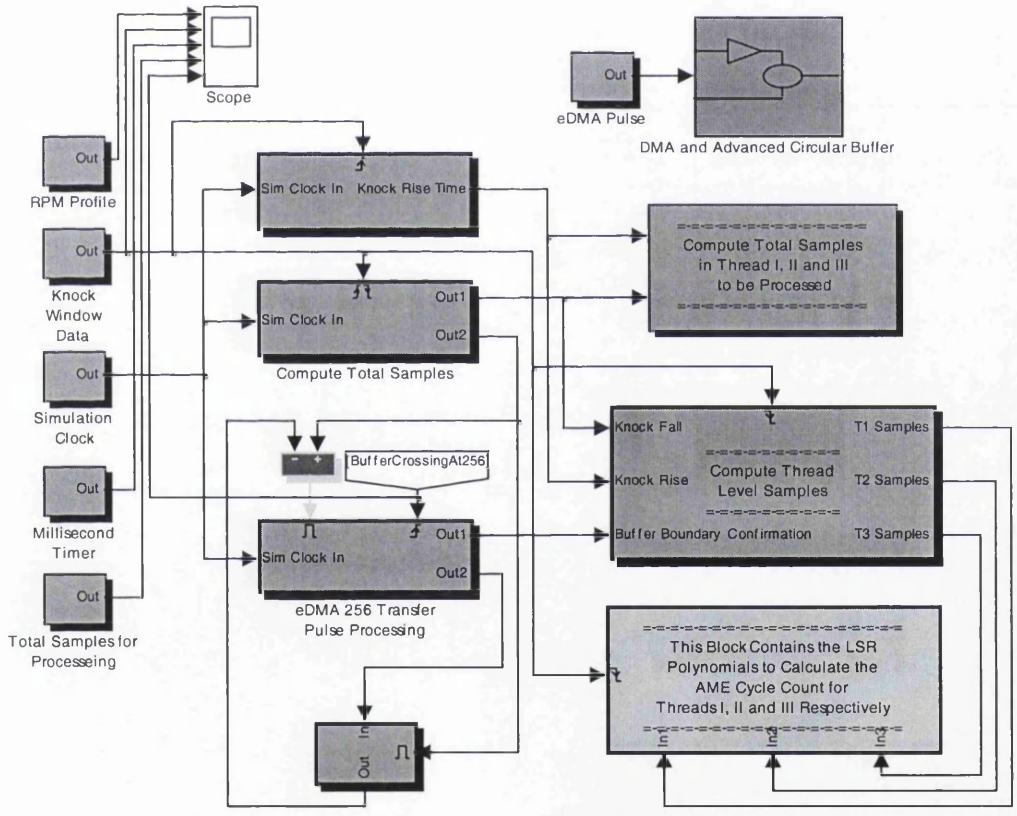

Figure 3.1: Pre-Silicon Knock Functional and Behavioural Simulation Platform

In Figure 3.1, AME based linear and non-linear regressed performance analysis tables of the developed SIMD knock signal processing kernels are also incorporated in order to simulate different knock signal acquisition windows as shown in Figure 1.2. As a result, performance optimisation of the entire system, eliminating laborious programming and delivering substantial time was achieved.

4 Knock Hardware and Software

4.1 Hardware Knock Evaluation Platform

A high level overview and a block diagram of the embedded SoC knock evaluation platform is shows in Fig. 2. It employs an MC33394 Power Supply IC [16] and connectivity to other basic optional communication protocols available on the SoC.

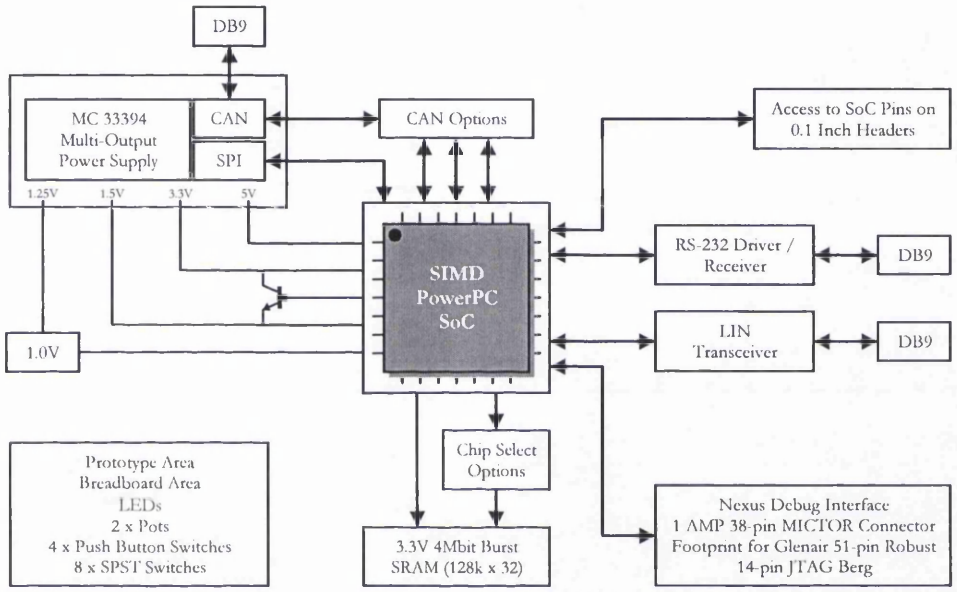

Figure 4.1: Overview of Hardware Knock Evaluation Platform

The SoC platform has two levels of memory hierarchy. The fastest accesses are to the 32 kB unified cache. The next level in the hierarchy contains the 64 kB on-chip SRAM and 2 MB of internal flash memory. Both the on-chip SRAM and the flash memory can hold instructions and data.

The complex I/O timer functions of the SoC are performed by two enhanced Time Processor Units (eTPU). Off-chip communication is performed by a suite of serial protocols including Controller Area Networks (CAN), enhanced Serial Peripheral Interfaces (SPI) and Serial Communications Interfaces (SCI). The SoC has an on-chip 40-channel enhanced Queued dual Analogue-to-Digital Converter (eQADC).

The SoC core embedded on the evaluation board complies with the classic PowerPC Book E architecture. It is 100% user mode compatible (with floating point library) with the classic PowerPC instruction set [16]. The SoC core consists of a register file with 32 64-bit registers. In addition to the vector instructions used for the development of the algorithm, the PowerPC 32-bit instructions used operate on the lower (least significant) 32 bits of the 64-bit register. The vector instructions defined view the 64-bit registers as vectors of two 32-bit elements, and some of the instructions also read or write 16-bit elements. These instructions are used to perform scalar operations in the algorithm developed.

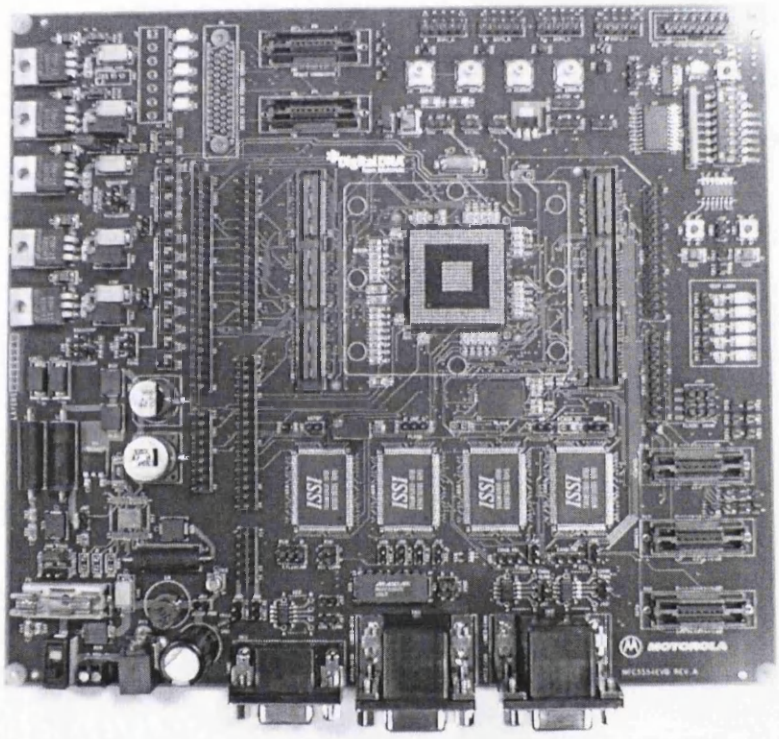

Figure 4.2: Fully Populated Real-time Knock Evaluation PCB

Figure 4.2 shows the fully populated printed circuit board (PCB) used for the real-time evaluation of the overall knock processing strategy developed. Where possible, components mounted on the PCB are automotive qualified to allow system evaluation over the full automotive temperature range (-40 °C to 125 °C).

4.2 Software Platform

4.2.1 SIMD Software

Key software elements of the knock processing strategy was developed based around the SIMD functionality supported by the SoC as SIMD works best with arrays of streaming data. This SIMD code was written using the developed high-level programming model which can be used in conjunction with the classic C and C++ languages. This programming model eliminates the issues associated with writing code at the assembly level: register allocation, scheduling, stack management and conformance to the underlying application binary interface (ABI). These SIMD instructions are supported by the SPE tightly coupled to the core of the SoC as shown in Figure 2.1.

The SIMD programming model introduced consists of a set of fundamental data types supporting parallel loading and storing of appropriate data into the 64-bit (__ev64__) vector registers as shown in Table 4.1.

SIMD 64-bit Data Type	Interpretation of Contents	Values
__ev64_u16__	4 unsigned 16-bit integers	0 to 65535
__ev64_s16__	4 signed 16-bit integers	-32768 to 32767
__ev64_u32__	2 unsigned 32-bit integers	0 to $2^{32} - 1$
__ev64_s32__	2 signed 32-bit integers	-2^{31} to $2^{31} - 1$
__ev64_u64__	1 unsigned 64-bit integer	0 to $2^{64} - 1$
__ev64_s64__	1 signed 64-bit integer	-2^{63} to $2^{63} - 1$
__ev64_fs__	2 floats	IEEE-754 single-precision values
__ev64_opaque__	any of the above	—

Table 4.1: SIMD Fundamental Data Types

4.2.2 Executing the Developed Knock Kernel

Figure 4.3 illustrates a 180° part of a 720° ignition cycle. As shown in this figure, the knock kernel is run once per cylinder combustion event, per engine cycle. The actions illustrated in Figure 4.3 are then repeated for each of the other three cylinders of the V4 for a total engine cycle of 720°. Normally, the start of the knock window and the ignition pre-schedule time are both hard coded. Whereas the end of the knock window, the ignition scheduling, and the start and end of the ignition pulse are calculated during the operation of the engine. For example, the end of the knock window is determined by what the speed of the engine is at the start of the window.

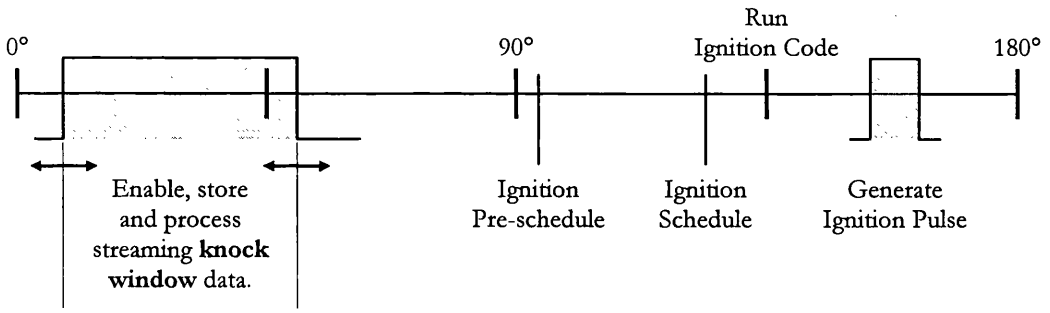

Figure 4.3: Key Events in an Ignition Cycle of a Cylinder

4.2.3 Advanced Streaming Software Circular Buffer

The software circular buffer scheme implemented to hold the streaming knock data involves the allocation of memory space on the on-chip SRAM of the SoC with sufficient space left over for most of the fast executable portions of the operating system (OS) and other active applications to reside. This buffer mechanism solves many of the problems associated with streaming high-throughput data acquisition on a multi-threaded / multi-tasking operating system. An eDMA channel is used to create the required circular buffer in the on-chip SRAM.

The 32-bit data word per knock window sample stored in the advanced circular buffer (ACB) consists of a 16-bit timestamp and its 16-bit data sample. The eDMA channel runs continuously without software intervention.

The ACB shown in Figure 4.4 is used for the transfer of data between two processes, the continuous uninterrupted streaming eDMA transfers and the knock software thread processing, explained in section 4.2.4. The producer process, namely the eDMA transfer, places items into the ACB and the consumer process, namely the knock software threads remove them for the algorithmic processing. The variable capacity of the ACB accommodates timing differences between the producer and the consumer processes.

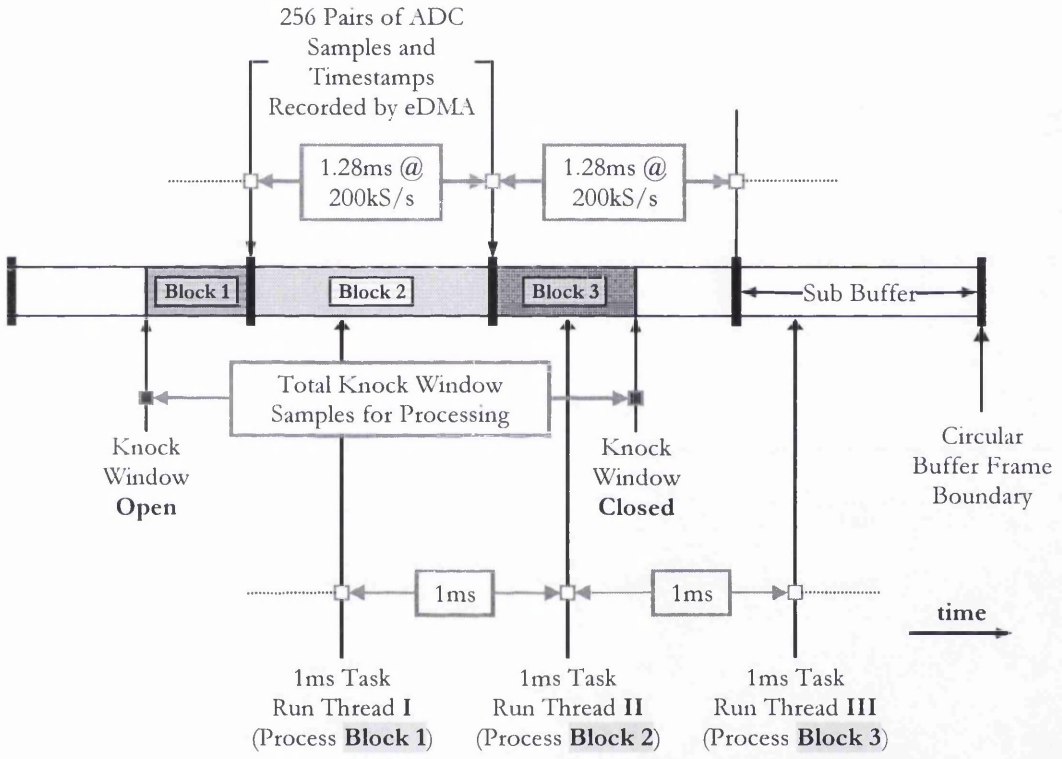

Figure 4.4: Advanced Software Circular Buffer and Knock Thread Processing

In the implemented strategy, the software ACB executes faster than other queues that hold a variable amount of data since a fixed size block of memory is allocated just once from memory management and then reused. This circular buffer can be visualised as a linear buffer with indices that wrap, modulo the buffer size, when the end of the buffer is reached.

For the knock detection strategy, the ACB suited the overall implementation due to the decoupling requirement of the independent processes with different speeds. For example, a faster eDMA process can "burst" data into the buffer and continue with its processing. A slower thread, i.e., the consumer of that data can then read it at its own rate without synchronising and slowing the producer. In this type of application, the average rate, over time, of both processes must be the same to avoid an over or under flow condition of the ACB and this is referred to as the synchronous mode of operation.

In addition, sequencing is critical unless an appropriate method is used with the producer process executing first. To prevent unintentional incrementing of the tail pointer, the pointer is incremented only after the application has finished reading the data in the buffer and has indicated that the buffer space is relinquished for the write operation.

4.2.4 Knock Processing Threads

The time available for processing the developed knock kernel consists of a scheduling mechanism which contains tasks requiring system resources that should meet all logical and temporal constraints. As the knock algorithmic task is of predictable behaviour, i.e., periodic in nature, static scheduling is used in order to reliably meet all timing constraints. This is of fundamental importance when defining an allocation for hard deadline tasks in such a real-time environment. The knock kernel is portioned into two segments - one containing three threads, which are activated by a 1ms periodic task and the other containing all the background processing. As shown in Figure 4.4, the execution of these threads is primarily dependent on the position of the circular buffer write pointer and the status of the knock window. Figure 4.4 and Figure 4.5 illustrate a typical scenario for thread level processing using the four frames of the segmented ACB. In order to keep the data frame boundaries at fixed buffer locations and to make optimal use of the algorithm, the overall circular buffer size should be a multiple of the frame size. As shown in Figure 4.5;

- Thread I: The millisecond task initially enters this thread to begin knock processing. This thread is processed only if the knock window is in the open phase and only after a corresponding eDMA data transfer crossing a 256 data frame boundary.

- Thread II: This thread only processes data packets consisting of 256 data points and is executed only when the knock window is in the open phase.

- Thread III: This concluding thread is responsible for completing the knock widow data processing. The knock window should be in the closed phase for this thread to be invoked and executed.

If a multi-channel eDMA transfer is required, then the frame size should also be a multiple of the number of channels used. Doing so keeps the pointer arithmetic from becoming unnecessarily complex, which also keeps the core processing cycles to a minimum. Knock window open and closed data points are obtained by subtracting timestamp of the two events from the time obtained using the pre-fixed buffer boundary contents.

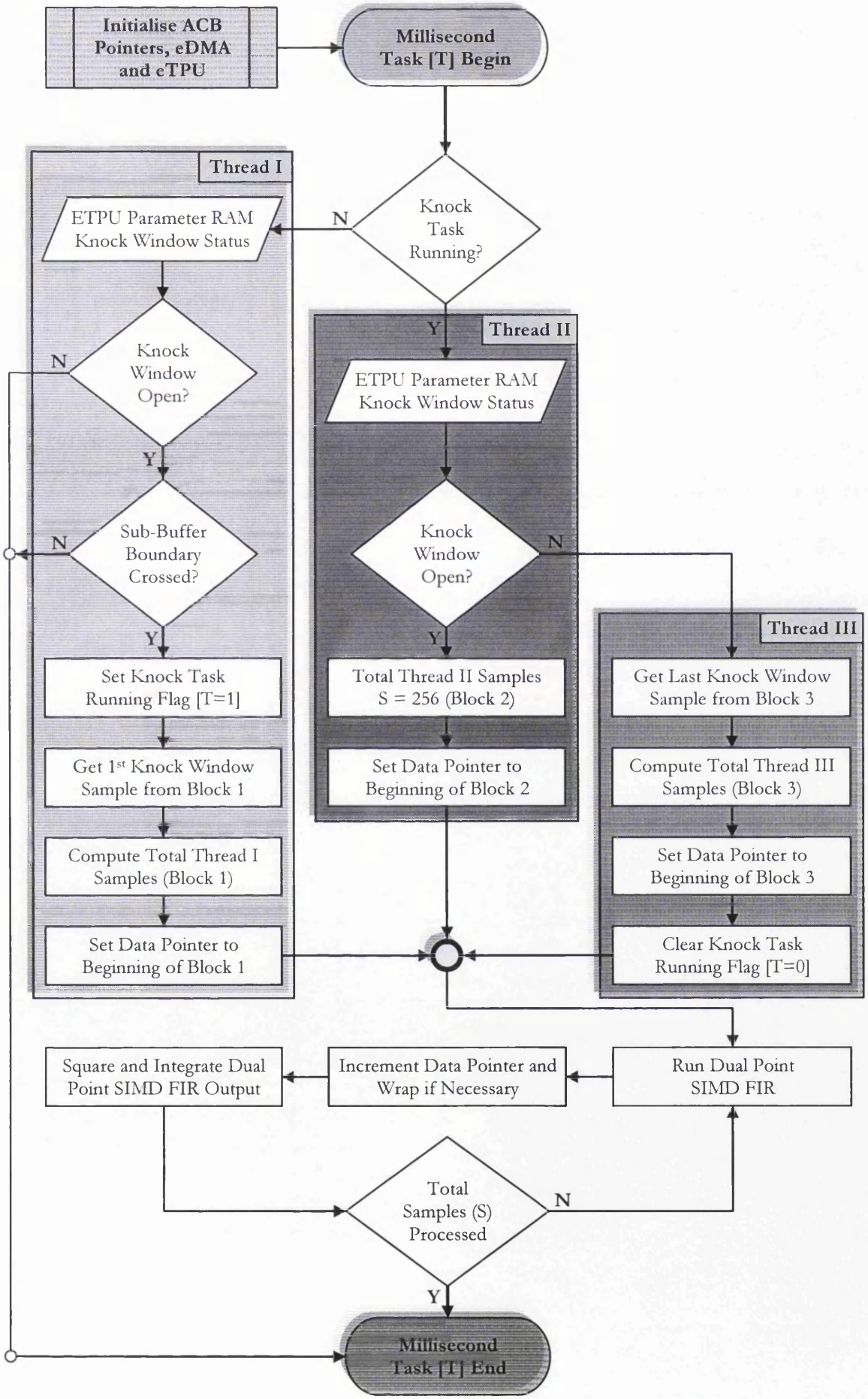

Figure 4.5: Thread Level Flowchart of Knock Algorithm

4.2.5 Implementation of the Dual Point SIMD FIR

Key bandwidth consuming function of the knock kernel is the dual point SIMD FIR. The FIR filter is implementable as a sequence of operations "multiply-and-accumulate", often called MAC. As shown in equation (2), in order to run an N^{th} order FIR filter it is necessary to have, at any instant, the current input sample together with the sequence of the N preceding samples. These N samples constitute the memory of the filter. In practical implementations, it is customary to allocate the memory in contiguous cells of the data memory or, in any case, in locations that can be easily accessed sequentially. At every sampling instant, the state must be updated in such a way that the data point x[k] becomes x[k – 1], and this seems to imply a shift of N data words in the filter memory. Indeed, instead of moving data, for computational efficiency, it is convenient to move the indexes that access the data as shown in the ACB in Figure 4.4.

```
/* include library with signal processing extensions */
#include <spe.h>

void fir(int N, short *x,  short *y, short *h)
{
  int i,j; /* loop counters */

  /* FIR o/p, load 2x32-bit input data into 64-bit reg */
  __ev64_opaque__ y0, z;

  /* Taps below indicates the total filter taps */
  for (i = Taps-1; i < (N + Taps-1); i += 2)
  {
    /* clear accumulator */
    __ev_set_acc_u64(long(0));

    for (j = 0; j < Taps; j++)
    {
      z = __ev_create_s32((int)x[i-j] , (int)x[i-j+1]);

      /* Vector Multiply Half Words, Odd, Signed,
      Saturate, Integer and Accumulate into Words */
      y0 = __ev_mhossiaaw(tap[j],z);
    }

    /* extract 2*32-bit outputs from accumulator */
    y[i]  = __ev_get_upper_u32(y0);
    y[i+1]= __ev_get_lower_u32(y0);
  }
}
```

Table 4.2: Unoptimised Dual Point SIMD FIR Code

```
lha          r12,0(r30)  ;load half-word algebraic
lha          r11,2(r30)  ;load half-word algebraic
evlddx       r10,r0,r9   ;vector load double word
evmergelo    r31,r12,r11 ;vector merge low
evmhossiaaw r11,r10,r31 ;vector multiply and accumulate
evmergelohi r31,r11,r11 ;vector merge
cmpi         0,0,r31,0   ;compare immediate
bc           4,0,.loop   ;branch conditional
neg          r31,r31     ;negate
```

Table 4.3: Unoptimised Dual Point SIMD FIR Assembly Code

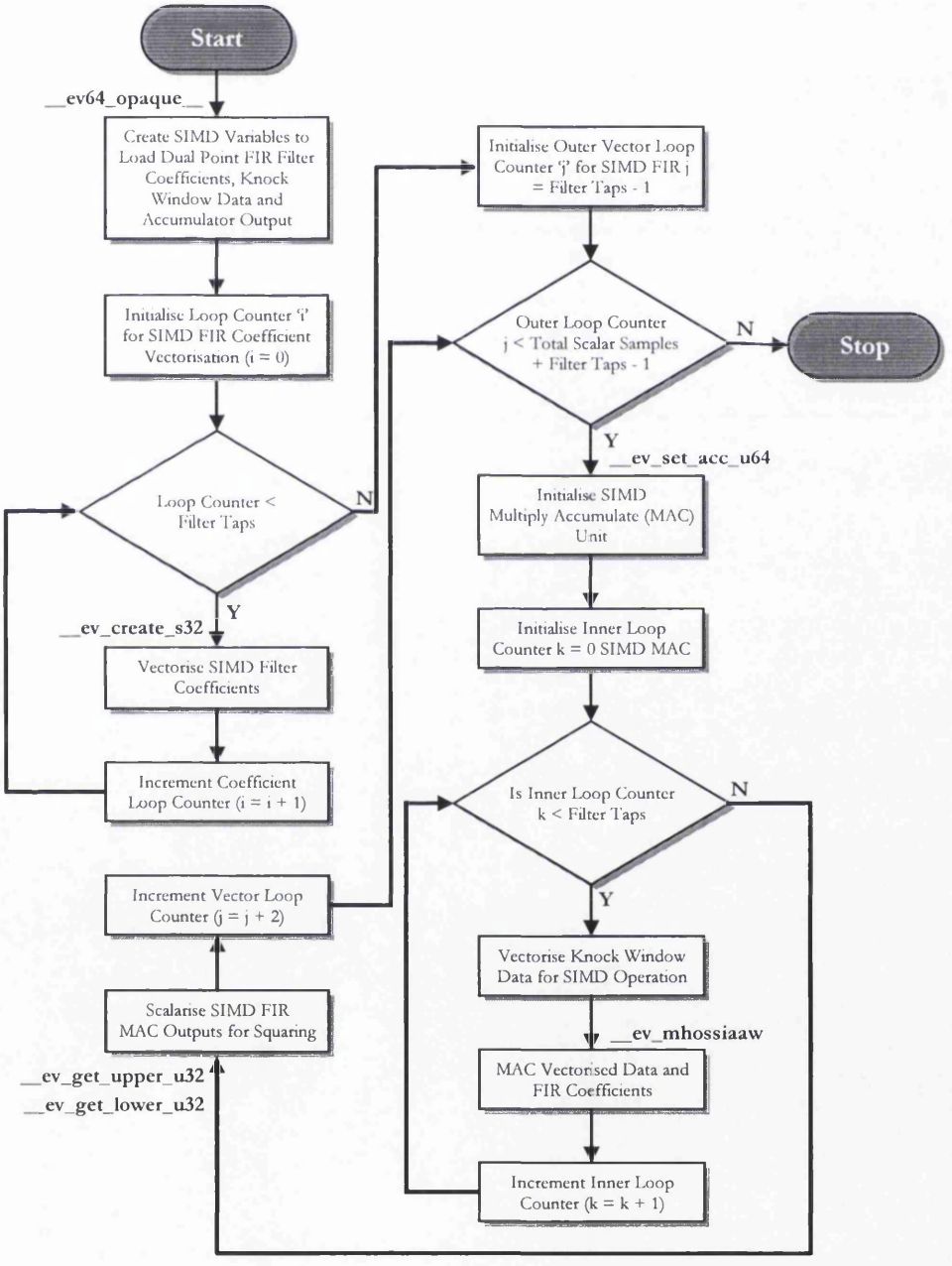

Figure 4.6: Software Implementation of the Dual Point SIMD FIR

As shown in Table 4.2 and Table 4.3, while SIMD extensions are capable of performing multiple computations in the same cycle, it is essential to provide data to the SIMD computation units in a timely fashion in order to make efficient use of the sub-word parallelism. Providing data in a timely fashion requires supporting instructions for address generation, address transformation (data reorganisation such as packing, unpacking, and permute), processing multiple nested loop branches, and loads/stores.

Table 4.2, Table 4.3 and Figure 4.6 illustrate the SIMD implementation of the bandpass FIR. The SIMD instruction, a.k.a., intrinsic, used for the implementation of the SIMD FIR inner loop utilises a dedicated fast hardware multiply accumulate (MAC) unit incorporated in the SPE, which allows back-to-back execution of dependent MAC instructions.

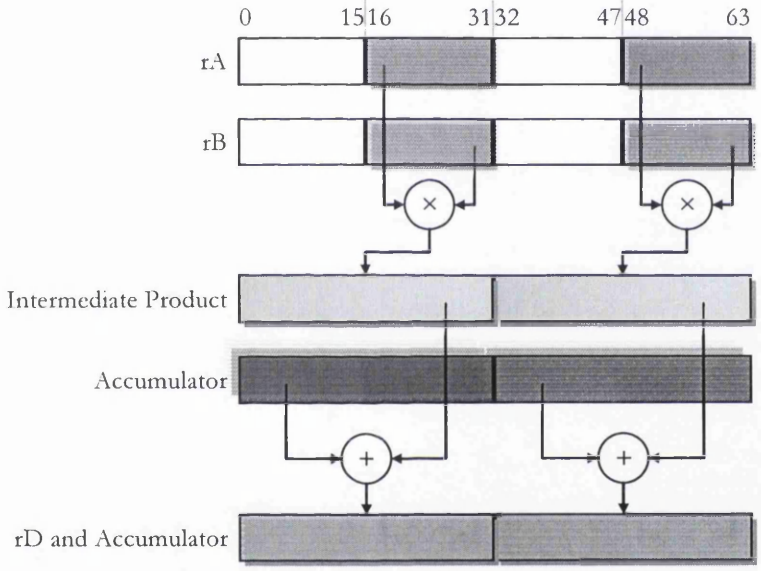

Figure 4.7: SIMD Dual Point Vector MAC

Figure 4.7 illustrates this hardware MAC instruction, evmhossiaaw rD, rA, rB (vector multiply half words, odd, signed, saturate, integer and accumulate into words). In this instruction, the least significant 16 bits of rA and rB are multiplied for both elements of the vector and the result is shifted left one bit and added to the accumulator and the result is possibly saturated to 32 bits in case of overflow.

The final result is placed both in the accumulator and also in rD so that the result of this instruction can be used by accessing rD. As shown in Fig. 10, The 64-bit architectural accumulator register, rD, holds the results of the MAC operation.

The accumulator is partially visible to the programmer in that its results do not have to be explicitly read to use them. Instead, for computational efficiency, they are always copied into a 64-bit destination GPR specified as part of the instruction. The accumulator however has to be explicitly cleared when starting a new MAC loop. Based upon the type of intrinsic used, an accumulator can hold either a single 64-bit value or a vector of two 32-bit elements.

5 Findings and Conclusions

Table 5.1 and Table 5.2 overleaf show the measured cylinder combustion event bandwidth required to process the knock kernel using the evaluation platform described in section 4. In both tables, performance was evaluated by running the SoC at 132MHz at a sampling rate of 200kSamples/sec.

For example, in Table 5.1, at 500rpm, a 15° knock window task requires 372 (22+284+66) μs to process the knock kernel, which is 0.62% of the overall cylindrical combustion event bandwidth of the V4. The results obtained are based on a single-band bandpass filter applied to the primary knocking resonance frequency component. The measured integrator output shown in Figure 5.1 consists of both the noise and the appropriate resonance components in the band of interest. The normalised time in this figure shows the time required to produce the appropriate integrator output, a.k.a., knock index.

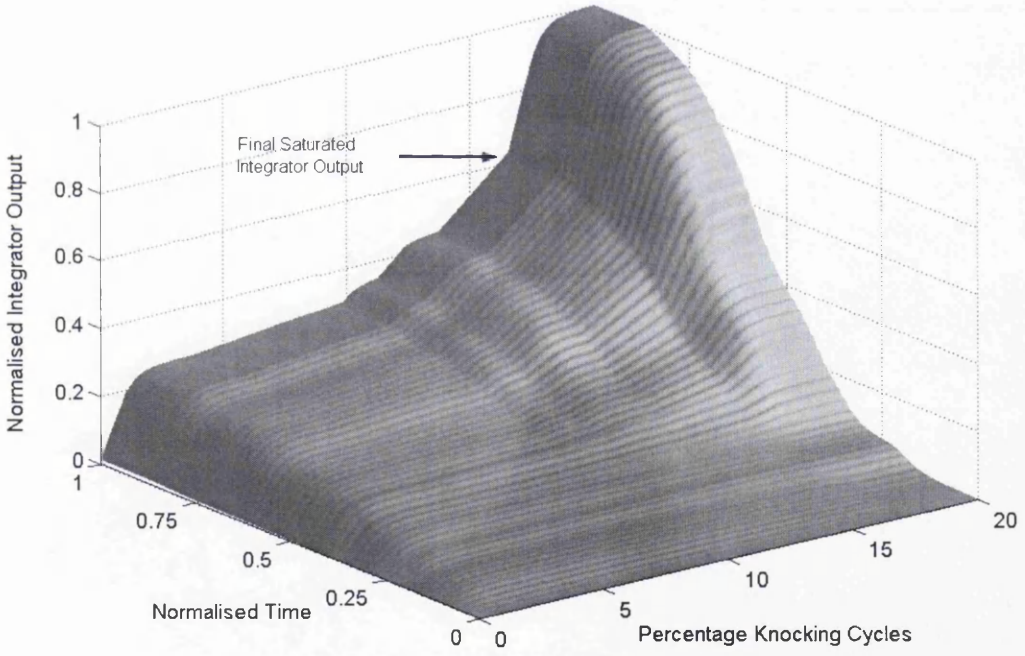

Figure 5.1: Dual Point SIMD Integrator Output

The knock free index, in other words, the pure background engine noise in the absence of knock is adaptively averaged and updated over time and this serves as the acceptable no knock threshold of the appropriate cylinder firing cycle [13]. As shown in [10], signal-to-noise-ratio of the overall process can be improved by selectively switching a multiband bandpass filter at higher engine speeds to extract all present knock resonance frequencies. Such a multiband filter can be designed using constrained least squares FIR filter design technique [18], [19].

RPM	Thread Level Execution Timing in μs						% Combustion Event Time Used for Thread Processing	
	15° Knock Window Threads			40° Knock Window Threads				
	I	II	III	I	II	III	15°	40°
500	22	284	66	58	843	77	0.62	1.63
2000	72	NA	12	28	191	32	0.63	1.67
2500	60		16	36	97	68	0.63	1.68
4000	28		21	25	97	6	0.64	1.71
6000	11		22	33	NA	51	0.66	1.69
8000	16		10	63		1	0.68	1.71

Table 5.1: Bandwidth Required to Process SIMD Knock Kernel

RPM	Thread Level Execution Timing in μs						% Combustion Event Time Used for Thread Processing	
	15° Knock Window Threads			40° Knock Window Threads				
	I	II	III	I	II	III	15°	40°
500	53	711	164	145	2000	192	1.55	4.11
2000	179	NA	55	68	474	78	1.55	4.13
2500	149		38	90	238	169	1.56	4.14
4000	68		50	60	238	14	1.57	4.16
6000	27		53	82	NA	127	1.60	4.18
8000	38		22	156		0	1.61	4.17

Table 5.2: Bandwidth Required to Process non-SIMD Knock Kernel

It has been shown that using a common architecture for both RISC and DSP instructions, in combination with autonomous on-chip peripherals, allows complex systems to be built around a single SoC platform, where previously two or more different processors would have been used together [13], [15]. Based on the results obtained, it is also evident that real SIMD computers need to have a mixture of single instruction single data (SISD) and SIMD instructions. Importance of SISD elements in the micro-core to perform operations such as branches and address calculations that do not need parallel operation is also highlighted. It is also worth nothing that for efficient dynamic power management and flexibility, unused individual execution units of the SoC are disabled during algorithmic execution.

Experimental analysis confirms that SIMD works best in dealing with arrays of streaming data. Additionally, in the proposed architecture, sustained MAC instructions are executed in a single CPU cycle. In contrast, in a typical fixed-point microprocessor, a multiply and an add typically executes in 15 to 20 CPU cycles [16].

The SIMD unit implemented also significantly increases execution speed by performing multiple operations in parallel. For instance, in the same instruction cycle that a MAC operation is performed, a parallel data move is carried out. SIMD enhancements in the SoC supplement the computational speed of present generation real-time processors used and make them ideal for high-performance real-time applications.

As shown in Table 5.3 and Figure 5.2, computational bandwidth is what separates the SIMD based core from the classic CPU – the ability to process an abundance of data, consistently, in an uninterrupted stream. Measured results in Table 5.3 confirms that the efficient coding and optimisation techniques used for the SIMD implementation of the knock kernel have improved performance by a minimum of x1.8. This figure is obtained by comparing the overall normalised knock task processing time, i.e., sum of time taken to run all three knock threads, shown in Fig. 8, with and without SIMD.

Vehicle RPM Range	SIMD Enabled Core		SIMD Disabled Core		Legacy RCPU	
	15° Knock Window	40° Knock Window	15° Knock Window	40° Knock Window	15° Knock Window	40° Knock Window
500	0.93%	2.45%	2.32%	6.17%	8.96%	23.82%
2000	0.94%	2.50%	2.33%	6.20%	8.98%	23.88%
2500	0.95%	2.52%	2.34%	6.21%	8.98%	23.85%
4000	0.96%	2.56%	2.35%	6.24%	9.06%	23.96%
6000	0.99%	2.54%	2.39%	6.28%	9.07%	23.99%
8000	1.02%	2.56%	2.42%	6.25%	9.12%	24.12%

Table 5.3: Processor Core Bandwidth Comparison of Knock Kernel

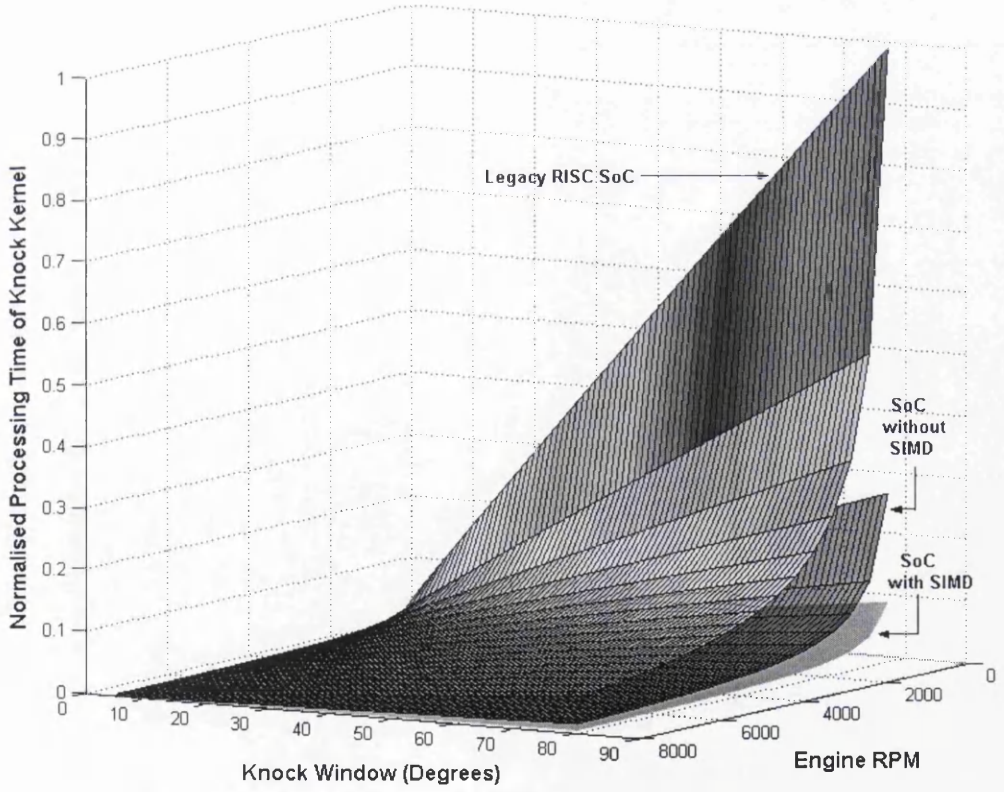

Figure 5.2: Knock Kernel Processing Time with and without SIMD

Streaming knock processing constitutes a significant part of current day average microprocessor workloads. To address this, a SoC combining SIMD DSP functionality with a classic microprocessor core has been developed. Autonomous peripherals on this SoC designed use fast register and memory based communication and synchronisation mechanisms to deliver high performance. Memory based communication and synchronisation is realised using the eDMA module. Parallelism in this application is exploited using a combination of orthogonal parallel processing techniques, namely instruction and data level parallelism (ILP and DLP).

A novel high performance knock detection strategy on an automotive SoC has been presented and the efficient use of various intelligent autonomous modules on the Motorola's next generation automotive qualified SoC combining ILP and DLP in the form of SIMD parallelism are described. The change from standard scalar 32-bit to SIMD vector based cores has been established, which allows the implementation of advanced algorithms with minimal processing time.

6 References

[1] J. B. Heywood, Internal Combustion Engine Fundamentals, New York: McGraw-Hill, 1988.

[2] Taylor, C. F., The Internal Combustion Engine in Theory and Practice, Volume II: Combustion, Fuels, Materials, Design, M.I.T. Press, MA, 1985

[3] Ferguson, C. R., Internal Combustion Engines: Applied Thermosciences, Wiley and Sons, NY, 1986.

[4] C. S. Draper, Pressure waves accompanying detonation in engines, Journal of Aeronautical Science, vol. 5, pp. 219-226, 1938.

[5] S. Ortmann, M. Rychetsky, M. Glesner, R. Groppo, P. Tubetti, G. Morra, Engine knock estimation using neural networks based on a real-world database, SAE Technical Paper 980513.

[6] E. C. Ifeachor and B.W. Jervis, Digital Signal Processing, A Practical Approach, Reading, MA: Addison-Wesley Ltd., pp. 278–373.

[7] F. Mintzer and B. Liu, "An estimate of the order of an optimum FIR bandpass digital filter," in Proc. IEEE 1978 Int. Conf. on Acoust., Speech, Signal Processing, May 1978, pp. 483-486.

[8] F. Mintzer and B. Liu, Practical Design Rules for Optimum FIR Bandpass Digital Filters, IEEE Transactions on Acoustics, Speech and Signal Processing, vol. ASSP-27, no. 2, April 1979.

[9] M. D. Checkel, J. D. Dale, Computerised knock detection from engine pressure records, SAE Technical Paper 860028.

[10] M. Kaneyasu et al., Engine knock detection using multi-spectrum method, SAE Technical Paper 920702.

[11] B. Samimy, G. Rizzoni, Engine knock analysis and detection using time frequency analysis, SAE Technical Paper 960618.

[12] Hickling, R., Feldmaier, D. A., Chen, F. H. K., Morel, J. S., Cavity Resonances in Engine Combustion Chambers and Some Applications, Journal of Acoustical Society of America, vol. 73, no. 4, pp. 1170-1178, 1983.

[13] J. Wagner, J. Keane, R. Koseluk, and W. Whitlock, Engine Knock Detection: Products, Tools, and Emerging Research, SAE Technical Paper 980522.

[14] S. Carstens-Behrens and J.F. Bohme, "Fast Knock Detection using Pattern Signals", IEEE Int'l Conf. on Acoustics, Speech, and Signal Processing Proceeding, v. 5, 2001, p 3145-3148.

[15] A. Kirschbaum, S. Ortmann, and M. Glesner, "Rapid Prototyping of a Co-processor based Engine Knock Detection System", Proceedings of the International Workshop on Rapid System Prototyping, 1998, p 124-129.

[16] Motorola Semiconductor Products Sector, [Online], Available: http://www.motorola.com/automotive/products.html

[17] Intersil Corporation, High Performance Analog Division [Online], Available: http://www.intersil.com/product_tree/product_tree.asp?x=9

[18] G. Cortelazzo, M. Lightner, and W. Jenkins, Frequency domain design of multiband FIR filters based on the minimax criterion," in Proc. IEEE International Symposium on Circuits and Systems, (Chicago, Illinois), pp. 532-535, April 1981.

[19] Ivan Selesnick, Markus Lang, and C. Sidney Burrus, A Modified Algorithm for Constrained Least Square Design of Multiband FIR Filters with Specified Transition Bands," IEEE Transactions on Signal Processing, vol. 46, issue:2, Feb. 1998, pp. 497 – 501.

[20] MotoTron [Online], Available: http://www.mototron.com/products/

Motorola TSPG
32-bit Embedded Controller division

Embedded Von-Neumann On-Chip PowerPC SoC Cache

Applicable to Processor Families with the ZEN Z650N3 core

Revision Number	Revision Date	Author	Comment
0.1	July 07, 2002	Mohamed Anas	Change of diagram for direct and indirect cache implementation.
0.2	July 21. 2002	Mohamed Anas	Expansion of section on using the cache as local memory.
0.3	November 01, 2002	Mohamed Anas	Cache lookup flow diagram added. Parity protection feature added.
0.4	November 11, 2002	Mohamed Anas	Changes in address calculation section and cache terminologies. Change of diab dld.
0.5	November 15, 2002	Mohamed Anas	Created and added new ".dld"/".lnk" for customer.
0.6	November 22, 2002	Mohamed Anas	Added comments in the .dld to facilitate understanding. Added new data cache group in the .dld.

TABLE OF CONTENTS

TABLE OF FIGURES

LIST OF TABLES

1. Abstract

The PowerPC™ instruction set provides opcodes that allow programmers to explicitly move items in or out of the processor's unified cache. This application note examines these cache control instructions and provides the user with sample routines to invalidate the cache and relocate object code in memory using the cache. Most applications only use these cache control instructions during power-on initialisation and when necessary to flush cache contents to system memory. However, in time-critical code segments these instructions can often improve throughput via preloading of required cache contents and by reducing unnecessary transfers between external memory and the unified cache.

2. Introduction

In order to maintain a high level of performance processor families with the ZEN Z650N3 core require access to instructions and data at the clock rate. In most circumstances, external memory cannot provide this level of data throughput and on-chip memory cannot fulfil all the system's memory requirements. An effective solution is to exploit the locality of instruction and data accesses present in most programs and retain frequently accessed items in an on-chip cache memory. Typically, caches greatly increase performance while minimally affecting system and program design. However, there are methods of accessing memory that can cause on-chip caches and external memory to become incoherent, resulting in data errors and possible system failure.

3. Z650n3 Cache Specific Terminologies

3.1. Cache Hit

A cache hit refers to an occurrence when the CPU asks for information from the cache, and gets it.

3.2. Cache Miss

A cache miss is an occurrence when the CPU asks for information from the cache, and does not get it. From this, we can derive the hit rate, or the average percentage of times that the processor will get a cache hit.

3.3. Hit Ratio / Hit Rate

The hit ratio is a metric that quantifies the fraction of processor accesses satisfied by the cache compared to the total number of accesses. Usually the hit ratio is determined individually for the instruction and data caches.

3.4. Write Policies

The cache's write policy determines how it handles writes to memory locations that are currently being held in cache. The two policy types are:

3.4.1. Write-Back Policy

When the system writes to a memory location that is currently held in cache, it only writes the new information to the appropriate cache line. When the cache line is eventually needed for some other memory address, the changed data is "written back" to system memory. This type of cache provides better performance than a write-through cache, because it saves on (time-consuming) write cycles to memory.

3.4.2. Write-Through Policy

When the system writes to a memory location that is currently held in cache, it writes the new information both to the appropriate cache line and the memory location itself at the same time. This type of caching provides worse performance than write-back, but is simpler to implement and has the advantage of internal consistency, because the cache is never out-of-sync with the memory the way it is with a write-back cache. Both write-back and write-through caches are used extensively, with write-back designs more prevalent in newer and more modern machines.

3.4.3. Valid, Invalid, Dirty (and Valid) and Locked Cache Lines

Valid cache lines contain valid data, which is consistent with main memory. Dirty cache lines also contain valid data, but it is not consistent with main memory. Invalid cache lines contain invalid data and are ignored during look-ups. A locked cache is not available for replacement

3.5. Coherency

A memory system is coherent when the value read from a memory address is always the value last written to that address.

4. Cache Mapping Strategies

4.1. Direct Mapped Cache

Each memory location is mapped to a single cache line, which could be shared with many others memory locations; only one of the many addresses that share this line can use it at a given time. This is the simplest technique both in concept and in implementation. Using this cache means the circuitry to check for hits is fast and easy to design, but the hit ratio is relatively poor compared to the other designs because of its inflexibility.

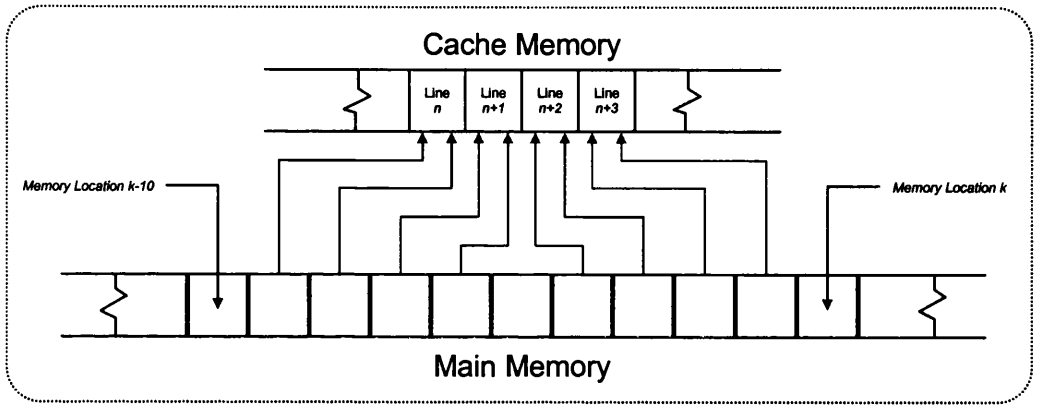

Figure 1: Direct Mapped Cache

4.2. Fully Associative Cache

Any memory location can be cached in any cache line. This is the most complex technique and requires sophisticated search algorithms when checking for a hit. It can lead to the whole cache being slowed down because of this, but it offers the best theoretical hit ratio since there are so many options for caching any memory address.

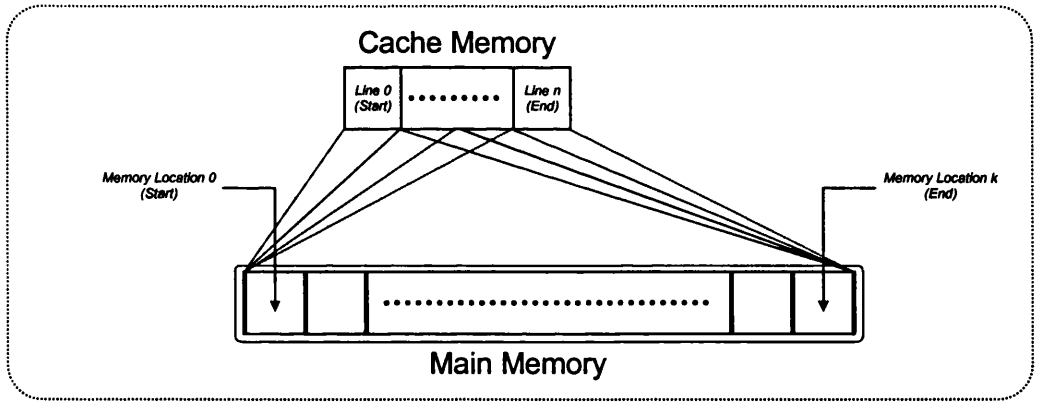

Figure 2: Fully Associative Cache

4.3. N-Way Set-Associative Cache

"N" is typically 2, 4, 8 etc. This technique is a compromise between the two previous designs. In this case, the cache is broken into sets of "N" lines each, and any memory address can be cached in any of those "N" lines. This improves hit ratios over the direct mapped cache, but without incurring a severe search penalty (since "N" is kept small). The 2-way, 4-way or 8-way set associative cache is common in modern processor L1 caches.

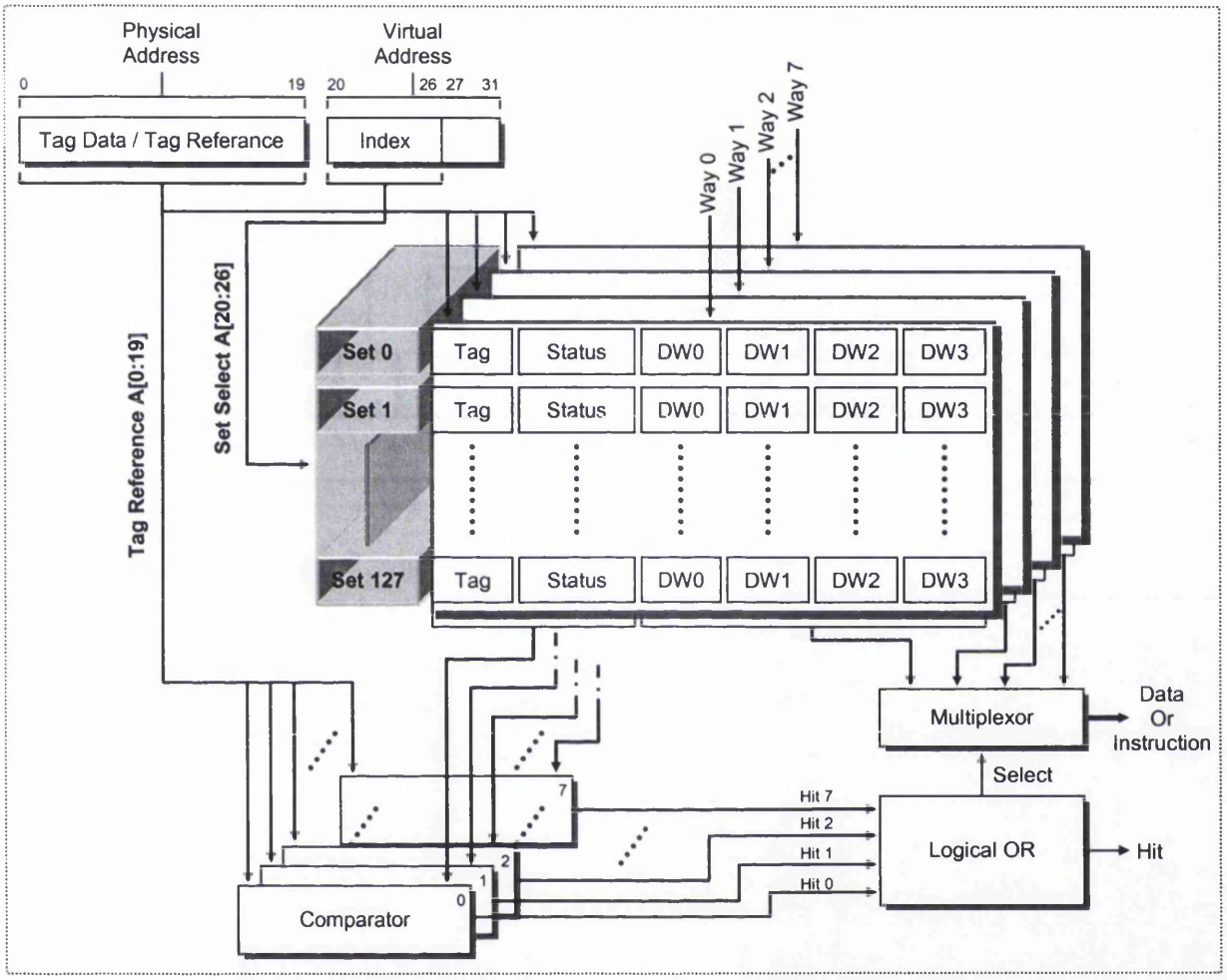

Figure 3: 8-Way, 32kB Set-Associative Cache Lookup Flow

Figure 3 illustrates the general flow of cache operation for the 32kB Cache. To determine if the address is already allocated in the cache,

1. The cache set index, virtual address bits A[20:26] are used to select one cache set. A set is defined as the grouping of eight lines (one from each way), corresponding to the same index into the cache array.

2. The higher order physical address bits A[0:19] are used as a tag reference or used to update the cache line tag field.

3. The eight tags from the selected cache set are compared with the tag reference. If any one of the tags matches the tag reference and the tag status is valid, a cache hit occurs. Please note that each tag is unique.

4. Virtual address bits A[27:28] are used to select one of the four doublewords in each line. A cache hit indicates that the selected doubleword in that cache line contain valid data (for a read access), or can be written with new data depending on the status of the access control bit necessary for a write access.

5. Cache Structure of Zen Z650N3

The L1 Cache described in this document incorporates the following features:

- 32 kB Unified Cache design (support for 16K and 8K options)
- Virtually indexed, Physically tagged
- 32-byte line size (4 doublewords)
- 64-bit data, 32-bit address
- Pseudo Round-Robin (PRR) replacement algorithm
- 8-entry store buffer
- 1 entry push (copyback) buffer
- 1 entry linefill buffer
- Hit under fill/copyback
- Parity protection

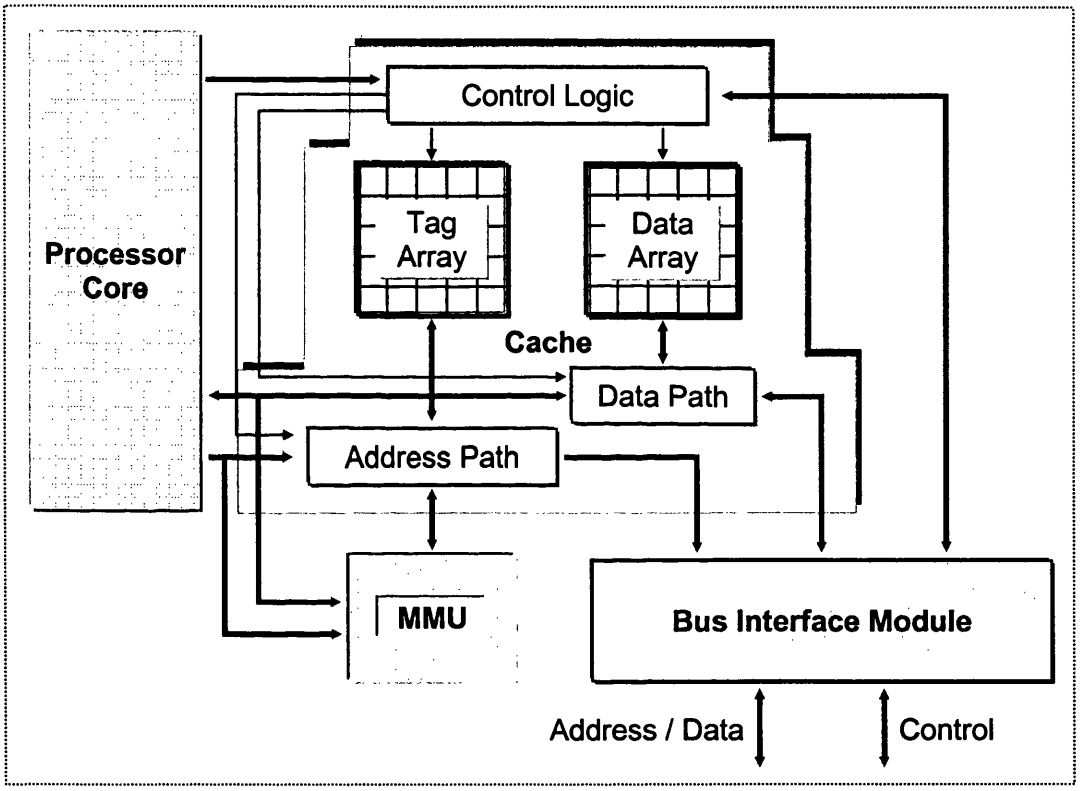

Figure 4: Overview ZEN Z650N3 Unified Cache

The Zen Z650N3 processor supports a 32 kB, 8-way set-associative, unified (instruction and data) cache with a 32 Byte line size. This cache can also be configured as 16 kB, 4-way or 8 kB, 4-way set-associative units. The cache is virtually indexed and physically tagged.

The Zen Z650N3 does not provide hardware support for cache coherency in a multi-master environment. Therefore, software must explicitly accomplish this task, explained in section 7.

6. 32 kB Unified Cache Organisation

The Zen cache is organised as eight ways of 128 sets with each line containing 32 Bytes (four doublewords) of storage. The following figure illustrates the cache organisation as well as the terminologies used, along with the cache line format.

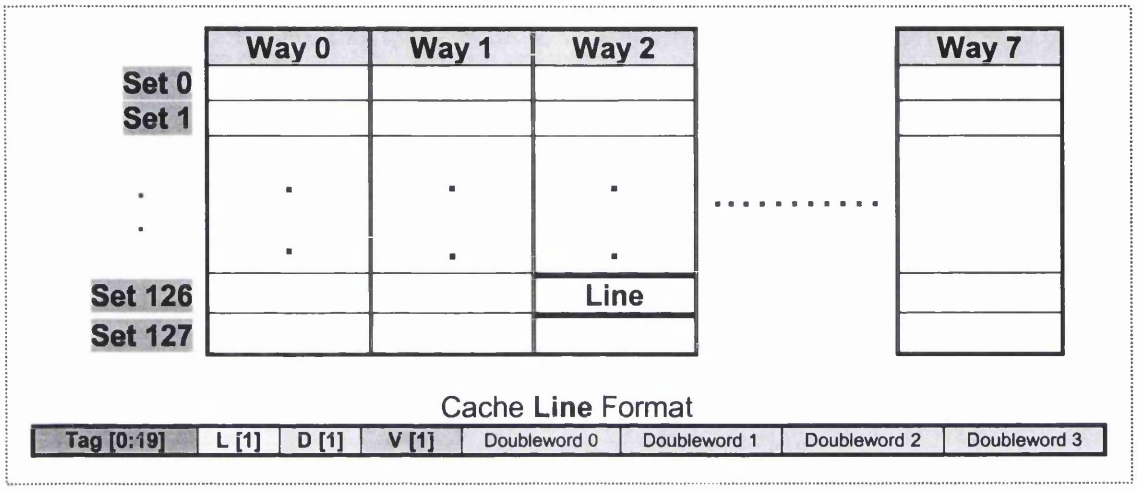

Figure 5: Cache Organisation and Line Format

Where,

Tag - 20-bit physical address corresponding to the data contained in this line

L - Lock bit
0 = The line has not been locked.
1 = The line has been locked and is not available for replacement.

D - Dirty bit
0 = The data contained in this entry has not been modified.
1 = The data contained in this entry has been modified and is not consistent with physical memory.

V - Valid bit
0 = This bit signifies that the cache line is invalid and a tag match should not occur
1 = This bit signifies that the cache line is valid

7. Cache Coherency

A memory system is coherent when the value read from a memory address is always the value last written to that address. Systems that perform all reads and writes directly to a common memory and those with caches that utilise specialised hardware techniques to enforce coherency are always coherent. Since the Zen Z650N3 processor features cache without dedicated hardware for enforcing coherency, situations may occur where software must explicitly accomplish this task. Prior to examining techniques for enforcing coherency, several situations that can cause a loss of coherency are examined below:

7.1. How is Coherency Lost?

When an area of memory exists in a cache, any access to that region by another device or cache has the potential to result in a loss of coherency. One of the most common examples is an external controller, such as a DMA channel, directly accessing a cacheable area of system memory. Following figure depicts such a system. Because the external controller directly reads from and writes to system memory, it neither obtains modified nor updates data present in the cache. Directly reading locations in system memory also present in a cache frequently yields incorrect data. This occurs because modified cache lines in a *write-back* cache do not update external memory until normal cache activity or an explicit cache control instruction displaces them from the cache.

In contrast, when the processor alters data in a cache that supports *write-through* operations that change is also written to external memory at the same time. In particular, when an external device changes a memory location also present in the cache, the cache is not updated. Additionally, write-through operations may cause significantly higher amounts of traffic to the memory system, thus decreasing overall system performance.

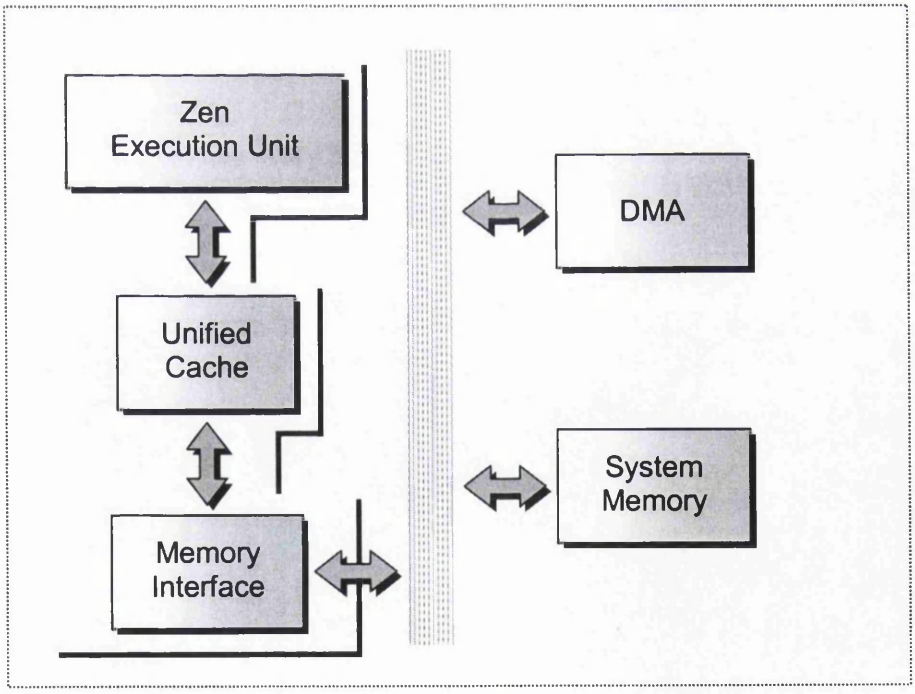

Figure 6: Maintaining Coherency with Shared Access

In the Zen Z650N3 processor, cache coherency is supported through software operations to invalidate, flush dirty lines to memory or invalidate dirty lines. The cache may operate in either write-through or write-back modes, and in conjunction with an MMU, may designate certain accesses as write-through or copy-back. Cache misses will force the copy-back and store buffers to empty prior to performing the access. No other hardware coherency support is provided.

8. Cache Operation

8.1. Cache Enable / Disable

The cache is enabled or disabled by using the Cache Enable bit, L1CSR0[CE]. L1CSR0[CE] is cleared on power-on reset or normal reset, disabling the cache. When the cache is disabled (L1CSR0[CE] = 0), the Cache Tag status bits are ignored, and the cache is not accessed for normal loads, stores, or instruction fetches. All normal accesses are propagated to the system bus as single-beat (non-burst) transactions.

Altering the CE bit must be preceded by an `isync` and `msync` to prevent the cache from being disabled or enabled in the middle of a data or instruction access. In other words, if the cache is to be changed from enabled to disabled, copy-back of dirty lines must be forced to ensure coherency. It is also worth noting that global flushing of cache is advisable before it is disabled to prevent coherency problems when it is re-enabled. All cache operations are affected by disabling the cache.

8.2. Cache Line Fills

Cache line fills are requested when a load miss occurs. Cache line fills load a four-doubleword linefill buffer, and actual updates to the cache array are delayed until and a free cycle is available to perform the cache update. The line fill buffer is burst loaded. In addition, the cache supports hit under fill to improve performance.

8.3. Cache Line Replacement Using the Pseudo Round Robin Algorithm

On a cache read miss, the cache controller uses a pseudo-round-robin replacement algorithm to determine which cache line would be selected to be replaced. There is a single replacement counter for the entire cache. Lines selected for replacement that are dirty (modified) must be copied back to main memory.

The replacement algorithm acts as follows: On a miss, if the replacement pointer is pointing to a way, which is not enabled for replacement by the type of the miss access (the selected line or way is locked), it is incremented until an available way is selected (if any). After a cache line is successfully filled without error, the replacement pointer increments to point to the next cache way. If no way is available for the replacement, the access is treated as a single beat access and no cache linefill occurs.

8.4. Cache Management Instructions

The Zen Z650N3 provides a number of instructions for managing the unified cache and potentially improving code performance. These cache control instructions can be used during power-on initialisation and when necessary to flush cache contents to system memory. In time-critical code segments these instructions can often improve throughput via preloading of required cache contents and by reducing unnecessary transfers between slower system memories.

Following are brief descriptions and the typical usage of common Zen cache control instructions. Since these instructions may vary by processor, more complete descriptions, syntax, architectural notes and exceptions for these and additional instructions are included in the User's Manual for the appropriate processor or core.

- icbi - Instruction Cache Block Invalidate (maps this instruction to a dcbf)
- dcbf - Data Cache Block Flush
- dcbi - Data Cache Block Invalidate
- dcbst - Data Cache Block Store
- dcbz - Data Cache Block set to Zero

9. Cache Control

Control of the cache is provided by bits in the L1 Cache Control and Status register (L1CSR0). Control bits are provided to enable/disable the cache and to invalidate it of all entries. In addition, availability of each way of the cache may be selectively controlled for use by instruction accesses and data accesses. This way control provides cache way locking capability, as well as controlling way availability on a cache line replacement.

9.1. L1 Cache Control and Status Register 0 (L1CSR0)

The L1CSR0 is a 32-bit register. The L1CSR0 register is accessed using a "move from special purpose register" (mfspr) or "move to special purpose register" (mtspr) instruction. The correct sequence necessary to change the value of LSCSR0 is as follows:

The `msync` instruction provides a synchronization function. This instruction waits for all preceding instructions and data memory accesses to complete before the `msync` instruction completes. Subsequent instructions in the instruction stream are not initiated until after the `msync` instruction ensures these functions have been performed. The `isync` instruction waits for all previous instructions to complete and then discards any fetched instructions, causing subsequent instructions to be fetched (or refetched) from memory and to execute in the context (privilege, translation, and protection) established by the previous instructions.

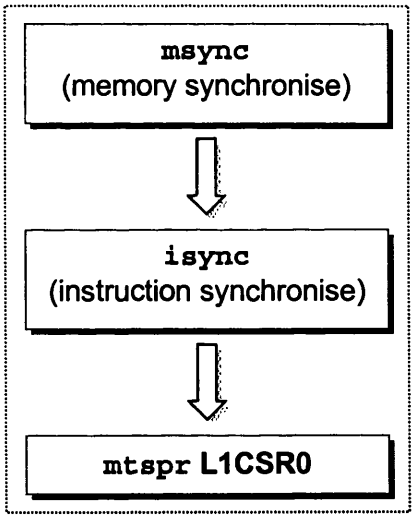

Figure 7: Sequence of Instructions Necessary to Change LSCSR0 Value

9.2. L1 Cache Configuration Register 0 (L1CFG0)

L1CFG0 is a 32-bit read-only register. L1CFG0 provides information about the configuration of the Zen Z650N3 L1 Cache design. The contents of the L1CFG0 register can be read using an `mfspr` instruction. The L1CFG0 register is shown below: The SPR number for L1CFG0 is 515 in decimal.

CARCH		CWPA	CFAHA	CFISWA	0		CBSIZE		CREPL		CLA	CPA	CNWAY							CSIZE										
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Table 1: SPR – 515: Read-only

The L1CFG0 bits are described below:

Bits	Name	Description
0:1	CARCH	Cache Architecture 01 – The cache architecture is unified
2	CWPA	Cache Way Partitioning Available 1 – The cache supports partitioning of way availability for I/D access
3	CFAHA	Cache Flush All by Hardware Available 0 – The cache does not support flush all in hardware
4	CFISWA	Cache Flush / Invalidate by Set and Way Available (via L1FINV0) 1 – The cache supports flushing / invalidation by set and way
5:6	—	Reserved Read as 0
7:8	CBSIZE	Cache Block Size 00 – The cache implements a block size of 32 bytes
9:10	CREPL	Cache Replacement Policy 10 – The cache implements a pseudo-round-robin replacement policy
11	CLA	Cache Locking APU Available 1 – The cache implements the line locking APU
12	CPA	Cache Parity Available 0 – The cache does not implement parity
13:20	CNWAY	Number of ways in the Unified Cache 0x03 – The cache is 4-way set-associative 0x07 – The cache is 8-way set-associative
21:31	CSIZE	Cache Size 0x008 – The size of the cache is 8kB 0x010 – The size of the cache is 16kB 0x020 – The size of the cache is 32kB

Table 2: Bit Fields of the L1CFG0

10. Cache Line Locking / Unlocking

Zen supports the *Motorola Book E* Cache Line Locking, which defines user-mode instructions to perform cache line locking or unlocking. Three of the instructions are for data cache locking control (dcblc, dcbtls, dcbtstls) and the remaining instructions are for instruction cache locking control (icblc, icbtls).

For the Zen Z650n3 unified cache, the instruction and data versions of these instructions operate similarly.

- dcblc - data cache block lock clear
- dcbtls - data cache block touch and lock set
- dcbtstls - data cache block touch for store and lock set

The data storage interrupt (DSI) handler decides whether or not to lock a given cache line based upon available cache resources. In general, a DSI occurs if no higher priority exception exists and one of the following exception conditions exists:

- Read or Write Access Control exception condition
- Byte Ordering exception condition
- Cache Locking exception condition

A Byte Ordering exception condition occurs for any misaligned access across a page boundary to pages with mismatched E (endianness) bits. Cache locking exception conditions occur for any attempt to execute a `dcbtls`, `dcbtstls`, `dcblc`, `icbtls` or `icblc` in user mode with MSR[UCLE (User Cache Lock Enable)] = 0, in which the execution of the cache locking instructions in user mode is disabled.

If the locking instruction is a set lock instruction, and if the handler decides to lock the line, the following 4 steps are done by the DSI:

1. Add the line address to its list of locked lines.
2. Execute the appropriate set lock instruction to lock the cache line.
3. Modify save/restore register 0 to point to the instruction immediately after the locking instruction that caused the DSI.
4. Execute an rfi (return from exception).

11. Cache Initialisation

Operating as local memory, the unified cache module provides the processor with access to recently referenced instructions and data. The cache should be explicitly invalidated after a hardware reset; reset does not invalidate the cache lines. Following initial power-up, the cache contents will be undefined. The L, D and V bits may be set on some lines, necessitating the invalidation of the cache by software before being enabled. Therefore, if the unified cache is to be enabled, it is necessary to explicitly invalidate the unified cache module by software. A generic PowerPC assembly routine to invalidate the unified cache is given in section 14.2.

12. Push and Store Buffers

The push buffer reduces latency for requested new data on a cache miss by temporarily holding displaced dirty data while the new data is fetched from memory. The push buffer contains 32 Bytes of storage (one displaced cache line or 4 doublewords).

On the other hand, the store buffer contains a FIFO that can defer pending write misses or writes marked as write-through in order to maximize performance. The store buffer can buffer 32 Bytes.

In the disabled push buffer case, dirty line replacement is performed by first generating a burst write transaction, copying out the entire dirty line, starting with the doubleword in the dirty line corresponding to the missed address. The store buffer may be disabled for debug purposes.

13. Programming the Unified Cache

13.1. Relocation of an Executable Object Code Using the Cache

Moving executable objects from one memory location to another with caching enabled requires cache control instructions. The following assembly routines forces coherency with such an operation:

```
                        ; r1 = source address (word aligned)
                        ; r2 = target address (word aligned)
                        ; r3 = number of words to move

addi   r1,r1,-4         ; allows use of lwzu and stwu
addi   r2,r2,-4
mtctr  r3               ; load count register with the value of r3

loop:  lwzu r4,4(r1)    ; read source
stwu   r4,4(r2)         ; write target
dcbf   0,r2             ; remove target from unified cache
bdnz   loop             ; repeat until done

msync                   ; ensure dcbst is completed
```

Figure 8: Routine to Relocate an Executable Object Using the Cache

This example moves one word and then flushes the corresponding address and then invalidates it in the unified cache. By using this sequence of operations, the cache remains coherent during the relocation process. However, if the application does not attempt to execute code in the target range until after moving the entire block, improved performance results from flushing the data cache and invalidating the instruction cache outside of the loop.

13.2. Flushing the Unified Cache

The size and placement of a shared memory area along with the size of the cache determines the best method for removing the region from the cache. Flushing the region by address is most efficient when the shared area is smaller than the data cache. An assembly routine to flush a specific region of the cache is as follows:

```
                                    ; r1 = start of region
                                    ; r2 = end of region

loop: dcbf 0, r1                    ; flush line at address r1
addi  r1, r1, <line size in bytes>  ; point to next line
cmpw  r1, r2                        ; finished?
ble   loop                          ; if not, continue until done
```

Figure 9: Routine to Flush the Cache

13.3. Invalidating the Unified Cache

The cache may be invalidated through the Cache Invalidate (CINV) control bit located in the L1CSR0 register. This function is available even when the cache is disabled. Reset does not invalidate the cache automatically. Software must use the CINV control for invalidation after a reset. Proper use of this bit is to determine that it is clear and then set it with a pair of `mfspr` and `mtspr` operations. During the process of performing the invalidation, the cache does not respond to accesses, and remains busy.

14. Using the Cache as Local Memory

Inability to reuse data and under utilisation of cache capacity are responsible for poor cache performance on various commonly used automotive applications. Data pre-fetching, blocking and data copying have been used to address these problems. These techniques, though effective, are directed towards solving one aspect of the overall problem. This section of the document proposes a comprehensive solution to these problems by employing the on-chip cache as an explicitly managed local memory using software. The cache can be statically or dynamically configured to act as a local memory block or a conventional cache; reconfiguration overhead is small. Therefore, if a time critical task is to be run from cache, it is necessary to acquire an exclusive lock to the cache line or way, which is done using the `dcbtls` instruction. This lock can then be cleared using the instruction `dcblc`.

14.1. Steps Needed for Emulation of the Cache as Local RAM

Instantiation of the cache block to be used as local memory should be done when the system comes out of a reset. The following steps explain how this can be done on a Zen based platform:

1. Read and clear error type bits in the L1CSR0.

2. Ensure the code that initialises the cache is executing from cache-inhibited memory, and interrupts are disabled. This guarantees that the sequence of cache operations execute as desired.

3. Set up the MMU to assign a dummy memory region with read and write permissions. The memory region should be the same start address and range as the data storage section defined by the linker for the data variables the user wishes to store in cache "local memory". It should be unimplemented memory to avoid conflicts with data references. Please see appendix A for sample of linker command file.

Setting up the MMU to assign a dummy memory region with read and write permissions can be done as follows:

The TLBCAM (Translation Lookaside Buffer – Content Addressable Memory) array must be first written to by writing the necessary information into MAS0-MAS3 (MMU Assist Registers - SPR 624-627) using mtspr and then executing the tlbwe instruction. To write an entry into the TLB, the TLBSEL field in MAS0 must be set to '01', and the ESELCAM (Entry Select for TLBCAM) bits in MAS0 must be set to point to the desired entry. When the tlbwe instruction is executed, the TLB entry information stored in MAS1-MAS3 will be written into the selected TLB entry. Please note that X used in the table needs to be set by the user based on the requirements.

```
mtspr $50XX00XX, MAS0  ;MMU Read/Write and Replacement Control
mtspr $4000XX00, MAS1  ;Descriptor Context and Configuration Control
mtspr $XXXXX002, MAS2  ;EPN(Effective Page Number) and Page Attributes
mtspr $XXXXXXXF, MAS3  ;RPN (Real Page Number) and Access Control
tlbwe RT, RA, WS   ;Specifies a word in the TLB entry being accessed
```

Figure 10: Setting Up the MMU for a TLB Write Operation

4. Execute `dcbz RA, RB`

```
add r1, 0, __DATA_CACHE_RAM_START ;load start address (Appendix A) to GPR1
dcbz 0, r1
```

dcbz establishes a cache line without accessing main memory. Since there are no memory bus cycles, the address range defined in the MMU in 2 above can be unimplemented memory. However, the instruction is treated as a store by the MMU, the address range must have user or supervisor write permissions (depending on which mode the code is executing in).

5. Execute `dcbtls CT, RA, RB`.

```
add r1, 0, __DATA_CACHE_RAM_START ;load start address (Appendix A) to GPR1
dcbtls 0, 0, r1
```

Because the previous instruction established the <ea> in the cache, the dcbtls hits in the cache, and therefore does not fetch the value from memory. So no memory bus cycles occur thus avoiding a bus error to our unimplemented memory. Note that in this case, the instruction is treated as a load by the MMU, so the address range must have user or supervisor read permissions (depending on which mode the code is executing in).

6. Increment address by a cache until the whole memory region is covered.

```
add r1, 0, __DATA_CACHE_RAM_START ;load start address (Appendix A) to GPR1
add r2, 0, __DATA_CACHE_RAM_END   ;load end address (Appendix A) to GPR2

loop:
addi r1,r1,32   ;point to next line.  The number of bytes per line = 32
cmpw r1,r2      ;is memory region covered?
ble loop        ;if not, continue until done
```

7. Continue with the rest of the initialisation code.

15. Conclusion

The Zen unified cache module greatly improves system performance. Although most programs do not directly manipulate the cache, cache control instructions provide programmers with the means to control the content of the L1 on-chip cache. This level of control is required in applications with devices that share memory with the processor or when a program relocates executable object from one memory area to another. By utilising cache control instructions these memory areas become cacheable and overall system performance increases.

Appendix A

```
/* ------------------------------------------------------------------------------------------*/
  * Sample Linker Command File to Set-up Dedicated Memory Segment to Handle Data to be cached
  * Please read diab 4.3 linker file documentation for more information.
  * Author: Mohamed Anas
/* ------------------------------------------------------------------------------------------*/

MEMORY
{
  ram:    org = 0x10000,  len = 0x6400
  rom:    org = 0x14000,  len = 0xA000
  stack:  org = 0x1E000,  len = 0x2000

  /* initialise data cache memory segment of 32kB to be used as local RAM on Copperhead. */

  datacache: org = 0x20000, len = 0x8000
}

SECTIONS
{

  GROUP : {
      .text (TEXT)    : {
    *(.text) *(.rodata) *(.init) *(.fini) *(.eini)
    . = (.+15) & ~15;
      }
      .sdata2 (TEXT) : {}
  } > rom


  GROUP : {
      .data  (DATA) LOAD(ADDR(.sdata2)+SIZEOF(.sdata2)) : {}
      .sdata (DATA) LOAD(ADDR(.sdata2)+SIZEOF(.sdata2)+SIZEOF(.data)) : {}
      .sbss  (BSS) : {}
      .bss   (BSS) : {}
  } >ram


GROUP : {

/* Note that the .bss section and .sbss sections (if present) never occupy any space in
the linked object file because they will be initialised by the system at execution time.

  .csbss (BSS) : {}
  .cbss (BSS)  : {}

} >datacache

}

  __DATA_CACHE_RAM_START = ADDR(.csbss);
  __DATA_CACHE_RAM_END   = ADDR(.csbss)+SIZEOF(.cbss);

__HEAP_START = ADDR(.bss)+SIZEOF(.bss);
__SP_INIT    = ADDR(stack)+SIZEOF(stack);
__HEAP_END   = ADDR(ram)+SIZEOF(ram);
__SP_END     = ADDR(stack);

__DATA_ROM   = ADDR(.sdata2)+SIZEOF(.sdata2);
__DATA_RAM   = ADDR(.data);
__DATA_END   = ADDR(.sdata)+SIZEOF(.sdata);
__BSS_START  = ADDR(.sbss);
__BSS_END    = ADDR(.bss)+SIZEOF(.bss);
```

NEXT GENERATION POWERTRAIN SOC PERFORMANCE REQUIREMENTS

MOTOROLA CONFIDENTIAL PROPERTY
FREESCALE CONFIDENTIAL PROPERTY

BY
MOHAMED ANAS
32-BIT EMBEDDED CONTROLLER DIVISION
MOTOROLA SPS
COLVILLES ROAD
KELVIN INDUSTRIAL ESTATE
EAST KILBRIDE
GLASGOW G750TG
SCOTLAND. U. K.

Table of Contents

1 Introduction

Ever increasing performance and increased memory requirements from various automotive powertrain applications radically increases the demand on powertrain ECU overhead and the bandwidth required to run such applications.

Powertrain applications not only take cost, performance and issues of real-time deterministic operation into account when choosing an SoC, but also lay emphasis on devices with sufficient flexibility and scalability to cope with complex, and modern development methodologies being employed. This document summarise customer requirements from the field along with feedback and research done so far, meeting all the demand required by various bandwidth intensive future powertrain applications with minimal cost.

In general, powertrain applications are driven by two main goals:

- Reduced fuel consumption
- Reduced emissions

These goals are to be achieved at the same or even higher engine power and, of course, at the same system cost level.

Our present customers and present trend confirm that new software development techniques presently being employed in powertrain applications increases the code size and requires higher computing power. Rapidly growing performance requirements are being primarily driven by applications developed with auto code generation using embedded real-time targets, inclusion of DSP functionality, model fitting to automatically generate calibration tables (Denso), frequent use of specialist math functions, especially in areas like model based development, excellent quality and reliability performance with increased safety requirements.

It is a very well known fact that implementing various number crunching and signal processing algorithms on a general-purpose CPU core can be very challenging. Issues such as numeric formats and precision, type conversion, cache behaviour, dynamic instruction scheduling, and data-dependent instruction execution times pose hazards for number crunching intensive applications, especially those working on applications with real-time constraints. These issues need to be address in the next generation core complex.

This document outlines the key requirements for next generation powertrain ECUs. This is done by profiling some key applications and particularly the critical computational blocks required for the execution of such applications. Emphasis is laid on model based control

Some of the critical computational blocks in the powertrain application code signatures are as follows:

- Table manipulation mechanisms that allows powertrain calibrations system to its most robust operating conditions
- Use of torque-spark curve to model port fuel injection (PFI)
- Fixed and floating point FIR filtering
- Fixed and floating point IIR filtering
- Various adaptive filters
- Fixed and floating point FFT including fast vector magnitude calculation
- Vector dot product
- Advanced motor control for x-by-wire
- Linear regression and radial basis function techniques

2 Brief Introduction to Some Key Powertrain Applications

Applications briefly described in this section details the specific functional modules used and required. Further details are available at request.

2.1 Motorola's SIMD Streaming Knock (MOSK) Processing

Figure 1 shows the measured performance of SIMD and scalar C based knock kernel I developed. See Table 1 for more details on specific functional performance requirements.

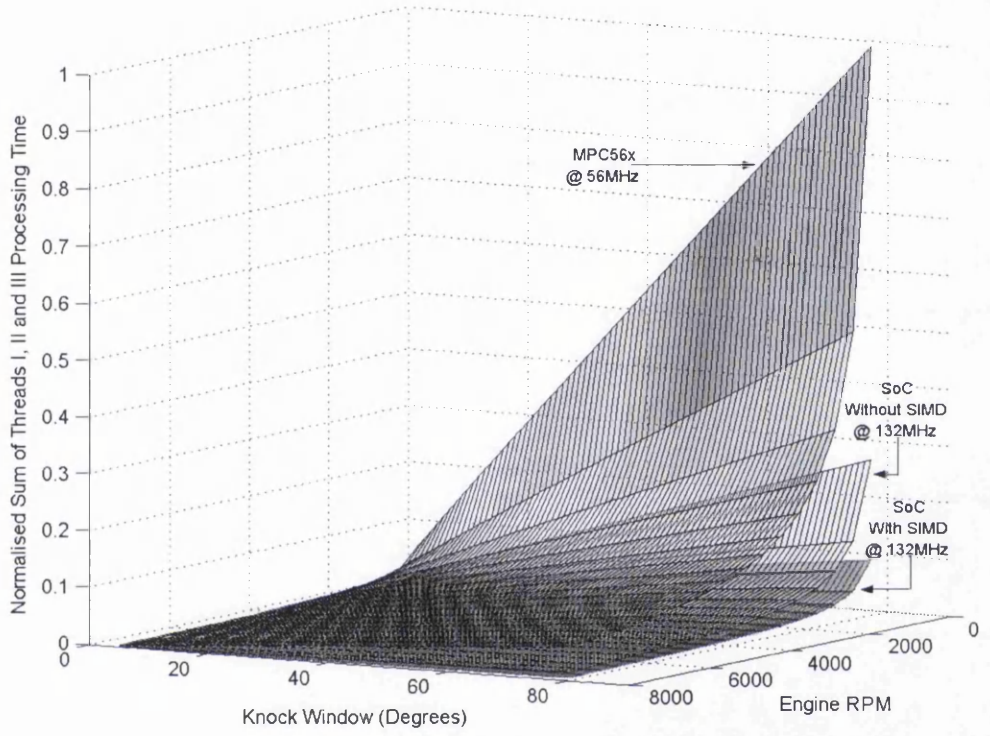

Figure 1: Measured MOSK Knock Kernel Performance

2.2 BMW's Knock Processing

During the Copperhead knock demonstration tour in Europe, BMW indicated that they use a 77^{th} order filter (still to be confirmed if it is floating or fixed point, although it is likely to be fixed point based on some literature) in their knock detection strategy. Based on the performance figures obtained from the AME for z650n3 with the instruction and the data cached, the SPE knock kernel developed internally requires < 2.6% of the combustion event timing bandwidth for a 40° knock window.

Therefore, for a 77^{th} order fixed-point filter, using the SPE, the bandwidth utilisation would be < 10% with a 40° knock window. As per customer information, bandwidth of < 10% is still required for their kernel with higher order filters, possibly up to orders 133 and an increased knock window of 70°. BMW have also indicated that a floating point SPFPMAC with performance similar to that of present fixed point MAC would be a definite requirement for the next generation.

2.3 Delphi's Knock Processing

This is a collection of powertrain algorithms, predominantly encapsulating DSP components.

2.4 BMW Electromechanical Valvetrain Control (EMVT)

This is a collection of valve control algorithms which fits in the 32kB unified cache. This can be used to test the raw performance of the core itself without memory domain delays.

2.5 Denso Powertrain Control

This is a collection of automatically generated code from Simulink for powertrain control.

2.6 GM Knock Processing

Based on discussions with GM, their next generation knock detection strategy would have a bank of FIR filters in parallel (possibly four) with centre frequencies of 8, 16, 24 and 32kHz. GM also prefers to utilise <10% of the combustion event timing for knock algorithmic execution. Initial indication is that they would be using knock windows of up to 50° and filters of order > 40 per filter. In addition, they are also investigating the possibilities of implementing knock with FFT.

2.7 Amroth (Mix of DCX Applications)

AME Profiled information already available in the repository. Fixed point customer code of size ~600kB.

2.8 PCOPS

This is a collection of representative application signatures which fits in the 32kB unified cache. This can be used to test the raw performance of the core itself without memory domain delays.

2.9 BMARK02

Reasonably sized (~95kB) complex customer algorithm.

2.10 ARWEN

This suite consists of integer DSP code.

2.11 Bosch's Central Chassis Controller

FAE is working on getting some indicative information on code signatures. However, this is expected to be very similar to standard chassis control code.

2.12 Visteon's Powertrain Development - Nimloth

Some similarities with to that of Denso's powertrain control code. Mixture of fixed and single precision floating point code.

2.13 Toyota's Powertrain Controller (MBC)

Same as Denso requirements.

2.14 Miscellaneous Applications

Advanced high performance motor control, including industrial applications and robotics, is especially considered to be a viable target for enhanced z6 cores. Brushless motors are gradually replacing commutation motors as they are smaller, less expensive and do not require as much maintenance. Moreover, brushless motors are much lower EMI than commutation motors. The math functions required for the precision control of motors are more applicable to the DSP functionalities embedded than the conventional microcontroller tasks.

3 Model Based Control (MBC) – An Introducton

Based on overall customer feedback, particularly Denso and Visteon, development throughout the powertrain supply chain is becoming bolder because fractional improvements no longer keep pace with legal requirements or with the desires of an increasingly well-informed auto enthusiast.

Vehicle manufacturers have long since shelved the idea of the five-year development cycle in favour of two or three. Several of the manufacturers in the Far East talk loudly about the 18 month development cycle, and in more hushed tones about cutting that back even further.

Denso and Visteon, as two of the largest tier one suppliers, have to bear the burden of its customers' ambition. These two tier ones are particularly well noted for its range of powertrain control products. Powertrain products, and in particular their calibration, is one of the areas where time has become most critical. The calibration effort rapidly increases as the number of control parameters increases. For previous engine designs, the number of parameters was small, whereas in the latest engines additional input factors such as injection timing, exhaust gas re-circulation rate (EGR), variable cam timing (VCT) and electromechanical camless valvetrain (EMCVT) take the number of parameters up to seven or eight.

The calibration process relies on the acquisition of data from the prototype engine on the test bed, but because the number of data points increases exponentially as new input parameters are added, the classical full factorial method of producing a calibration grid is no longer a feasible solution. Using this method it would take years to calibrate a prototype engine – and that time is simply not available. At best it would require a significant increase in the calibration effort at a time when a movement in the opposite direction is required in order to reduce costs and time to market.

Instead, prominently, Denso, Visteon and a few OEMs have been using The MathWorks' Model-Based Calibration Toolbox (MBCT) for the past few years. This new strategy, originally developed by Denso, calibrates the next generation engines by employing various piecewise mathematical models, with the understanding on what is going on in the engine followed by calibrating it using the model. This is all done using the MBCT.

The general concept of model-based design involves a front-loading of calibration efforts by using plant models and a HIL (hardware in the loop) system. As much simulation as possible is performed ahead of time – before expensive physical prototypes are built. Additionally, Denso has introduced automatic map (or look-up table) generation techniques by using the MBCT: in effect automatically generating the required calibrations.

Using these MBCT tools, DENSO selects experiments to run on the engines. Data is acquired from the engine on the test bed (using an automated operation and measuring system) and a model is built from it. The look-up tables that form the maps can then be filled using the models. Traditionally, simple linear interpolation is performed on the data points; with this new process more sophisticated models used, which generate more accurate calibrations.

If a model is not used, the airflow-manifold pressure relationship is measured at a number of points in each of the other variable directions, i.e. at a large grid of points. This is very expensive, because of the sheer amount of data that has to be collected. A huge number of tables are required for expressing the behaviour and it takes a long time to measure the huge data sets for the multiple tables. In addition, a large amount of ***costly storage space*** is required on the ECU.

The alternative is to build a model of manifold pressure from a smaller set of data points and then to use that model to understand its behaviour at all the other input points. This reduces the data acquisition effort (the number of data points measured) which reduces cost and time to market. The model can be validated by taking more measurements from the engine and seeing how well the model predicts the data at those points. By modelling the relationship, the inherent error that comes from measuring the data is removed. The accuracy of the model depends on the number of data points collected and whether it is flexible enough to pick up all the trends in the data without being adversely affected by noise (error).

Rather than just selecting any set of data points to be measured, a set of points is chosen that will provide the most useful information from which to build an accurate model. Using MBCT, the model built can then be viewed and evaluated in a number of visual and statistical ways, so that its accuracy can be assessed or a detailed analysis or investigation of its robustness can be carried out. It is then validated with data from the engine. Finally the model can be used to develop an ECU strategy.

Optimal design where the placement of prior knowledge on the shape of the response expected in order to see and then come up with the optimal points to collect data is, in some ways the ideal design to use. In this case Denso are looking at a new type of engine and did not know the shape of the response that they were expecting to see – so they have decided to go for the space filling design which reduces the number of points that need to be collected, losing the exponential dependence on the number of variables.

Traditionally on the test bed the calibrator would move around the control parameters and have a look at the data. You can only get so much of the picture if you are moving just one or two control parameters at a time. If you have three or more input parameters and are trying to find where the response is maximum, it is easier to build a model and work out mathematically where the maximum is rather than manually move test bed controls to find that response.

Application Description Requirements	Code Density	Execution Time on z6 @ 132MHz	Execution Time on RCPU @ 56MHz	DSP Performance and Functions Required		Special, Fast Embedded Math Functions	System Clock	Required Performance	Additional Information
				Integer MMACs	Floating Point MMACs				
Motorola MOSK SIMD Streaming Knock Processing	Similar to that of z6 footprint	110μs (Using MW Binary) - Currently Satisfies <10% BW	1.28ms (Scalarised SIMD algorithm)	≥ 600 sustained (Estimated)	> 600 sustained (Estimated)	SP Sqrt (Not much emphasis on DP)	TBD	> 2.01 x z6 @ 132MHz	Ideally, this code would be locked in cache. Performance required is based on a 101st order filter, dual point backward difference integrator and with a 70° knock window
BMW's EMVT	Similar to that of z6 footprint	140μs for 4 valves (Based on measured MGT5100 performance)	NA	NA	NA	SP Sqrt << 400ns (Measured 375ns @ 233MHz from MGT5100 cache)	TBD	> 2.83 x z6 @ 132MHz	The controller of this application is to be locked in the cache. Control loop time is 5μs
BMW's Knock Processing	Similar to that of z6 footprint	Does not Satisfy <10% BW	NA	≥ 600 sustained (Estimated)	> 600 sustained (Estimated)	None	TBD	> 2.75 x z6 @ 132MHz	Ideally, this code would be locked in cache. BMW have indicated filters of order up to 133 and an increased knock window of 70°. SP FP algorithm is being planned.
Amroth (Internal SRAM and External Flash)	< 65 – 70% of z6 footprint	AME Data Already Available	NA	AME Data Already Available	AME Data Already Available	AME Data Already Available	TBD	> 1.00 x z6 @ 132MHz	Complete set of AME profiled data is available with various scenarios
Amroth (Internal SRAM and Internal Flash)	< 65 – 70% of z6 footprint	AME Data Already Available	NA	AME Data Already Available	AME Data Already Available	AME Data Already Available	TBD	> 1.70 x z6 @ 132MHz	
BMARK02 (Internal SRAM and External Flash)	< 65 – 70% of z6 footprint (TBC)	TBD	TBD	TBD	TBD	TBD	TBD	> 1.00 x z6 @ 132MHz	Complete set of AME profiled data is available with various scenarios
BMARK02 (Internal SRAM and Internal Flash)	< 65 – 70% of z6 footprint (TBC)	TBD	TBD	TBD	TBD	TBD	TBD	> 1.70 x z6 @ 132MHz	
Denso's MBC Applications	Ideally 75 – 80% of z6 footprint (TBC by customer)	See Table II	NA	≥ 600 sustained (Estimated)	> 600 sustained (Estimated)	Fast Sqrt (SP)	TBD	See §4 for more details	Some Cache Locking required
ARWEN	Similar to that of z6 footprint	SIMD Code. Can be directly compared with knock kernel developed internally.						> 1.00 x z6 @ 132MHz	Knock detect function with a 32 tap FIR on 100 samples of data. Runs on an 8 cylinder engine @ 8000rpm, i.e. once every 1.875 ms
256 Point Radix 2 Complex FFT	Similar to that of z6 footprint	10000 cycles approximately	NA	≥ 800 sustained (Estimated)	> 600 sustained (Estimated)	NA	TBD	> 2.30 x z6 @ 132MHz	Ideally the twiddle factors would be loaded and locked in the cache
Single Precision Squareroot	Similar to that of z6 footprint	60-70 cycles using assembly SPE instructions	NA	NA	NA	NA	TBD	> 1.80 x z6 @ 132MHz	Preferably Newton-Raphson
Double Precision Squareroot	Similar to that of z6 footprint	1100 cycles using assembly SPE instructions	NA	NA	NA	NA	TBD	> 2.33 x z6 @ 132MHz	Preferably Newton-Raphson
Bosch's Central CC	< 60 – 65% of z6 footprint	Representative code was used to analyse this on the ARMulator and AME						15 x ARM7 @ 20MHz or 3.10 x z6 (Estimated)	More profiling of customer specific code signatures would be useful for solidification of performance
Visteon Engine Control	< 65 – 70% of z6 footprint	Discussions are still ongoing with FAEs to solidify this performance factor. But it should be in the range of Denso requirement as Visteon are actively involved with MBC.						Similar to that of Denso's MBC applications	Some cache locking required
Mitsubishi Engine Control	Ideally 60 – 65% of z6 footprint (To be solidified by customer)	Happy with present roadmap. Price driven customer unlike Denso.						~1 x z6 @132MHz	Not much emphasis on performance. Happy with 80MHz Moccasin. Price driven customer.
GM Knock Processing	Similar to that of z6 footprint	One of the applications presently being looked into directly maps onto the streaming knock algorithm mentioned above. Initial discussions suggests need for floating point MAC with same performance levels as mentioned for streaming knock above.						> 2.71 x z6 @ 132MHz	Implementation with 4 parallel 33rd order SIMD FIRs and 40° knock window. Bandwidth expectation is ≤ 10% of combustion event timing
Multi-Point EFI									
GDI									
DDI									
CVT									
ISA									
Cruise Control									
Traction Control									
Suspension Control									
Neutral Idle									
Performance Activated Shift									
Electronic Throttle Control									
Cylinder Deactivation									
ABS					x				

Table 1: Summary of Performance Requirements

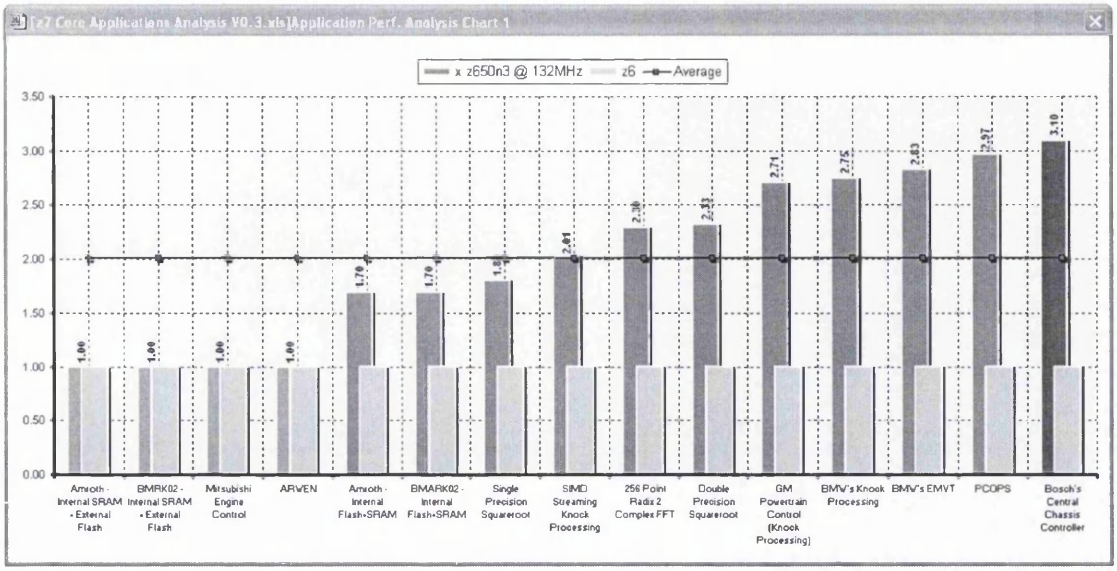

Figure 2: Overall Performance Required

4 DENSO's Performance Requirement Profiling

MBC based code signatures generated by Denso were analysed and profiled to obtain the following results. All linked binaries of the code signatures used fit in the cache and, in addition to the standard BTAC mode enabled, following z6 configuration was used for AME profiling.

```
// Configuration Resources
Config           = zen::z6      // Configuration keyword and name
// ------------------ Memory Definitions ---------------------
MemAccessCost    = 2            // Memory Access Time in clock ticks
MemBusWidth      = 8            // Memory data path width in bytes
// ------------------ Zen Model Parameters -------------------
ZenUnit          = On           // Zen unit - on, off
ZenArchType      = Ip7          // Zen Architecture - Iss,IssBt,Ip5,Ip7
ZenBusArch       = Unified      // Unified, Harvard (Unified sets iu=du)
ZenCpuIFSchSize  = 8            // Instruction fetch scheduling size in bytes
ZenCpuBdtSchSize = 8            // Block-data-transfer scheduling size in bytes
ZenCpuWriteBackLimit = 2        // Number of results that can be written per cycle
ZenCpuDBusWidth  = 8            // Data-side bus width in bytes
ZenCpuDBusPipelineDepth = 1     // Data Bus Pipeline Depth
ZenCpuIBuffSlots = 6            // Number of instruction prefetch buffer slots
ZenCpuIBuffIssue = 1            // Number of instructions per clock to issue
ZenCpuIBuffCofDecSlotMap = 0x0 // Cof pre-decode 0x0=Off,...,0x7=slot2_1_0
ZenCpuIBuffCcLookAhead = Off    // Condition-Code look-ahead for Bcc
ZenStats         = On           // Statistic counters
ZenPorAssertion  = 3            // Number of clocks for power-up reset to assert
ZenCpuIBuffInst2Fold = none     // Instruction to fold
```

Application Signature	Compilers with Highest Speed Optimisation Criterion Enabled	**Platform I (μs)**	**Platform II (μs)**	
		z650n3 132MHz	**Dual Issue z650n3**	
			200MHz	**266MHz**
I	Target	**80**	**50**	**50**
	GHS *v4.0.1*	385.10	223.00	167.67
	MW *v1.5* 55xx	384.67	231.22	173.85
II	Target	**20**	**10**	**10**
	GHS *v4.0.1*	15.61	8.86	6.66
	MW *v1.5* 55xx	19.78	11.78	8.85
III	Target	**300**	**200**	**200**
	GHS *v4.0.1*	744.40	406.01	305.27
	MW *v1.5* 55xx	657.23	378.58	284.64

Table 2: Performance Details of Denso's Benchmark

For all three targets, although the link binaries generated are cache friendly, various compiler optimisation criteria had to be enabled in the speed optimised mode to achieve the highest performance. Excerpts of critical segments of code signatures is shown below:

```
/* Scaling loop - single precision floating point*/
for(iL = 0; iL < ANALYSYS_LEVELS; iL++)
{
     /* Time-shift loop */
     for(iT = 0; iT < TARGET_DATA_NUM; iT++)
     {
       /* Convolution unit */
       tmp = 0.0;
       tmp2 = 0.0;
       for(iK = 0; iK < KERNEL_SIZE; iK++)
       {
         tmp += tmp2;
         tmp2 = Kernel[iK] * InData[iT + iK];
       }
       tmp += tmp2;

       /* Pattern Matching */
       sum += tmp * PatternMap[iT][iL];
     }
   }

   Result_Data = sum;
}
```

Table 3: Denso Time Shift and Convolution Unit

```
float A0_001, A0_002, A0_003, A0_004, A0_005, A0_006, A0_007;
float A0_008, A0_009, A0_010, A0_011, A0_012, A0_013, A0_014;
float A0_015, A0_016, A0_017, A0_018, A0_019, A0_020, A0_021;
float A0_022, A0_023, A0_024, A0_025, A0_026, A0_027, A0_028;

float A1_001, A1_002, A1_003, A1_004, A1_005, A1_006, A1_007;
float A1_008, A1_009, A1_010, A1_011, A1_012, A1_013, A1_014;
float A1_015, A1_016, A1_017, A1_018, A1_019, A1_020, A1_021;
float A1_022, A1_023, A1_024, A1_025, A1_026, A1_027, A1_028;

a0 =
   A0_001 - A0_002*A + A0_003*V + A0_004*E + A0_005*nn - A0_006*na -
   A0_007*ns - A0_008*nv - A0_009*ne + A0_010*aa - A0_011*av +
   A0_012*ae - A0_013*ss - A0_014*sv - A0_015*vv + A0_016*ve -
   A0_017*N*na - A0_018*N*ns - A0_019*N*nv + A0_020*N*aa -
   A0_021*N*as - A0_022*N*av + A0_023*N*ss + A0_024*N*sv -
   A0_025*N*vv + A0_026*N*ve + A0_027*A*aa - A0_028*A*as ;
a1 =
   A1_001*A*ss - A1_002*A*sv - A1_003*A*se + A1_004*A*ve +
   A1_005*A*ee + A1_006*S*ss - A1_007*S*sv + A1_008*S*se +
   A1_009*S*ve + A1_010*V*ve - A1_011*E*ee - A1_012*nn*nn +
   A1_013*nn*ns + A1_014*nn*ne - A1_015*nn*aa - A1_016*nn*ss -
   A1_017*nn*sv - A1_018*nn*se - A1_019*nn*ve - A1_020*nn*ee -
   A1_021*na*as - A1_022*na*av - A1_023*na*ae + A1_024*na*ss -
   A1_025*na*vv + A1_026*na*ee + A1_027*ns*ss - A1_028*ns*se ;
a2 =
   A2_001*ns*vv - A2_002*ns*ee + A2_003*nv*vv + A2_004*nv*ve -
   A2_005*ne*ee - A2_006*aa*aa + A2_007*aa*av + A2_008*aa*ae -
   A2_009*aa*ss + A2_010*aa*sv - A2_011*aa*se - A2_012*aa*vv -
   A2_013*as*ss - A2_014*as*sv - A2_015*as*se - A2_016*as*ee -
   A2_017*av*vv - A2_018*av*ve - A2_019*ae*ee + A2_020*ss*ss +
   A2_021*ss*sv + A2_022*ss*se + A2_023*ss*vv + A2_024*ss*ve +
   A2_025*ss*ee + A2_026*sv*ee + A2_027*se*ee - A2_028*vv*vv -
   A2_029*vv*ve + A2_030*vv*ee - A2_031*ee*ee ;
a_total = a0 + a1 + a2;
```

Table 4: Denso Vectoriseable Module

Based on the analysis given in Table 2, it is evident that platforms I and II satisfy the requirements of only target II. However, the performance of targets I and II is far from satisfactory. It is also blatantly obvious that a single precision floating point MAC and a compiler capable of vectorisation would certainly have achieved better performance on the modules shown in Table 3 and Table 4. Note that Platform II's profiling is done with a dual issue machine with the rest of the configuration similar to that of z650n3. See conclusion for more comments and suggestions. Some additional performance enhancements achieved are listed below:

Metrowerks generated Denso binary for the third application signature improves performance by approximately 2.5% by changing the BTAC to a BTIC with 8 entries. No change was observed in the performance by varying the number of BTIC entries to be greater than 8.

Metrowerks generated Denso binary for the first application signature improves performance by approximately 1.3% by changing the BTAC to a BTIC with 8 entries. In the BTIC mode, an improvement of 2% was observed when the entries were increased up to 16. Increasing the BTIC entries to 32 yields 10% performance improvement.

Therefore, in summary, for all three Denso signatures, there is a lot of room for performance improvement by vectorising the code, which needs to be supported in our tool flow. Better scheduling mechanisms also need to be introduced in order to exploit the architectural benefits of z6 and its enhanced APUs.

The above analysis on the branching mechanism confirms that there is also a lot of room for improvement by varying various characteristics of the branch acceleration unit.

5 Summary of Analysis and Conclusions

It is evident that aforementioned applications require various enhancements to the performance that is already available on the z650n3 core. The features listed in this §5.1 are required to primarily performance accelerate fixed and single precision floating point number crunching algorithms including DSP and other bandwidth intensive applications *without sacrificing a clean, intuitive programming model.* Development tools issues aside, most competitors have built-in features that powertrain engineers have wanted in microcontrollers for years.

5.1 Key Features Required

- < ~70% of z6 code density to meet customer requirements and to particularly beat competition
- An enhanced BTAC / BTIC APU supporting speculative execution and branch prediction.
- Reasonably low capacity, on chip tightly coupled SRAM memory, aka scratch pad memory is required to place and manipulate frequently used data. This memory will be used to access frequently used data, create circular buffers which are often used for inter-thread communications as developed in the SIMD based streaming knock processing methodology.
- Multiple fast hardware SPFPMAC operations per cycle
- Modulo addressing mode supporting circular buffer management
- Misaligned load store support in SIMD
- Fast hardware math functions, particularly single precision squareroot
- Enhanced MMU granularity
- 100% binary compatibility with z6
- Zero-overhead looping mechanism supporting a minimum of 4K iterations
  Most powertrain superscalar architectures including competition consist of three pipelines: an integer pipeline, a load/store pipeline and a zero-overhead loop pipeline. As shown in Figure 3, the zero-overhead loop hardware keeps track of the looping which combined with the other two units, allows execution of up to 3 instructions in parallel.

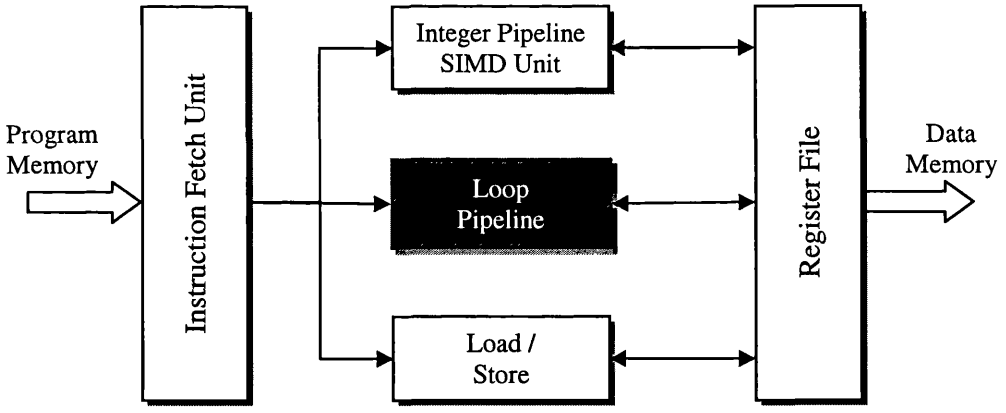

Figure 3: Present Superscalar Pipeline Architecture

- Bit reversing hardware (already implemented on z6)
- Barrel-shifters (already implemented on z6)
- Branch predication in order to avoid misprediction (wider issue suits this mechanism)
- Hardware cache coherency mechanism
- Most of all it would be nice to have a super compiler to vectorise and exploit full potential of SIMD
- Multithreading and Multiprocessing – If the required performance can be met with wider issue, this would obviously be the successor to z6

5.2 Unified MCU and DSP Functionality

Based on customer requirements and present MCU based powertrain application trends, the developers involved would rather stick with a known entity; that is, the classis MCU. It is evident that the DSP and specialist computation intensive tasks required establishes the need for an SoC with traditional control tasks in addition to providing enough performance to satisfy the signal processing requirements, and it is obvious that the developers would favour this solution.

If the two critical issues of price/performance and development tool quality are met, the microcontroller with DSP capability would be the first choice for the application, especially with the engineers who are already familiar with the development tools and/or the microcontroller. Applications being developed presently necessitate higher than z650n3's clock speeds in order to brute-force appropriate tasks while still maintaining real-time performance.

Systems that require only basic control to determine which path of data flow the program will take are simple and commonly implemented in many digital signal processing systems. All that is required is efficient implementation of program control routines (conditional jumps and test instructions) and an I/O port for control and detection of external events.

5.3 Why not Dual Core or a Dedicated DSP for Number Crunching?

To many powertrain applications developers, DSP processing in a microcontroller is an unfamiliar entity and they prefer the present way forward for adding tightly coupled DSP functionality as opposed to having a hybrid. It is evident that dual-core implementation of a microcontroller, i.e. e separate DSP on the same die is a less than optimal development solution. Dual-core architectures can suffer from the vagaries of inter-processor communications and pipeline interlocks. As to development tools, it is difficult using conventional debugging techniques to simulate such an architecture, to say nothing of the cost of a special in-circuit emulator.

The step we have taken with the z6+ platforms, i.e. the simplicity of merging the microcontroller and DSP architectures into a single instruction stream seems the natural way forward. By implementing parallel execution units, high clock speeds, and glueless interfaces for common memories, deterministic real-time signal processing performance for DSP can exist simultaneously with the multilevel interrupt hierarchy and rapid context switching required for the controller needs of the system.

5.4 What's Next?

Barriers with some of the customers depend on configuration of the peripheral set, ability to move multiple pieces of data, and quality of the development tools. Today's powertrain applications developers requiring both real-time control as well as analogue signal processing and are looking for ways to reduce cost and speed development time in a market that becoming increasingly competitive. Both technical and personal considerations rule. For a microcontroller engineer to consider a DSP or for a DSP engineer to use a microcontroller three strict criteria must be met.

- Price versus performance
- Peripheral set
- Development tool quality

The absolute convergence of these two architectures are partly implemented by z6. However, market intelligence suggests that most powertrain controllers presently being offered by our competitors consist of hardware and instruction set enhancements designed to increase the execution speed of computations associated with signal processing tasks as listed in §5.1.

As in z6, the commonest addition is the single-instruction, multiple-data (SIMD) unit. It is estimated that the performance improvement achievable by implementing the Denso's application in SIMD along with addressing alignment issues would be in the region of 1.8x.

Penetration of the embedded-code technology, based on auto coding tools such as Matlab and Simulink, Targetlink are being used to develop powertrain Electronic Control Units (ECUs). Visteon recently described their progress in advancing efficient automatic code generation for production applications. Highlights from the presentation included smaller code size through autocoding and the ability to leverage a growing library of reusable models.

Production code generation is a key component for companies using a model-based design approach and we need a solid strategy to meet performance requirements needed to support the applications being developed using MBT. It is also required to support fast embedded library functions or preferably hardware implementation of such functions to remain competitive in the powertrain controller market.

It is also evident that, after the z6, superscalar cores would be the natural way forward although much emphasis is being laid in the industry for multithreading and multiprocessor support.

An Architectural Modelling Environment for the PowerPC System-on-Chip

Mohamed Anas

A portfolio submitted to

The Universities of

Edinburgh

Glasgow

Heriot Watt

Strathclyde

for the Degree of

Doctor of Engineering in System Level Integration

March 2005

Abstract

The Architectural Modelling Environment discussed in this portfolio targets the middle ground simulation requirements of performance evaluation by recognising that typical application binaries do not require a fully functional model to run and that only a vital few key system-on-chip core architectural elements determine a systems performance characteristics. Typical benchmark code does not target a specific architecture which allows the benchmark to run across many different system implementations. Therefore, most detailed 'nooks-and-crannies' of a fully functional simulation model are never used due to the generalised nature of benchmarks. Development time devoted to functional features which are never used by a benchmark is thus wasted effort in regards to performance evaluation. Similarly, most of the development time associated with creating a very precise, cycle accurate model is lost since a system's performance is generally controlled by a very few core architectural elements or bottlenecks.

Application binaries profiled on the architectural modelling environment include but are not limited to performance demanding powertrain applications. Optimised assembly libraries and compiler intrinsics are used to create the single-instruction-multiple-data binaries. As discussed in the enclosed document *"Next Generation Powertrain SoC Performance Requirements"*, the applications binaries used contain large amounts of available parallelism; however, most of it is inter-iteration parallelism.

In an ideal world, there would be one system-on-chip core simulation model, or at least one unified model database, that contains all the information required to perform any type of desired simulation.

Table of Contents

List of Figures and Tables

1 Introduction

Over the years, simulation models have migrated to either fully functional models or to cycle accurate models [1, 2, 3, 4, 5, 6]. Fully functional models are characterised by modelling system features without regard to hardware timing. Functional models are typically preferred by software developers since they tend to be the highest performance models available for developing application code. The drawback is that they offer very little information for performance evaluation. On the other hand, cycle accurate models provide very precise behaviour in regards to timing but tend to be relatively slow in performance due to the extreme detailed nature of the model [6, 7, 8].

The area of performance evaluation is in a middle ground – requiring the simulation performance of a functional model but needing the information provided by a cycle accurate model. Ideally, performance evaluation studies would like to simulate proposed system architectures over a wide range of benchmarks to determine system level performance – then, adjust the system architecture and re-evaluate the performance.

The Architectural Modelling Environment (AME) developed allows both fully functional and cycle accurate models to be developed. The intent of the AME is to target the system-on-chip (SoC) core complex specific requirements for performance evaluation. The current area of study that AME is being used for is in architectural trade-off studies at the processor, cache, and memory interface level of system design – or what is being called the core-complex for embedded, SoC designs.

2 An Overview of the AME

The AME is a third generation modelling environment written in the C, C++, UNIX shell scripts and using object oriented programming techniques to build hybrid simulation models containing both functional and cycle accurate model characteristics.

2.1 The Realm of Performance Evaluation

Simulation models used for performance evaluation provide the ability to determine how a system will perform before committing resources to implement and verify a particular SoC design. Being able to validate concepts for SoC core designs provides an ability to explore various implementations allowing optimisation of key architectural elements. Performance evaluation is, therefore, not a point solution to a particular design but must provide an environment in which models can be built to test design thoughts. Indeed, it is often the case that after a successful simulation, there are more questions raised than answered, requiring modifications, additions, or extensions to architectural and system simulation models to allow validation of new design concepts.

The intent of AME is to tune the simulation models and model development process to the unique requirements of performance evaluation while keeping the resources devoted to AME model development to a minimum. AME already includes a number of models to explore SoC implementations and provides an environment to create and develop new models or to extend existing ones. In addition, AME is specifically geared toward studies around architectural trade-offs at the processor and SoC levels. AME has been used successfully to profile dynamic instruction execution, collect statistics on instruction usage and instruction sequences, compare various instruction pipeline sequence units, study the efficiencies of bus structures, and to analyse memory configurations.

AME models are typically characterised by being very flexible, providing a large number of run-time parameters which change the behaviour of processor and system elements. By varying the run-time parameters, an AME model can then be run several times using benchmark code or specific application binary to determine performance trade-offs.

2.1.1 A Generic Performance Evaluation Environment

The current trend in simulation models is to provide a functional model and a cycle accurate model. Figure 2.1 overleaf shows a typical simulation performance spectrum of available simulation models for processors. The functional models are typified by the Instruction Set Simulators (ISS) and perform simulations at the instruction level with a typical simulation throughput in the megahertz range. Cycle accurate, or cycle precise, models perform simulations at the clock cycle granularity and tend to have a throughput in the low kilohertz range. AME models are built to work in the simulation gap between these two extremes. In addition, AME simulation performance is regulated by the amount of detail enabled in an AME model. If only an ISS level of detail is required, then an AME simulator will achieve the same level of performance as an ISS. As more and more details are enabled in an AME model, the slower the performance will be for an AME simulator. The important concept is that for an AME simulator, this sliding level of performance is dynamic and can be adjusted by the user at run-time by adjusting the amount of detailed information which is enabled. To achieve a high degree of flexibility and reuse, typical AME models do not provide a detailed description of a specific hardware implementation. Instead, AME models are an abstraction of hardware concepts which allows parameter assignments at run time to configure the model environment to measure performance of various implementations over a series of simulations.

The intent of AME is not to model every detailed aspect of the design but only key design concepts tuned to running application binaries. As such, typical AME models do not have the design details required for verification but do provide performance feedback in regards to architectural trade-offs around key system design constraints.

Architectural features selected for AME are modelled as accurately as possible and with as little detail as possible – but not too simple. Architectural elements which are selected for modelling are based on the requirements of the application binaries selected for performance evaluation.

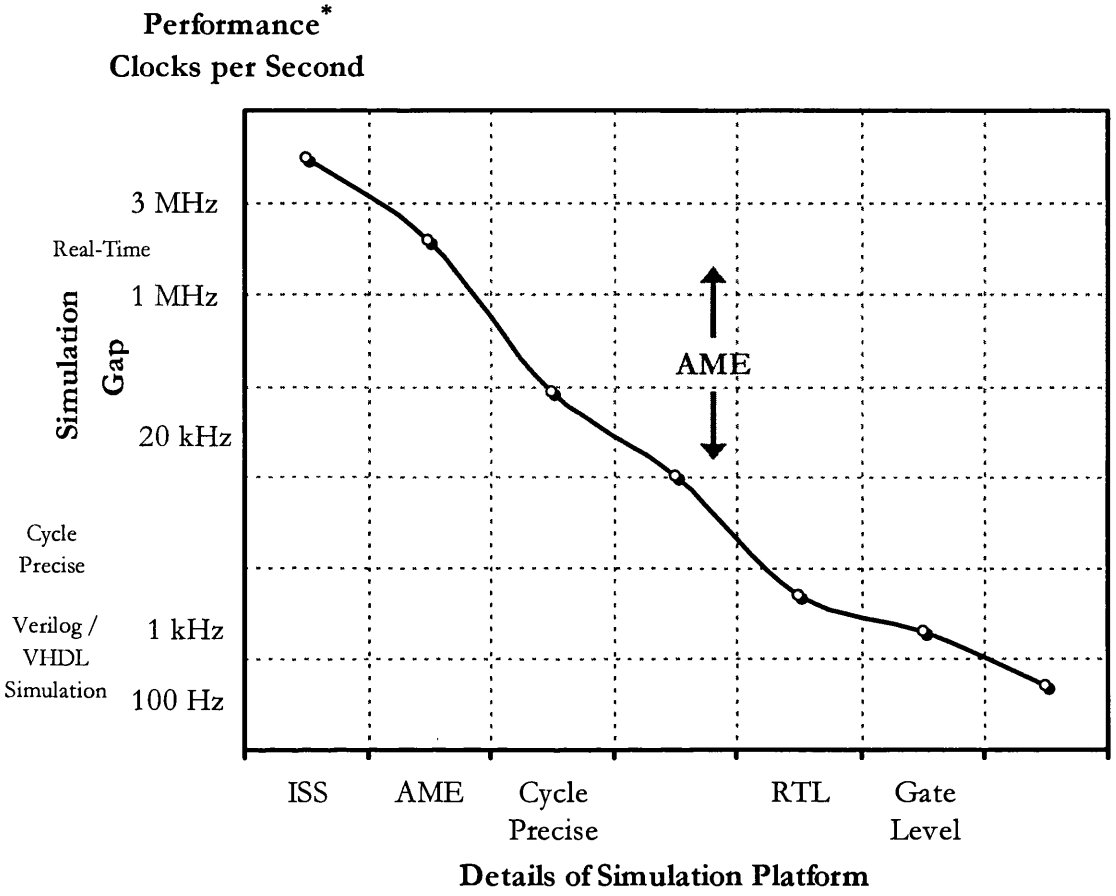

Figure 2.1: Simulation Performance[1] Spectrum

Since there tends to be a complex interaction between architectural features, it is not enough to only model one aspect of the design. An example of an AME simulation may include, running a 7-stage instruction pipeline with instruction pre-fetch buffering using 9-1-1-1 flash memory in both a Harvard and Unified bus configuration.

[1] Performance typical of Sun Workstation – 266 MHz Sparc Ultra-30

As shown in Figure 2.2 overleaf, in order to efficiently performance profile an application using the AME, the modules around it require the designer to work in two fields; on the one hand, the development of the software part including compiler, assembler, linker, and simulator and, on the other hand, the development of the target architecture itself. The AME produces the characteristics of the core architecture specific application binary performance and, thus, may answer questions concerning the instruction set, the performance of an algorithm, and the required size of memory and registers. The required silicon area or power consumption can only be determined in conjunction with a synthesisable hardware description language (HDL) model.

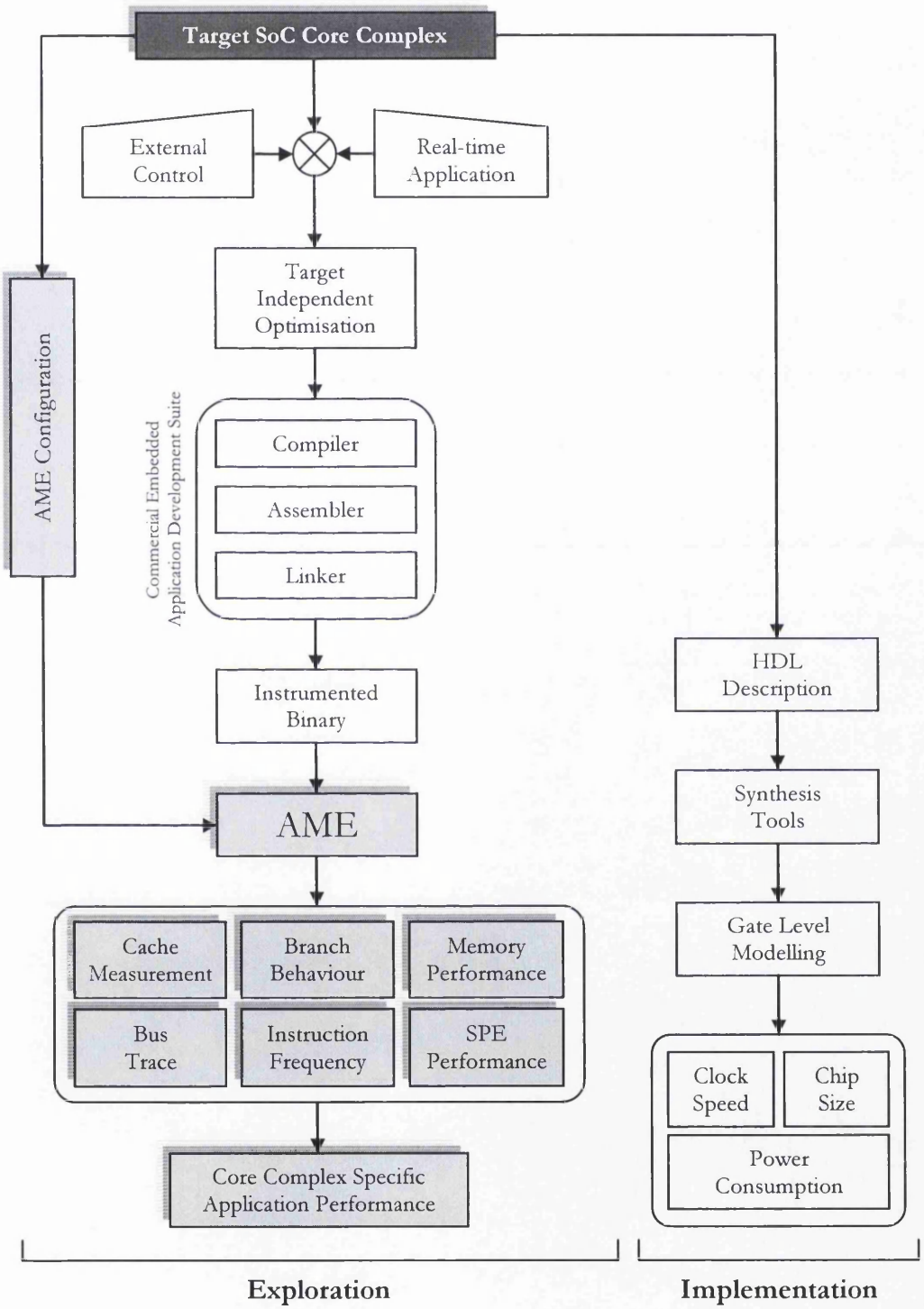

Figure 2.2: Application Binary Characterisation Process Using the AME

2.1.2 What Platform Does AME Run On

The current development for AME is being done using the UNIX operating system on a Sun Workstation running Sun Solaris (SunOS 5.7). There are currently no plans to port AME to other platforms. However, the predecessor package to AME was available on Sun Workstations, HP Workstations running HP-UX 10, and PCs running Cygwin 1.1.0, so it may be possible to port AME to other platforms with some effort – there will be some porting issues – there always are.

2.1.3 What Compiler is needed to Develop Models for AME

The current AME software is being developed using the Gnu compiler version 2.95.2. There are currently no plans to use any other compiler. Migrating to another compiler will probably require some work, even just upgrading to the latest Gnu compiler will probably require some modifications. The AME software package is available as either a precompiled binary for the Sun Solaris (SunOS 5.7) platform or in source code form.

2.1.4 AME Features

The following is a list of some of the key features of AME.

- Execution driven model supporting
  - Executable and Linking binary Format (ELF)
  - UNIX a.out binary format
  - Verilog memory image format (vmem)
- Support for modelling multiple clock domains
- Command line interface easily connects to UNIX scripts to automate the process of data collection across simulation runs of benchmark suites.

- Built-in debug monitor supporting
  - Instruction execution control: run, stop, and single-step
  - Software breakpoints, memory modify and display
  - Symbolic entry of address information for breakpoint and memory commands
  - Disassembly of memory and dynamic code execution
  - Dynamic loading of debug command set allowing new AME models to contain new debug commands.
- Profiling, Logging, and Statistical commands supporting
  - Capture of dynamic instruction flow, collection of instruction usage
  - Details of instruction-to-instruction dependencies.
- Support for multiple Instruction Set Architectures (ISA)
- Flexible Memory model supporting
  - Memory latency definitions on a per memory block basis
  - Flash memory interfaces, ROM, and RAM definitions
  - Dynamic allocation of memory based on benchmark executable
- Bus Structures
  - Harvard versus Unified
  - Bus width
  - Bus pipelining
- Instruction Pipelines
  - Pipeline depth
  - Instruction pre-fetch buffering
  - Instruction-to-instruction dependencies
  - Branch look-ahead support
- Cache configurations supporting
  - Harvard versus Unified
  - Cache size, organisation, and associativity

2.2 Getting Started

The AME simulator uses object oriented design principles to support cycle based models of various processor cores and core-complexes. The simulator is capable of supporting an assortment of on-chip and off-chip peripherals and memories. This section provides a quick-start reference to build the AME simulator and run some example code.

2.2.1 Building the Simulator

The AME Simulator is built using the Gnu compiler (Version 2.95.2). The simulator is packaged using the UNIX tape-archive (tar) format and the Gnu compress file (gzip) utility. The simulator package comes with source code and a precompiled binary executable for the Sun Solaris (SunOS 5.7) platform.

The simulator package is delivered as a compressed archive where the name of the archive is of the form ame-{version}.tgz. The {version} tag is used to track information between releases of the AME simulator package. All of the files contained in the ame-{version}.tgz archive are expanded into a subdirectory called ame-{version}. The AME simulator can then be built using the make command. The following example expands the ame-3.0.0.tgz simulator files and uses the make command to build the AME simulator executable (sim-ame).

```
[] tar -zxvf ame-3.0.0.tar
[] cd ame-3.0.0/src
[] make
```

The AME simulator can be started by invoking the sim-ame executable. If the verbose (-v) command line option is used, the AME simulator outputs some header information reflecting the AME simulator version. Entering the quit ('q') command terminates the AME simulator and returns to the unix prompt. The following is an example of starting the simulator under the Sun Solaris environment.

```
[] sim-ame -v
Architectural Modeling Environment - Version 3.0.0
Compiled on: Wed Jan 16 19:23:13 CST 2002
with: gcc version 2.95.2
under: SunOS 5.7 Generic_106541-16 sun4u
ame> q
[]
```

2.2.2 Running the Simulator

The simulator works with a number of command line options or interactively through the built-in debugger. A configuration file tells the AME simulator which models are to be used in the simulation. Included in the ame-{version}directory are a number of example configuration files located in the configsub-directory. As an example, the following hello.cprogram will be used which simply prints "Hello, World" to standard-out and then terminates.

```
/*************************/
/* hello.c program */
#include <stdio.h>
int
main()
{
printf("Hello, World\n");
return 0;
}
/*************************/
```

The following example compiles the hello.c program for the SoC core architecture, loads the compiled hello ELF binary into the AME simulator configured, simulates the program, and then terminates the AME simulator. Notice that the AME simulator provides the required system interface calls emitted by the compiler to support file io operations to standard-out.

```
[] ppc-elf-gcc -msim hello.c -o hello
[] sim-ame config/selected_core_config hello
Hello, World
program exit(0)
Sim.  Loop Time: 0.000 Seconds; Clock Ticks: 2045; CPS 0.000
[]
```

Additionally, the AME simulator uses a number of command line options and a simulator configuration file to setup the simulation environment. In addition, memory configurations may be setup by using a memory configuration file. There are also a number of commands which the debugger supports.

2.3 Simulator Configuration File

The simulator avoids hard-coding programmable options by allowing users to specify resource options through the use of a simulator configuration file (simcfg). The simcfg file must begin with the keyword 'Config' for the simulator to recognise the file as a configuration file. Configuration parameters are read until either end-of-file or the keyword 'ConfigEnd' is encountered. Comments are allowed in the simcfg file and are denoted by either '//' for line comments or by the block comment delimiter pair '/*' and '*/'. To get a full listing of the current resource options, use the -D command line option. For example:

```
[] sim-ame -D
```

Options supplied on the command line take precedence over resource options listed in the simcfg file.

3 SoC Core Configuration and Profiling

3.1 Core Configuration Parameters

The following is a list of parameters that control the operation of the SoC core simulation model.

- `CoreUnit = [On,Off] // enable or disable core complex`
- `CoreArchType = [Iss, Ip5, Ip7] // Core Architecture`

The parameter controls the instruction execution unit of the SoC core model. ISS - puts the core into the instruction set simulation mode which is the highest performance mode. While in the ISS mode, there is no timing information available, as such, the cpu model will execute 1 instruction every time through the AME simulator timing loop – this will result in a 1.00 cpi (clocks per instruction). Also note that while in the ISS mode, the core execution unit is tied directly to the memory unit. As a result, there are no bus transactions generated and, therefore, simulation models that rely on bus traffic will not record any events. One of these simulation models is the cache model. Ip5 - enables the 5-stage instruction pipeline mode. Ip7 - enables the 7-stage instruction pipeline mode.

- `CoreBusArch = [Unified,Harvard] // Unified, Harvard`

The CoreBusArch parameter controls the bus structure of the cpu. Unified - both instruction fetches and data load-store traffic communicate over a single bus connection to downstream models such as cache and memory. All bus traffic is tagged with du (data unit) to indicate that a single bus is being used. Harvard - instruction fetches and data load-store traffic each have their own bus connection to downstream models. This allows an instruction fetch and data transaction to occur in the same clock cycle. Bus traffic is tagged with iu (instruction unit) for instruction fetches, and with du (data unit) for load-store bus transactions.

- `SoCCpuIFSchSize = num // Instruction fetch scheduling`

The SoCCpuIFSchSize parameter controls the size of the bus transactions being scheduled for instruction prefetch operations. The size of the bus transaction is scheduled using num bytes per bus transaction. If num is less than the width of an instruction, then multiple bus transactions will be scheduled to complete the fetching of a whole instruction.

- `SoCCpuBdtSchSize = num // Block-data-transfer scheduling`

The SoCCpuBdtSchSize parameter controls the size of the bus transactions scheduled for load and store multiple operations. A num size of 4 allows the core to transfer one register at a time over the bus. A num size of 8 allows the cpu to schedule the transfer of 2 registers at a time.

- `SoCCpuWriteBackLimit = num // Number of results that can be written per cycle`

The SoCCpuWriteBackLimit parameter defines the number of update operations that can be performed into the general purpose registers during a single clock cycle. Normally, during single instruction execution, only 1 register update operation is performed. However, for multiple issue machines (SoCCpuIBuffIssue) or during folded instruction operations (SoCCpuIBuffInst2Fold) multiple write-back results per clock may be needed to complete the operation. If the number of write-back results is greater than the defined limit (num), then the instruction pipeline stalls until all the results have been saved.

- `SoCCpuLd2StData = [On,Off] // Ld-data to St-data feedforward path`

The SoCCpuLd2StData parameter, when enabled (On), allows the SoC cpu to schedule a back-to-back load instruction followed by a store instruction where the data being loaded is the same data that is being used by the store instruction without stalling the instruction pipeline. Otherwise, when disabled (Off), an extra clock is required.

- `SoCCpuIBusWidth = num // Instruction bus width in bytes`

The SoCCpuIBusWidth parameter defines the bus width of the instruction side bus. The width is specified in bytes (num). When a unified bus is specified by the SoCBusArch parameter, this parameter (SoCCpuIBusWidth) is not used. Instead, the SoCCpuDBusWidth parameter is used to define the bus width for both instruction and data transactions.

- `SoCCpuDBusWidth = num // Data-side, data bus width in bytes`

The SoCCpuDBusWidth parameter defines the bus width of the data side bus. The width is specified in bytes (num). When a unified bus is specified by the SoCBusArch parameter, then this parameter (SoCCpuDBusWidth) is used to define the bus width for both instruction and data transactions.

- `SoCCpuIBusPipelineDepth = num// Instruction Bus Pipeline Depth`

The SoCCpuIBusPipelineDepth parameter defines the number of outstanding bus transactions that can be pipelined on the instruction bus. When a unified bus is specified by the SoCBusArch parameter, this parameter (SoCCpuIBusPipelineDepth) is not used. Instead, the SoCCpuDBusPipelineDepth parameter is used to define the bus pipeline depth for both instruction and data transactions. A pipeline depth of 0 indicates that only one bus transaction can be outstanding at a time – a subsequent bus transaction must wait until the data from the previous transaction is received. A pipeline depth of 1 allows a subsequent bus transaction to be issued before data has been received from the previous bus transaction. Only one bus transaction will be issued per clock..

- `SoCCpuDBusPipelineDepth = num// Data Bus Pipeline Depth`

The SoCCpuDBusPipelineDepth parameter defines the number of outstanding bus transactions that can be pipelined on the data-side bus. When a unified bus is specified by the SoCBusArch parameter, then this parameter (SoCCpuDBusWidth) is used to define the bus pipeline depth for both instruction and data transactions. A pipeline depth of 0 indicates that only one bus transaction can be outstanding at a time – a subsequent bus transaction must wait until the data from the previous transaction is received. A pipeline depth of 1 allows a subsequent bus transaction to be issued before data has been received from the previous bus transaction. Only one bus transaction will be issued per clock.

- `SoCCpuIBuffSize = num // Number of instruction prefetch buffer slots`

The SoCCpuIBuffSize parameter defines the number of instructions (num) that can be held in the instruction prefetch buffer. If the bus parameters are setup to fetch multiple instructions per bus transaction, then the SoCCpuIBuffSize parameter should be defined to provide enough buffer space to prevent the cpu from having to re-fetch instructions that had to be discarded because there was no place to save them.

- `SoCCpuIBuffIssue = num // Number of instructions per clock to issue`

The SoCCpuIBuffIssue parameter defines the number of instructions (num) that can be issued into the execute phase during a single clock cycle. A single issue machine would set SoCCpuIBuffIssue to 1, while a dual issue machine would set this to 2.

- `SoCCpuIBuffBrDecSlotMap = hexnum// Branch pre-decode 0x0=Off,...,0x7=slot2_1_0`

The SoCCpuIBuffBrDecSlotMap parameter defines which instruction buffer slots, defined by the SoCCpuIBuffSize parameter, should pre-decode unconditional branch instructions (b, and ba). The hexnum provides a bit map of the buffer slots which should look for unconditional branch instructions. A hexnum value of 0x0 disables this option. A hexnum value of 0x1 defines instruction buffer slot 0 to pre-decode instructions looking for unconditional branches. If an unconditional branch is detected, then the instruction buffer is adjusted to start prefetching at the target address of the branch. To adjust the instruction prefetch buffer, all instructions in the buffer behind the detected unconditional branch are flushed and prefetching resumes at the target of the branch. A hexnum value of 0x7 indicates that buffer slots 2, 1, and 0 are to pre-decode looking for unconditional branches.

- `SoCCpuIBuffBcLookAhead = [On,Off] // Condition Register look-ahead for Bc`

The SoCCpuIBuffBcLookAhead parameter, when enabled (On), allows the instruction prefetch buffer (defined by the SoCCpuIBuffSize parameter) to use the condition register contents during a pre-decode phase to determine whether a conditional branch (bc) is taken or not. The contents of the condition register are only allowed to be used if there are no pending, or potential pending, updates to the condition register. This parameter works in conjunction with the SoCCpuIBuffBrDecSlotMap to define the instruction buffer slots which look for conditional branch instructions.

- `SoCCpuIBuffInst2Fold = inst // Instruction to fold`

The SoCCpuIBuffInst2Fold parameter defines an instruction (inst) which is allowed to execute in parallel (folding) with the instruction preceding it in the instruction pipeline. The instruction (inst) is only allowed to fold if there are no register dependencies between the inst instruction and any outstanding, or potential outstanding, write-back results. The SoCCpuIBuffInst2Fold parameter may be listed multiple times in a simcfg file to specify more than one instruction to fold.

- `SoCXStats = [On,Off] // Statistic counters`

The SoCXStats parameter, when enabled (On), allows the SoC model to capture various statistics on instructions and instruction-to-instruction usage. By default, the SoCXStats parameter is set to On. Simulation speed may improve by disabling (Off) this parameter. The highest level of simulation performance can be achieved by setting the SoCArchType parameter to Iss and the SoCXStats parameter to Off.

- `SoCPorAssertion = num // Number of clocks for power-up reset to assert`

The SoCPorAssertion parameter defines the number of clocks (num) that the SoC cpu will remain in reset before starting to execute instructions.

- `SoCIOFunction = targetio_req // IO Function name - for Diab dcc printf,etc.`

The SoCIOFunction parameter defines the name of the function which the Diab compiler uses for file io operations.

3.2 Cache Model Configuration

There are a number of cache parameters controlling the cache organisation and behaviour. Parameters controlling the instruction cache begin with the ICache prefix, and parameters controlling the data cache begin with the DCache prefix. If a unified cache is specified, the ICache parameters are not used. Instead, the DCache parameters are used for both instruction and data accesses.

- `CacheOrganisation = type // Off, Unified, Instruction, Data, Both`

The CacheOrganisation parameter specifies the type of cache or caches to be included in the simulation. The type may be Off, Unified, Instruction, Data, or Both. Off - specifies that caching should not be used in the simulation. This is the default value if cache configuration parameters are not supplied. Unified - specifies that a unified cache should be used. A unified cache caches both instruction and data accesses into a single cache unit. The Unified option uses the data cache parameter set. Data cache parameters start with the DCache prefix. Instruction - specifies that an instruction cache should be used. An instruction cache only caches instruction accesses, data accesses are not cached. Data - specifies that a data cache should be used. A data cache only caches data accesses, instruction accesses are not cached. Both - specifies that an instruction and data cache should be used. The Both option allows two caches to be used in a simulation, one cache for instruction accesses and the other cache for data accesses. The instruction cache behavior is controlled by the parameters with the ICache prefix while the data cache behavior is controlled by the parameters with the DCache prefix.

- `CacheDumpStatsClock = N // Display stats after every N clocks`

The CacheDumpStatsClock specifies that the simulator's execution statistics should be displayed during simulation after every N clocks. A zero value turns this option off. The statistics displayed by this parameter are the same statistics displayed by the simulator's x command. Note that the clock count specified by this parameter is not guaranteed to be an exact match with the simulation clock count -- only a close estimate. The cache model only checks this parameter during events that affect the cache, not on every clock event.

```
- ICacheBaseAddr = addr // Cache Base Address for active range
- DCacheBaseAddr = addr // Cache Base Address for active range
- ICacheEndAddr = addr // Cache End Address for active range
- DCacheEndAddr = addr // Cache End Address for active range
```

The BaseAddr works in conjunction with the EndAddr to specify an address (addr) range for memory transactions that are cacheable. Transactions outside this address range are considered non-cache events and are allowed to pass through the cache model. The address range provides a mechanism to excluded memory transactions from being cached such as memory mapped control registers or semaphore locations for synchronizing multiprocessor events.

```
- ICacheSets = num // Number of sets
- DCacheSets = num // Number of sets
```

The CacheSets parameter specifies the number (num) of cache tag sets to be used when creating the cache. The CacheSets parameter works in conjunction with the CacheTags parameter to define the associativity of the cache. A direct mapped cache (no associativity) would have 1 CacheTag per CacheSet; whereas, a fully associative cache would have 1 CacheSet with multiple CacheTags. Note: the number of cache sets will be adjusted upward by the cache model to make it a power of 2. For example, specifying 3 for CacheSets will be adjusted to 4.

```
- ICacheTags = num // Number of tags per set - maximum of 256
- DCacheTags = num // Number of tags per set - maximum of 256
```

The CacheTags parameter specifies the number of cache tags per cache set. The CacheTags parameter works in conjunction with the CacheSets parameter to define the associativity of the cache (refer to CacheSets).

- `ICacheSubBlocks = num // Number of sub-blocks per tag - maximum of 64`
- `DCacheSubBlocks = num // Number of sub-blocks per tag - maximum of 64`

The CacheSubBlocks specifies the number (num) of sub-blocks per cache tag. The cache model allows multiple sub-blocks to be controlled with a single cache tag. It is therefore possible to get a cache tag hit but a miss on a sub-block. The CacheSubBlocks parameter works in conjunction with the CacheSubBlockSize, CacheFillSize, and CacheSubBlockMissCost parameters to define the size and behavior of sub-blocks. The number of cache sub-blocks will be adjusted upward by the cache model to make it a power of 2. For example, specifying 3 for CacheSubBlocks will be adjusted to 4.

- `ICacheSubBlockSize = num // Number of Bytes per Sub_block (max. 4096)`
- `DCacheSubBlockSize = num // Number of Bytes per Sub_block (max. 4096)`

The CacheSubBlockSize parameter specifies the size of a sub-block - size is specified in number (num) of bytes. Note: the CacheSubBlockSize will be adjusted upward by the cache model to make it a power of 2. For example, specifying 3 for CacheSubBlockSize will be adjusted to 4.

- `ICacheFillSize = num // Sub_blocks to fill on a miss`
- `DCacheFillSize = num // Sub_blocks to fill on a miss`

The CacheFillSize parameter specifies the number (num) of sub-blocks to fill on a cache miss. Not all of the sub-blocks associated with a cache tag have to be filled on a cache tag miss. It is therefore possible, on a subsequent memory transaction, to get a cache tag hit with a miss on a sub-block. The HitCost and CacheSubBlockMissCost affect the clock behavior in this latter case.

- `ICacheTagReplAlgo = type // Tag replacement algorithm: Linear, LFSR, LRU`
- `DCacheTagReplAlgo = type // Tag replacement algorithm: Linear, LFSR, LRU`

The CacheTagReplAlgo specifies the type of tag replacement algorithm. When there are multiple tags per cache set (associativity) and all of the tags are filled within a set, but there is a current cache miss, then one of the tags has to be selected for replacement. The CacheTagReplAlgo is used to either use a Linear, a least recently used (LRU), or pseudo-random replacement algorithm (LFSR). The linear replacement algorithm (Linear) uses one counter for all cache sets (this is opposed to one counter for each set). When a tag has to be replaced (regardless of the set number), the replacement counter is used to provide the tag number and then the replacement counter is advanced. The tag number is then used to select the tag within the set for replacement. The least recently used (LRU) replacement algorithm selects the LRU tag within a set. The algorithm first locates the correct set, and then identifies the LRU tag within that set. The pseudo-random replacement algorithm uses a linear-feedback-shift-register (LFSR) to generate a pseudo-random tag number. This tag number is then used to select the tag within the set for replacement.

- `ICacheRdHitCost = num // Number of clocks for a read hit`
- `DCacheRdHitCost = num // Number of clocks for a read hit`

The CacheRdHitCost parameter specifies the number (num) of clock cycles for a read transaction that hits in the cache.

- `DCacheWrHitCost = num // Number of clocks for a write hit`

The DCacheWrHitCost parameter specifies the number (num) of clock cycles for a write transaction that hits in the data or unified cache. This is the overhead cost to perform the cache tag look-up and does not include costs associated with bus traffic to main memory. Any costs associated with bus traffic to main memory will be added to the total transaction cost. The effects of bus traffic to main memory on total transaction cost can be eliminated by using the CacheZeroMemCost parameter.

```
▪ ICacheTagMissCost = num // Number of clocks for a tag
  miss
▪ DCacheTagMissCost = num // Number of clocks for a tag
  miss
```

The CacheTagMissCost parameter specifies the number (num) of clock cycles for a read or write transaction that misses in the cache. This is the overhead cost to perform the cache tag look-up and does not include costs associated with bus traffic to main memory. Any costs associated with bus traffic to main memory will be added to the total transaction cost. The effects of bus traffic to main memory on total transaction cost can be eliminated by using the CacheZeroMemCost parameter.

```
▪ ICacheSubBlockMissCost = num // Number of clocks for a
  sub_block miss
▪ DCacheSubBlockMissCost = num // Number of clocks for a
  sub_block miss
```

The CacheSubBlockMissCost parameter specifies the number (num) of clock cycles of any additional overhead associated with a sub-block miss. A sub-block miss implies that there was a cache tag hit, so the SubBlockMissCost is added into the total transaction cost which also includes the cost of a cache tag hit. In addition, any costs associated with bus traffic to main memory will be added to the total transaction cost. The effects of bus traffic to main memory on total transaction cost can be eliminated by using the CacheZeroMemCost parameter.

```
▪ ICacheZeroMemCost = [On,Off] // Don't include memory
  costs with cache costs
▪ DCacheZeroMemCost = [On,Off] // Don't include memory
  costs with cache costs
```

The CacheZeroMemCost parameter specifies whether to include, or not include, the cost of bus traffic to main memory. If CacheZeroMemCost is On, then transaction costs associated with main memory are not included. Therefore, the total cost of a transaction is controlled by the costs associated with the cache -- CacheRdHitCost, DCacheWrHitCost, CacheTagMissCost, CacheSubBlockMissCost, and DCacheCopyBackOverheadCost.

- `DCacheCopyBackOverheadCost = num // Number of clocks per Modified (M) entry`

The DCacheCopyBackOverheadCost parameter specifies the number (num) of clock cycles of overhead associated with looking at the modified bits to determine whether a sub-block has to be copied out to main memory. In this case, the total transaction cost is the accumulation of the CacheTagMissCost, the DCacheCopyBackOverheadCost, and the cost of any bus traffic to main memory. However, the effects of bus traffic to main memory on total transaction cost can be eliminated by using the CacheZeroMemCost parameter.

- `DCacheWriteBackPolicy = type // WrThru, Copyback`

The DCacheWriteBackPolicy specifies the type of write-back policy for data and unified caches. The write-back policy is either write-through (WrThru) or copyback (Copyback). In the write-through mode, write transactions are allowed to go through the cache to update main memory. In the copyback mode, write data is stored in the cache and only pushed to memory when a cache line, having modified data, has to be replaced or flushed.

- `DCacheWriteAllocate = [On,Off] // Write Allocate Policy (On or Off)`

The DCacheWriteAllocate specifies the write allocate policy. If write-allocate is enabled (On), then on write transactions that miss in the cache, the cache line must first be filled (allocated) before performing the write. Once the cache line is filled, the DCacheWriteBackPolicy determines whether the write transactions is allowed to update main memory (or copied-back at a latter time).

- `ICacheFlushClock = N // Flush cache after every N clocks`
- `DCacheFlushClock = N // Flush cache after every N clocks`

The CacheFlushClock parameter specifies that the cache should be flushed after every N clocks - where N is the number of clocks between flush events. A zero value turns this option off. The CacheFlushClock parameter provides a mechanism for estimating the effects of task switching on benchmark code without having to simulate an entire operating system.

3.2.1 Example Cache Model Configurations

Following AME configuration code segment creates a "Direct Mapped Cache" – a 256-byte direct mapped instruction cache with 4 words (32-bit words) per cache line that operates in the first 1 Megabyte of memory. These cache parameters can be included in a simulation configuration file (simcfg) to define the cache.

```
CacheOrganisation = Instruction
CacheLiterals = Instruction
ICacheBaseAddr = 0x0
ICacheEndAddr = 0x000fffff
ICacheSets = 64
ICacheTags = 1
ICacheSubBlocks = 4
ICacheSubBlockSize = 4
ICacheTagReplAlgo = 0
ICacheFillSize = 4
ICacheRdHitCost = 1
ICacheTagMissCost = 2
ICacheSubBlockMissCost = 0
ICacheZeroMemCost = Off
ICacheFlushClock = 0
```

3.3 Memory Model Configuration

The AME simulator's memory model handles memory allocation and the loading of the target executable file – usually an Executable and Linking Format (ELF) binary file. Typically, to keep things as simple as possible, there is only one instance of the memory model created in an AME simulation. Having multiple instances leads to coherency issues and problems related to locating a specific instance of the memory model when looking for data. To avoid these issues, while allowing different views or interfaces into the memory model, the concept of a memory window was developed. The data still resides in the single instance of the memory model but a facade or new front-end into the memory model is created. This allows the AME simulator to model on-chip ROM, RAM, and FLASH memory blocks while still maintaining central control over the data and memory management.

3.3.1 System Memory Model Configuration

The system memory model defines the system memory (or main memory) characteristics. The following configuration parameters are included in the simulator configuration (simcfg) file to define the default behavior of the system memory model.

```
▪ MemUnit = [On,Off] // Enable system memory
```

The MemUnit parameter enables (On) or disables (Off) the memory model. Usually, the memory model is always enabled (On).

```
▪ MemPageSize = num // Memory page size for allocating
  memory
```

The MemPageSize parameter is used by the memory model when allocating memory for the simulation. The memory model allocates memory in pages with num bytes per page until the requested memory range has been allocated. In general, the larger the page size, the more efficient the simulation. Typical page sizes range from 64 Kbytes (0x10000) to 1 Mbytes (0x100000). The MemPageSize parameter is for tuning the memory model to work more efficiently with the host platforms memory management and not for defining the memory mapping of the processor system being simulated.

```
▪ MemBaseAddr = addr // Memory default base address
▪ MemEndAddr = addr // Memory default end address
```

The MemBaseAddr works in conjunction with the MemEndAddr to specify a default address (addr) range for the memory model. When the simulator starts-up, the memory model allocates memory to populate this address range of the processor system being simulated. The default address range defined by MemBaseAddr and MemEndAddr are over-ridden by the memory requirements defined by an executable file (usually an ELF file) being loaded, or by a memory configuration file.

```
▪ MemoryAccessTime = num // Memory Access Time in clock
  ticks
```

The MemoryAccessTime parameter defines the default memory latency in number (num) of clock cycles. This default memory latency is over-ridden when using a memory configuration file (refer to "Memory Configuration").

- `MemoryBusWidth = num // Memory data path width in bytes`

The MemoryBusWidth parameter defines the width of the bus in number (num) of bytes to the system memory model. This default memory bus width is over-ridden when using a memory configuration file (refer to "Memory Configuration")

3.3.2 On-Chip ROM Configuration

The following list provides the configuration parameters which define Read-Only-Memory (ROM) memory ranges. Multiple sets of the OnChipRom parameters may be used to define multiple ROM memory regions.

- `OnChipRomUnit = [On,Off ] // Enable On-chip rom unit`

Off - specifies that the on-chip rom unit should not be used in the simulation. This is the default value if the OnChipRomUnit configuration parameter is not supplied. On - enables the on-chip rom unit. Write bus transactions are terminated with a transfer error signal potentially causing an exception to be taken.

- `OnChipRomBase = addr // On-Chip ROM base address`
- `OnChipRomEnd = addr // On-Chip ROM end address`

The OnChipRomBase address (addr) works in conjunction with the OnChipRomEnd address (addr) to specify an address range for the on-chip rom. Bus transactions in this address range will be directed to the on-chip rom unit. Bus transactions outside this address range are not directed to the on-chip rom unit.

- `OnChipRomBlkSize = num // On-Chip ROM block size in bytes`

The OnChipRomBlkSize parameter provides a mechanism to allow the on-chip rom to be multiply mapped within an address range. (The memory range is controlled though the OnChipRomBase and OnChipRomEnd addresses discussed above.) The OnChipRomBlkSize must be set to a power of 2. For example, if the OnChipRomBlkSize is set to 0x1000 (i.e. 4K bytes) then at every 4K boundary the same data will be seen. If there is only one block boundary, then the OnChipRomBlkSize should be set to the entire OnChipRomBase to OnChipRomEnd address range.

3.3.3 On-Chip RAM Configuration

The following list provides the configuration parameters which define Random-Access-Memory (RAM) memory ranges. Multiple sets of the OnChipRam parameters may be used to define multiple RAM memory regions.

- `OnChipRamUnit = [On,Off] // Enable On-chip ram unit`

Off - specifies that the on-chip ram unit should not be used in the simulation. This is the default value if the OnChipRamUnit configuration parameter is not supplied. On - enables the on-chip ram unit. The ram unit supports both read and write bus transactions.

- `OnChipRamBase = (addr) // On-Chip RAM base address`
- `OnChipRamEnd = (addr) // On-Chip RAM end address`

The OnChipRamBase address (addr) works in conjunction with the OnChipRamEnd address (addr) to specify an address range for the on-chip ram. Bus transactions in this address range will be directed to the on-chip ram unit. Bus transactions outside this address range are not directed to the on-chip ram unit.

- `OnChipRamBlkSize = num // On-Chip RAM block size in bytes`

The OnChipRamBlkSize parameter provides a mechanism to allow the on-chip ram to be multiply mapped within an address range. (The memory range is controlled though the OnChipRamBase and OnChipRamEnd addresses discussed above.) The OnChipRamBlkSize must be set to a power of 2. For example, if the OnChipRamBlkSize is set to 0x1000 (i.e. 4K bytes) then at every 4K boundary the same data will be seen. If there is only one block boundary, then the OnChipRamBlkSize should be set to the entire OnChipRamBase to OnChipRamEnd address range.

- `OnChipRamProtect = [On,Off] // Protect On-Chip RAM from program loader`

The OnChipRamProtect parameter, when enabled (On), prevents the program loader from initializing the on-chip ram space. This provides a mechanism to make sure that application programs work with on-chip ram where the ram is not assumed to be initialised as part of the executable file loading process, usually and ELF binary file.

3.3.4 Flash Memory Configuration

The following list provides the configuration parameters which define Flash memory ranges. Multiple sets of the Flash parameters may be used to define multiple Flash memory regions.

- `FlashMemUnit = mode // Mode: Off,Page,SeqAddr, Seq, ASeq, ISeq, Burst`

The mode of operation work in conjunction with other flash parameters to control the cycle count information for bus transactions directed to the flash memory unit. For each mode of operation, the FlashMemBusWidth and FlashMemBusCyclingCost can add additional cycles to bus transaction costs. The Seq, ASeq, ISeq, and Burst modes represent specific modes which some processors may support. Refer to the processor's reference manual for more information on the use and availability of these modes.

Off - specifies that the flash unit should not be used in the simulation. This is the default value if the FlashMemUnit parameter is not supplied.

Page - page mode works with other flash parameters to control the cycle cost information for bus transactions. In page mode, an initial access requires a cycle cost defined by FlashPageAccessCost and establishes a page boundary (defined by FlashMemPageSize) for subsequent accesses. Pages are aligned to their natural boundaries defined by the FlashMemPageSize parameter (i.e. aligned starting with page 0 at address 0x0). If a subsequent access is made which falls within the same page as the previous access, then the cycle cost information is given by FlashPageHitCost. If an access falls outside the current page, a new page is loaded and the cycle cost is again defined by FlashPageAccessCost. Note that the FlashMemBusWidth in conjunction with the FlashMemBusCyclingCost can add additional cycle costs to the access.

SeqAddr - Sequential Address mode. The SeqAddr mode is from the point of view of the flash memory -- as long as accesses that arrive at the flash unit are sequential, the cycle cost is given by FlashPageHitCost. If addresses are not sequential, then the cycle cost is given by FlashPageAccessCost. Note that the FlashMemBusWidth in conjunction with the FlashMemBusCyclingCost can add additional cycle cost to the access.

Seq - Sequential mode. The Seq mode, if supported, is controlled by signals from the processor. If the Seq mode is signaled by the processor and if the access does not cross the page boundary controlled by FlashMemPageSize, then the cycle cost is given by FlashPageHitCost; otherwise the cycle cost is given by FlashPageAccessCost. Normally, to avoid a sequential access that misses a page boundary, the FlashMemPageSize is set to a value that covers the entire address range of the flash unit – if the page boundary is missed, the cycle cost is given by FlashPageAccessCost. Note that the FlashMemBusWidth in conjunction with the FlashMemBusCyclingCost can add additional cycle cost to the access.

ASeq - Accurate Sequential Access mode. The ASeq mode, if supported, is controlled by signals from the processor. If the ASeq mode is signaled by the processor and if the access does not cross the page boundary controlled by FlashMemPageSize, then the cycle cost is given by FlashPageHitCost; otherwise the cycle cost is given by FlashPageAccessCost. Normally, to avoid a sequential access that misses a page boundary, the FlashMemPageSize is set to a value that covers the entire address range of the flash unit -- if the page boundary is missed, the cycle cost is given by FlashPageAccessCost. Note that the FlashMemBusWidth in conjunction with the FlashMemBusCyclingCost can add additional cycle costs to the access.

ISeq - Instruction Sequential Access mode. The ISeq mode, if supported, is controlled by signals from the processor. If the ISeq mode is signaled by the processor and if the access does not cross the page boundary controlled by FlashMemPageSize, then the cycle cost is given by FlashPageHitCost; otherwise the cycle cost is given by FlashPageAccessCost. Normally, to avoid a sequential access that misses a page boundary, the FlashMemPageSize is set to a value that covers the entire address range of the flash unit -- if the page boundary is missed, the cycle cost is given by FlashPageAccessCost. Note that the FlashMemBusWidth in conjunction with the FlashMemBusCyclingCost can add additional cycle cost to the access.

Burst - Cache Line Burst mode. The Burst mode, if supported, is controlled by signals from the cache model. If the Burst mode is signaled by the cache model and if the access does not cross the page boundary controlled by FlashMemPageSize, then the cycle cost is given by FlashPageHitCost; otherwise the cycle cost is given by FlashPageAccessCost. Normally, to avoid a sequential access that misses a page boundary, the FlashMemPageSize is set to a value that covers the entire address range of the flash unit – if the page boundary is missed, the cycle cost is given by FlashPageAccessCost. Note that the FlashMemBusWidth in conjunction with the FlashMemBusCyclingCost can add additional cycle cost to the access.

- `FlashMemBase = (addr) // On-Chip flash memory base address`
- `FlashMemEnd = (addr) // On-Chip flash memory end address`

The FlashMemBase address (addr) works in conjunction with the FlashMemEnd address (addr) to specify an address range for the on-chip flash memory unit. Bus transactions in this address range will be directed to the on-chip flash unit. Bus transactions outside this address range are not directed to the on-chip flash unit.

- `FlashMemBlkSize = num // On-Chip flash memory block size in bytes`

The FlashMemBlkSize parameter provides a mechanism to allow the on-chip flash to be multiply mapped within an address range. (The memory range is controlled though the FlashMemBase and FlashMemEnd addresses discussed above.) The FlashMemBlkSize is specified in number (num) of bytes and must be set to a power of 2. For example, if the FlashMemBlkSize is set to 0x1000 (i.e. 4K bytes) then at every 4K boundary the same data will be seen. If there is only one block boundary, then the FlashMemBlkSize should be set to the entire FlashMemBase to FlashMemEnd address range.

- `FlashMemPageSize = num // Number of bytes for flash page size`

The FlashMemPageSize parameter controls the page size for an access. The page size is given in number (num) of bytes and must be a power of 2. The page size works in conjunction with FlashPageAccessCost and FlashPageHitCost to control the cycle cost of bus transactions.

- `FlashPageAccessCost = num // Number of clocks for initial page access`
- `FlashPageHitCost = num // Number of clocks for access to same page`

If an access to the flash memory misses the current page, or misses one of the sequential modes (i.e. SeqAddr, Seq, ASeq, ISeq, or Burst), then the cycle cost is given by FlashPageAccessCost; otherwise the cycle cost is given by FlashPageHitCost. Note that the FlashMemBusWidth in conjunction with the FlashMemBusCyclingCost can add additional cycle costs to the access.

- `FlashIFSize = type // Instruction Fetch Size Off, single, multi`

Some processors provide a dynamic mode to switch from pre-fetching multiple instructions per bus transaction to pre-fetching a single instruction per bus transaction. If the flash memory resides on a narrow bus, performance may be optimised in some systems by signaling the processor to prefetch instructions using the single mode of operation as opposed to the multi mode. The single option provides a signal on every bus transaction from the flash memory that the next bus transaction should only be requested as a single access. The multi option signals each bus transaction that the next access can be made using multiple instruction prefetching. The Off option is the default mode which does not signal the processor.

- `FlashMemBusWidth = num // Number of bytes for flash memory bus width`

The FlashMemBusWidth specifies the width of the bus connecting the flash memory to the rest of the system. The bus width is specified in number (num) of bytes and is used in conjunction with the FlashMemBusCyclingCost parameter to add additional cycle costs to bus transactions.

- `FlashMemBusCyclingCost = num // Number of clocks to cycle flash bus`

The FlashMemBusCyclingCost specifies the number (num) of clock cycles required to run multiple bus transactions in the event that the requested data size is greater than the FlashMemBusWidth parameter. For example, if a 4-byte (32-bit) word was requested over a 2-byte (16-bit) flash bus then the flash bus would have to be cycled 2 times to get all of the requested data. The first access cost is supplied by the FlashPageAccessCost and FlashPageHitCost parameters while the second access cost is supplied by FlashMemBusCyclingCost.

The cycle cost of the first access is always determined by the FlashPageAccessCost and FlashPageHitCost parameters. The FlashMemBusCyclingCost provides the additional cycle cost, for cycling the bus to obtain the requested access width. For a 4-byte access over a 1-byte bus, the first access cost is supplied by the FlashPageAccessCost or the FlashPageHitCost. The total access cost is the cost of the first access plus 3 times the FlashMemBusCyclingCost. The 1-byte bus has to be cycled an additional 3 times to complete a 4-byte access width.

3.4 Memory Configuration

By default, the memory model uses the MemBaseAddr and the MemEndAddr parameters in conjunction with the MemoryAccessTime and the MemoryBusWidth parameters to defined the memory map (refer to "System Memory Model Configuration") which can be displayed using the simulator's map command. There are two ways to modify the default memory map. One method is to load an executable file – usually an ELF binary file. The ELF binary file provides header information which specifies the memory segments needed for the program to execute. The second method uses a memory configuration file (memcfg) to define memory segments along with an optional memory latency and memory attributes entry for each specified memory segment.

The memory configuration information contained in a memcfg file can be combined with the simulator configuration (simcfg) file to form a single file. The memory configuration information must appear outside the Config-ConfigEnd parameter pair (refer to "Simulator Configuration File"). All numbers in a memcfg file are in hexadecimal. The first entry specifies the number of segments contained in the memcfg file.The subsequent lines contain memory segment entries. Each memory segment entry specifies the starting address location, the size of the segment in bytes, followed by an optional wait state entry. The wait state entry specifies the number of additional clock cycles required for the memory segment to respond to an access. If the wait state entry is not present, the memory object is expected to supply its own memory latency information. The following is an example of a memcfg file consisting of 2 memory segments.

```
// All values are in hexadecimal
3    // number of memory segments
// Start       Segment size
// Address     in bytes       [Waits]   [attributes]
//---------    ------------   -------   ------------
00000000       400            0                      // 1 clock access
400            c00            4                      // 5 clock access
1000           200                                   // use defaults
```

The Waits entry takes precedence over the default MemoryAccessTime parameter associated with the system memory model (refer to "System Memory Model Configuration"). If the system memory model to control wait state information is required, then the wait state entry is left blank as shown in the last entry of the example. There are additional attributes that can be specified with each memory segment declaration. The general format of the memcfg file is as follows.

```
Segments Address Size [Waits] [bus8:bus16:bus32:bus64] [if1:if2:if4]
```

Segments – The first hexadecimal number in the memcfg file represents the number of memory segments being declared.

Address – The Address entry is the starting address location of the memory segment being declared. This entry is assumed to be in hexadecimal format.

Size – The Size entry is the size of the memory segment being declared. The size is specified as a hexadecimal number and represents the number of bytes in the segment.

Waits – The Waits is an optional entry and represents the number wait states for memory accesses to the segment being declared. If a wait state entry is made, then the MemoryAccessTime parameter associated with the system memory model is over-ridden (refer to "System Memory Model Configuration"). The waits entry is specified as a hexadecimal number.

bus8:bus16:bus32:bus64 – This optional attribute entry specifies the bus width for the memory segment being declared. This entry defaults to bus32, a 32-bit bus width.

if1:if2:if4 – Some processor architectures provides dynamic instruction fetch sizing allowing the architecture to optimise performance in systems having narrow bus widths. The [if1:if2:if4] entry allows each memory segment being declared the ability to dynamically adjust the instruction size being requested by the processor. For a memory region being defined and to optimise instruction fetch bus transactions, the if1 option specifies that one instruction should be requested, if2 indicates that two instructions can be requested, and if4 indicates that four instructions can be requested. This entry defaults to if4. Note that this entry has no affect on processor architectures that do not support this architectural feature. Also note that this entry does not prevent a processor from using any instruction fetch size that the architecture requires.

4 Profiling Binaries

The AME simulator provides special interface and start-up features which simplifies the process of preparing application code for simulation. The AME simulator uses information provided by the executable file along with simulator configuration information to initialise the stack pointer, heap pointer, program counter, and registers used to pass the standard argc and argv arguments used by C programs. In addition, the AME simulator provides a set of system calls which interface to the host platform's operating system for file io operations.

4.1 Simple Compile and Simulate

The example hello program shown below is used throughout this portfolio to demonstrate techniques for compiling target programs and simulating. The hello program simply prints the Hello, World message to the display and terminates.

```
/*******************/
/* hello.c program */
#include <stdio.h>
int
main()
{
printf("Hello, World\n");
return 0;
}
/*******************/
```

When working with C programs, it is sometimes useful to compile and run them natively on the host platform. For example, the hello program can be compiled with the Gnu compiler and executed as follows.

```
[] gcc hello.c -o hello
[] hello
Hello, World
[]
```

Ideally, target programs or benchmarks should compile and execute just as simply for the AME simulator. For example, using the Gnu compiler for PowerPC with the -msim command line option builds the hello program which can be simulated on the AME simulator.

```
[] ppc-elf-gcc -msim hello.c -o hello
[]
```

The -msim option on the PowerPC Gnu compiler links the compiled code with a library and start-up code supporting the system calls used for simulation. This simulation support is part of the PowerPC Gnu package. A summary of available target options for the PowerPC Gnu compiler can be displayed by using the --target-help command line option.

```
[] ppc-elf-gcc --target-help
:
:
```

Once the hello program has been cross-compiled for SoC, it can be simulated as follows.

```
[] sim-ame SoC hello
Hello, World
program exit(0)
Sim.  Loop Time: 0.020 Seconds; Clock Ticks: 2045; CPS 102250.000
[]
```

The AME simulator executes the program printing the Hello, World message and then displays the program exit status -- in this case, an exit status of 0. Before terminating, the AME simulator displays some execution statistics. By default, the AME simulator runs in the Instruction set simulation (Iss) mode executing one instruction per simulated clock cycle. Therefore, there were 2045 instructions executed in this example. The CPS number indicates the simulation speed that was achieved by showing the number of simulated clocks per second on the host machine. The CPS number will vary depending on how loaded the host machine is and on the length of the program being simulated. For very short programs, like the hello program, there may not be enough measured time (0.020 Seconds in this case) to get an accurate CPS number.

4.2 Binding Code Segments to Memory Regions

When compiling code for a specific target system, it is often the case that code segments will need to be placed within certain memory address boundaries. For example, a target system may have read-only-memory (rom) to store static code and data, and random-access-memory (ram) for dynamic variables and stack space. The following "Linker Control Script File - map.lnk" provides an example which controls the link process to bind code segments to specific memory regions. Any exception table code is stored in the vec space starting at address 0x00000, static code and data is stored in the rom space starting at address 0x10000, and dynamic variables and stack space

```
/* ------------------------------------------------------------------
* The MEMORY command is used to describe the physical memory available
* to bind the program to memory.  There are three areas of memory:
* "vec", "rom", and "ram" as pictured below.  On the right is a symbol
* '_end' used to initialise a pointer which the system call brk uses
* to dynamically allocate memory for data segments.  This symbol is
* defined at the end of this file.
*
* 0x00000000  +--------------------------------+
*   "vec"     | .vec : Vector Table Code       |
* 0x00010000  +--------------------------------+
*       |     | .text: crt0 code               |
*    "rom"    | library code                   |
*       |     | program code                   |
*       |     | initialised data               |
*       |     +--------------------------------+
*       |     | / / / / / / (unused) / / / / / |
* 0x00040000  +--------------------------------+
*       |     |      uninitialised dynamic     |
*       |     |           variables            |
*    "ram"    |--------------------------------| <- _end
*       |     |                                |
*       |     | stack, grows down in memory    |
*       |     | toward address 0x0             |
* 0x00041000  +--------------------------------+
* ------------------------------------------------------------------*/
MEMORY
{ vec : org = 0x00000, len = 0x10000
  rom : org = 0x10000, len = 0x8000
  ram : org = 0x40000, len = 0x1000
}
/* Place code sections into appropriate memory types */
SECTIONS
{ .vec : { *(.gcc_except_table) } > vec
  .rom : { *(.text) *(.data)
           *(.sdata) *(.sdata2)
           *(.rodata)
           *(.init) *(.fini) *(.eh_frame) *(.fixup)
           *(.got) *(.got1) *(.got2)
           *(.ctors) *(.dtors) } > rom
           .ram : { *(.sbss) *(.sbss2) *(.bss) } > ram
}
/* Assign _end pointer to a free memory region */
_end = __SBSS_END__;
```

Any exception table code is stored in the vec space starting at address 0x00000, static code and data is stored in the rom space starting at address 0x10000, and dynamic variables and stack space are allocated to the ram space starting at address 0x40000. The symbol _end is used by the system call brk to change dynamically the amount of space allocated for the data segment. The _end symbol is set to the end of the uninitialised variable space in the ram memory region.

Note that more complex programs will need both read and write access to the initialised data space requiring some added start-up code to first copy the initialised data space to ram memory before starting execution. Also note that the program will need to be linked with the initialised data space in the ram region to allow program references to evaluate correctly. Then, as a post process, moving initialised data to the rom memory region. The following command line can be used to compile the simple hello program using the linker control script map.lnk. A comma separated option list (-Wl,-T,map.lnk,-e,_start) is passed to the linker to provide the name of the linker control script (map.lnk) and the name of the entry point (_start).

```
[] ppc-elf-gcc -msim -Wl,-T,map.lnk,-e,_start hello.c -o hello
```

4.3 Automatic Stack and Heap Pointer Initialisation

The AME simulator uses the memory model configuration parameters (refer to "System Memory Model Configuration") and information from the AME program loader to assign initial values to the stack pointer register and the heap pointer. The stack pointer value is loaded into register r1 of the SoC processor while the heap pointer is an internal variable kept by the AME memory model and potentially used by system calls for dynamic memory allocation during run-time. A program may choose to use these default initialisation values or provide its own start-up and memory allocation routines which over-ride these default values. The intent of providing default initialisation values is to simplify the process of preparing benchmark code for simulation – ideally no additional start-up code will be needed.

Figure 4.1 overleaf shows the relationship between the MemBaseAddr and MemPageSize memory configuration parameters and the assignment of the heap pointer and stack pointer. After the AME simulator completes the loading of the program, the heap pointer is assigned to the end of the initialised memory region which contains the program code and data.

When memory allocation is requested by the program, the heap pointer will be increased toward higher memory (i.e. away from address 0x0). The stack pointer is initialised to the end of the memory page boundary (as defined by the MemPageSize parameter). When stack space is required by the program, the stack pointer moves toward lower memory (i.e. toward address 0x0).

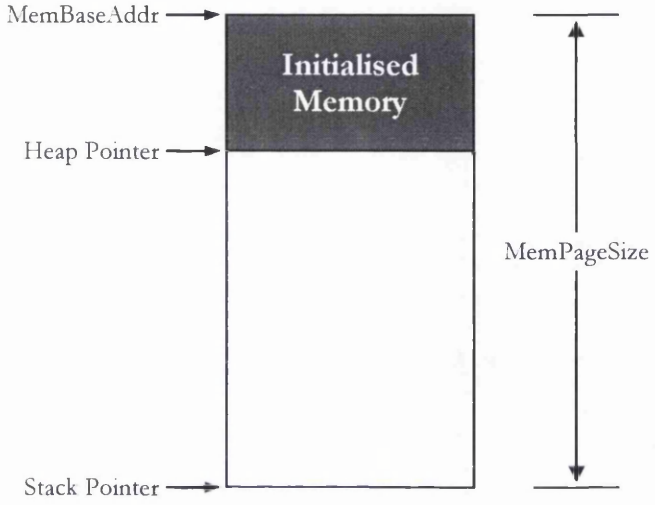

Figure 4.1: Stack Pointer and Heap Pointer Initialisation

If program code has been compiled and bound to specific memory regions, then the MemBaseAddr and MemPageSize memory configuration parameters can be used to adjust the pointers for heap and stack assignments. For example, in the previous section the hello program was bound to memory using a linker control script. Since the AME program loader allocates simulation memory based on the requirements of the executable file, the MemBaseAddr and MemPageSize parameters can be adjusted as follows to control the automatic assignment of the stack and heap pointers to point to unused sections of the ram region.

```
// Example simulation configuration file
Config      = SoC
MemPageSize = 0x0800
MemBaseAddr = 0x40800
MemEndAddr  = 0x40fff
ConfigEnd
```

5 Conclusions

This portfolio introduced the usefulness and infrastructure of a modelling and profiling environment for the evaluation of single and dual-issue PowerPC SIMD SoCs with tightly coupled signal processing capabilities and a hierarchical memory architecture. Using the AME, it was observed that:

SIMD techniques provide a significant speedup for advanced automotive powertrain applications with high data throughput. Using the AME, the observed average speedups over a 2-way single and dual issue SIMD in order execution machine range from 1.0 to to 3.6. For optimised application kernels, such as the knock processing platform developed, a factor of 2.5 improvement was achieved with perfect branch prediction.

Application binary characterisation reduces the cost of exploring the design space, focussing on the experimental system toward the intended workload, and curtails simulation and redesign requirements.

In order to realise the true potential of AME, application binaries need to be developed with improved embedded compiler technology supporting SIMD architectures even though intrinsics provide some ease of programming when compared to hand-coded assembly, it is still up to the code developer to find the data and instruction parallelism.

Identical application binaries profiled on both the real-time SoC platform and the AME confirm that the accuracy of simulated results is better than 95% in the features modelled.

6 References

[1] J. L. Hennessy and D. A. Patterson, "Computer Architecture: A Quantitative Approach", Morgan Kaufmann, Second Edition, 1996

[2] WindRiver, providers of high performance compilers, software development tools and real-time operating systems for developers of embedded systems, Web home-page: http://www.widriver.com/

[3] Green Hills Software Inc., providers of high performance compilers, software development tools and real-time operating systems for developers of embedded systems, Web home-page: http://www.ghs.com/

[4] Metrowerks, providers of high performance compilers, software development tools and real-time operating systems for developers of embedded systems, Web home-page: http://www.metrowerks.com/

[5] ARM (Advanced RISC Machines), designers of digital products for wireless, networking, consumer entertainment, imaging, automotive, security and storage devices, Web home-page: http://www.arm.com/

[6] IAR Systems, suppliers of development tools and services for embedded systems, Web home-page: http://www.iar.com/

[7] Motorola Semiconductor Products Sector, [Online], Available: http://www.motorola.com/automotive/products.html

[8] Intel Corporation, Web home-page: http://www.intel.com/

Submitted to IEEE VTS

System Level Design of a Controller Platform for a Camless, Electromagnetic Actuator Driven Next Generation Car Engine

M. Anas, Member IEE, Member, IEEE, D. Cumming, Member, IEEE, R. Paling, B. Nailon

Abstract – **Besieged by demands for better fuel economy, more power, and less pollution, motor engineers around the world are pursuing a radical "camless" design that promises to deliver the internal combustion engine's (ICE) biggest efficiency improvement in years. For the automobile driver, the camless engine might offer many of the advantages of hybrid gasoline-electric drive trains recently introduced by Toyota and Honda, but at less cost.**

Like a very simple software program that contains only one set of instructions, the cams always open and close the valves at the same precise moment in each cylinder's constantly repeated cycle of fuel-air intake, compression, combustion, and exhaust. They do so regardless of whether the engine is idling or spinning at maximum rpm.

The highest variability is obtainable by a system through which each valve is controlled separately. The elctromechanical valavetrain enables the optimised timing of the individual valve offering a reduction in fuel consumption by about 10 per cent and an additional peak torque of 5 percent [1].

We propose a comprehensive real-time System Level Development Platform (SLDP) and a Design Methodology (DM) to test and validate electromechanical valvetrain controlled (EMVC) camless engines with rapid prototyping and development borne in mind.

Index Terms – **camless engines, system level design (SLD), electromechanical actuators (EMA), single instruction multiple data (SIMD) units, camless engines, variable valvetrain timing (VVT), Processor-in-the-Loop (PIL), rapid prototyping.**

Manuscript received October 15, 2003.

M. Anas is with the Institute for System Level Integration, Livingston and Motorola SPS, East Kilbride. (e-mail: mohamed.anas@ieee.org),

D. Cumming is with the University of Glasgow. Glasgow. Scotland. (e-mail: d.cumming@elec.gla.ac.uk)

R. Paling is with Motorola SPS. East Kilbride. Scotland. (e-mail: robin.paling@motorola.com)

B. Nailon is with the University of Strathclyde. Scotland. (e-mail: W.Nailon@eee.strath.ac.uk)

Publisher Item Identifier S xxxx-xxxx(xx)xxxx-x

I. Introduction

Automotive engines equipped with camless valvetrains (so called camless engines) have been studied for over thirty years but production worthy vehicles with engines of this type are still not available due to difficulties in ensuring adequate and reliable electromagnetic valve performance. For an electromechanical camless valvetrain (EMCV), the actuator noise caused by high contact velocities of the moving parts has been identified as a key problem [2].

With the idea of introducing a real time rapid prototyping methodology, in this paper, a system level model to design, develop and validate a model for an electromechanical camless valve actuator, including the internal combustion engine (ICE) is discussed.

Conventional internal combustion engines use mechanically driven camshafts to actuate intake and exhaust valves. While this system is convenient and reliable, the fixed timing of the valve events with respect to the piston motion is typically selected as a compromise among fuel economy, emissions, maximum torque output, valvetrain noise, vibration and harshness (NVH) [2].

The growing need to improve fuel economy and reduce emissions led to the introduction of an alternative valvetrain technology, namely a camless valvetrain [1]. In the camless valvetrain, the valve motion is controlled directly by a valve actuator, without mechanical linkage to the crankshaft. As a result, the timing of the exhaust and intake valve opening and closing can be optimised for each engine operating condition [1]. Various studies have shown that a camless valvetrain can alleviate many otherwise necessary engine design tradeoffs by supplying extra degrees of freedom to the overall powertrain system [2].

Specifically it has been shown that controlling the intake valve events can eliminate the need for throttled operation in gasoline engines [3], thereby reducing pumping losses, and improving fuel economy [2]. Other benefits of camless engines include, higher maximum torque output, which is optimised for different driving conditions, cylinder de-activation, and elimination of external exhaust gas recirculation (EGR), etc. [4].

While VVT can be obtained using a wide spectrum of different technologies (see a review of various VVT technologies in [1] and [4], the highest degree of flexibility and the fastest VVT capability is achieved in truly camless engines with either electro-hydraulic [4] or electromechanical actuators [1]. These camless actuator technologies are under intensive development by several car OEMs, with the electromechanical technology currently considered by many to be in a relatively more developed stage. The issues that have to be addressed in the actuator design include cost, reliability, packaging, power consumption, noise and vibrations. The noise has been identified as the main problem with the

electromechanical actuator technology. It may, in fact, preclude the usage of such systems, if satisfactory solutions are not found [4].

Research proves that high contact velocities of the moving parts of the actuator cause the noise in electromechanical actuator [9]. The noise can be reduced if the contact velocities are reduced, i.e. the so-called *soft-landing* is achieved. In a conventional valvetrain the fixed valve profiles are carefully optimised to reduce the noise and the optimal solution is mechanically embedded into the precision valvetrain design during manufacturing of the camshaft lobes. In a camless valvetrain, it is the responsibility of the electronic controller platform to ensure that adequate actuator performance at varying engine operating conditions is achieved. To facilitate the analysis and rapid prototyping real-time controller platform development for an electromechanical camless valvetrain (EMCV), an actuator model is developed in this document.

This paper is organised as follows: First, the real-time system level development platform built for the rapid prototyping of the actuator is described. This platform is primarily built around Matlab®, Simulink and the Real Time Workshop. Unlike in the work done by various OEMs using a variety of hardware and software modules [2], consistency is maintained by building the development environment around a single real-time simulation environment. Then an overview of the structure of the controller developed to achieve soft-landing is explained in section IV. Section V captures construction and the operation of a typical electromechanical actuator currently being used by various OEMs in camless engines.

II. Elements of the Electromagnetic Actuator Rapid Prototyping System Level Development Platform

In order to considerably reduce the development time, decision was made to use the Real-Time Workshop Embedded Coder (RTWEC) to generate, test, and deploy production worthy C code for use in the PowerPC based VVT embedded platform [8]. One of the features required was the integration of legacy code. Therefore, it was necessary to establish thorough understanding of the RTWEC to provide the setting up of the real time actuator electromagnetic controller framework shown in Figure 1.

III. Nomenclature

Signals and Parameters

Symbol	Unit	Explanation
(Y_d), Y	m	(desired armature position
V	ms^{-1}	armature velocity
$V_{u/l}$	V	voltage upper / lower coil
$I_{u/l}$	A	current upper / lower coil
$\Phi_{u/l}$	NmA^{-1}	flux upper / lower coil
$F_{u/l}$	N	magnetic force
F_{flow}	N	force due to gas airflow
m	kg	mass of moving part of actuator
G	kgs^{-1}	friction coefficient
D	kgs^{-2}	Spring constant
2h	m	thickness of armature disc
R	Ω	resistance or a coil

Notations

S(t)	continuous time signal
S[n]	discrete time signal
S^0	signal at equilibrium point
$s[n] = S[n] - S^0$	deviation between signal and equilibrium point
s[n, k]	discrete signal of the k^{th} cycle
ν	vector
M	matrix

Table 1: Signals, Parameters and Notations

IV. Actuator Controller Design, Development and Analysis

The actuator control system is required to ensure accurate valve closing and opening events (timing). Variable valve timing is used to optimise the engine operation with respect to emissions, fuel economy, and driveability [3]. The engine management system typically generates these commands based on the drivers torque demand and other vehicle variables. One of the key objectives of the controller is to reduce the armature-coil and valve-cylinder contact velocities, which in turn reduce noise and component wear. Modern engine manufacturers design camshafts to achieve a low 0.04 m/s contact velocity [9] at low engine speeds. The contact velocity in conventionally driven valves increase linearly with engine speed [6].

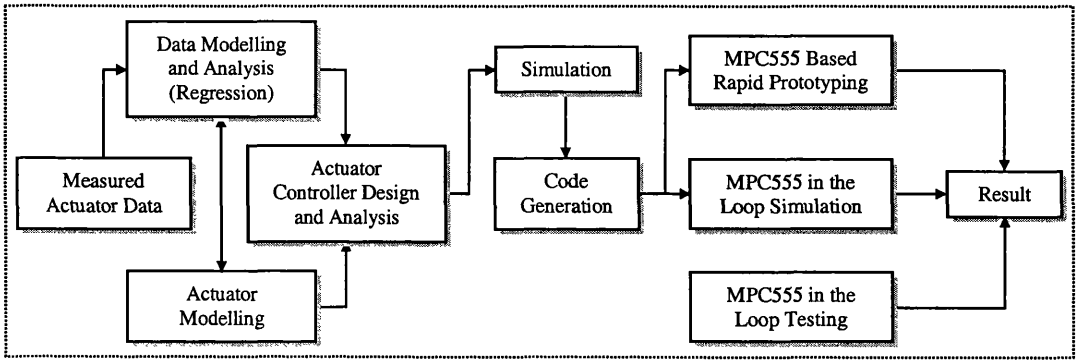

Figure 1: Electromechanical Camless Engine Controller Development Platform

In order to achieve the above requirements, a controller that achieves tracking of a reference trajectory $Y_d[n]$ [7] with the desired timing and contact velocity is designed.

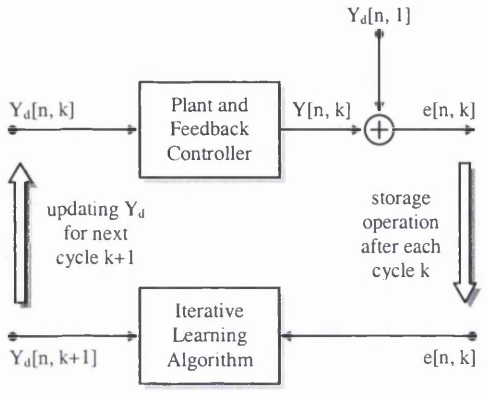

Figure 2: The controller structure with the iterative learning controller, applied to a closed loop system consisting of the actuator and an observer based feedback controller

Figure 2 shows the controller structure. The block "Plant and Feedback Controller" consists of an observer based feedback controller used to stabilise the plant at an equilibrium point close to the contact point. In order to improve the transient behaviour of the feedback controller, a feedforward controller changes the input of the closed loop system $Y_d[n]$. The new input is calculated by an Iterative Learning Controller (ILC), updating Y_d between consecutive cycles (full armature travel) k and k+1. This ILC is processing the error between the desired position $Y_d[n,1]$ and the actual position Y[n, k]. Detailed information about the learning controller is given in section VII.

V. Construction and the Operation of the Electromechanical Actuator

Figure 3 shows an electromagnetic actuator with a valve at three typical positions. There are two magnets (upper and lower magnets), two springs (actuator and valve springs) and an armature in the actuator. The two magnets are coils wound on ferromagnetic material. The coils are driven by currents generated by electronics. The electronics are driven by a pulse-width modulated voltage and the duty cycle of this voltage signal determines the steady-state current value. When the coil is activated, a magnetic field is generated and consequently the magnetic force is applied on the armature in the magnetic field. The two springs are adjusted such that both are always compressed for any position of the armature in between the two magnets. The armature is subjected to the magnetic force and actuator spring force and passes the forces to the valve. The actuator uses the spring force to accelerate the masses (the actuator spring bolt, the armature, the valve and the valve spring bolt), then uses the electromagnetic force to attract and dwell the valve and the armature. The equilibrium of the spring-mass system is at the middle (neutral) position between the two magnets. Therefore, when there is no current on coils, the valve will stay in the middle position. When the valve is closed there is a voltage applied on the upper coil, which generates a holding current. The holding current depends on the spring force and the pressure difference between the cylinder and the exhaust/intake manifold.

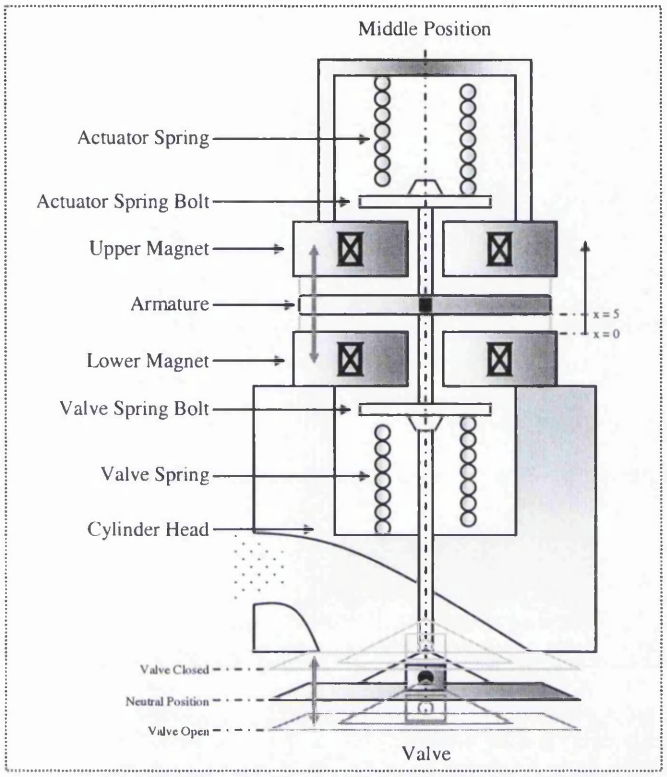

Figure 3: Construction of an electromagnetic actuator with the valve at neutral, open and closed positions

The driving current of the EMCV actuator and the corresponding valve profile is shown in Figure 4. If the valve is scheduled to open at t_0, the voltage applied to the upper coil should be regulated to zero at, $t_0 - \tau_m$; the time τ_m is necessary for the discharge of the magnetic field. Then a voltage with high duty cycle, d_c, is applied to the lower coil at t_1. A magnetic field is generated that attracts the armature to contact the lower coil and maintains the maximum valve lift. The current generated by this high voltage, d_c, is denoted as the *catching current*. Once the contact is ensured and quasi-static conditions of the mechanical subsystem are reached, the voltage applied to the lower coil can be reduced. The time that the lower duty cycle voltage, d_h, is applied to the lower coil is denoted by the t_2. Controlling t_2 varies the power consumption of the electrical subsystem. When the valve is closing, the operation is similar with the voltage applied on the lower coil regulated to zero at t_3. In Figure 4, the time intervals are defined as follows: $\tau_1 = t_1 - t_0$ and $\tau_2 = t_2 - t_0$.

To summarise, the actuator consists of electrical, magnetic and mechanical subsystems, which are interconnected with each other as shown in Figure 3. The variables and the subsystems used for the control of the EMA are explained in the following sections. Because of the symmetry, analysis is done only on the valve-opening event.

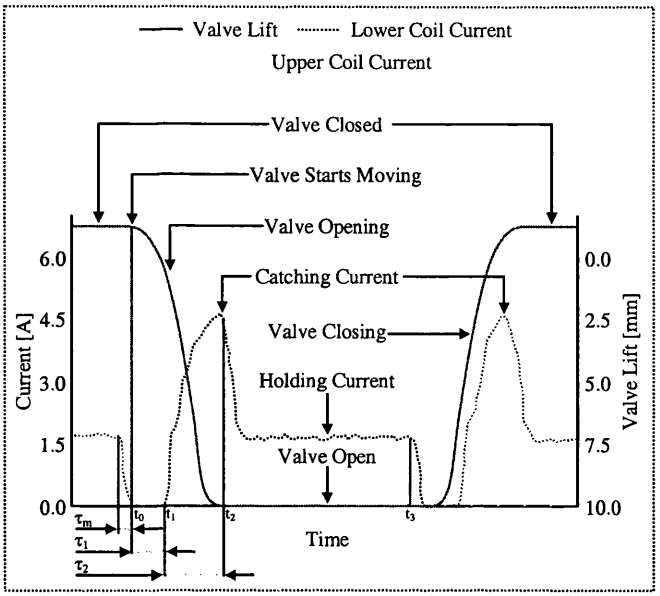

Figure 4: The Driving Current of the EMCV Actuator and the Corresponding Valve Profile

VI. Tracking Using Iterative Learning Control

In order to achieve better tracking of the desired position, the cyclic nature of the process is exploited by use of the Iterative Learning Controller (ILC) introduced in section IV. The next paragraph explains the ILC briefly, followed by a paragraph dedicated to the specific learning algorithm used in this paper.

A. *Modelling the Actuator*

The model of the actuator consists of a mechanical, electrical and a magnetic subsystem. For an explanation of symbols and parameters, see Table 1. The mechanical subsystem is modelled as a spring-mass-damper system including the external magnetic forces F_u of the upper and F_l of the lower electromagnet. A force balance yields

$$m\frac{dV(t)}{dt} = -DY(t) - GV(t) + F_l(t) + F_u(t) \tag{1}$$

The two coils are modelled by an electrical subsystem, consisting of a resistance/reluctance-circuit. The coil reluctances are inversely proportional to the armature gaps Y-Y_l-h or Y_u-Y-h, respectively. The coil currents $I_{l/u}(t)$ are modelled with a non-linear function f_I of the armature gap and the flux, yielding

$$V_l(t) = Rf_I(\Phi_l, Y - Y_l - h) + \frac{d\Phi_l(t)}{dt} \tag{2}$$

and

$$V_u(t) = Rf_I(\Phi_u, Y_u - Y - h) + \frac{d\Phi_u(t)}{dt} \tag{3}$$

as the two equations for the lower and upper coil, respectively. The mechanical and electrical subsystems are linked by the magnetic force equations of the two electromagnets,

$$F_l = -f_{mag}(\Phi_l, Y - Y_l - h) \tag{4}$$

and

$$F_u = f_{mag}(\Phi_u, Y_u - Y - h) \tag{5}$$

To summarise, the actuator has two inputs, upper voltage $V_u(t)$ and lower voltage $V_l(t)$, respectively. The actuator output is the armature position Y(t). The four elements of the state vector are position Y(t), velocity V(t), lower flux $\Phi_l(t)$ and the upper flux $\Phi_u(t)$. Thus, the state space description of the model is given by

$$\frac{dY}{dt} = V \tag{6}$$

$$\frac{dV}{dt} = -\frac{D}{m}Y - \frac{G}{m}V + \frac{F_u}{m} + \frac{F_l}{m} \tag{7}$$

$$\frac{d\Phi_u}{dt} = -Rf_I(\Phi_u, Y_u - Y - h) + V_u \tag{8}$$

$$\frac{d\Phi l}{dt} = -Rf_I(\Phi_l, Y_l - Y - h) + V_l \tag{9}$$

VII. Design of the Iterative Learning Controller

The input and the output sequences of the closed loop system are $Y_d[n]$ and Y[n], respectively. To formulate an ILC in a compact way, it makes sense to describe this mapping by defining the operator

$$\Gamma: \Re^N \mapsto \Re^N \text{ as } y = \Gamma(y_d) \tag{10}$$

$$y_d = \begin{bmatrix} y_d[n_{fb}] \\ \cdot \\ \cdot \\ y_d[n_{fb} + N - 1] \end{bmatrix} \tag{11}$$

and

$$y = \begin{bmatrix} y[n_{fb} + 1] \\ \cdot \\ \cdot \\ y[n_{fb} + N] \end{bmatrix} \tag{12}$$

n_{fb} is defined as the indice of the first sample after switching on the feedback controller. N is the number of values later involved in the ILC. As mentioned above, the lower case notation of a signal stands for its deviation from the equilibrium point.

The purpose of the ILC is to find some vector y_d^* with the property $y_d \approx \Gamma(y_d^*)$. In order to solve this problem, the cyclic opening and closing of the valve is exploited. Let the cycles be numbered with k. In the first cycle, the input vector $y_d[1] \equiv y_d$.is applied to the system. This vector and the corresponding output vector y[1] are used to generate an improved input vector $y_d[2]$ for the next cycle, and so forth. Thus, a linear formulation of the ILC algorithm reads as [7].

$$y_d[k+1] = Sy_d[k] + E(y_d[1] - y[k]) \tag{13}$$

where the matrices S and E weight the previous input $y_d[k]$ and the previous error $e[k] = y_d[1] - y[k]$, respectively. They have to be chosen in a way that the sequence $\{y_d[k]\}$ converges

$$y_d^* = \lim_{k \to \infty} y_d[k] \tag{14}$$

A. Design of the Learning Controller

Similar to the feedback controller, the learning controller is designed using the linearised model of the plant calculated in the section VI. Define the convolution matrix

$$P = \begin{bmatrix} h[1] & 0 & 0 & 0 & . & . & 0 \\ h[2] & h[1] & 0 & 0 & . & . & 0 \\ h[3] & h[2] & h[1] & 0 & . & . & 0 \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ h[N-1] & h[N-2] & . & . & . & h[1] & 0 \\ h[N] & h[N-1] & h[N-2] & . & . & h[2] & h[1] \end{bmatrix} \quad (15)$$

The matrix entries are the elements of the impulse response sequence {h[n]} of the linearised, discretised closed loop system, therefore $\Gamma(y_d) \approx Py_d$ holds true close to the equilibrium point [7]. To derive the ILC used in this paper, the singular value decomposition (svd) is applied to the convolution-matrix

$$P = L\Lambda R^T \quad (16)$$

where R and L are orthonormal matrices, Λ is a diagonal matrix with the elements

$$\sigma_0 > \sigma_i > 0 \ \forall i \in [1, N-1] \quad (17)$$

The largest singular value σ_0 is the L_2-norm of P. The learning algorithm is determined by setting [7]

$$S = I \quad (18)$$

and

$$E = \frac{1}{\sigma_0} R L^T \quad (19)$$

This learning algorithm is now analysed with the help of the discrete, linearised model. Using in equation (13) the linear model equation $y = Py_d$ yields

$$y_d[k+1] = Sy_d[k] + E(y_d[1] - Py_d[k]) \quad (20)$$

With equations (16), (18) and (19), equation (20) can be written in the form

$$R^T y_d[k+1] = \left(I - \frac{1}{\sigma_0}\Lambda \right) R^T y_d[k] + \frac{1}{\sigma_0} L^T y_d[1] \quad (21)$$

Let $v[k] = R^T y_d[k]$ and $\mu = L^T y_d[1]$, equation 21 reads as

$$v[k+1] = \left(I - \frac{1}{\sigma_0}\Lambda \right) v[k] + \frac{1}{\sigma_0}\mu \quad (22)$$

or rewritten in N separate equations

$$v[k+1] = \left(I - \frac{\sigma_i}{\sigma_0}\Lambda \right) v_i[k] + \frac{1}{\sigma_0}\mu_i \ \forall i \in [0, N-1] \quad (23)$$

The above choice of E and S yields a decoupled learning algorithm. Thus, to determine the convergence of the learning controller the convergence properties of N scalar equations can be studied instead of a matrix equation. Solving the recursive equation (23) yields

$$v_i[k] = \left(1 - \bar{\sigma}_i\right)^k v_i[0] + \frac{1 - (1 - \bar{\sigma}_i)^k}{\sigma_i}\mu_i \ \forall i \in [0, N-1] \quad (24)$$

with $\bar{\sigma}_i = \frac{\sigma_i}{\sigma_0}$. The equation converges to $v_i^- = \frac{\mu_i}{\sigma_i}$ for $|1 - \bar{\sigma}_i| < 1$. This is always true due to the svd property $\sigma_0 > \sigma_i > 0$. The convergence speed is determined by $1 - \bar{\sigma}_i$.

VIII. Modelling of the Car Engine to Evaluate the Performance of Internal Combustion Engine with Electromagnetic Actuator

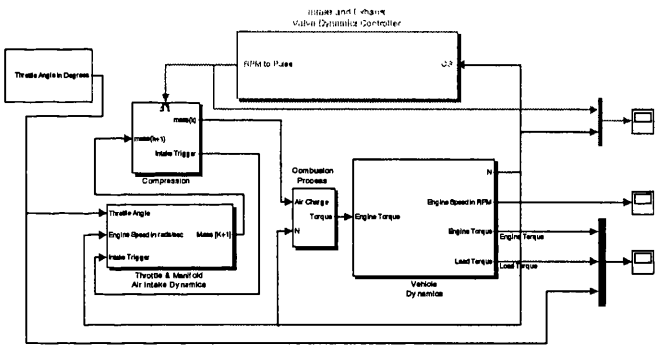

Figure 5: Snapshot of the Internal Combustion Engine Modelled in Simulink

The use of graphical dynamic system simulation software is becoming more popular as engineers strive to reduce the time to develop new control systems [5]. Dynamic system simulation software is an important tool for developing advanced reliable and high quality products. Thus, the skill of real-time complex systems modelling is the inherent goal of many modern mechatronic systems.

As mentioned in section I, MATLAB® SIMULINK modelling tool was chosen because of the wide exposure of this software in both academia and industry. The ICE model, snapshot of which is shown in Figure 5, created was used to evaluate the benefits of employing the electromechanical actuator controller algorithm developed. In addition, the model is used to simulate a sequential-fuel-injected, spark ignition engine and includes air and fuel dynamics in the intake manifold as well as the process delays inherent in a four-stroke cycle engine. Primarily, this engine model is used as follows:

1. As an embedded model within a control algorithm or observer
2. As a real-time engine model for hardware-in-the-loop testing
3. As a system model for evaluating engine sensor and actuator models
4. As a subsystem in a powertrain or vehicle dynamics model

Modular programming techniques were introduced to reduce model complexity by dividing the engine into hierarchical subsystems [8].

IX. Simulations and Concluding Remarks

An electromechanical camless valvetrain controller was developed for a camless engine. Simulations confirm the functional ability of the electromechanical actuator to vary valve timing, lift, velocity and event duration, as well as to perform selectively variable deactivation in a four-valve multi-cylinder engine.

Figure 6 and Figure 7 show simulations of the actuator controller with the learning algorithm and a comparison between engine torques obtained with a dual overhead camshaft (DOHC) engine and an EMCV arrangement respectively.

The contact velocity reduces as k is increased. Low contact velocity is associated with low noise and high component reliability.

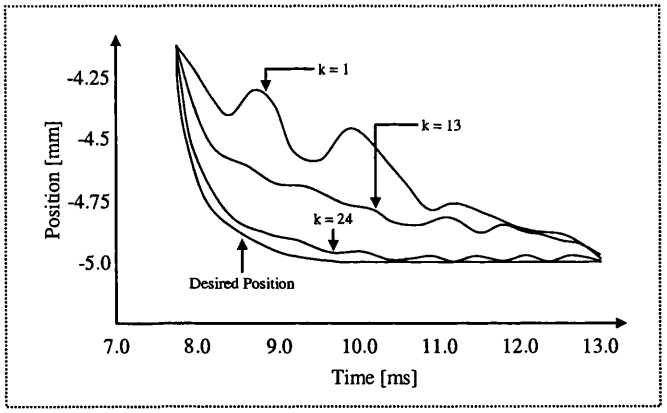

Figure 6: The position Y(t) tends to the desired position $Y_d(t)$ (solid) with increasing k. The figure shows $k \in [1, 13, 24]$

Throttle-free load control; whether with electromechanical or electro-hydraulic valvetrain, offers modern internal combustion engines a fuel consumption benefit of 10 per cent. Based on the present trends toward the employment of the variable valve timing methodology, it is evident that BMW's VALVETRONIC system is the next step in the foreseeable future.

This paper also presents a rapid prototyping approach to complex internal combustion engine modelling. The use of model based control methods designed to meet future emission and diagnostic regulations has also increased the need for validated engine models.

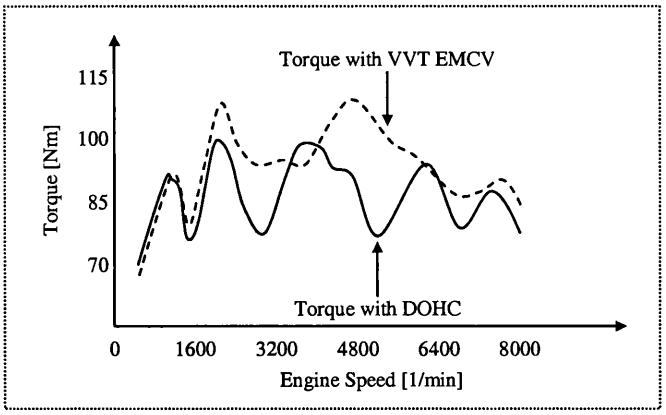

Figure 7: Simulated Torque Generated with the DOHC vs. EMCV

References

[1] R. Flierl and M. Kluting, BMW Group, "The Third Generation of Valvetrains for Throttle-Free Load Control", *SAE 2000 World Congress*, SAE Technical Paper Series, Paper No. 2000-01-1227.

[2] Ahmad T. and Theobald M. A., "A Survey of Variable Valve Actuation Technology", SAE Paper No. 891674.

[3] J. H. Tuttle, "Controlling Engine Load by Means of Early Intake-Valve Closing" SAE Paper 820408, *SAE Transactions*, Vol. 91, 1982.

[4] Gray C., "A Review of Variable Engine Valve Timing", SAE Paper No. 880386.

[5] R. W. Weeks and J. J. Moskwa, "Automotive Engine Modelling for Real-time Control Using Matlab/Simulink", In: *The SAE International Congress and Exposition*, SAE paper No. 950417, 1995.

[6] Dr. Martin Pischinger, "A New Opening", In: *Engine Technology International 2001*.

[7] Xu, J.-X. and Z. Bien (1998). The frontiers of iterative learning control. In: *Iterative Learning Control (Z. Bien and J.-X. Xu, Eds.)*. pp. 9-35. Kluwer Academic Press, Boston.

[8] The MathWorks, "Processor-in-the-Loop (PIL) Co-Simulation", In: http://www.mathworks.com/access/helpdesk_r12p1/help/toolbox/mpc555dk/ppc_pil2.shtml#5703

[9] Butzmann S., Melbert J., and Koch A., "Sensorless Control of Electromagnetic Actuators for Variable Valve Train", SAE Paper No. 2000-01-1225.

Fast Internal Combustion Engine Knock Processing Algorithm Using an Automotive PowerPC System-on-a-Chip

Mohamed Anas, R. J. Paling, W. H. Nailon and D. R. S. Cumming

***Abstract*—This paper discusses a novel high performance knock detection strategy using a next generation automotive System-on-a-Chip. The proposed algorithm is based on autonomous on-chip modules and an auxiliary signal processing extension to the main System-on-a-Chip core. Real-time software development techniques with an advanced software circular buffer implementation for processing the streaming knock sensor data have been developed. Various single instruction multiple data software optimisation techniques are employed to reduce the real-time knock algorithmic execution time. Real-time and simulation results are presented for the detection of knock on a four cylinder internal combustion engine, but, the approach is widely applicable. The efficient coding and optimisation techniques used for the single instruction multiple data implementation of the algorithm have been shown to improve computational performance and as a result utilises minimum combustion event timing.**

***Index Terms*—Internal combustion engine knock, SoC, signal detection, signal analysis, software performance, real time systems, FIR digital filters, SIMD, integrators**

I. Introduction

Knock in internal combustion engines (ICE) refer to the premature self or auto ignition of the air-fuel mixture in the engine when the unburnt mixture's temperature and pressure have exceeded a critical point. Frequent occurrence of this knock phenomenon causes permanent damage to the ICE and should be avoided. However, in order to obtain maximum power, modern engines are run at their borderline limit of incipient knock using closed-loop control of spark timing based on knock sensor feedback [1], [2], [3].

In a four stroke engine, during normal drive cycles, the air-fuel mixture in the cylinder is compressed by the engine's piston to a high pressure. A timely spark generated by the mounted sparkplug then ignites the fuel near sparking point, which then creates a flame front. As this flame front propagates throughout the air-fuel mixture, the remainder of the air-fuel mixture is burnt in a highly controlled manner, which gradually increases the cylinder pressure to push the piston downwards. As the piston moves away, the pressure eases. Engine knock occurs when the air-fuel mixture ignites before the flame front can reach it [1]. This uncontrolled ignition of the air-fuel mixture causes it to burn in an irregular and explosive manner. This rapidly expanding mixture exerts a sudden pressure wave, which produces a sizeable force on the surroundings of the combustion chamber [1], [2], [3].

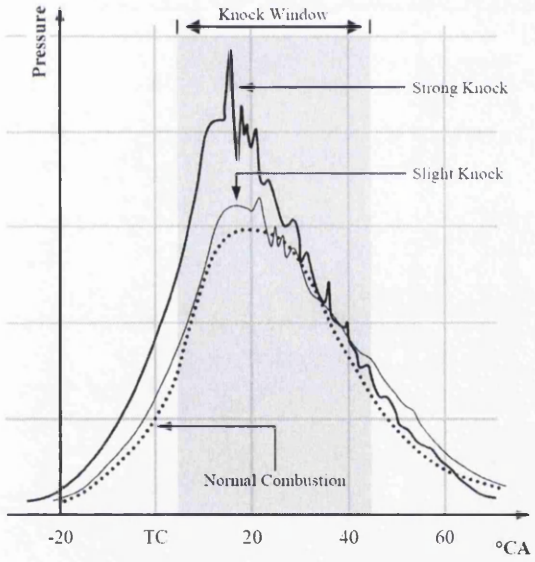

Fig. 1. Cylinder Pressure versus Crank Angle (CA)

In severe cases, the piston may still be moving upwards to compress the air-fuel mixture. As such, it cannot move away to ease the build up of pressure wave. This results in severe stress on the engine and should be prevented as permanent engine damage can occur. Fig. 1 shows three plots of cylinder pressure against crank angle of a single cylinder engine with ignition timing three degrees apart between each trace [1]. ICE knock usually occurs under wide-open-throttle (WOT) operating conditions. As shown in Fig. 1, if the normal combustion cycle is advanced by some three degrees, a slight knock occurs which indicates the engine has reached its limit.

Manuscript received November 02, 2004. This work was fully supported by the 32-bit Embedded Controller Division of Motorola SPS Limited under the industrial EngD sponsorship scheme.

Mohamed Anas and Robin Paling are with the 32-bit Embedded Controller Division, Motorola SPS, Glasgow G75 0TG, U.K. (e-mail: Mohamed.Anas@ieee.org).

W. H. Nailon is with the University of Strathclyde, Glasgow G1 1XQ, U.K.

D. R. S. Cumming is with the University of Glasgow, Glasgow G12 8LT, U.K.

Further timing increase produces severe knock. A common reason for engine knock is using poor quality gasoline with too low an octane rating, which has the tendency to ignite prematurely [1].

Normally, all internal combustion engines are designed to run with a minimum octane rating gasoline. Another source of combustion knock is insufficient cooling of the engine. When the temperature in the engine gets too high, it can trigger fuel to suddenly self-ignite. Even with good cooling of the engine, a poorly designed engine may have "hot spots" that do not get cooled properly. This could be due to recirculation zones, crevices or failure to properly exhaust burnt gases. Yet another possibility is the use of a turbocharger on an engine that is not designed and recommended by the engine manufacturer. Since turbochargers compress air at the engine inlet, pressures in the engine get much higher than the engine was originally designed for. Like high temperatures, an overly high pressure (above 14:1 for gasoline) also triggers the fuel to prematurely self-ignite.

The rest of the paper is organized as follows. Section II provides an overview of the state-of-the-art knock detection systems and the impact of knock in ICEs. Section III describes the developed Single Instruction Multiple Data (SIMD) PowerPC system-on-a-chip (SoC) based hardware platform. Section IV introduces the developed knock detection algorithm based around the SoC. This section particularly highlights the knock sensor, its conditioning circuit and the efficient use of the micro-architectural elements. Section V introduces the overall software platform including SIMD functionality. We present results and discussions with measured data in section VI before concluding the paper highlighting the benefits of the proposed SIMD SoC micro-architecture.

II. Impact of Knock and its Real-time Processing

Impact of knocking in an engine depends on its intensity and duration. Trace knock has no significant effect on engine performance or durability. Heavy knock can lead to extensive engine damage. The engine can be damaged by knock in different ways: piston ring sticking; breakage of the piston rings and lands; failure of the cylinder head gasket; cylinder head erosion; piston crown and top land erosion; piston melting and holing. Knocking is one important factor limiting the efficiency of an engine and is therefore of great importance to the engine manufacturers [1], [2], [3], [4].

There are several different approaches to detect the presence of knock in engines, see e.g. [5], [9] – [15]. One of the classic techniques presently used in production engines is based on application specific integrated circuits (ASIC) with limited programmability, such as the ProSAK™ knock control ASIC [16] and The HIP9011 ASIC [17].

Due to the high cost of direct knock sensors, most of the current knock detection systems are based on structural vibration signals obtained using an accelerometer [13].

This signal is then processed by aforementioned ASICs that include the functionalities, analogue filtering, rectification and integration. The final integrated value obtained is then compared to a heuristically determined threshold to determine the presence of engine knock [5]. Several laboratory based methods[5], [9], [11], [13] – [15] have also been proposed for the extraction of the energy in the resonance frequencies generated by combustion knock, however such methods are not useful for engine manufacturers as they are not cost effective and particularly computationally demanding in production engines. Additionally, there is very little information available on the real-time implementation of such detection methods.

III. SoC Based Hardware Knock Evaluation Platform

A high level overview and a block diagram of the embedded SoC knock evaluation platform is shows in Fig. 2. It employs an MC33394 Power Supply IC [16] and connectivity to other basic optional communication protocols available on the SoC.

The SoC platform has two levels of memory hierarchy. The fastest accesses are to the 32 kB unified cache. The next level in the hierarchy contains the 64 kB on-chip SRAM and 2 MB of internal flash memory. Both the on-chip SRAM and the flash memory can hold instructions and data.

The complex I/O timer functions of the SoC are performed by two enhanced Time Processor Units (eTPU). Off-chip communication is performed by a suite of serial protocols including Controller Area Networks (CAN), enhanced Serial Peripheral Interfaces (SPI) and Serial Communications Interfaces (SCI). The SoC has an on-chip 40-channel enhanced Queued dual Analogue-to-Digital Converter (eQADC).

The SoC core embedded on the evaluation board complies with the classic PowerPC Book E architecture. It is 100% user mode compatible (with floating point library) with the classic PowerPC instruction set [16]. The SoC core consists of a register file with 32 64-bit registers. In addition to the vector instructions used for the development of the algorithm, the PowerPC 32-bit instructions used operate on the lower (least significant) 32 bits of the 64-bit register. The vector instructions defined view the 64-bit registers as vectors of two 32-bit elements, and some of the instructions also read or write 16-bit elements. These instructions are used to perform scalar operations in the algorithm developed.

IV. Proposed SoC Based Knock Processing Methodology

A. *Resonance Frequencies in a Knocking Engine*

The resonant frequencies excited by the presence of knock depend on the geometry of the combustion chamber and the speed of sound in the cylinder charge [1], [2], [3], [4]. These resonant frequencies are typically estimated by assuming an acoustic model for the combustion chamber.

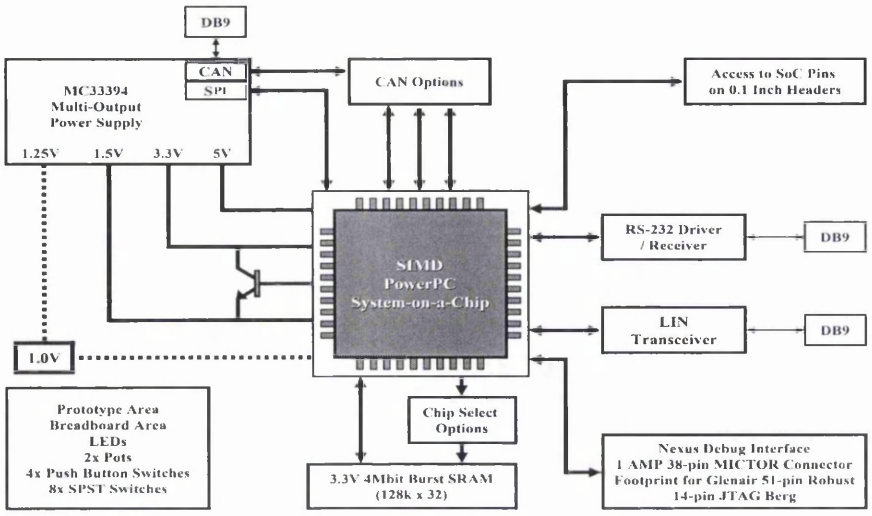

Fig. 2. Overview of Hardware Knock Evaluation Platform

For a homogeneous gas filled, acoustically hard walled ideal cylinder, the resonance frequencies are given by Draper's equation [4].

$$f_{m,n} = \frac{c_0 \sqrt{T} \eta_{m,n}}{\pi B} \quad (1)$$

where $f_{m,n}$ is resonant frequency, $\eta_{m,n}$ is a non-dimensional mode number, c_0 is the phase velocity constant, T is the combustion mixture temperature, B is the cylinder bore diameter and the integer subscriptions m and n denote radial and circumferential mode numbers. Generally, the axial mode is neglected as knock generally occurs when the piston is just past the top dead centre (TDC) position and at this instance; axial dimension is negligible compared to the radial dimension.

B. Knock Sensors

Knock sensors can be partitioned into the categories of direct and remote measurement devices. Direct sensors measure the pressure inside the combustion chamber necessitating each cylinder to encompass a dedicated sensor and such sensors operating in harsh conditions tend to be expensive, whereas the remote tuned or broadband measurement sensors use the vibrations transmitted through the structure of the ICE [13]. Typically, the resonant point of the tuned sensor is centred on the fundamental knocking frequency, which is between 5 kHz and 7 kHz, whereas the bandwidth of broadband sensors tend to be between 1 kHz and 25 kHz and this tends to be the most cost effective since a single sensor could cover a range of resonance frequencies [13].

The basis for the implemented knock processing methodology is based on broadband sensors. However, with minor modifications, the proposed methodology can be used with data obtained using pressure and acoustic information.

Optimal location of accelerometers is heuristically chosen so that the sensor is not in a "dead" area and the transmitted quality of the signal has the highest SNR. The interface circuitry shown in Fig. 5 provides impedance matching for the knock sensors and a highpass filter attenuating frequencies below 1 kHz. An on-chip digital multiplexer, selects the appropriate knock input of the active cylinder. In our evaluation, the sensor was placed in the threaded screw hole on the frame rib intersection of the upper part of the engine as prescribed by the manufacturer.

The developed knock kernel was validated using the structural vibration signal obtained using a piezoelectric accelerometer generated by a four cylinder engine (V4). This signal was acquired at a sampling rate of 50 kHz using a broadband knock sensor with a 25 kHz flat frequency response. In order to avoid engine damage, only 20% of knocking cycles were introduced at 4000 RPM in the engine in a strictly controlled manner. Fig. 3 and Fig. 4 show an example set of the accelerometer signals, highlighting the fundamental knock frequency of the engine used and their power spectral densities respectively.

C. Knock Sensor Data and Algorithmic Flow

Fig. 5 illustrates the knock sensor data and the algorithmic flow. User programmed eQADC commands are contained in the on-chip memory in a user defined data structure. The eQADC command data is moved from the command queue to the command FIFO buffer by either the host CPU or by the enhanced Direct Memory Access (eDMA) controller. Once the command FIFO is triggered and is transferred into the ADCs on chip, the ADC executes the command, and the result, i.e. a pair of time stamp and data is moved through the result FIFO by the eDMA or the host CPU in to the circular buffer in the on chip memory.

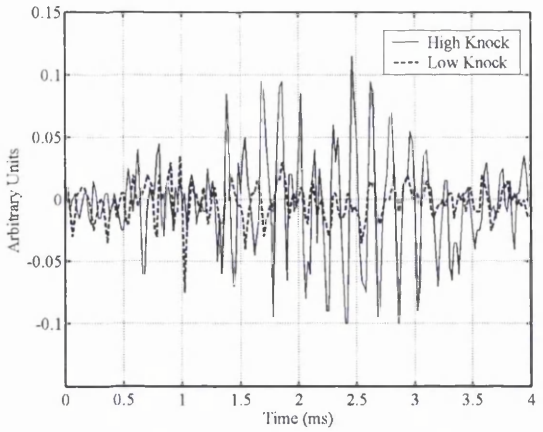

Fig. 3. Accelerometer Signal from External Engine Block

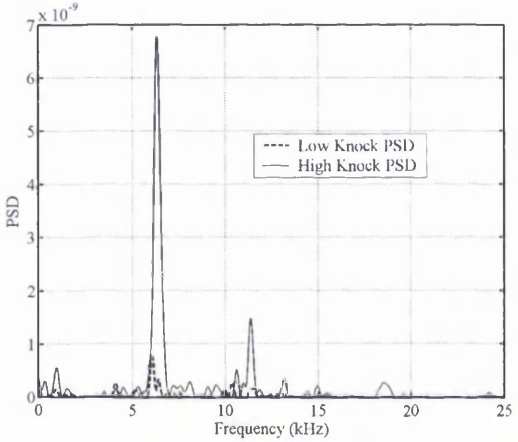

Fig. 4. Power Spectral Densities (PSD) of Fig. 3's Accelerometer Signals

As shown in Fig. 5 , each sampled data point x[n] follows its time stamp t[n] into the circular buffer.

The data in the circular buffer is then processed and presented by the knock task management software threads to the key SIMD knock signal energy extraction elements. As explained in section V, based on the status of the knock task, the stored streaming data is then bandpass SIMD FIR filtered to extract the signal of interest. This bandpassed signal is then squared, integrated and compared to determine the presence of knock.

D. Computationally Efficient Minimum Length Bandpass Filter Design and Analysis

The FIR was chosen because it can provide linear phase over a specified range of frequencies. The input, x(n) and the output y(n) signals to the FIR filter are related by the convolution sum [6].

$$y(n) = \sum_{k=0}^{N-1} h(k)x(n-k) \tag{2}$$

where h(k), k = 0, 1, …, N-1, are the impulse response coefficients of the filter and N is the filter length, that is the number of filter coefficients.

To achieve highest computational efficiency and to reduce design and evaluation time, accurate estimation of the minimal FIR bandpass filter length required for the algorithm was estimated using the following empirical relationship [7], [8]. The parameters used to specify the bandpass filter are, δp, δs – passband and stopband ripples, ΔF – transition bandwidth normalised to sampling frequency, b1 = 0.01201, b2 = 0.09664, b3 = -0.51325, b4 = 0.00203, b5 = -0.57054 and b6 = -0.44314.

$$N = \frac{C_\infty(\delta_p,\delta_s)}{\Delta F} + g(\delta_p,\delta_s)\Delta F + 1 \tag{3}$$

where

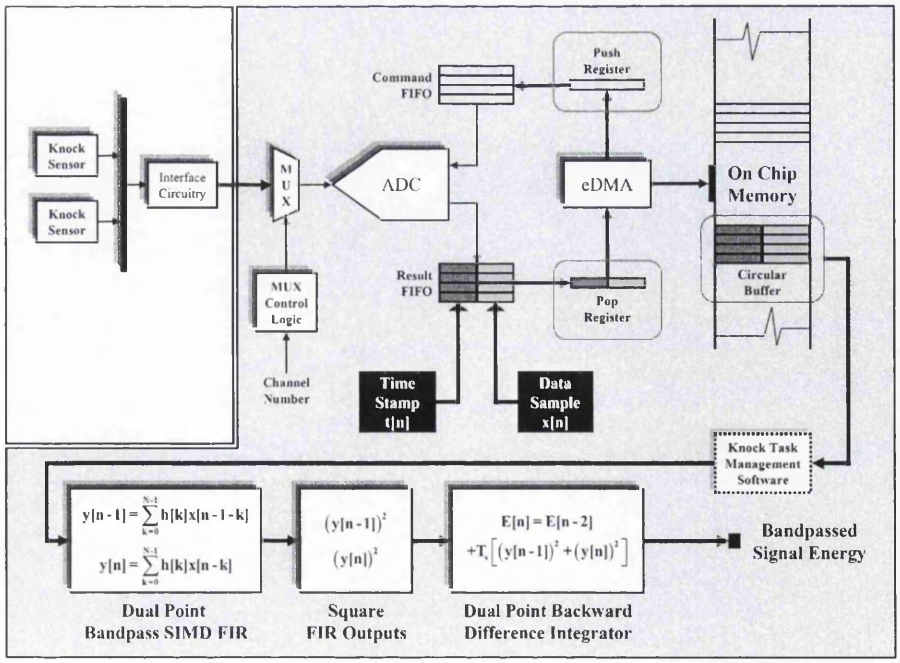

Fig. 5. Data and Algorithmic Flow of the Proposed Knock Kernel

$$C_{-}(\delta_p,\delta_s)=\log_{10}\delta_s\left[b_1\left(\log_{10}\delta_p\right)^2+b_2\log_{10}\delta_p+b_3\right]+\left[b_4\left(\log_{10}\delta_p\right)^2+b_5\log_{10}\delta_p+b_6\right] \quad (4)$$

$$g(\delta_p,\delta_s)=-14.6\log_{10}\left(\frac{\delta_p}{\delta_s}\right)-16.9 \quad (5)$$

FIR coefficients were calculated using windowing, equi-ripple and least square error minimization techniques. The least-squares error minimization scheme produced the desired frequency response and the coefficients obtained were used for the real-time implementation of the bandpass filter.

E. Dual-Point Backward Difference Integrator

The square of the bandpassed FIR output is subjected to a backward difference integrator. For a single point integrator, the output $y[k]$ is given by,

$$y[k]=y[k-1]+T_s x[k] \quad (6)$$

where $y[k-1]$ is the previous integrator output, T_s is the sampling interval and x[k] is the square of the filter output. Computation of present integrator output requires a single or scalar data point and the previous integrator output. However, the SIMD bandpass FIR block developed computes two back-to-back filtered output points simultaneously, which are then squared in parallel. In order to exploit the concurrent availability of two such data points and to reduce the computational burden involved with conversion of vector data points into scalar format for the single point integrator, a two point backward difference integrator was developed and this is given by,

$$y[k]=y[k-2]+T_s\left(x[k]+x[k-1]\right) \quad (7)$$

This integrator saves ~15% of the computational bandwidth compared to that of a single point integrator.

V. Implementation of Key Software and Hardware Elements of the Knock Kernel

A. SIMD Software

Key software elements of the knock processing strategy was developed based around the SIMD functionality supported by the SoC as SIMD works best with arrays of streaming data. This SIMD code was written using the developed high-level programming model which can be used in conjunction with the classic C and C++ languages. This programming model eliminates the issues associated with writing code at the assembly level: register allocation, scheduling, stack management and conformance to the underlying application binary interface (ABI). These SIMD instructions are supported by the signal processing extension (SPE) tightly coupled to the core of the SoC.

The SIMD programming model introduced consists of a set of fundamental data types supporting parallel loading and storing of appropriate data into the 64-bit (__ev64__) vector registers as shown in Table 1.

SIMD 64-bit Data Type	Interpretation of Contents	Values
__ev64_u16__	4 unsigned 16-bit integers	0 to 65535
__ev64_s16__	4 signed 16-bit integers	-32768 to 32767
__ev64_u32__	2 unsigned 32-bit integers	0 to 2^{32}-1
__ev64_s32__	2 signed 32-bit integers	-2^{31} to 2^{31}-1
__ev64_u64__	1 unsigned 64-bit integer	0 to 2^{64}-1
__ev64_s64__	1 signed 64-bit integer	-2^{63} to 2^{63}-1
__ev64_fs__	2 floats	IEEE-754 single-precision values
__ev64_opaque__	any of the above	—

Table 1. SIMD Fundamental Data Types

B. Executing the Developed Knock Kernel

Fig. 6 illustrates a 180° part of a 720° ignition cycle. As shown in this figure, the knock kernel is run once per cylinder combustion event, per engine cycle. The actions illustrated in Fig. 6 are then repeated for each of the other three cylinders of the V4 for a total engine cycle of 720°. Normally, the start of the knock window and the ignition pre-schedule time are both hard coded. Whereas the end of the knock window, the ignition scheduling, and the start and end of the ignition pulse are calculated during the operation of the engine. For example, the end of the knock window is determined by what the speed of the engine is at the start of the window.

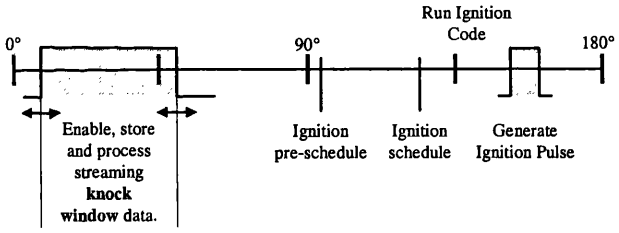

Fig. 6. Key Events in an Ignition Cycle of a Cylinder

C. Implementation of The Frame Based Advanced Software Circular Buffer (ACB) on The On-Chip SRAM Using An Enhanced Direct Memory Access Controller (eDMA) Channel

The software circular buffer scheme implemented to hold the streaming knock data involves the allocation of memory space on the on-chip SRAM of the SoC with sufficient space left over for most of the fast executable portions of the operating system (OS) and other active applications to reside. This buffer mechanism solves many of the problems associated with streaming high-throughput data acquisition on a multi-threaded / multi-tasking operating system. An eDMA channel is used to create the required circular buffer in the on-chip SRAM.

The 32-bit data word per knock window sample stored in the ACB consists of a 16-bit timestamp and the appropriate 16-bit knock window sample. The eDMA channel runs continuously without software intervention.

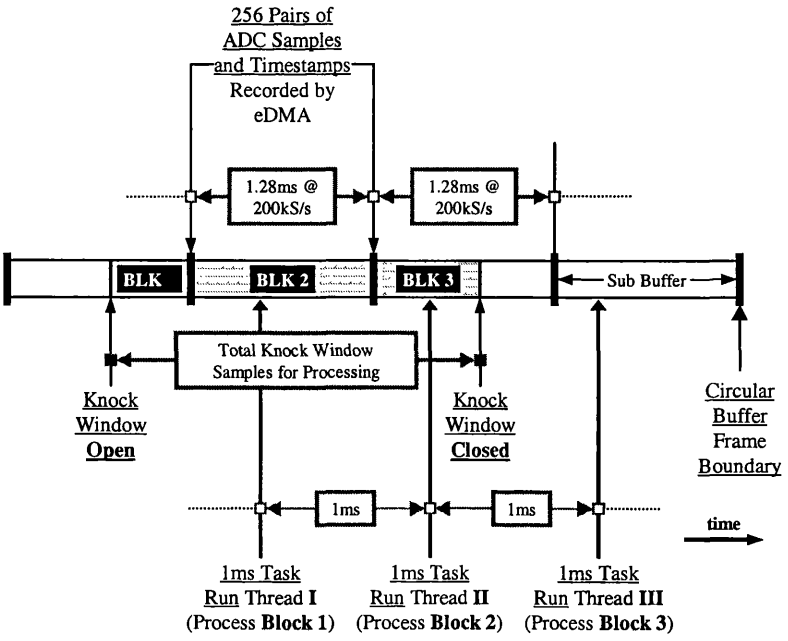

Fig. 7. Advanced Software Circular Buffer and Knock Thread Processing

The ACB shown in Fig. 7 is used for the transfer of data between two processes, the continuous uninterrupted streaming eDMA transfers and the knock software thread processing, explained in section D. The producer process, namely the eDMA transfer, places items into the ACB and the consumer process, namely the knock software threads remove them for the algorithmic processing. The variable capacity of the ACB accommodates timing differences between the producer and the consumer processes.

In the implemented strategy, the software ACB executes faster than other queues that hold a variable amount of data since a fixed size block of memory is allocated just once from memory management and then reused. This circular buffer can be visualised as a linear buffer with indices that wrap, modulo the buffer size, when the end of the buffer is reached.

For the knock detection strategy, the ACB suited the overall implementation due to the decoupling requirement of the independent processes with different speeds. For example, a faster eDMA process can "burst" data into the buffer and continue with its processing. A slower thread, i.e., the consumer of that data can then read it at its own rate without synchronising and slowing the producer. In this type of application, the average rate, over time, of both processes must be the same to avoid an over or under flow condition of the ACB and this is referred to as the synchronous mode of operation.

In addition, sequencing is critical unless an appropriate method is used with the producer process executing first. To prevent unintentional incrementing of the tail pointer, the pointer is incremented only after the application has finished reading the data in the buffer and has indicated that the buffer space is relinquished for the write operation.

D. Knock Processing Threads

The time available for processing the developed knock kernel consists of a scheduling mechanism which contains tasks requiring system resources that should meet all logical and temporal constraints. As the knock algorithmic task is of predictable behaviour, i.e., periodic in nature, static scheduling is used in order to reliably meet all timing constraints. This is of fundamental importance when defining an allocation for hard deadline tasks in such a real-time environment. The knock kernel is portioned into two segments - one containing three threads, which are activated by a 1ms periodic task and the other containing all the background processing. As shown in Fig. 7, the execution of these threads is primarily dependant on the position of the circular buffer write pointer and the status of the knock window. Fig. 7 and Fig. 8 illustrate a typical scenario for thread level processing using the four frames of the segmented ACB. In order to keep the data frame boundaries at fixed buffer locations and to make optimal use of the algorithm, the overall circular buffer size should be a multiple of the frame size. As shown in Fig. 8;

- Thread I: The millisecond task initially enters this thread to begin knock processing. As shown in Fig. 8, this thread is processed only if the knock window is in the open phase and only after a corresponding eDMA data transfer crossing a 256 data frame boundary.
- Thread II: As shown is Fig. 7 and Fig. 8, this thread only processes data packets consisting of 256 elements and is executed only when the knock window is in the open phase.
- Thread III: This concluding thread is responsible for completing the knock widow data processing. The knock window should be in the closed phase for this thread to be invoked and executed.

If a multi-channel eDMA transfer is required, then the frame

size should also be a multiple of the number of channels used. Doing so keeps the pointer arithmetic from becoming unnecessarily complex, which also keeps the core processing cycles to a minimum. Knock window open and closed data points are obtained by subtracting timestamp of the two events from the time obtained using the pre-fixed buffer boundary contents.

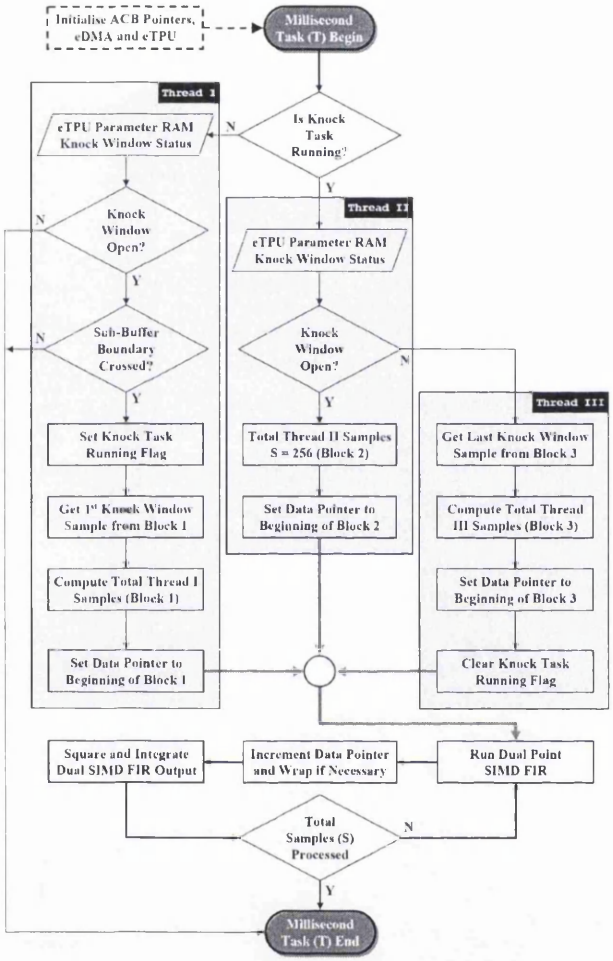

Fig. 8. Thread Level Flowchart of Knock Algorithm

E. Implementation of the SIMD FIR

Key bandwidth intensive function of the knock kernel is the SIMD FIR. The FIR filter is implementable as a sequence of operations "multiply-and-accumulate", often called MAC. As shown in equation (2), in order to run an N^{th} order FIR filter it is necessary to have, at any instant, the current input sample together with the sequence of the N preceding samples. These N samples constitute the memory of the filter. In practical implementations, it is customary to allocate the memory in contiguous cells of the data memory or, in any case, in locations that can be easily accessed sequentially. At every sampling instant, the state must be updated in such a way that the data point x[k] becomes x[k – 1], and this seems to imply a shift of N data words in the filter memory.

Indeed, instead of moving data, for computational efficiency, it is convenient to move the indexes that access the data as shown in the ACB in Fig. 8.

Fig. 9 illustrates the SIMD implementation of the bandpass FIR. The SIMD instruction, a.k.a., intrinsic, used for the implementation of the SIMD FIR inner loop utilises a dedicated fast hardware multiply accumulate (MAC) unit incorporated in the SPE, which allows back-to-back execution of dependent MAC instructions.

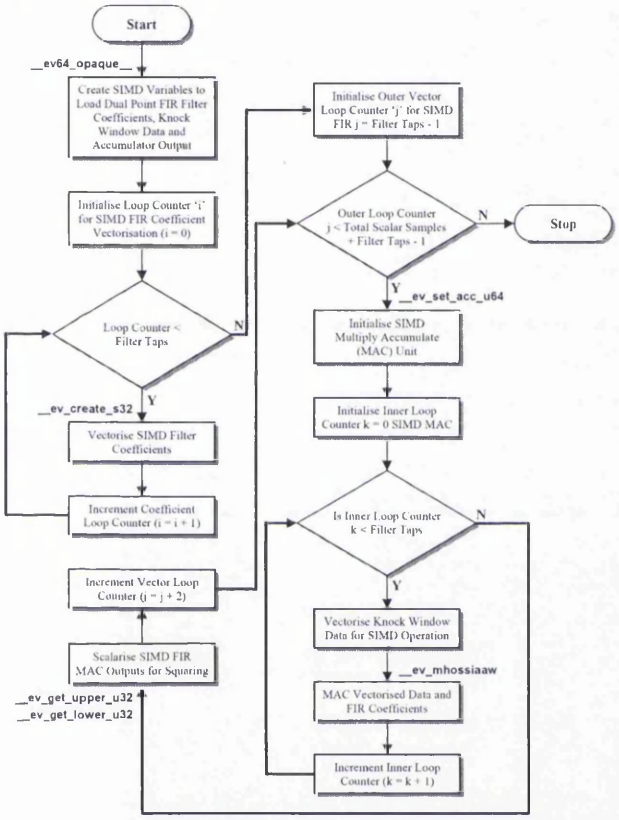

Fig. 9. Software Implementation of the Dual Point SIMD FIR

Fig. 10 illustrates this hardware MAC instruction, evmhossiaaw rD, rA, rB (vector multiply half words, odd, signed, saturate, integer and accumulate into words). In this instruction, the least significant 16 bits of rA and rB are multiplied for both elements of the vector and the result is shifted left one bit and added to the accumulator and the result is possibly saturated to 32 bits in case of overflow.

The final result is placed both in the accumulator and also in rD so that the result of this instruction can be used by accessing rD. As shown in Fig. 10, The 64-bit architectural accumulator register, rD, holds the results of the MAC operation. The accumulator is partially visible to the programmer in that its results do not have to be explicitly read to use them. Instead, for computational efficiency, they are always copied into a 64-bit destination GPR specified as part of the instruction. The accumulator however has to be explicitly cleared when starting a new MAC loop. Based upon the type of intrinsic used, an accumulator can hold either a single 64-bit value or a vector of two 32-bit elements.

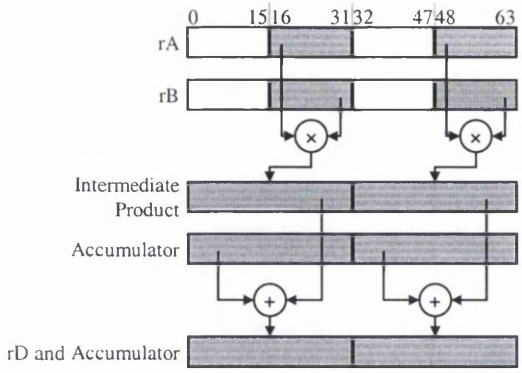

Fig. 10. Fast SIMD Vector Hardware MAC

VI. Measured Results and Discussions

Table 2 and Table 3 show the measured cylinder combustion event bandwidth required to process the knock threads using the evaluation platform described in section III. In both tables, performance was evaluated by running the SoC at 132MHz at a sampling rate of 200kSamples/sec.

RPM	Thread Level Execution Timing in μs						% Combustion Event Time Used for Thread Processing	
	15° Knock Window Threads			40° Knock Window Threads				
	I	II	III	I	II	III	15°	40°
500	22	284	66	58	843	77	0.62	1.63
2000	72		12	28	191	32	0.63	1.67
2500	60		16	36	97	68	0.63	1.68
4000	28	NA	21	25	97	6	0.64	1.71
6000	11		22	33	NA	51	0.66	1.69
8000	16		10	63		1	0.68	1.71

Table 2. Bandwidth Required to Process Knock Algorithmic Threads with SIMD Code

RPM	Thread Level Execution Timing in μs						% Combustion Event Time Used for Thread Processing	
	15° Knock Window Threads			40° Knock Window Threads				
	I	II	III	I	II	III	15°	40°
500	53	711	164	145	2000	192	1.55	4.11
2000	179		55	68	474	78	1.55	4.13
2500	149		38	90	238	169	1.56	4.14
4000	68	NA	50	60	238	14	1.57	4.16
6000	27		53	82	NA	127	1.60	4.18
8000	38		22	156		0	1.61	4.17

Table 3. Bandwidth Required to Process Knock Algorithmic Threads with Standard C and Assembly Code

For example, in Table 2, at 500rpm, a 15° knock window task requires 372 (22+284+66) μs to process the knock kernel, which is 0.62% of the overall cylindrical combustion event bandwidth of the V4.

The results obtained are based on a single-band bandpass filter applied to the primary knocking resonance frequency component. The measured integrator output shown in Fig. 11 consists of both the noise and the appropriate resonance components in the band of interest. The normalised time in this figure shows the time required to produce the appropriate integrator output, a.k.a., knock index.

The knock free index, in other words, the pure background engine noise in the absence of knock is adaptively averaged and updated over time and this serves as the acceptable no knock threshold of the appropriate cylinder firing cycle [13].

As shown in [10], signal-to-noise-ratio of the overall process can be improved by selectively switching a multiband bandpass filter at higher engine speeds to extract all present knock resonance frequencies. Such a multiband filter can be designed using constrained least squares FIR filter design technique [18], [19].

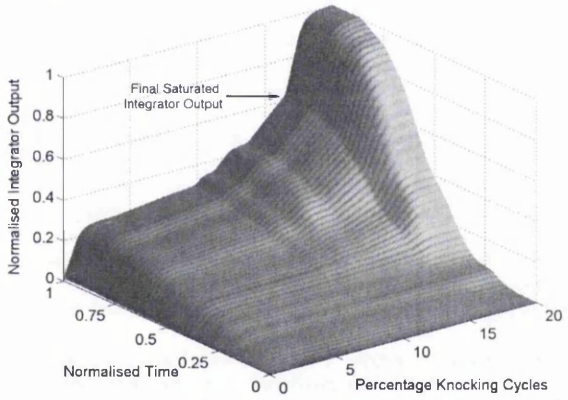

Fig. 11. Dual Point Integrator Output

We have shown that using a common architecture for both RISC and DSP instructions, in combination with autonomous on-chip peripherals, allows complex systems to be built around a single SoC platform, where previously two or more different processors would have been used together [13], [15]. Based on our results, it is also evident that real SIMD computers need to have a mixture of single instruction single data (SISD) and SIMD instructions. Importance of SISD elements in the micro-core to perform operations such as branches and address calculations that do not need parallel operation is also highlighted. It is also worth nothing that for efficient dynamic power management and flexibility, unused individual execution units of the SoC are disabled during algorithmic execution.

Our analysis confirms that SIMD works best in dealing with arrays of streaming data. Additionally, in the proposed architecture, sustained MAC instructions are executed in a single CPU cycle. In contrast, in a typical fixed-point microprocessor, a multiply and an add typically executes in 15 to 20 CPU cycles [16].

The SIMD unit implemented also significantly increases execution speed by performing multiple operations in parallel. For instance, in the same instruction cycle that a MAC operation is performed, a parallel data move is carried out. SIMD enhancements in the SoC supplement the computational speed of present generation real-time processors used and make them ideal for high-performance real-time applications.

As shown in Fig. 12, computational bandwidth is what separates the SIMD based core from the classic CPU – the ability to process an abundance of data, consistently, in an uninterrupted stream. Our measured results in Fig. 12 confirms that the efficient coding and optimisation techniques used for the SIMD implementation of the knock kernel have improved performance by a minimum of x1.8. This figure is obtained by comparing the overall normalised knock task processing time, i.e., sum of time taken to run all three knock threads, shown in Fig. 8, with and without SIMD.

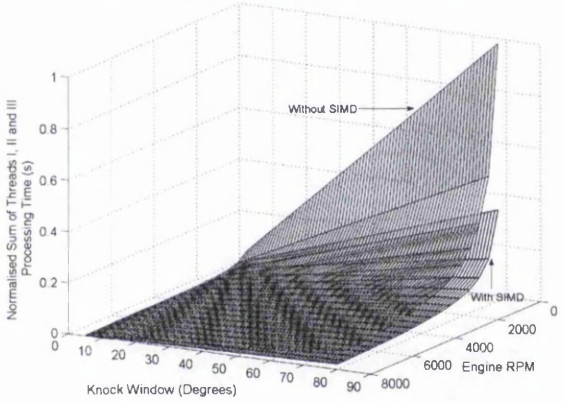

Fig. 12. Comparison of Knock Kernel Processing Time with and without SIMD.

VII. Conclusions

Streaming knock processing constitutes a significant part of current day average microprocessor workloads. To address this, a SoC combining SIMD DSP functionality with a classic microprocessor core has been developed. Peripherals on this SoC designed use fast register and memory based communication and synchronization mechanisms to deliver high performance. Memory based communication and synchronization is realised using the eDMA module. Parallelism in this application is exploited using a combination of orthogonal parallel processing techniques, namely instruction and data level parallelism (ILP and DLP).

A novel high performance knock detection strategy on an automotive SoC has been presented and the efficient use of various intelligent autonomous modules on the Motorola's next generation automotive SoC combining ILP and DLP in the form of SIMD parallelism are described. The change from standard scalar 32-bit to SIMD vector based cores has been established, which allows the implementation of advanced algorithms with minimal processing time.

References

[1] J. B. Heywood, Internal Combustion Engine Fundamentals, New York: McGraw-Hill, 1988.

[2] Taylor, C. F., The Internal Combustion Engine in Theory and Practice, Volume II: Combustion, Fuels, Materials, Design, M.I.T. Press, MA, 1985

[3] Ferguson, C. R., Internal Combustion Engines: Applied Thermosciences, Wiley and Sons, NY, 1986.

[4] C. S. Draper, Pressure waves accompanying detonation in engines, Journal of Aeronautical Science, vol. 5, pp. 219-226, 1938.

[5] S. Ortmann, M. Rychetsky, M. Glesner, R. Groppo, P. Tubetti, G. Morra, Engine knock estimation using neural networks based on a real-world database, SAE Technical Paper 980513.

[6] E. C. Ifeachor and B.W. Jervis, Digital Signal Processing, A Practical Approach, Reading, MA: Addison-Wesley Ltd., pp. 278–373.

[7] F. Mintzer and B. Liu, "An estimate of the order of an optimum FIR bandpass digital filter," in Proc. IEEE 1978 Int. Conf. on Acoust., Speech, Signal Processing, May 1978, pp. 483-486.

[8] F. Mintzer and B. Liu, Practical Design Rules for Optimum FIR Bandpass Digital Filters, IEEE Transactions on Acoustics, Speech and Signal Processing, vol. ASSP-27, no. 2, April 1979.

[9] M. D. Checkel, J. D. Dale, Computerised knock detection from engine pressure records, SAE Technical Paper 860028.

[10] M. Kaneyasu et al., Engine knock detection using multi-spectrum method, SAE Technical Paper 920702.

[11] B. Samimy, G. Rizzoni, Engine knock analysis and detection using time frequency analysis, SAE Technical Paper 960618.

[12] Hickling, R., Feldmaier, D. A., Chen, F. H. K., Morel, J. S., Cavity Resonances in Engine Combustion Chambers and Some Applications, Journal of Acoustical Society of America, vol. 73, no. 4, pp. 1170-1178, 1983.

[13] J. Wagner, J. Keane, R. Koseluk, and W. Whitlock, Engine Knock Detection: Products, Tools, and Emerging Research, SAE Technical Paper 980522.

[14] S. Carstens-Behrens and J.F. Bohme, "Fast Knock Detection using Pattern Signals", IEEE Int'l Conf. on Acoustics, Speech, and Signal Processing Proceeding, v. 5, 2001, p 3145-3148.

[15] A. Kirschbaum, S. Ortmann, and M. Glesner, "Rapid Prototyping of a Co-processor based Engine Knock Detection System", Proceedings of the International Workshop on Rapid System Prototyping, 1998, p 124-129.

[16] Motorola Semiconductor Products Sector, [Online], Available: http://www.motorola.com/automotive/products.html

[17] Intersil Corporation, High Performance Analog Division [Online], Available: http://www.intersil.com/product_tree/product_tree.asp?x=9

[18] G. Cortelazzo, M. Lightner, and W. Jenkins, Frequency domain design of multiband FIR filters based on the minimax criterion," in Proc. IEEE International Symposium on Circuits and Systems, (Chicago, Illinois), pp. 532-535, April 1981.

[19] Ivan Selesnick, Markus Lang, and C. Sidney Burrus, A Modified Algorithm for Constrained Least Square Design of Multiband FIR Filters with Specified Transition Bands," IEEE Transactions on Signal Processing, vol. 46, issue:2, Feb. 1998, pp. 497 – 501.

[20] MotoTron [Online], Available: http://www.mototron.com/products/

Mohamed Anas (M'00) received the B.Eng. (Hons) degree in Aberdeen, U.K., in 1998. He is currently with the 32-bit Embedded Controller Division of Motorola SPS completing the Doctor of Engineering (EngD) with System Level Integration, an academic collaborative venture of four of the UK's universities: Edinburgh, Glasgow, Heriot-Watt, and Strathclyde.

His current research involves the design of Systems-on-a-Chip, CPU architectural enhancements, real-time software and DSP algorithms for the next generation automotive platforms.

He has previously worked on various design and development of ASICs, real-time DSP algorithms and operating systems.

Dr. Anas is a member of the IEE and SAE. He received the IEE award for best undergraduate student, the Scottish-Hydro Electric award for best overall performance and the IEE international research award in 1997, 1998 and 2002 respectively.

Robin Paling received the B.Sc. (Hons) degree from the University of Glasgow, Glasgow, U.K., in 1984 and the MBA degree from the University of Edinburgh, Edinburgh, U.K., in 1990.

He is currently Systems Engineering Manager at Motorola in East Kilbride, Scotland. He presently specialises in the definition of 32-bit microcontrollers targeted at automotive applications. He has previously worked on ALU design for digital audio processing at Philips Electronics and ISDN communications device design at Fujitsu Microelectronics.

D. R. S. Cumming (M'97) received the B.Eng. degree from University of Glasgow, Glasgow, U.K., in 1989 and the Ph.D. degree from Cambridge University, Cambridge, U.K., in 1993.

Professor Cumming has worked on mesoscopic device physics, radio frequency (RF) characterization of novel devices, fabrication of diffractive optics for optical and submillimeter-wave applications, and microelectronic design. He is presently a Senior Lecturer and EPSRC Advanced Research Fellow in Electronics and Electrical Engineering at the University of Glasgow, where he leads the Microsystem Technology Group.

W. H. Nailon (M'04) received the B.Eng. degree from the University of Glasgow, Glasgow, U.K., in 1994 and the Ph.D. degree from the University of Edinburgh, Edinburgh, U.K., in 1997.

He is presently a Lecturer in the Department of Electronic and Electrical Engineering, University of Strathclyde, Glasgow, U.K. His current field of research is in medical image and signal processing, biochemical diagnostic detection methods, microelectronic design, and investigation of the diverse patterns of pathology characteristic of neurological disorders at microscopic and macroscopic levels.