Performance Analysis of ARM Cores and Use of Cortex-M3 for realisation of IoT application

Major Project Report

Submitted in fulfillment of the requirements for the degree of

Master of Technology
in
Electronics & Communication Engineering
(Embedded Systems)

By

Manas Dodya
(13MECE04)

Electronics & Communication Engineering Branch
Electrical Engineering Department
Institute of Technology
Nirma University
Ahmedabad-382 481

May 2015

Performance Analysis of ARM Cores and Use of Cortex-M3 for realisation of IoT application

Major Project Report

Submitted in fulfillment of the requirements for the degree of

Master of Technology in Electronics & Communication Engineering (Embedded Systems)

By

Manas Dodya (13MECE04)

Under the guidance of

External Project Guide:

Mr. ShankarGanesh Kandasamy
ARM Embedded Technologies Pvt. Ltd.
Mr. Kedar Sovai
Marvell India Pvt. Ltd.

Internal Project Guide:

Prof. Dhaval Shah
Assistant Professor, EC Department,
EE Branch, Institute of Technology,
Nirma University, Ahmedabad.

Electronics & Communication Engineering Branch
Electrical Engineering Department
Institute of Technology
Nirma University
Ahmedabad-382 481
May 2015

Declaration

This is to certify that

a. The thesis comprises my original work towards the degree of Master of Technology in Embedded Systems at Nirma University and has not been submitted elsewhere for a degree.

b. Due acknowledgment has been made in the text to all other material used.

- Manas Dodya

13MECE04

Disclaimer

Certificate

This is to certify that the Major Project entitled **“Performance Analysis of ARM Cores and Use of Cortex-M3 for realisation of IoT application”** submitted by **Manas M. Dodya (13MECE04)**, towards the partial fulfillment of the requirements for the degree of Master of Technology in Embedded Systems, Nirma University, Ahmadabad is the record of work carried out by him under our supervision and guidance. In our opinion, the submitted work has reached a level required for being accepted for examination.The results embodied in this major project, to the best of our knowledge,haven’t been submitted to any other university or institution for award of any degree or diploma.

Date: Place: Ahmedabad

Prof.Dhaval Shah
Internal Guide

Dr. N.P. Gajjar
Program Coordinator

Dr. D.K.Kothari
Section Head, EC

Dr. P.N.Tekwani
Head of EE Dept.

Dr. K. Kotecha
Director, IT

Acknowledgements

I would like to express my gratitude and sincere thanks to **Dr.P.N.Tekwani**, Head of Electrical Engineering Department, and **Dr.N.P.Gajjar**, PG Coordinator of M.Tech Embedded Systems program for allowing me to undertake this thesis work and for his guidelines during the review process.

I take this opportunity to express my profound gratitude and deep regards to **Prof.Dhaval Shah**, guide of my major project for his exemplary guidance, monitoring and constant encouragement throughout the course of this thesis. The blessing, help and guidance given by him time to time shall carry me a long way in the journey of life on which I am about to embark.

I would take this opportunity to express a deep sense of gratitude to my Project Manager **Mr.ShankarGanesh Kandasamy**(ARM) and **Mr.Kedar Sovani**(Marvell) for their cordial support and for providing valuable information regarding the project and guidance, which helped me in completing this task through various stages.I would also thank to **Mr.Praveen Arkeri**, **Mr.Biswa Singh**, **Mr.Raj Zalaria**, **Mr.Nirav Desai**, **Mr.Shantanu Potdar**, **Mr.Mahavir Jain**, **Ms.Nikita Nerkar** and **Mr.Amit Sheth** for always giving good suggestions and solving my doubts to complete my project in better way.

Lastly, I thank almighty, my parents and friends for their constant encouragement without which this assignment would not be possible.

- Manas Madhusudan Dodya
13MECE04

Abstract

As x86-64 processors have already became the most chosen CPU for the personal computers or desktops, it leads us to understand the performance impact of migrating to 64-bit environment from 32-bit environment in cases of mobile and portable devices. The applications that can explore large memory address space always get benefit from 64-bit environment in terms of performance. But still for the applications which does not require such large memory space and are able to fit into 32-bit environment, it is interesting to know the effect on the overall performance, if it can get improvement or not when shifted to 64-bit environment platform. In this report, the description regarding various performance metrics for benchmarks compiled for 32-bit platform and 64-bit platform that run on the same platform having ARM v8-A architecture based system is given.

In case of IoT devices, the higher speed of execution and low power consumption is always desirable. Here, we use two versions of cortex-M3 micro-controllers running on 200 MHz and 300 MHz to realize a serial to wi-fi converter IoT application and see how the performance is benefited by the use of 300 MHz cortex-M3 micro-controller in terms of power consumption. The serial to wi-fi converter IoT application is brought up on 300 MHz chip and its performance is validated for various test scenarios which can check the functionality of serial to wi-fi converter IoT application.

Contents

List of Tables

List of Figures

Abbreviation Notation and Nomenclature

BX	Branch Execution
CPI	Cycles Per instructios
CX	Complex Execution
DTIM	Data Trafic Indication Map
LS	Load Sore
MX	Multi-cycle Execution
PS	Power Save
PMU	Performance Monitoring Unit
SP	Special Purpose Registers
SPEC	Standard Performance Evolution Corporation
SX	Single-cycle Execution
TBTT	Target Becaon Transmission Time

Chapter 1

Introduction

1.1 Motivation

Personal computers have already shifted from 32-bit to 64-bit environment and providing the better performance in terms of speed of execution. It creates the anxiety to know if the same speed-up enhancement can be obtained by moving to 64-bit environment in case of portable devices and mobile platform. This defines the need to do the performance analysis of ARM v8 architecture to know if any benefit can be exploited from this architecture working in 64-bit mode.

IoT is a new and emerging thing in todays world. New devices that will come in consumer electronics market will already be capable of interacting with other devices over the internet as an IoT device. But what can be done to bring the devices that are existing and not able to connect and communicate with other devices. This curiosity leads to create an application serial to wi-fi converter which can bring the existing devices into an IoT eco-system.

1.2 Problem Statement

Main objective of this project is to do analysis of ARM cores and try to make use of them for enhancement of IoT application. Performance analysis of ARM v8-A architecture is done using SPEC CPU2000 integer benchmarks. While doing the performance analysis, benchmarks are compiled on one single platform for 32-bit and 64-bit environment and the focus is on various performance metrics. Performance metrics such as Speed up, Instruction mix, Cycles break-up, CPI, Cache miss rate and branch miss-prediction rate are to be captured and analysed. Using all these performance metrics, predict the influence of moving to a 64-bit mode from a 32-bit mode.

ARM Cortex M Series cores can be used for bringing up the existing device into a new eco-system of connected devices under IoT with the use of serial to wi-fi converter application. The main objective is to bring up the serial to wi-fi application on to a new platform with understanding the need of upgrading from one development platform to another for reducing the overall power consumption and increasing the performance efficiency of IoT chip from Marvell.

1.3 Overview

SPEC suite has become de facto standard for benchmarking the recent high performance processors. SPEC stands for Standard Performance Evaluation Corporation. It is the organization that defines and maintains a set of some standard benchmarks for computer systems and makes them available to the users of such systems as a common reference point in the evaluation of computer performance. It is participated by computer manufacturers, system integrators, consultants, publishers, universities and research organizations [1]. Since its foundation in 1988, the SPEC consortium has developed and distributed technically reliable benchmarks based on real applications. The selection of inputs and workload is performed by the consensus amongst consortium members willing for a transparent, comparable, reproducible and non-proprietary solution.

The details of this thesis report are confined to integer benchmark suit because they are helpful for CPU designers and researchers. This report presents detailed analysis of SPEC CPU 2000 integer benchmarks running on ARM v8A processors and showcases the important characteristics of workloads. We compare performance of 32-bit and 64-bit architecture with the help of SPEC CPU2000 integer benchmarks and identify the key performance behaviour of various workloads.

Single platform is able to provide an environment to run 32-bit code and 64-bit code. It will be good to find out the major benefits of upgrading to 64-bit. The aim is to analyse the performance of a particular set of popular applications in these modes.

New devices that are being manufactured in recent times are coming with integrated Wi-Fi chip to communicate with other devices. In coming future, there will be plethora of electronics devices that can be controlled over internet forming an eco-system named Internet of Thing. In this scenario, existing devises may fall back in race of inventions. But to bring such devices in the same harmony, there is need of a utility which can allow them to communicate and exchange data with other devices. This can be done for the devices which are already having micro-controllers embedded in them with the help of serial to Wi-Fi converter. Serial to Wi-Fi converter is a module that provides WLAN connectivity to other micro-controller based systems. A host microcontroller can communicate with this module over UART interface. The communication happens with the use of simple ASCII text based communication.

1.4 Thesis Organization

The rest of the thesis organized as follows.

Chapter 2 deals with brief description of ARM v8-A architecture, some of its features, some details about SPEC CPU2000 Integer Benchmarks, Cortex-M3 features, serial to wi-fi converter application and wi-fi power measurement techniques.

Chapter 3 deals with the measurement set-up and the technique used for performance analysis of ARM v8-A architecture and performance analysis of Cortex-M3 used in an IoT application.

Chapter 4 deals with the performance analysis of ARM v8-A architecture based on performance metrics and performance analysis of Cortex-M3 for the power consumption in various wi-fi modes.

Chapter 5 deals with the individual benchmark performance characterization giving analysis of five benchmarks which have shown remarkable performance variation between 64-bit mode and 32-bit mode.

Chapter 6 gives analysis summary of performance analysis of ARM v8-A architecture and points out some common benefits and some potential pitfalls of moving to 64-bit environment. Also summary of performance in terms of power consumption of cortex-M3 micro-controller used for IoT application is mentioned.

Chapter 7 gives various test scenarios that are necessary to check the functionality of serial to wi-fi converter application.

Chapter 8 concludes this report regarding performance analysis.

Chapter 2

Literature Survey

This chapter gives the brief overview of ARM v8-A Architecture and its evolution from its earlier versions and this chapter also explains about cortex M3 microcontroller. The ARM architecture is Reduced Instruction Set Computer (RISC) architecture. Since some recent years, there is tremendous increase in number of various smart phones and tablets. Because of this increase, ARM processors are becoming very popular. The main reason behind this is reduction in costs and higher power efficiency.

Complex Instruction Set Computer (CISC) processors, have a rich instruction set which can perform tedious operations using a single instruction. In contrast to CISC, RISC architectures have lesser number of general purpose instructions that might be executed with significantly fewer transistors making the silicon cheaper and more power efficient. ARM cores have a higher count of general-purpose registers and many instructions execute by consuming a single clock cycle. RISC architectures main features [2]:

- The **load/store** architecture only allows memory to be accessed by load and store operations and other operations uses values stored in registers only.
- Another specific difference with CISC architectures is that, when there is a

call to **Branch and Link** then instead of storing the return address in the stack, it is stored in a special register.

2.1 Processors

A short ARM processors list [2]:

1 **Cortex-A:** with DSP, Floating Point, TrustZone, Jazelle extensions. ARMv5, ARM6(2001)

2 **Cortex-M:** ARM Thumb-2 technology which provides excellent code density. With Thumb-2 technology, the Cortex-M processors support a fundamental base of 16-bit Thumb instructions, extended to include more powerful 32-bit instructions. First Multi-core.(2004)

3 **Cortex-R:** ARMv7 deeply pipelined micro-architecture, Flexible Multi-Processor Core (MPCore) configurations: Symmetric Multi-Processing (SMP) and Asymmetric Multi-Processing (AMP), LPAE extension.

4 **ARMv8-A:**64bit with load-acquire and store-release features, which are an excellent match for the C++11, C11 and Java memory models. (2011)

2.2 Extensions

With every new version of ARM, therere new extensions provided, the v8 architecture has these [2]:

- **Floating Point:** for floating point operations
- **Neon:** the ARM SIMD 128 bit (Single instruction, multiple data) engine and **DSP** the SIMD 32bit engine useful to make linear algebra operations

- **Cryptographic Extension** is an extension of the SIMD support and operates on the vector register file. It provides instructions for the acceleration of encryption and decryption to support the following: AES, SHA1, SHA2-256.
- **Virtualization Extensions** with the Large Physical Address Extension (LPAE) enable the efficient implementation of virtual machine hypervisors for ARM architecture compliant processors.
  - The Large Physical Address extension provides the means for each of the software environments to utilize efficiently the available physical memory when handling large amounts of data.

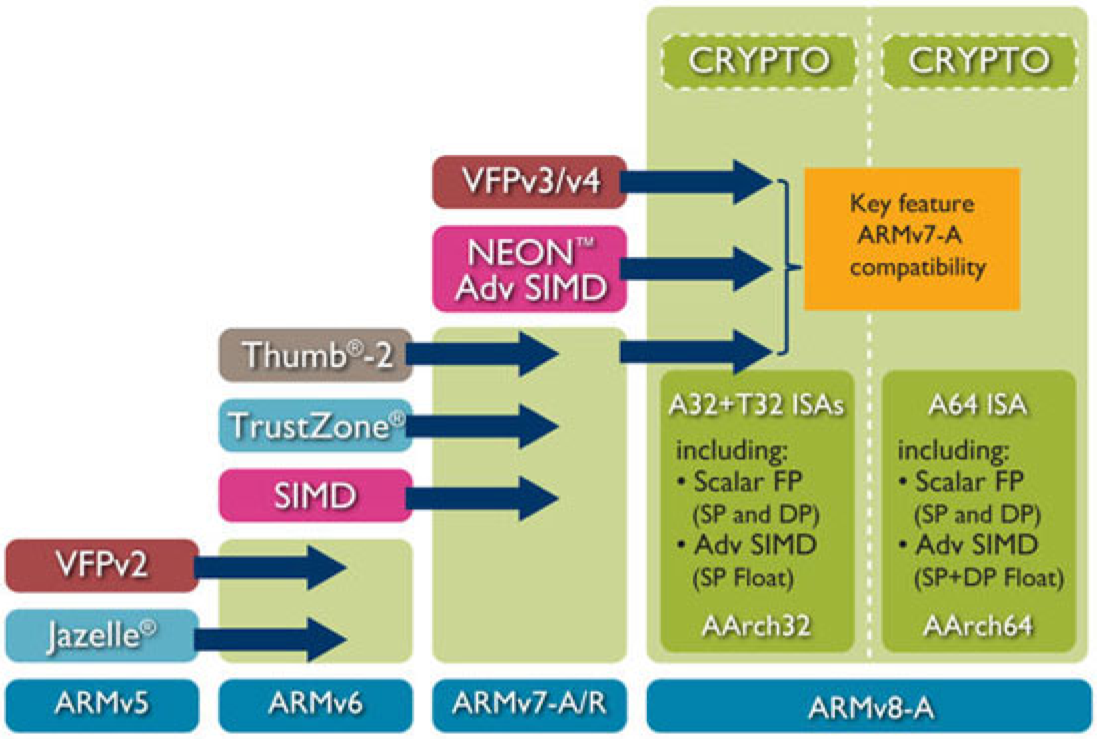
CRYPTO
CRYPTO
VFPv3/v4
NEON™ Adv SIMD
Key feature ARMv7-A compatibility
Thumb®-2
TrustZone®
SIMD
VFPv2
Jazelle®
A32+T32 ISAs including: • Scalar FP (SP and DP) • Adv SIMD (SP Float)
AArch32
A64 ISA including: • Scalar FP (SP and DP) • Adv SIMD (SP+DP Float)
AArch64
ARMv5
ARMv6
ARMv7-A/R
ARMv8-A

Figure 2.1: ARM Architecture Evolution [3]

2.3 Architecture Features

- **AArch64** the ARMv8-A 64-bit execution state, that uses 31 64-bit general purpose registers (R0-R30), and a 64-bit program counter (PC), stack pointer (SP), and exception link registers (ELR). Provides 32 128-bit registers for SIMD vector and scalar floating-point support (V0-V31) A64 instructions have a fixed length of 32 bits and are always little-endian.
- **AArch32** is the ARMv8-A 32-bit execution state, that uses 13 32-bit general purpose registers (R0-R12), a 32-bit program counter (PC), stack pointer (SP), and link register (LR). Provides 32 64-bit registers for Advanced SIMD vector and scalar floating-point support. AArch32 execution state provides a choice of two instruction sets, A32 (ARM) and T32 (Thumb2). Operation in AArch32 state is compatible with ARMv7-A operation.

2.4 Instruction Set Features

- New fixed length instruction set.
  - Instructions are 32-bits in size.
  - Clean decode table based on 5-bit register specifiers.
- Instruction semantics broadly the same as AARCH32.
- 31 general purpose registers accessible all the time.
- Improved performance and energy
- General purpose registers are 64-bits wide.
- No banking of general purpose registers
- Stack pointer and PC are not general purpose registers.

- Additional dedicated zero registers are available for most instructions.
- New instructions support 64-bit operands
- Addresses are assumed to be 64-bit in size.
- LP64 and LLP64 are primary data models targeted
- Far fewer conditional instructions than in AArch32 such as conditional Branches, compares, selects
- No arbitrary length load/store multiple instructions.
- LD/ST 'P'for handling pairs of registers added.
- A64 Advanced SIMD and FP semantically similar to A32
  - Advanced SIMD shares the floating point register files as in AArch32.
- A64 provides three major functional enhancements.
  - More 128 bit registers: 32 x 128 bit wide registers
  - Advanced SIMD supports DP floating-point execution
  - Advanced SIMD supports full IEEE754 execution
- Instruction level support for cryptography.

2.5 Performance Monitoring Unit (PMU)

Performance Monitoring Unit is found on all medium and high end processors. This unit is actually hardware unit that is situated inside processor to measure its hardware and software performance measurement events. We can measure parameters like branch miss, clock cycle, total instruction executed, cache miss and many other based on the hardware provided by the processor.

2.5.1 PMU Functional Description

The figure 2.2 below shows the PMU Block Diagram.

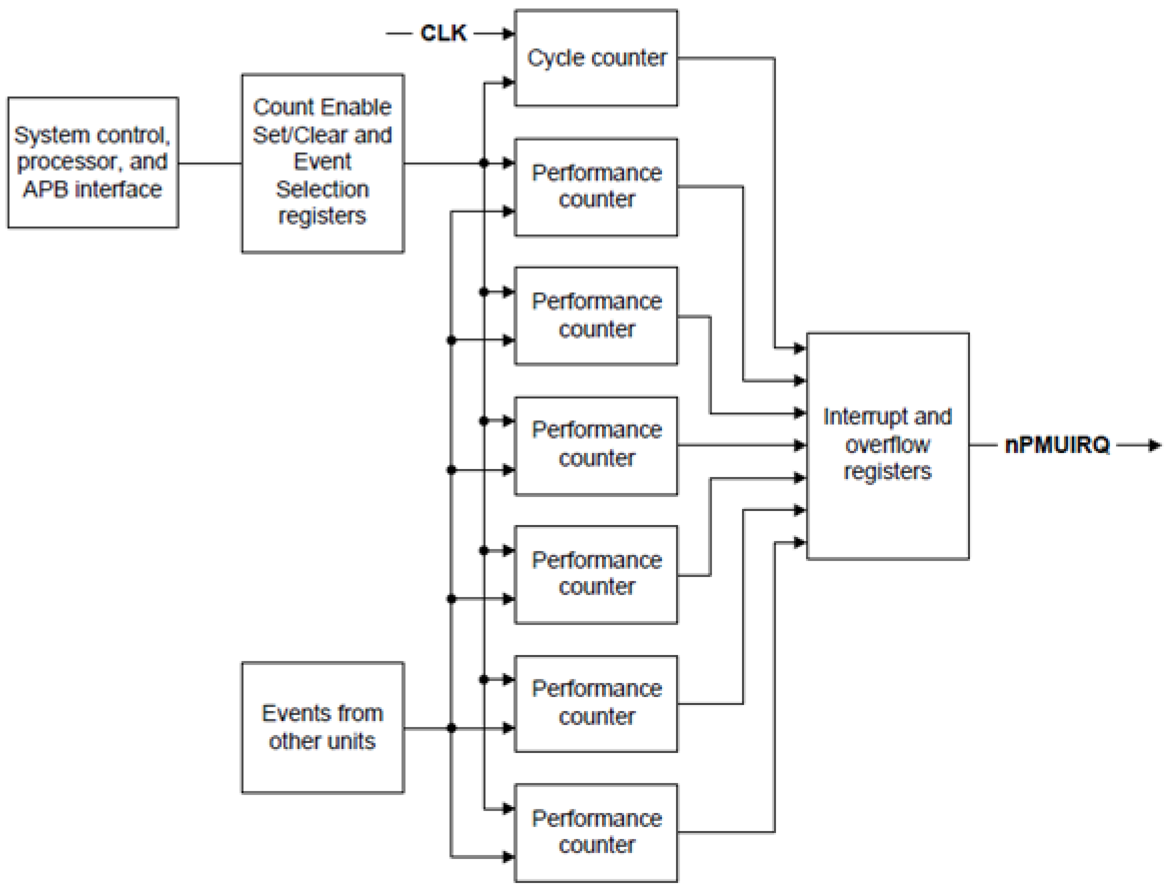

Figure 2.2: PMU Block Diagram [4]

- **Event Interface:** Events from all other units from across the design are provided to PMU.
- **System Register and APB interface:** We can program PMU register using system registers or external APB interface.
- **Counters:** There are 6 counters available. Each counter can count any of the PMU event available for the processor

2.5.2 PMU Events

There are different PMU events which are used for monitoring the performance. Basically the PMU events are divided into two types as shown in figure 2.3 below.

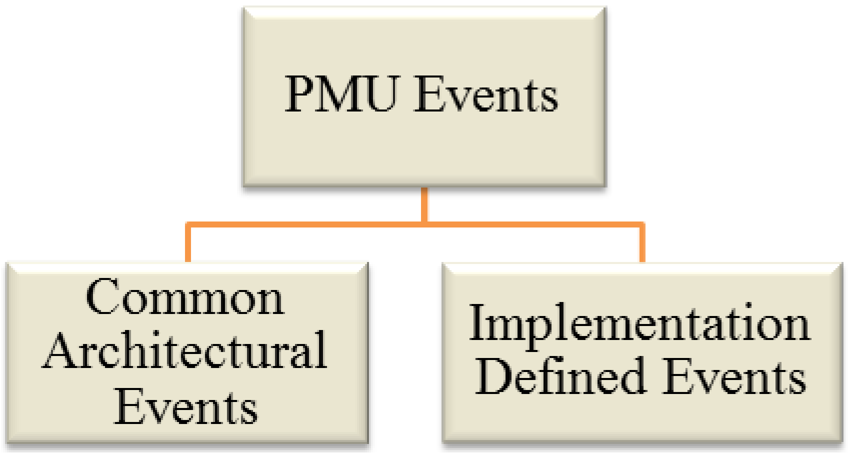

Figure 2.3: PMU Events Classification

Following table 2.1 describes some common architectural PMU events used for performance characterization of SPEC CPU2000 integer benchmarks and table 2.2 describes some of the implementation defined PMU events that are used for performance analysis of SPEC CPU 2000 integer benchmarks.

Table 2.1: Common Architectural PMU Events [4]

Name of Event	**Description**
CPU Cycles	Counts total no. of CPU cycles being used
Instructions Retired	Counts total no. of Instructions Executed
Memory Access	Counts total no. of memory access instructions
DP_Spec	Counts total no. of integer operation instructions
AES_Spec	Count of Advanced SIMD instructions
VFP_Spec	Count of Vector Floating Point instructions
Branch Predicted	Total no. of branch predictions
Branch Mispredictions	Total no. of branch mispredicted
I Cache Access	Counts total no. of Instructions caches
I Cache Refill	Counts no. of Instruction cache misses
D Cache Access	Counts no. of data cache access
D Cache Refill	Counts no. of data cache misses

Table 2.2: Implementation Defined PMU Events [4]

Name of Event	**Description**
BX_Stall	Gives no. of stall cycles due to branch execution
CX_Stall	Gives no. of stall cycles due to complex operation execution
LS_Stall	Gives no. of stall cycles due to load-store operations
MX_Stall	Gives no. of stall cycles due to Multiple-cycles operations execution
SP_Stall	Gives no. of stall cycles due to use of special purpose registers
SX_Stall	Gives no. of stall cycles due to single cycle operations

2.6 SPEC CPU2000 Integer Benchmarks

Following table 2.3 gives general idea about the SPEC CPU2000 integer benchmarks and the main task that they are designed for.

Table 2.3: SPEC CPU2000 Integer Benchmark Description [1]

Benchmark	**Language**	**Category**
164.gzip	C	Algorithm of Different Compression Techniques
175.vpr	C	Algorithm of FPGA Circuit Placement and Routing
176.gcc	C	Algorithm of C Programming Language Compiler
181.mcf	C	Algorithm of Combinational Optimization
186.crafty	C	Algorithm of Chess Game Playing
197.parser	C	Algorithm of Word Processing
252.eon	C++	Algorithm of Computer Visualization
253.perlbmk	C	PERL Programming Language
254.gap	C	Algorithm of Group, Theory Interpreter
255.vortex	C	Algorithm of Object Oriented Database
256.bzip2	C	Algorithm of Specific Compression technique
300.twolf	C	Algorithm of a Place and Route Simulator

2.7 ARM Cortex M3

The Cortex-M3 processor is based on the ARMv7-M architecture. It integrates the CM3 core which is central processor core with various advanced system peripherals for enabling different integrated capabilities like interrupt control, memory protection and system debug and trace. All these peripherals are highly configurable so that they facilitate a wide range of applications. The Cortex-M3 core and the integrated components are specifically designed to satisfy various requirements such as minimum memory implementation, reduced number of pin count and lower power consumption.

Table 2.4: ARM7TDMI and Cortex M3 Comparison [7]

Feature	**ARM7TDMI**	**Cortex-M3**
Architecture	ARMv4T (von Neuman)	ARMv7-M (Harvard)
ISA Support	Thumb/ARM	Thumb/Thumb-2
Pipeline	3-Stage	3-stage + branch speculation
Interrupts	FIQ/IRQ	NMI+ 1 to 240 Physical Interrupts
Interrupt Latency	24-42 Cycles	24-42 Cycles
Sleep Modes	None	Integrated
Memory Protection	None	8 region Memory Protection Unit
Power Consumption	0.28mW/MHz	0.19mW/MHz
Area	0.62mm2 (core only)	0.86mm2 (Core and Peripherals)

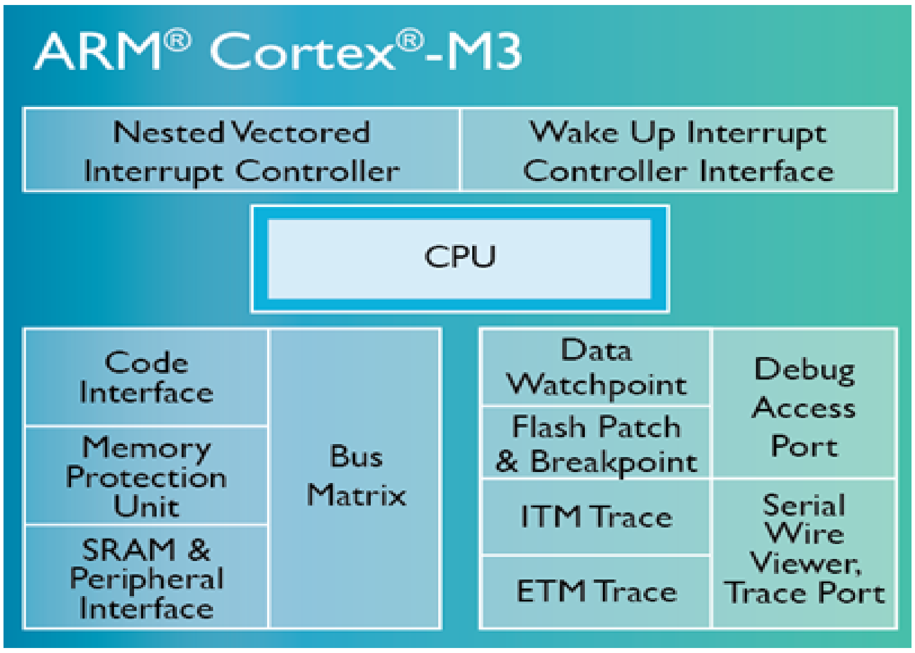

Figure 2.4: ARM Cortex-M3 Core
[8]

Cortex-M3 can be mainly useful for activity trackers, wearable and wi-fi transceivers. Some of the well-known features of cortex-M3 are as follows [9].

- 32-bit architecture optimised for embedded application
- Cortex M3 core can operate at up to 200 MHz
- Thumb-2 instruction set with mixture of both 16-bit and 32-bit instructions

- Hardware division and fast multiplier
- Little-endian memory space
- Memory Protection Unit (MPU) for protected operating system functionality
- Includes Nested Vectored Interrupt Controller (NVIC)
- SysTick Timer provided by cortex-M3 core
- Wakeup Interrupt controller (WIC) for waking up the CPU from reduced power modes.
- Standard JTAG debug interfaced
- Serial wire JTAG debug port (SWJ-DP)
- Enhanced system debug with extensive breakpoint
- 3 Stage enhanced pipeline

2.8 Serial to Wi-Fi Converter

Serial to wi-fi converter is a module that provides WLAN connectivity to the other micro-controller based systems. A host micro-controller can communicate with this module over an UART interface. The host micro-controller system issues commands and receives response from the serial to wi-fi converter module using simple ASCII text based communication. Following figure 2.5 shows the block diagram of serial to wi-fi converter module.

2.8.1 Features

- **WLAN Station:** This module acts as an 802.11 b/g/n station interface that can associate with any wi-fi alliance certified access point. It supports open, WEP, WPA and WPA2 security based association with these access points.

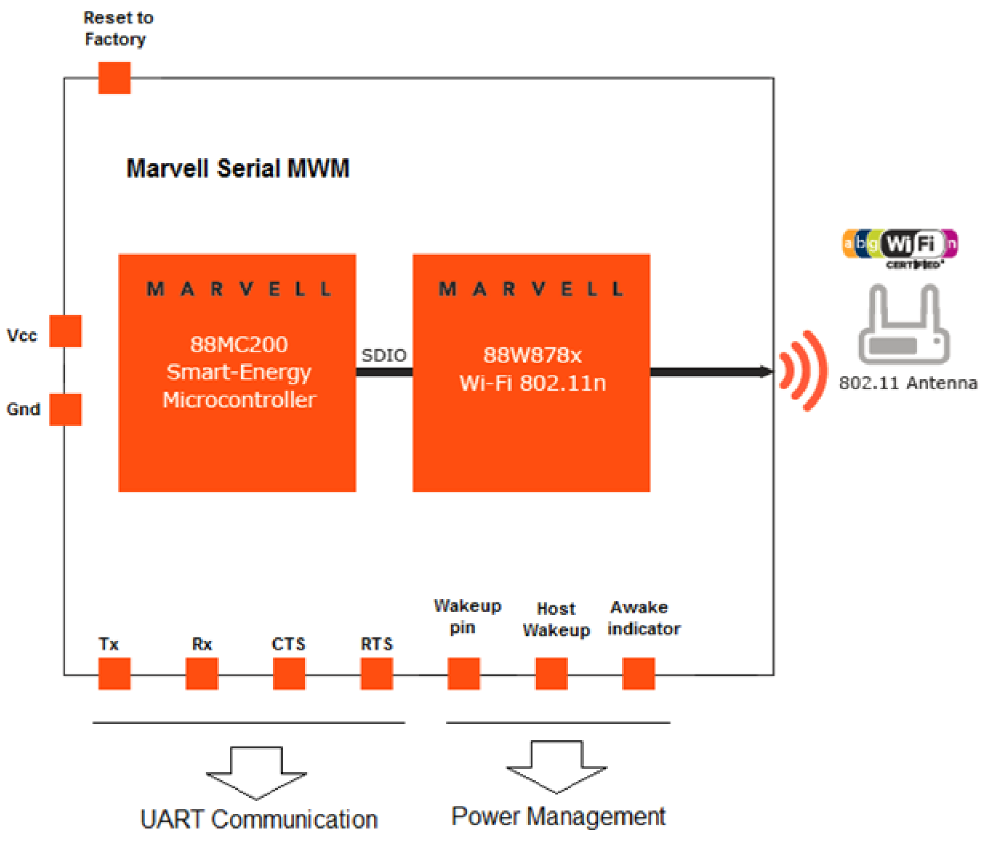

Figure 2.5: Serial to Wi-Fi Converter Block Diagram

- **Provisioning:** Provisioning is the process of configuring the setting of the home wireless network/access point in serial to wi-fi converter module. Once this configuration is performed, this module will automatically associate and maintain connectivity with the configured AP.
  - **Micro-AP and WPS based provisioning:** In this method of provisioning, this module hosts as a micro-AP network itself that clients can connect to. Once connected, they can program the provisioning information into this module. WPS is supported by many access points. A WPS-push button based provisioning is as easy as pressing a button on the AP and this module establishes association. Alternatively, a WPS pin based provisioning includes entering the same WPS pin on the AP and this module.

- **Configuration and Manufacturing:** Persistent configuration parameters are written in flash partition called PSM. Also a support is provided to have the manufacturing time default configuration data.
- **File System:** The file system image that contains the HTML5/CSS/JS code that implements the provisioning using web application.
- **Asynchronous Events:** Asynchronous events can be generated from this module to the host. This allows host to go into low power modes and this module can wake host up appropriately.
- **TCP and UDP sockets:** This module support communication over network using TCP/UDP sockets. This can be used to implement both clients and servers.
- **HTTP and HTTPS Client:** HTTP Client support provides an easy way of commutation with any web server supporting HTTP protocol. Secure HTTP HTTPS is also supported.
- **Websockets:** It provides full duplex communication over HTTP sockets by upgrading it to websockets.
- **Power Management:** This module is designed to opportunistically enter low power modes on detection of idle intervals.

2.9 Wi-Fi Power Management

2.9.1 Basic Terminology

To help the stations for saving the power, Access Points (APs) are designed such that they can buffer frames for a station when that station is kept in power save mode and to transmit them later to the station, when it is out of power save mode and ready to listen. In power save mode, transmitter and receiver of station are turned off to save the energy. It's more power-efficient for an AP to inform a station if it has buffered frames present on the AP than to have the station polling the AP and querying if frames are present.

- **Target Beacon Transmission Time (TBTT) and Beacon Interval:** The time at which AP must send a becon is Target Beacon Transmission Time (TBTT). The beacon interval is time difference between two TBTTs. The beacon interval is given in Time Units (TU), each TU represents 1024 microseconds.

- **Listen Interval:** The listen interval is given in beacon interval units, so essentially it tells the AP how many beacons it wants to ignore before turning the receiver on.

- **Delivery Traffic Indication Map (DTIM) Period :** This is a special type of TIM which announces that the AP is about to transmit all buffered broadcast and multicast frames called the Delivery Traffic Indication Map (DTIM). The time interval after which DTIM will be sent is called as DTIM period. The DTIM period represents the number of beacon intervals that must go by before a new DTIM is sent.

2.9.2 Modes of Operation

When talking about current consumption of an embedded WLAN device, the mode of operation puts a current consumption number in its proper context. For example, when comparing two current consumption numbers whose only other piece of attached information is the unit (in milliamps) of measurement, it becomes necessary to also include the operation mode in which the numbers were obtained. Different modes require different parts of the design to be turned on, which makes the operation mode a key to observed differences in the amount of current consumed.

Knowing the current consumption of the design in these basic modes allows the system designed to make accurate calculations for current consumption during various power saving modes. For example, IEEE power save mode consists of staying asleep and waking up periodically to receive DTIMs. The sleep portion is the same as the deep-sleep mode defined below, and receiving the DTIM beacons is an active receive mode. As a result, knowing the current consumption for these two modes and the length of time the device will be in each mode (essentially the duty cycle of the waveform for IEEE power save) allows the designer to calculate an average current consumption over a time interval. The modes of interest and their definitions are as follows.

- **Receive Idle:** DUT is powered on and had completed firmware download. DUT is ready to receive packets, but is not actively decoding packets because none are being transmitted to it.
- **Deep-Sleep:** This is a low-power state used in the sleep state of many power save modes. It is a low-power state where the external reference clock and many blocks in the chip are switched off. Only a slow sleep clock is used to maintain register and memory states. Wake-up does not require a firmware re-download.
- **Power Down:** Lowest power state achieved by completely powering down the WIFI subsection of the SoC. All internal WIFI clocks are shut down, and register and memory states are not maintained. Upon exiting power down mode, a WIFI reset is automatically performed and a firmware re-download is required to re-enter any of the above mentioned operation modes.
- **IEEE Power Save:** This mode is specially designed for wi-fi stations. In this power saving mode some of the hardware blocks are turned off (like hardware MAC, RF chip). A wakeup timer is programmed such that is should wakeup WLAN device when there is next expected beacon transmit from AP. When

wakeup occurs there is no need for establishing an association again with AP. The device automatically wakes up on beacons. This is dependent on the DTIM value of the AP it is connected to. If it is a DTIM value of 1 along with a beacon interval of 100ms, the device wakes up every 100ms. Similarly, with DTIM=2, the device wakes up at every 200ms and so on.

When the device is asleep in IEEE-power save mode, only a slow sleep clock is used to maintain register and memory states. However, there is a definite wake-up pattern. The device wakes up at regular intervals determined by the DTIM count and/or the listen interval.

2.10 Summary

This chapter gives some details about the ARM v-8A architecture and the PMU events which are useful for the performance characterization. Also the SPEC CPU2000 integer benchmarks details are listed out. Using these PMU events we can do some analysis for ARM-v8A architecture and we can find which factors will enhance the overall performance in 64-bit mode and for which factors 32-bit mode is still the better one. Also this chapter gives some brief details about ARM Cortex-M3 micro controller and introduction to serial to wi-fi converter application and background to wi-fi power measurement.

Chapter 3

Measurement Set-up and Techniques

This chapter gives idea about the experimental set-up and the measurement technique used for performance analysis of ARM v8-A architecture and ARM Cortex M-3 micro-controller used along with Marvells 8801 wi-fi chip.

3.1 ARM v8-A Architecture

For the performance analysis, ARMs test board having the ARM v8A architecture processor is used. Single platform is able to provide an environment to run 32-bit code and 64-bit code. It will be good to find out the major benefits of upgrading to 64-bit. The aim is to analyse the performance of a particular set of popular applications in these modes.

Following are some key points of that test board that have been set at the time of whole experiment.

- **GCC compiler version:** 4.9.1
- **OS:** SuSE Linux 9.2

- **No. of cores active:** 1

3.1.1 Perf Tool

perf is powerful tool. It can instrument CPU performance counters, tracepoints, kprobes, and uprobes (dynamic tracing). It is capable of lightweight profiling. It is also included in the Linux kernel, under tools/perf, and is frequently updated and enhanced.

perf began as a tool for using the performance counters subsystem in Linux, and has had various enhancements to add tracing capabilities.

Trace points are instrumentation points placed at logical locations in code, such as for system calls, TCP/IP events, file system operations, etc. These have negligible overhead when not in use, and can be enabled by the perf command to collect information including timestamps and stack traces. perf can also dynamically create tracepoints using the kprobes and uprobes frameworks, for kernel and userspace dynamic tracing. The possibilities with these are endless [5]. The user space perf command presents a simple to use interface with commands like [5]:

- **perf stat:** obtain event counts
- **perf record:** record events for later reporting
- **perf report:** break down events by process, function etc.
- **perf top:** see live event count
- **perf bench:** run different kernel micro-benchmarks

In this experiment, perf stat is majorly used for collecting data of various PMU events. The perf stat command can be used as follows [5].

perf stat [**options**] [**command**]

options consists of

- **-e** event selector
- **-p** stat events on process id
- **-t** stat events on thread id

command consist of executable binary name followed by some command line input to that binary
Ex:
perf stat -e r001 -e r002 -e r003 -e r004 -e r005 -e instructions -e cycles ./executable
Here,

- r001, r002, r003, r004, r005 are some PMU events
- instructions give count of instructions executed.
- cycles give CPU cycles required

In this case, executable can be any binary for various benchmarks from the SPEC CPU2000 integer benchmark suit.

3.1.2 Performance Metrics

The classification of various performance metrics that are to be measured can be done as shown in fig 3.1 below.

1 **Program Behavior:**

Program behaviour consists of measuring following parameters which gives basic idea of the performance.

- **Speed up:** Gives idea about how fast is the performance.
- **Static Code Size:** Gives the size of binaries.

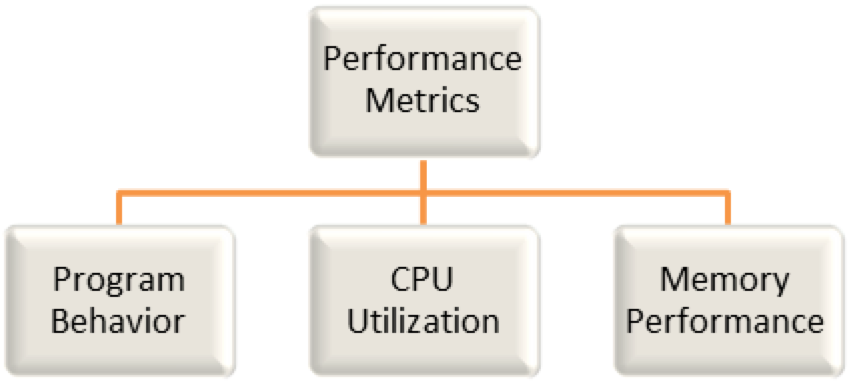

Figure 3.1: Performance Metrics Classification

- **Dynamic Instruction Count:** Total no. of instructions being executed
- **Instruction Mix:** Types of instructions and their weightage

2 **CPU Utilization:**

CPU utilization gives us the idea about the various parameters that can affect the performance of CPU and up to what extent CPU is used for various operations.

- **CPI:** CPI gives idea about the no. of cycles required to execute any instruction.
- **Stall Cycles:** Stalls gives us idea about which kind of cycles will affect the pipeline execution and which kind of operations are going to add stall in the pipeline.

3 **Memory Performance:**

Memory Performance can briefly give idea about memory subsystem utilization. The instruction cache and data cache utilization various branch instructions that affect memory utilization can be deduced from this metric.

- **Cache Behaviour:** Cache behaviour gives idea about L1 and L2 data and instruction cache misses and cache access. This will give the idea about the load on the memory subsystem.

- **Branch Miss:** This metric gives idea about all the branches being missed over the various branches being predicted.
- **Indirect Branches:** This metric tell about the indirect branch instructions executed among the total branch instructions. Indirect branches can be a key metric to analyse the behaviour of branch instructions

3.2 Cortex-M3 in Marvell 88MC200 chip

For power measurement, Marvell 88MC200 micro-controller chip is used along with the 8801 wi-fi chip. The following diagram shows these both chips.

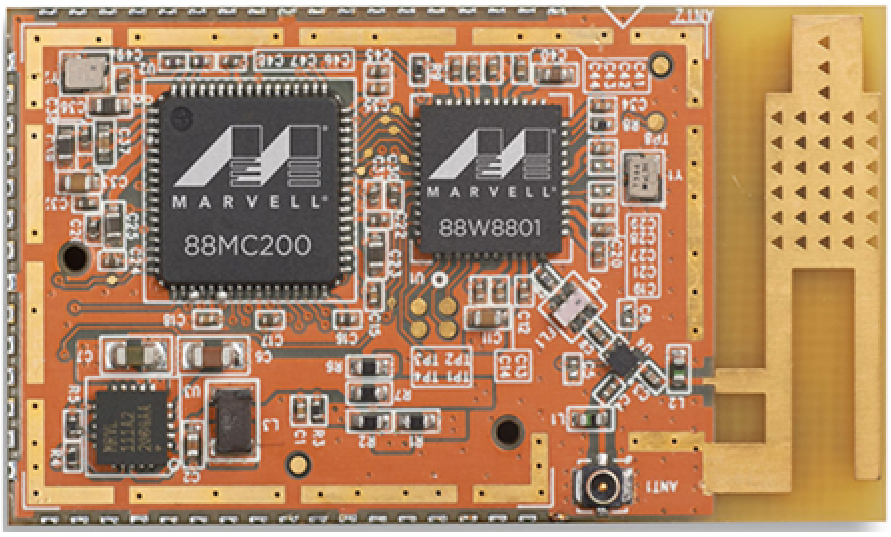

Figure 3.2: Marvell Wi-Fi Micro-Controller Chip
[10]

3.2.1 Power Measurement Techniques

The power consumption is directly related to the current drawn. The average power consumption can be found out using the following formula.

Electrical power equation:

$$\text{Power } P = I * V$$

Where,

power **P** is measured in watts,

current **I** is measured in amperes.

voltage **V** is measured in volts,

The device used to measure the current is called as Ammeter. Ammeter should be connected in series with the circuit, as it is required that the current which is needed to be measured must pass directly through the ammeter. The basic schematic to connect the ammeter is as follows.

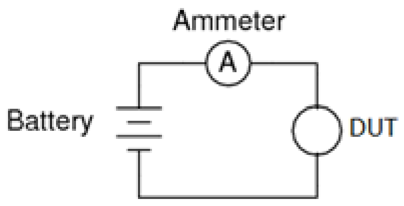

Figure 3.3: Basic Schematic for current measurement

3.2.2 Measurement Set-Up

The board is powered by USB and is connected to a laptop. It is connected to a laptop to give commands to the chip to switch its various power modes to take the readings.

A digital averaging multimeter is used for taking the readings. Usually all the readings are average of 1000 readings. Another advantage of using the averaging multimeter over the normal digital multimeter is that it can capture the instantaneous current.

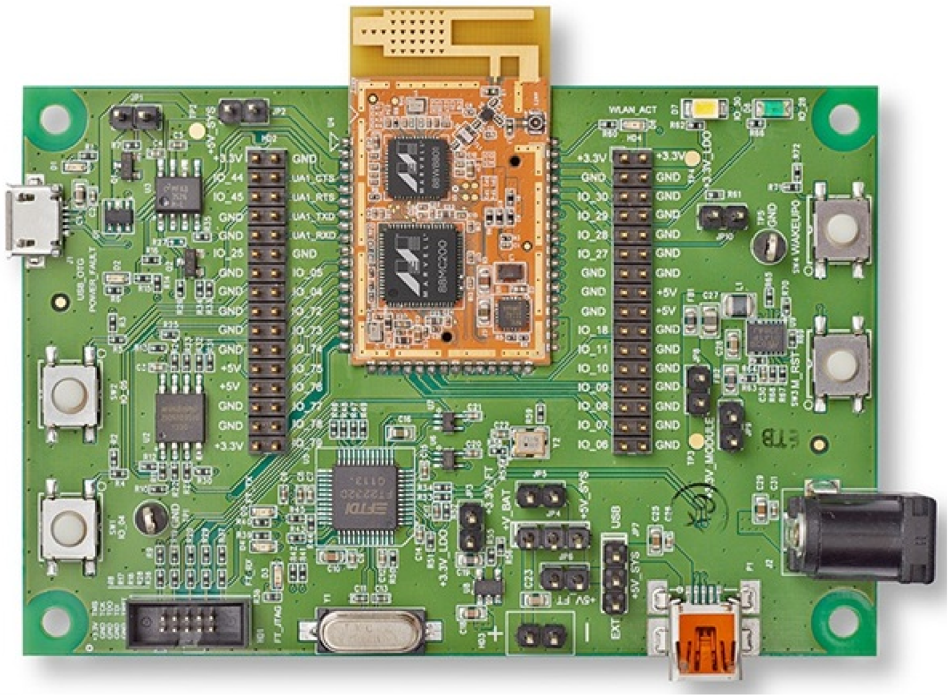

Figure 3.4: Test Board MC200 used for power measurement

To measure the current consumption of the board, the current meter is connected across JP8 of the DUT. JP8 measures the entire current of DUT at voltage 3.3V

3.2.3 Various Power Modes

Power Management unit (PMU) is responsible for power management and reset functions of MCU including power management of Cortex M3 and SRAM. Power states of MCU controlled by PMU are referred as PM States. PM State is a combination of states of Cortex M3, SRAM and flash. Before moving to PM states in detail a brief on Cortex M3 states and SRAM states is as follows.

Table 3.1: Cortex M3 States [9]

Run (C0)	**Idle (C1)**	**Standby (C2)**	**Off (c3)**
HCLK:On	HCLK:On	HCLK:Off	Power Removed
FCLK:On	FCLK:Off	FCLK:On	Power Removed

Table 3.2: SRAM States [9]

Run (M0)	**Standby (M2)**	**Off (M3)**
CLK:On	PWDN:Leakage reduction mode	Power Removed

Now moving further to 88MC200 MCU System Power States (PM states) are based on states of Cortex M3, SRAM and various PLL states.

Table 3.3: Power States of 88MC200[9]

Component	**PM0**	**PM1**	**PM2**	**PM3**	**PM4**
Cortex M3	C0	C1	C2	C3	C4
SRAM	M0	M0	M2	M2	M3
XTAL	On/Off	On/Off	On/Off	Off	Off
SPLL	On/Off	On/Off	Off	Off	Off
AuPLL	On/Off	On/Off	Off	Off	Off

All these power modes can be explained in brief as follows.

- **PM0:** Active Mode
- **PM1:** idle Mode
- **PM2:** Standby Mode
- **PM3:** Sleep Mode
- **PM4:** Shut Down Mode

3.3 Summary

This chapter gives details about the experiment set-up used for performance analysis of ARM cores and various methodologies to measure down the accurate readings required to do the performance analysis.

Chapter 4

Performance Analysis

In this chapter, analysis of the overall performance of SPEC CPU2000 INT benchmarks on ARM v8-A architecture and analysis of power measurement of various MCU states of ARM Coretex-M3 microcontroller is done. The outcome will be an overall understanding of the performance of ARM v8-A architecture for 64-bit mode and 32-bit mode and the need to switch to new micro-controller for realisation of the IoT application serial to wi-fi converter.

4.1 ARM v8-A Architecture

4.1.1 Speed Up

Speed-up is the difference between time taken for execution of a particular benchmark while running on 64-bit mode as well as 32-bit mode. By the observations, it is found that in most of the benchmarks while running in 32-bit mode provides speed-up.

Figure 4.1 represents the speedup achieved while running the CPU2000 integer benchmarks in 64-bit mode against 32-bit mode. Both of these modes run with the reference inputs in the suit. It is observed that, on an average 32-bit mode provides 5.85% performance boost over 64-bit mode. Highest performance boost is observed for crafty benchmark as it is designed around 64-bit word whereas the mcf benchmark performance is slowest for 64-bit mode as compared to 32-bit mode because it uses long data types and pointer data types which increases the code size in 64-bit and many branch instructions cause the higher BX stalls reducing the overall speed of operations.

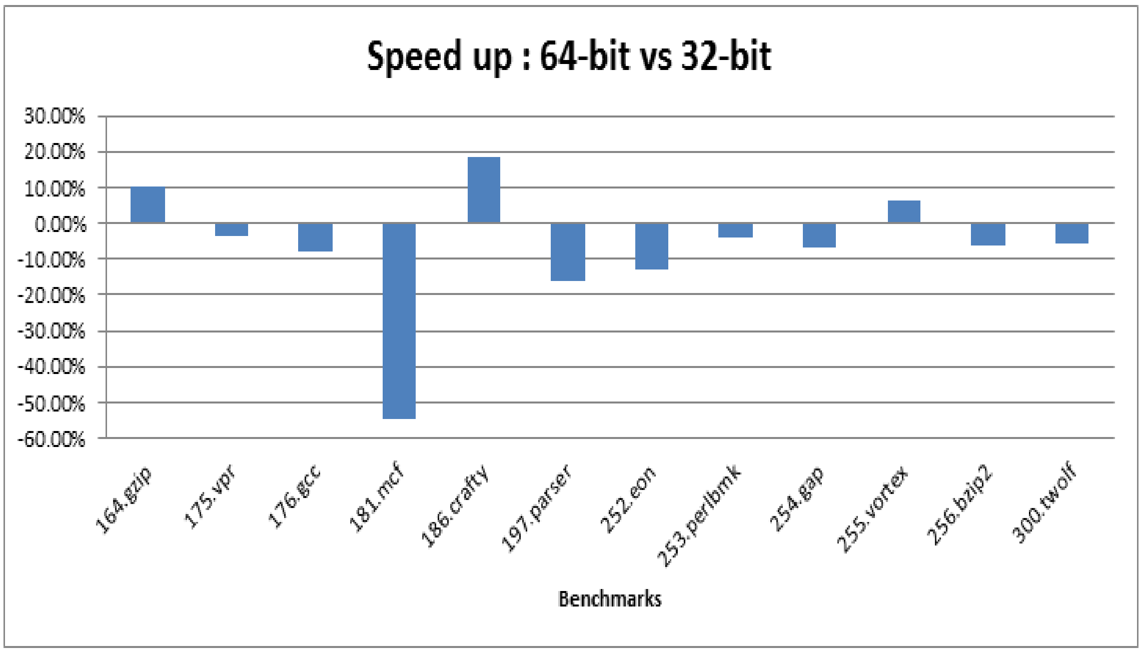

Figure 4.1: Speedup: 64-bit mode vs. 32-bit mode

4.1.2 Dynamic Instruction Count

Dynamic Instruction Count gives the idea about no. instructions required to execute the overall benchmark suit in any specific mode i.e. either 64-bit mode or 32-bit mode.

Figure 4.2 shows the dynamic instruction count of CPU2000 integer benchmark while running on 64-bit mode against 32-bit mode. It is observed that, on average the dynamic instruction count is decreased by 10.65% in 64-bit mode as compared to 32-bit mode. The significant decrease in dynamic instruction count is observed in benchmarks like crafty, eon, vortex, whereas significant increase in dynamic instruction count is found in benchmarks like bzip2 and some input sets of perlbmk.

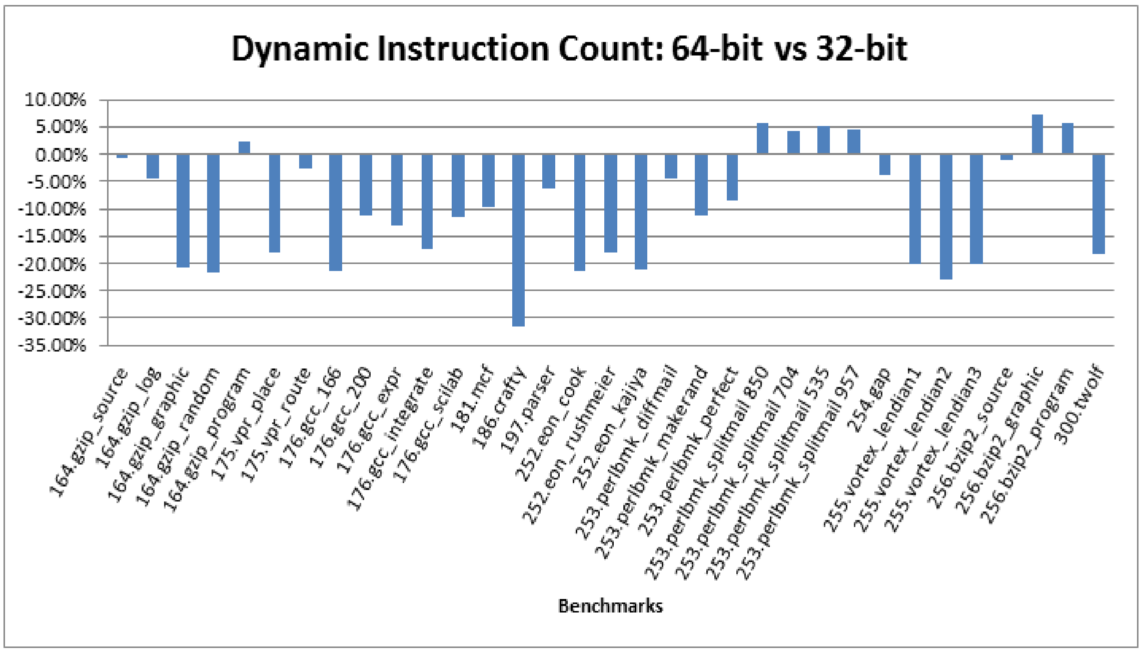

Figure 4.2: Dynamic Instruction Count: 64-bit mode vs. 32-bit mode

The main reason for decrease in instruction count in 64-bit mode of crafty benchmark is that, it is built around 64-bit mode, in 32-bit mode it needs extra registers to address the same data as each register is 32-bit wide. But in 64-bit mode it can be fitted into less no. of registers as the register is 64-bit wide and availability of more general purpose registers. The load-store instruction count is lowered in 64-bit mode. This is one of the major reasons of reduction in overall dynamic instruction count in 64-bit mode.

4.1.3 Instruction Mix

This performance metric explains the percentage of various types of instructions occurred such as memory, branch, neon, vfp and integer operations instruction out of total no. of instructions being executed. In figure 4.3 and 4.4, we can see that the Instruction mix for 64-bit mode as well as 32-bit mode. As observed the major part of instruction mix consists of integer operations and memory instructions for both 64-bit mode and 32-bit mode and very little of neon and vfp instructions.

There is increase of 2.38% in the integer operation instructions as we move from 32-bit mode to 64-bit mode and decrease of 6.17% in the memory instructions which significantly improves the performance in 64-bit mode as compared to that in 32-bit mode. But the branch instructions are increased by 8.29% in 64-bit mode on an average, this slows down performance of some benchmarks like parser. Vfp instructions are observed in eon and vpr benchmarks only in both 64-bit mode and 32-bit mode. This is because of some floating point operations being performed in these benchmarks.

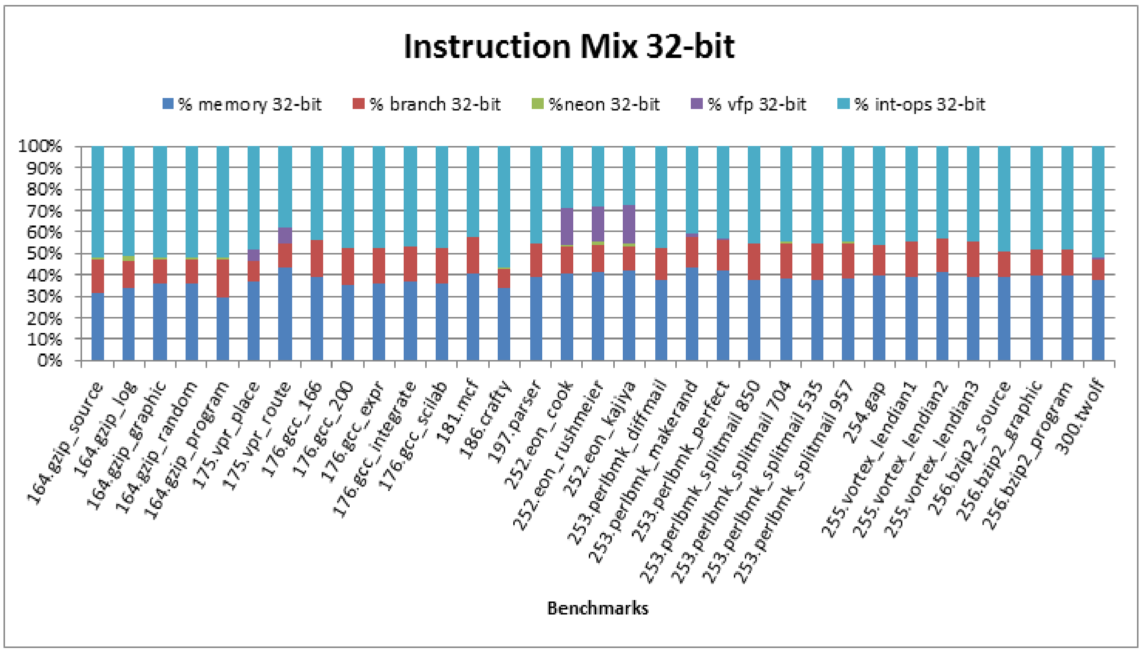

Figure 4.3: Instruction Mix: 32-bit mode

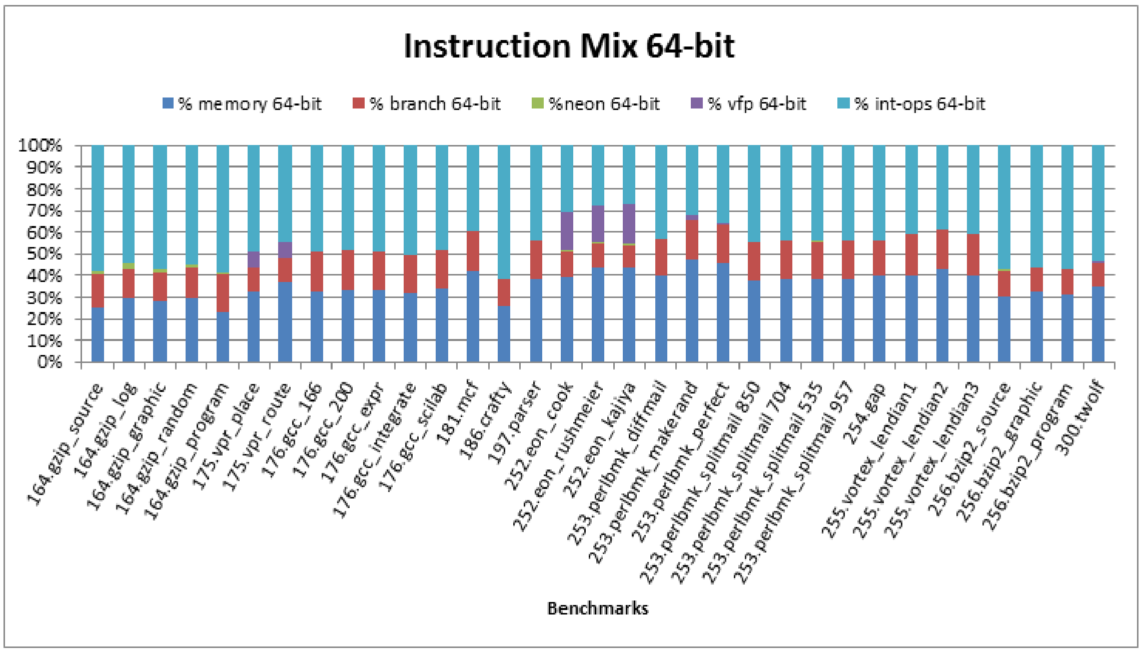

Figure 4.4: Instruction Mix: 64-bit mode

4.1.4 CPI

CPI is another major important performance parameter to differentiate between 64-bit mode and 32-bit mode. Lower the CPI, the better is the performance.

Figure 4.5 shows the cycles-per-instruction (CPI) for the CPU2000 integer benchmarks run in 64-bit mode against 32-bit mode. The impact on the number of dynamically instructions executed has been already showed in figure 4.2. From these figures, a basic idea regarding the components that are responsible for the observed performance difference is obtained. This will help in analysing individual benchmark.

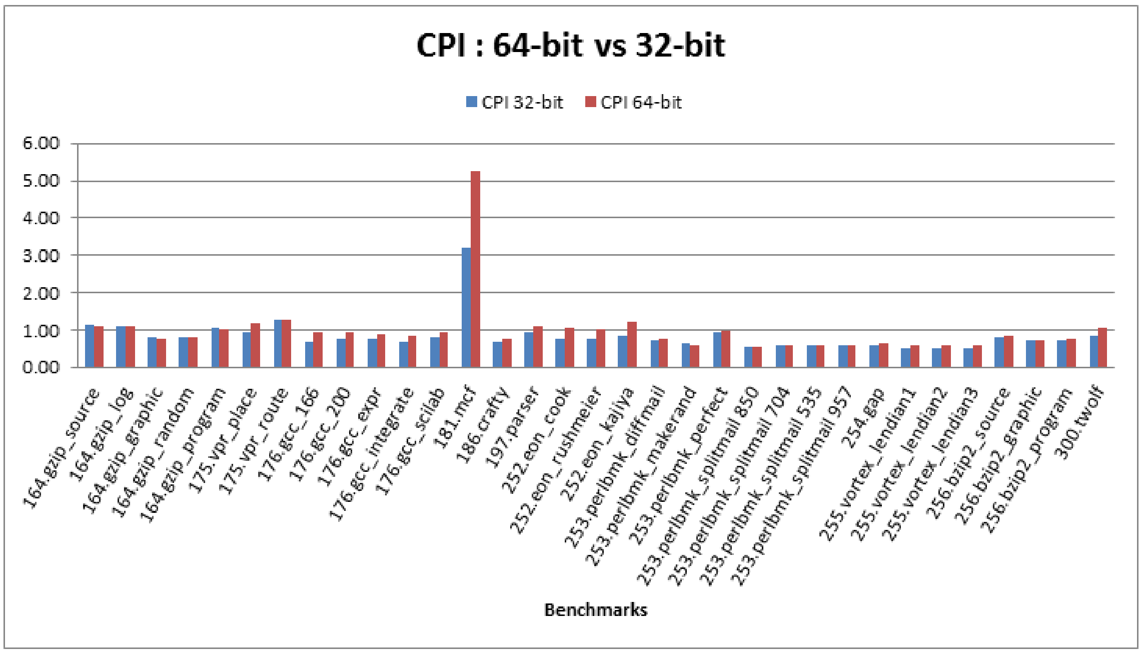

Figure 4.5: CPI: 64-bit mode vs. 32-bit mode

It is observed that CPI increased by 17.67% in 64-bit mode against to that in 32-bit mode. The average CPI for 32-bit mode is 0.84 and that for 64-bit mode is 0.99. While moving from 32-bit to 64-bit the highest increase in CPI is observed in the mcf benchmark. mcf benchmark consists of more of branch instructions as compared to other benchmarks hence the increase in CPI is expected as the execution of mcf benchmarks needs more CPU cycles.

4.1.5 Stall Cycles

To analyse increased pressure that was placed on instruction fetch and decode unit due to the factor of longer opcodes and the longer instructions in 64-bit mode, comparison of the stall cycles is done in both modes. We compared these six stall cycles BX, CX, LS, MX, SP and SX stall cycle.

In figure 4.6, we can see the BX stall cycles for 64-bit mode as well as 32-bit mode. It is found that on an average there is 35.31% increase in BX stall when we move from 32-bit mode to 64-bit mode though the overall BX stall is very less for each individual benchmark. Only significantly higher BX stall rate is observed in case of benchmarks like mcf and parser as these benchmarks consists of more number of branch instructions.

In figure 4.7, we can see the CX stall cycles for 64-bit mode as well as 32-bit mode. The CX stall rate value is very less for all benchmarks except eon and gzip benchmarks. eon benchmark consists of neon and vfp instructions hence the significant CX stall is observed in case of eon benchmark in both 32-bit mode and 64-bit mode.

In figure 4.8, we can see the LS stall cycles for 64-bit mode and 32-bit mode. The significant increase in the LS stall is found in benchmarks like mcf and gcc. mcf has higher stall rate in 64-bit mode because it has to perform more memory access as compared to other benchmarks in order to accumulate all the data for integer operations that are being performed under it. And these stall are increased in 64-bit mode because it puts additional pressure on memory subsystem to fetch the data and load-store operations being performed are significantly higher for these benchmarks in 64-bit mode.

In figure 4.9, we can see the MX stall cycles for 64-bit mode as well as 32-bit mode. The MX stall is decreased in 64-bit mode as compared to that of 32-bit mode. Overall for all the benchmarks MX stall is very less. Only in case of gzip benchmarks it is having significant value in 32-bit mode. This is mainly because the register width is less in 32-bit mode as compared to 64-bit mode, since gzip has to perform many arithmetic and load-store operations which consume multiple cycles for the single operation and hence the MX stall is higher for gzip.

In figure 4.10, we can see the SP stall cycles for 64-bit mode as well as 32-bit mode. Only significant increase in SP stall rate is observed in case of benchmarks eon and vpr while migrating from 32-bit mode to 64-bit mode. eon mostly consists of neon and vfp operations. Hence to perform these operations special purpose registers are used quite often which are resulting in the SP stalls for these benchmarks.

In figure 4.11, we can see the SX stall cycles for 64-bit mode as well as 32-bit mode. It is observed that SX stall rate is increased by 6.7% in 64-bit mode as compared to that in 32-bit mode. The key outlier in SX Stalls is gzip benchmark. In gzip the compression and decompression is being performed at different levels and many arithmetic operations which require single cycle for their execution are not performed properly. Hence the SX stall is more in gzip benchmark as compared to others.

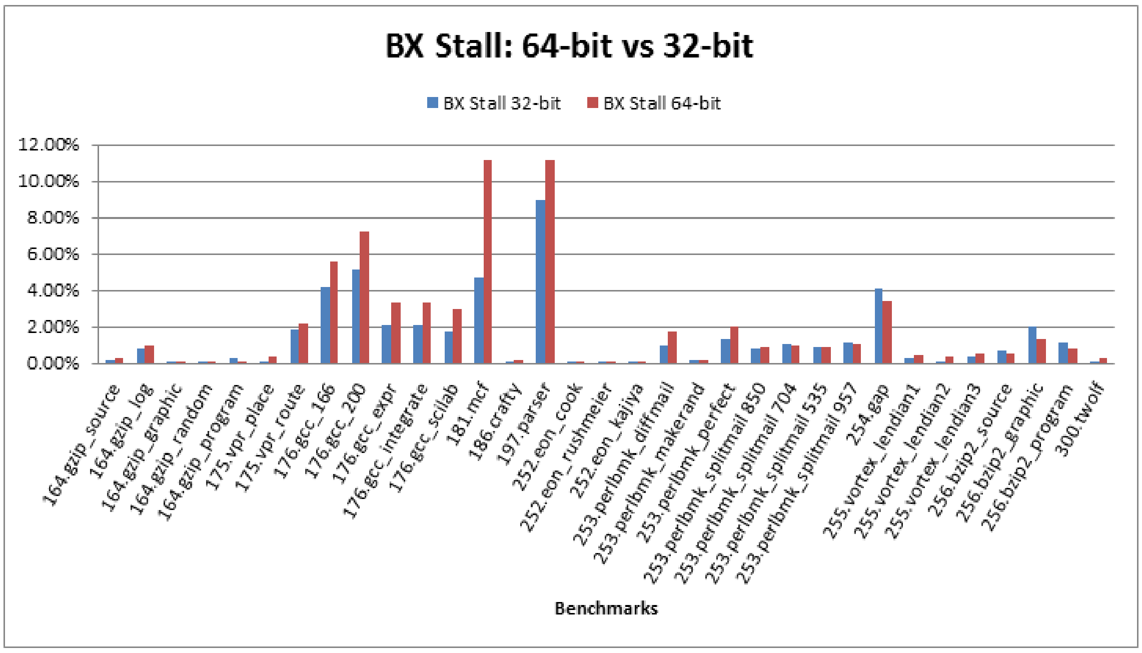

Figure 4.6: BX Stall: 64-bit mode vs. 32-bit mode

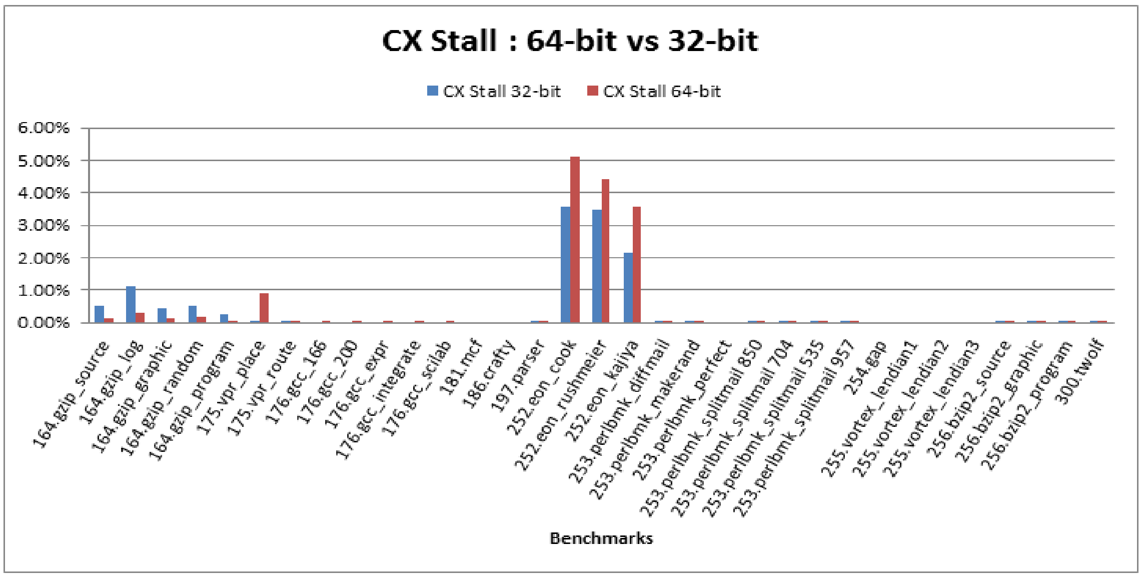

Figure 4.7: CX Stall: 64-bit mode vs. 32-bit mode

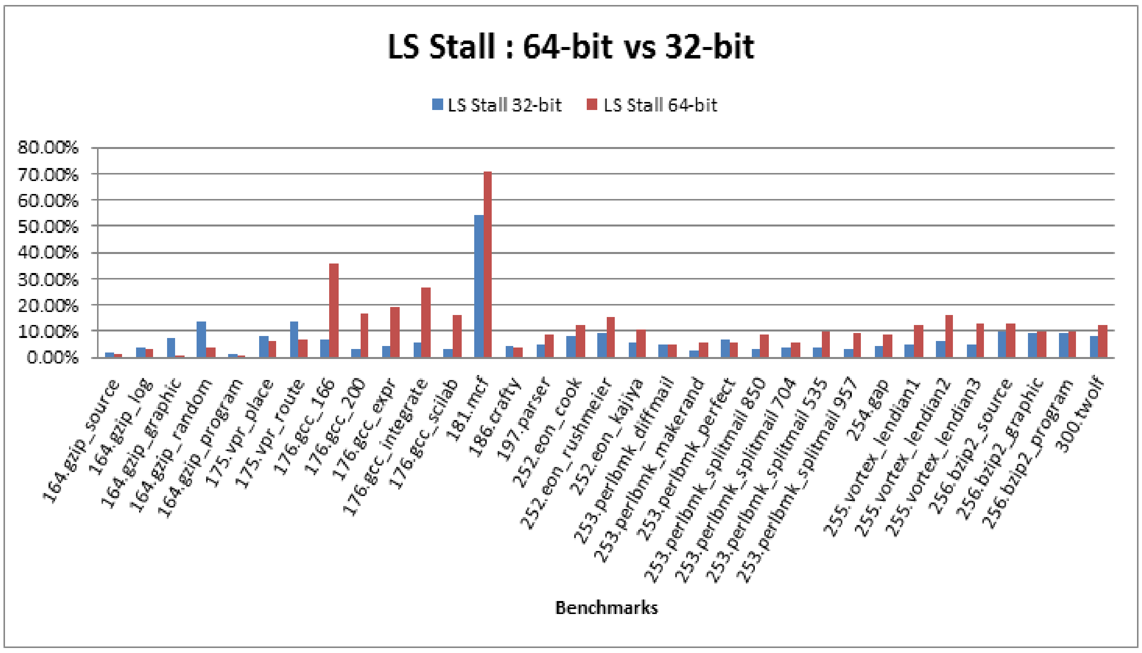

Figure 4.8: LS Stall: 64-bit mode vs. 32-bit mode

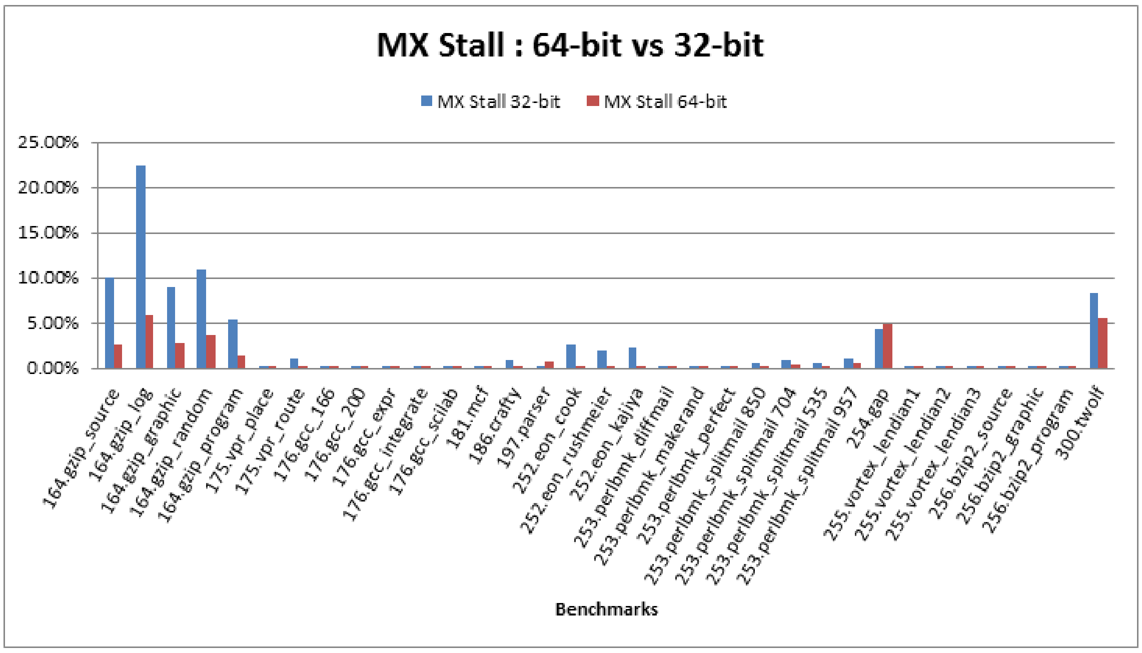

Figure 4.9: MX Stall: 64-bit mode vs. 32-bit mode

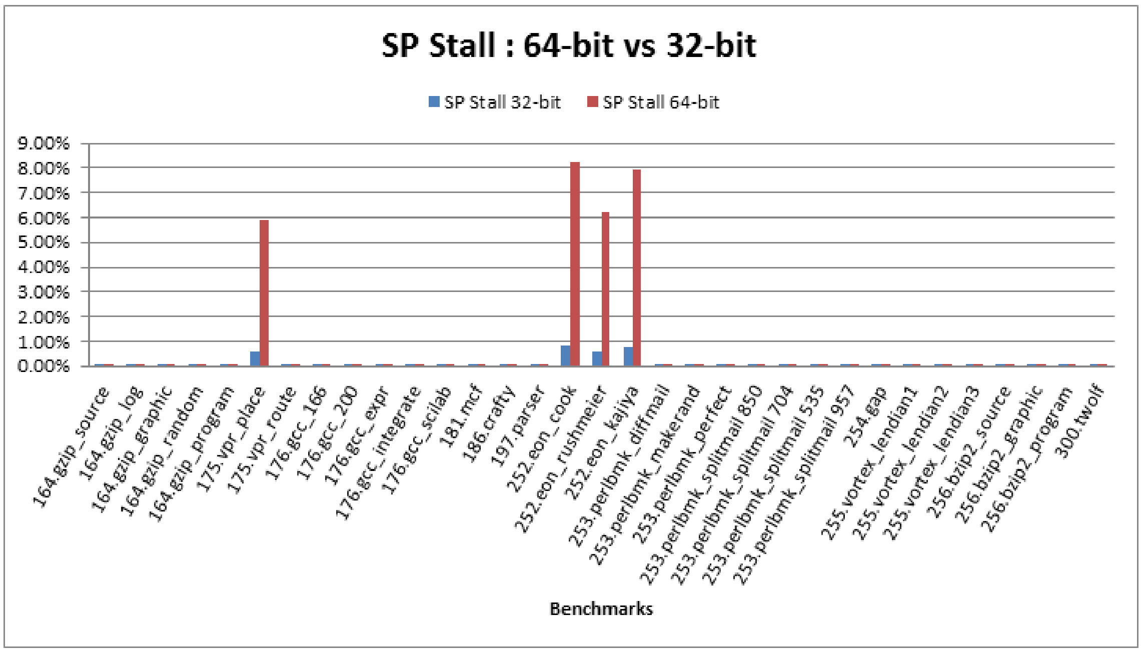

Figure 4.10: SP Stall: 64-bit mode vs. 32-bit mode

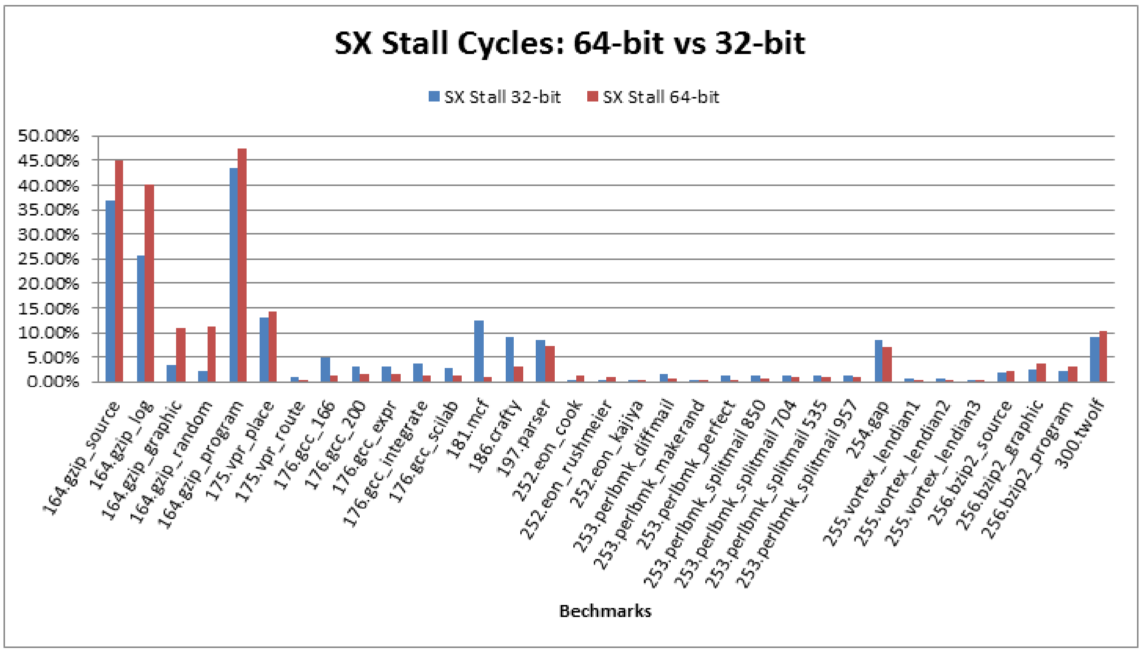

Figure 4.11: SX Stall: 64-bit mode vs. 32-bit mode

4.1.6 Instruction Cache Miss/KI

Instruction cache miss/KI gives the idea about how many cache miss instructions have occurred among all the instructions that are executed. Figure 4.12 shows the instruction cache miss observed per thousand instructions in both 64-bit mode and 32-bit mode. It is observed that in the 64-bit mode instruction cache miss rate increased by 16.82% on an average against to that in 32-bit mode. But overall the count of instruction cache miss is infinitesimally small in both modes for all benchmarks. Lower the cache miss rate better is the performance. Some significant amount of cache miss is observed in perlbmk perfect. This is because of all the branch and load store instructions which add the load on memory subsystem.

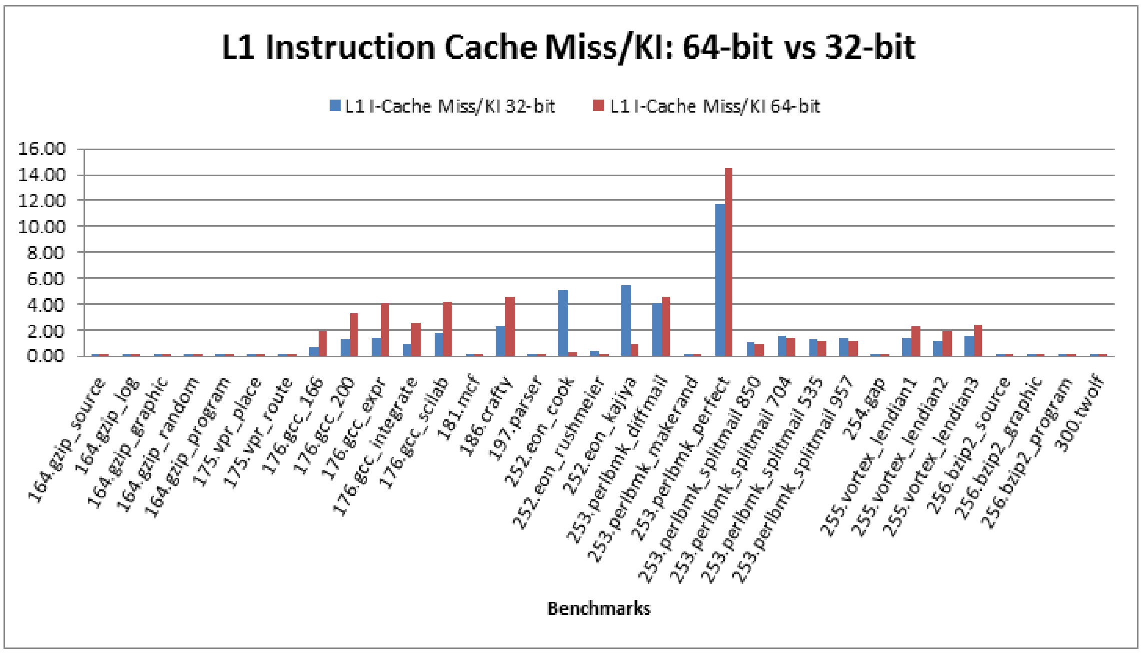

Figure 4.12: Instruction Cache Miss/KI: 64-bit mode vs. 32-bit mode

4.1.7 Instruction Cache Miss Rate

Instruction Cache Miss Rate gives the idea about how many instruction cache misses occurred among the all instruction cache access instructions. Figure 4.13 shows the instruction cache miss rate observed in both 64-bit mode and 32-bit mode. It is observed that in the 64-bit mode instruction cache miss rate increased by 1.43% on an average against to that in 32-bit mode. But overall the instruction cache miss is infinitesimally small in both modes for all benchmarks. Higher instruction cache miss rate is observed in perlbmk for the input set of perfect.pl.

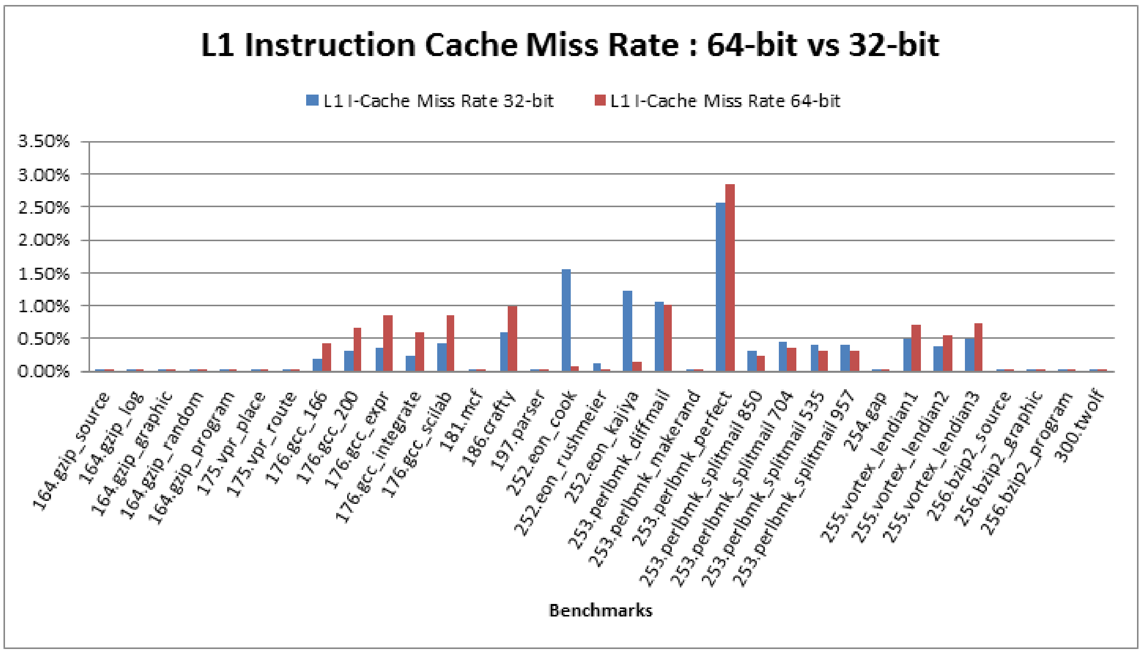

Figure 4.13: Instruction Cache Miss Rate: 64-bit mode vs. 32-bit mode

4.1.8 Data Cache Miss/KI

Data cache miss/KI gives the idea about how many data miss instructions have occurred among all the instructions that are executed. Figure 4.14 and 4.15 shows the L1 and L2 data cache miss observed per thousand instructions in both 64-bit mode as well as 32-bit mode respectively. Overall data cache miss is very less in all the benchmarks. The key outlier in L1 and L2 data cache miss is mcf benchmark.

The remarkable increase in data cache miss rate when upgrading from 32-bit mode to 64-bit mode in mcf benchmark illustrates that as the size of pointer and long data types is doubled, it causes an ill effect on the data cache behaviour. mcf puts excessive load on memory subsystem as its data cache request is already very high as compared to others, that results in higher data cache miss. Hence mcf performs slows down on 64-bit mode as against to that in 32-bit mode.

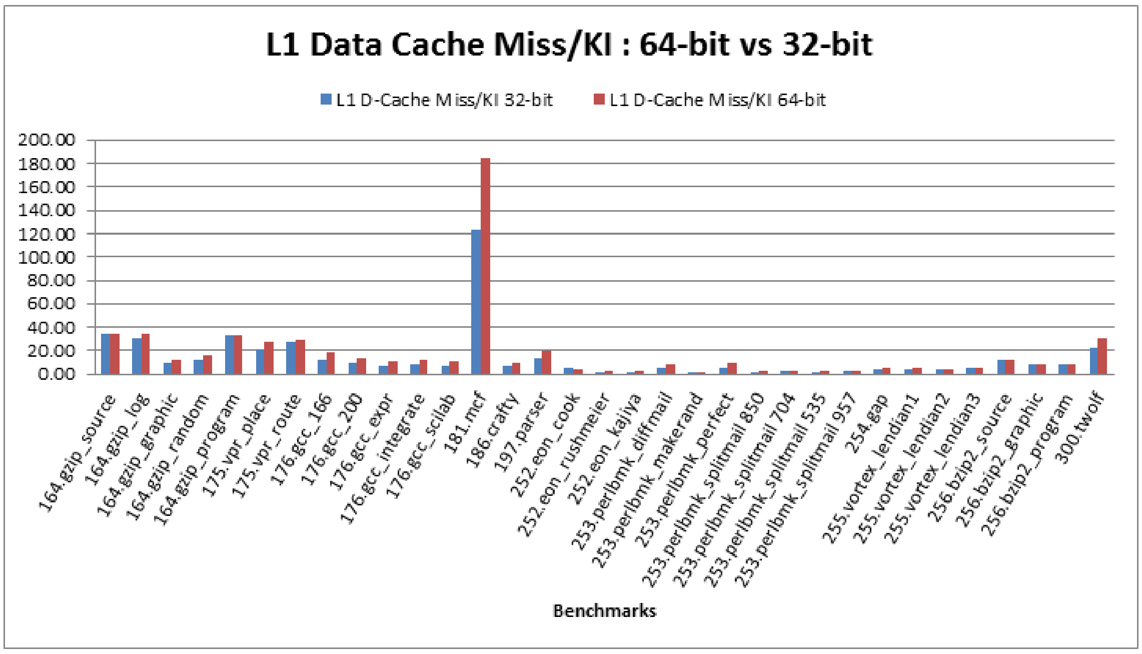

Figure 4.14: L1 Data Cache Miss/KI: 64-bit mode vs. 32-bit mode

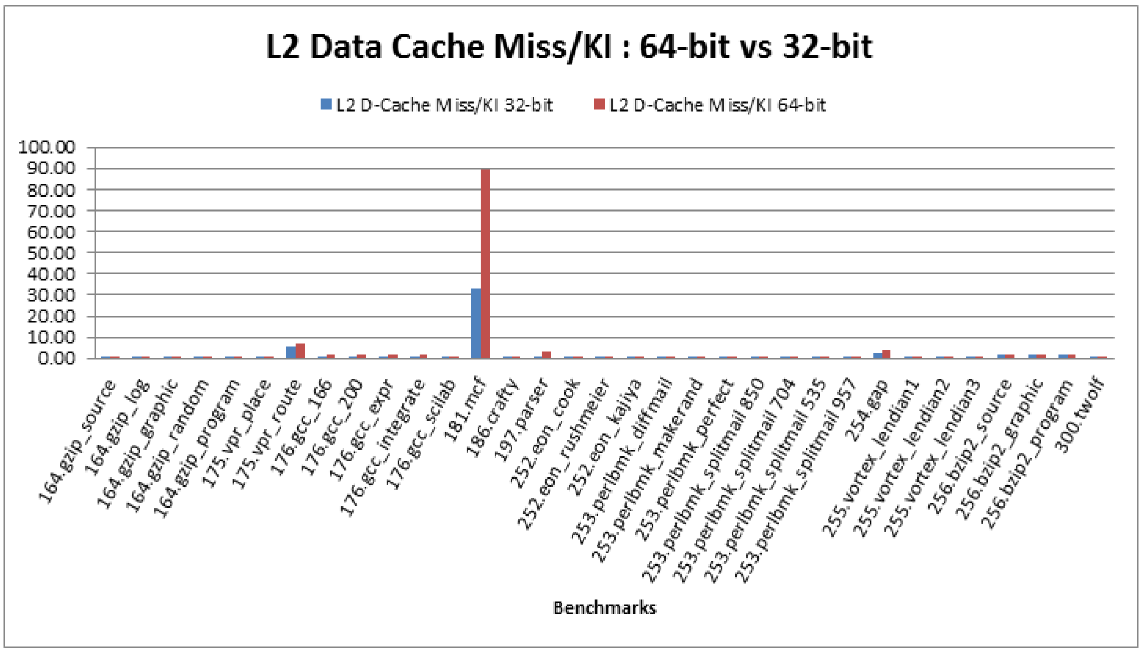

Figure 4.15: L2 Data Cache Miss/KI: 64-bit mode vs. 32-bit mode

4.1.9 Data Cache Miss Rate

Data Cache Miss Rate gives the idea about how many data cache misses occurred among the all data cache access instructions. Figure 4.16 shows the L1 data cache miss rate observed in both 64-bit mode and 32-bit mode. It is observed that in the 64-bit mode instruction cache miss rate increased by 34.21% on an average against to that in 32-bit mode. But overall the instruction cache miss is infinitesimally small in both modes. The main reason of higher data miss rate in mcf benchmark is the doubling of the size of pointer and long data types in 64-bit mode.

Figure 4.17 shows the L2 data cache miss rate observed in both 64-bit mode and 32-bit mode. It is observed that in the 64-bit mode instruction cache miss rate increased by 18.47% on an average against to that in 32-bit mode. Key outliers in L2 data cache miss rate are mcf and gap benchmarks. Size of L2 cache is higher as compared to L1 cache so secondary data to be loaded in L2 cache is more. The arithmetic operations in mcf benchmark and array operations to be performed in gap benchmarks that require higher memory access to load the data from main memory. This creates more no. of load store operations which results in data cache miss. The data cache miss rate slows down the overall performance of mcf and gap benchmarks in 64-bit mode as compared to that in 32-bit mode.

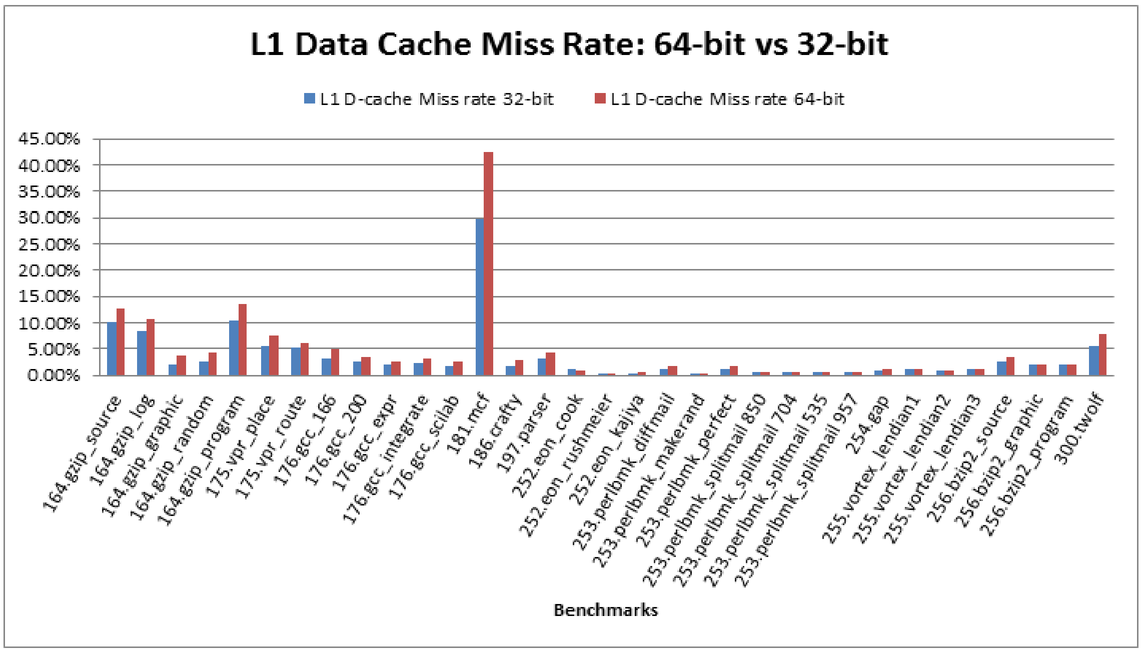

Figure 4.16: L1 Data Cache Miss Rate: 64-bit mode vs. 32-bit mode

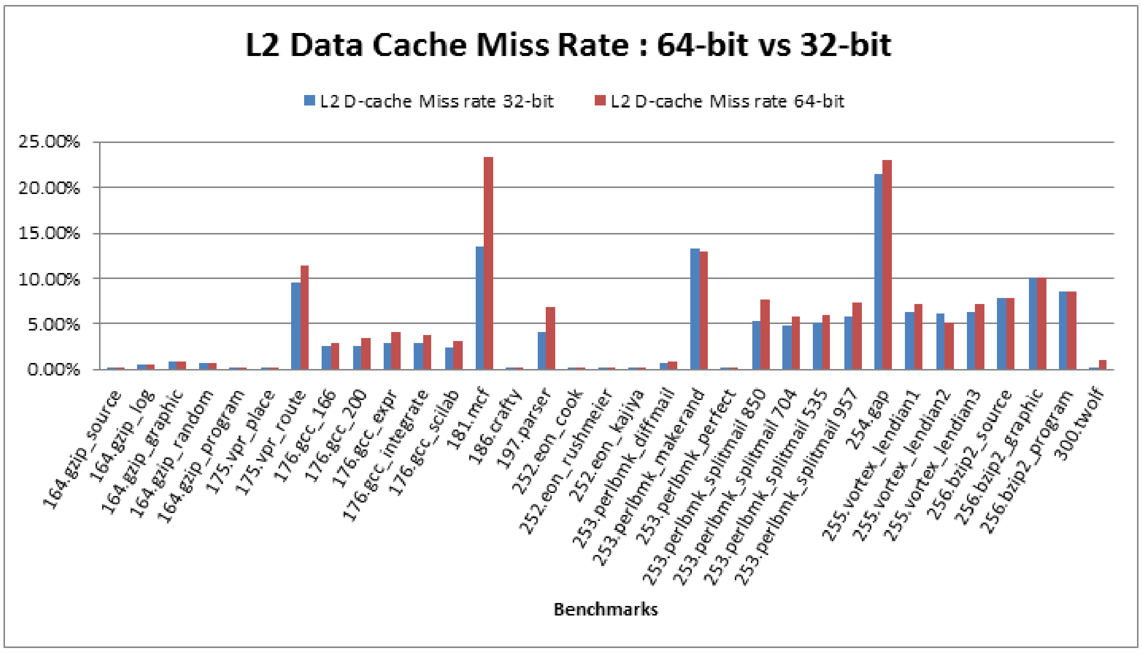

Figure 4.17: L2 Data Cache Miss Rate: 64-bit mode vs. 32-bit mode

4.1.10 Branch Miss/KI

Branch Miss is the performance metric to measure the no. of branch miss predictions occurred. Branch instructions are predicted so that it can be used to fasten the execution of total no. of instructions being executed. Lower the branch miss rate the better is the performance.

In figure 4.18, we can see the branch miss occurred per thousand instructions for both 64-bit mode as well as 32-bit mode. It is observed that in 64-bit mode the branch miss rate increased by 9.84% on an average as against to that with 32-bit mode.

It can be seen that, the branch miss increased while moving from 32-bit mode to 64-bit mode for benchmarks like vpr, mcf, bzip, twolf and the branch miss decreased in case of benchmarks like perlbmk and gap.

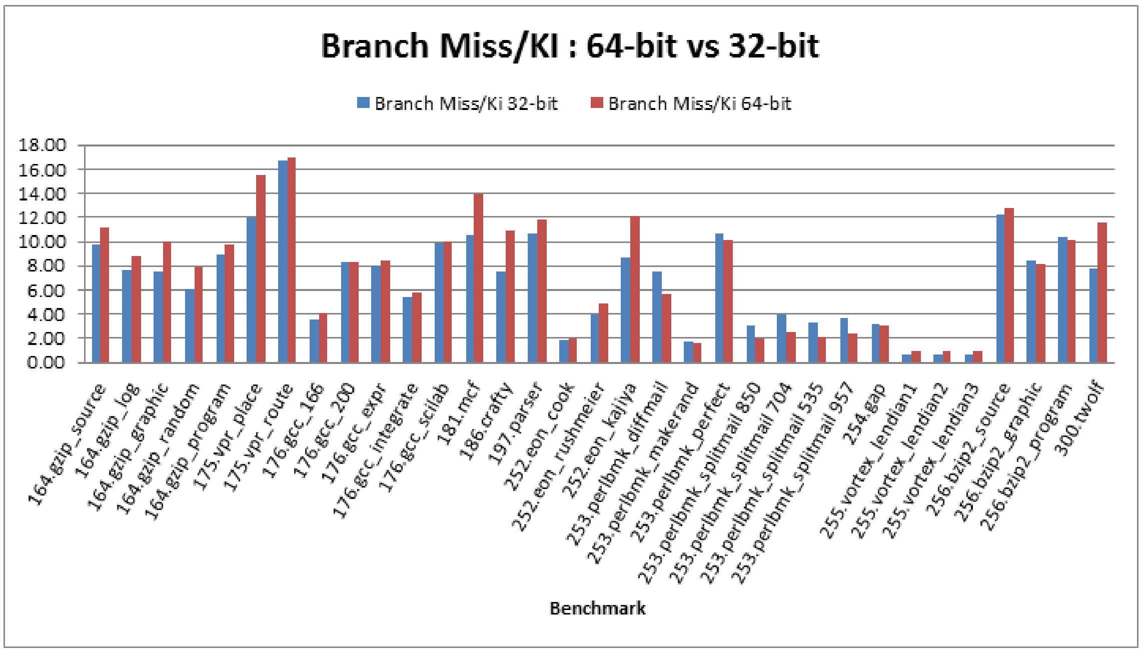

Figure 4.18: Branch Miss/KI: 64-bit mode vs. 32-bit mode

4.1.11 Branch Miss Rate

Branch Miss Rate gives the idea about how many branch misses occurred among the all branch instructions. Figure 4.19 shows the Branch miss rate illustrated in both 64-bit mode and 32-bit mode. It is observed that in the 64-bit mode branch miss rate increased by 1.7% on as against to that in 32-bit mode. Highest branch miss rate is in vpr benchmark because of its placement and routing algorithm used. For routing purpose various branch predictions has to be done and hence for planning out optimum routing branch miss occurs. Overall branch miss rate is low which gives performance benefit for branch specific applications.

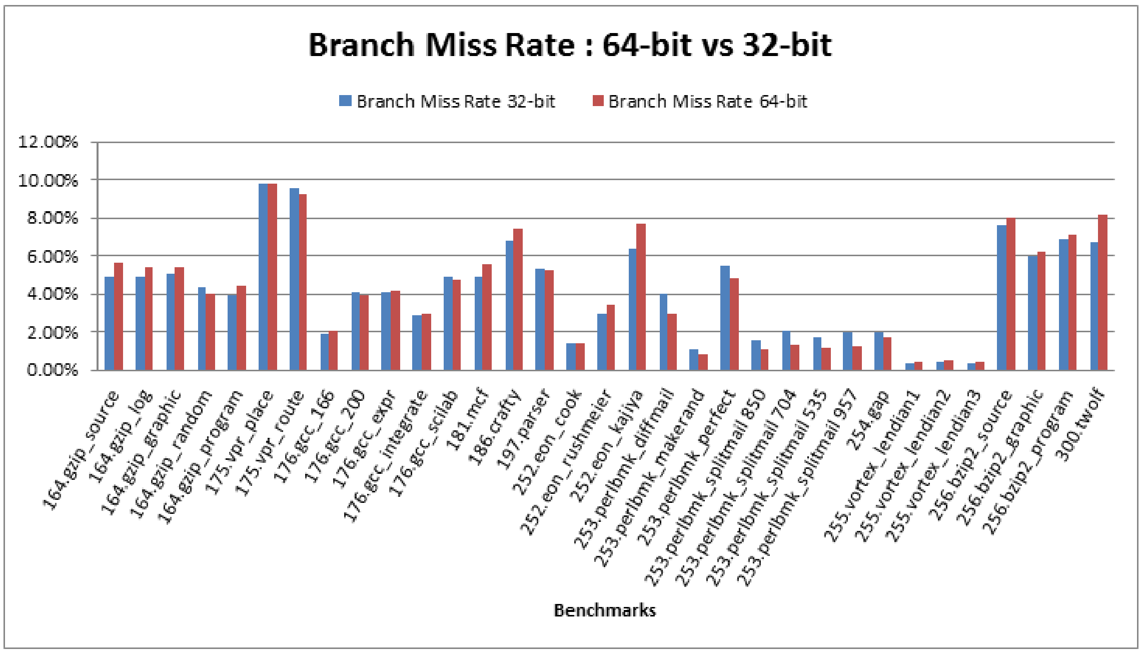

Figure 4.19: Branch Miss Rate: 64-bit mode vs. 32-bit mode

4.1.12 Indirect Branches/Total No. of Branches

This metric gives the idea about how many indirect branch instructions are present among the overall branch instructions that are being executed.

Figure 4.20 shows the indirect branches per total branches observed in both 64-bit mode and 32-bit mode. It is observed that in the 64-bit mode indirect branches are decreased by 9.56% on an average versus that in 32-bit mode. This reduction in indirect branches reduces the frequent memory access and helps in improving the performance.

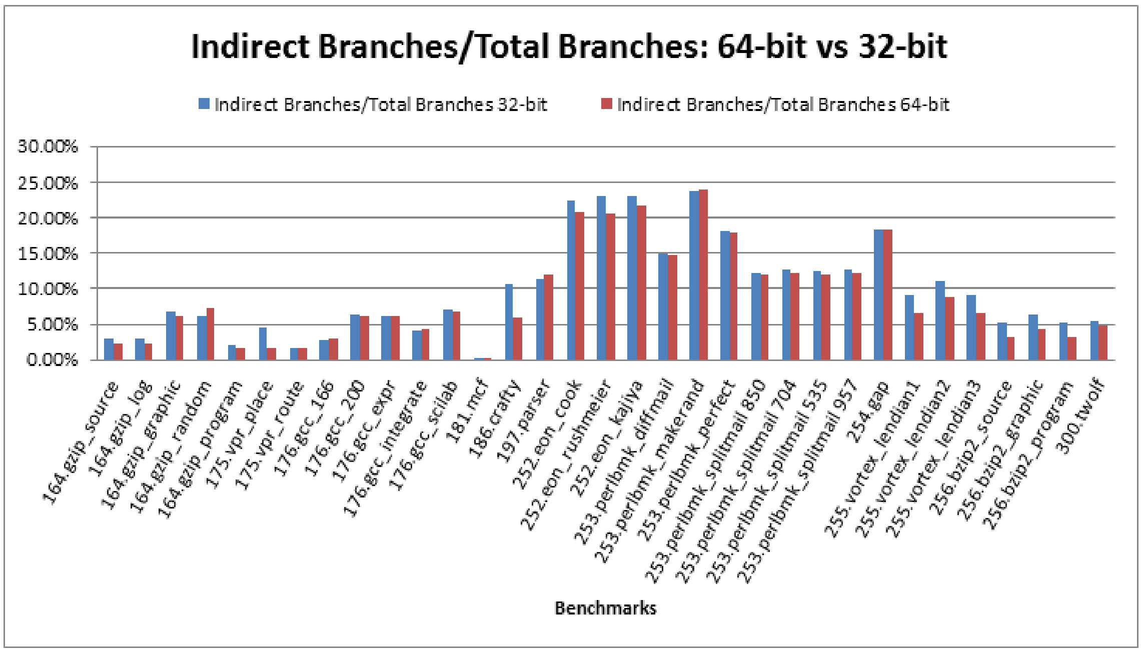

Figure 4.20: Indirect Branches/Total No. of Branches: 64-bit vs. 32-bit mode

4.2 Cortex-M3 in Marvell 88MC200 chip

In this section, the comparison of power consumption of ARM Cortex-M3 cores of 88MC200 and 88MW300 is done for various MCU states. The basic aim of this comparison is to know the need to bring up the serial to wi-fi converter application on 88MW300 chip. The main difference between MW300 and MC200 chip is that MW300 is made up of 45nm technology whereas MC200 is made up of 65nm technology and MC200 runs on 200MHz clock speed whereas MW300 runs on 300MHZ clock speed.

4.2.1 Wi-Fi Deep Sleep Mode

This is a low-power state used in the sleep state of many power save modes. It is a low-power state where the external reference clock and many blocks in the chip are switched off. Only a slow sleep clock is used to maintain register and memory states. Wake-up does not require a firmware re-download

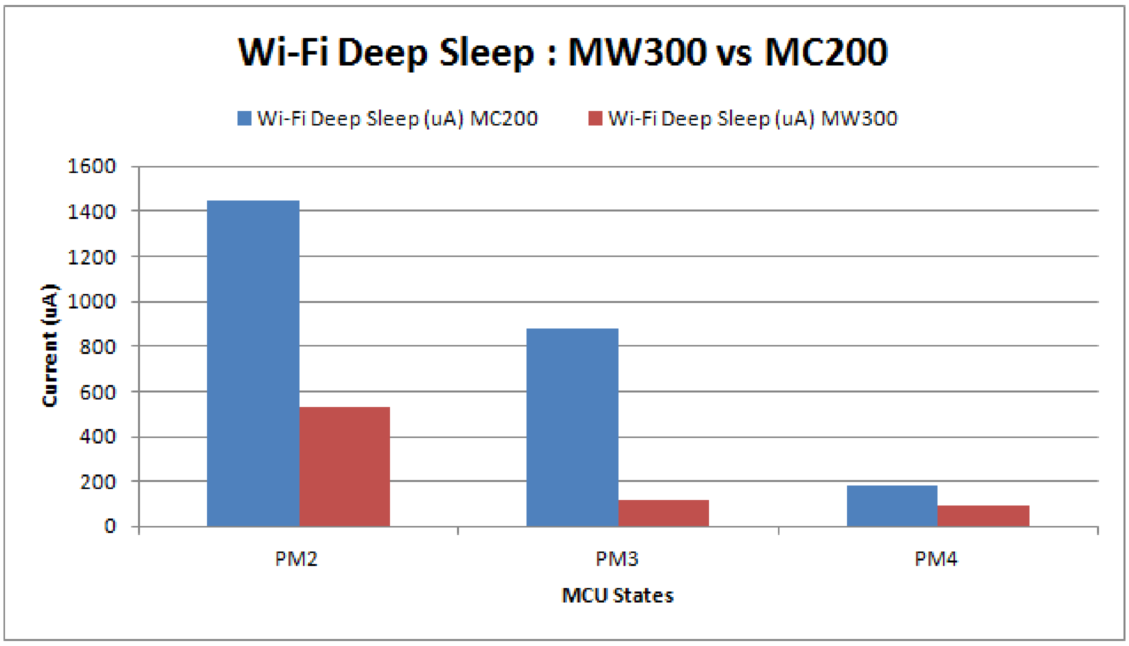

Figure 4.21: Wi-Fi Deep Sleep mode MW300 vs MC200

The fig. 4.21 shows the comparison of current consumed by various MCU states in the Wi-Fi deep sleep mode for MW300 and MC200 chipset. It can be clearly seen that in the standby mode i.e. PM2, the current consumed by MW300 is 63.59% lesser than that by MC200, in sleep mode i.e. PM3 current consumed by MW300 is 84.36% lesser than that by MC200 and in shut down mode i.e PM4 current consumed by MW300 is 48.31% lesser than that by MC200. Overall in wi-fi deep sleep mode the current consumed by MW300 is 66.09% lesser than that of MC200 chip. This implies that overall power consumed by MW300 is less as compared to MC200 in case of wi-fi deep sleep mode.

4.2.2 Wi-Fi IEEE PS DTIM=1 Mode

When the device is asleep in IEEE-power save mode, only a slow sleep clock is used to maintain register and memory states. However, there is a definite wake-up pattern. The device wakes up at regular intervals determined by the DTIM count and/or the listen interval.

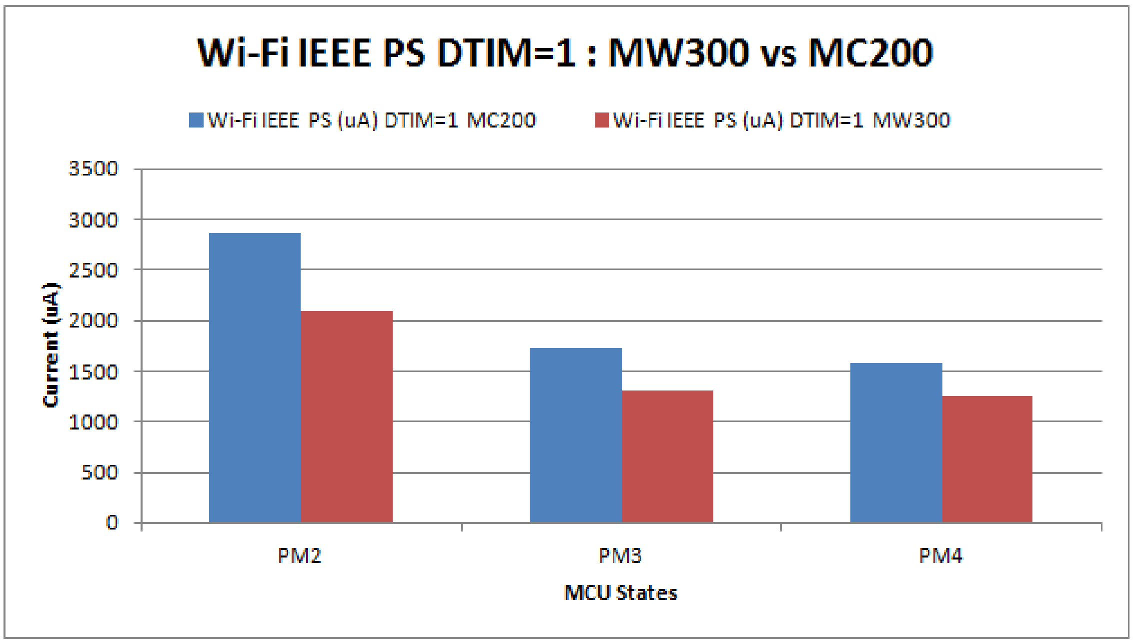

Figure 4.22: Wi-Fi IEEE PS (DTIM=1) mode MW300 vs MC200

The figure 4.22 shows the current consumption of MW300 and MC200 in wi-fi IEEE PS mode when the DTIM value is set to 1. For standby mode (PM2), there is 26.57% decrease in current consumption for MW300 as compared to MC200, for sleep mode (PM3), the current reduced by 24.28% with MW300 when compared to that of MC200 and for shut down mode (PM4), the current reduced by 21.13% with MW300 when compared to that of MC200 Overall there is 23.99% decrement in the current consumption for all the MCU states for MW300 over MC200.

4.2.3 Wi-Fi IEEE PS DTIM=3 Mode

The figure 4.23 shows the current consumption of MW300 and MC200 in wi-fi IEEE PS mode when the DTIM value is set to 3.

For Standby mode (PM2), there is 51% decrease in current consumption for MW300 as compared to MC200, for the sleep mode (PM3), the current consumption is reduced by 28.88% with MW300 when compared to that with MC200 and for shut down mode (PM4), the current consumption is reduced by 40% with MW300 when compared to that with MC200. Overall there is 39.96% decrement in the current consumption for all the MCU states for MW300 over MC200.

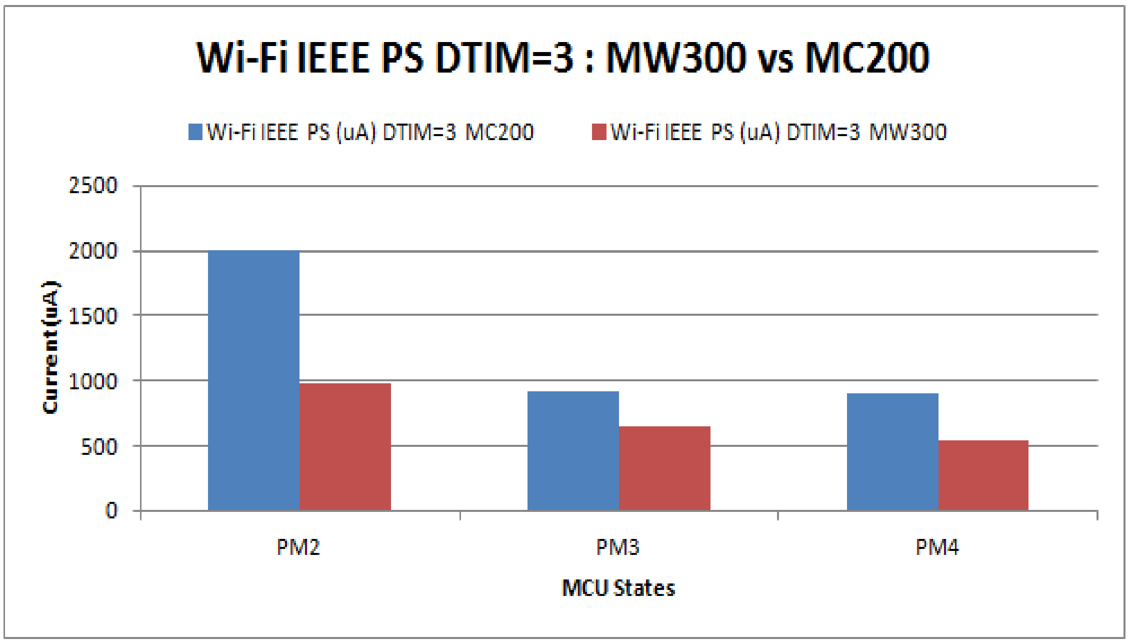

Figure 4.23: Wi-Fi IEEE PS (DTIM=3) mode MW300 vs MC200

As the value of DTIM is increased the overall difference significantly improves for MW300 as compared to MC200. All these indicate that MW300 consumes less current as compared to MC200 which means overall power consumed by MW300 is much lesser than that of MC200. Hence there is need to shift to MW300 chip for serial to wi-fi converter, to enhance the performance.

4.3 Summary

This chapter gives details about the result of performance analysis of ARM v8-A architecture with help of SPEC2000 integer benchmarks and performance analysis of Cortex M3 used with Marvell 8801 wi-fi chip.

Chapter 5

Significant Benchmark Analysis

In this chapter, the cynosure of analysis is on five selected benchmarks from SPEC2000 integer benchmark suit and they are gzip, mcf, crafty, parser and vortex. These five benchmarks showed vital performance distinction between 64-bit mode and 32-bit mode. Three of them (gzip, crafty and vortex) have performed higher in 64-bit mode and two (mcf and parser) performed higher in 32-bit mode.

Following table 5.1 shows the speed-up obtained by five benchmarks operating in 64-bit mode as against that in 32-bit mode.

Table 5.1: Benchmarks with significant performance difference

Benchmark	**Speedup of 64-bit mode over 32-bit mode**
gzip	10.30%
mcf	-54.68%
crafty	18.65%
parser	-15.99%
vortex	6.38%

5.1 gzip

gzip is a popular data compression program. SPEC's version of gzip performs only single file I/O which is reading the input. All compression and decompression takes place entirely in memory. This is to segregate the work done by the CPU and the memory subsystem.

gzip benchmarks reference workload has five ingredients : a large TIFF image, a webserver log, a program binary, random data, and a source tar file. Each input set is compressed and decompressed at various blocking factors ("compression levels"). The end result of the process is simultaneously compared to the original data after each step [1].

gzip performs highest no. of arithmetic operations as compared to all the other benchmarks in SPEC 2000 integer suit. When we move from 32-bit to 64-bit environment, from profiling of the benchmark, it is observed that the number of load-store instructions has been lowered in 64-bit mode which reduces the pressure on memory subsystem. This is one of the major points that help in speed up in 64-bit mode. As arithmetic operations are more, the wider register width can accumulate more data and speeds up the execution.

5.2 mcf

mcf is based on a program used for single-depot vehicle scheduling in public mass transportation [1]. This benchmark uses almost exclusively integer arithmetic. From the profiling, it is found that mcf has two main data structures which consume lot of overhead for its execution. These data structures consist of long data type and pointer data type operands. In 64-bit environment, size of long and pointer data types are doubled. As a result the code size has been increased in 64-bit mode. Same results have been observed for performance metrics such as instruction cache miss and data cache miss. Most of the instructions used for mcf benchmark are of integer operations and memory instructions. The mcf benchmark is key outlier in the spec 2000 suit and it executes faster on 32-bit environment as compared to 64-bit environment. And this is the only reason for overall degradation in the speed-up of whole benchmark suit in 64-bit mode against the 32-mode.

5.3 crafty

crafty is a high-performance Computer Chess program that is designed around a 64-bit word [1]. It runs on 32-bit machines exploiting the “long long ”data type. It is primarily an integer code, with a significant number of logical operations As the benchmark is designed around 64-bit word, there is no doubt that it provides speed up in 64-bit mode against the 32-bit mode. The code size of crafty benchmark reduces tremendously in 64-bit mode as compared to that in case of 32-bit mode.

Use of 64-bit word perceptibly causes overall reduction of dynamic instruction count in 64-bit mode. No. of indirect branches among the overall branch instructions are very less in case of crafty benchmark in 64-bit mode as against to 32-bit mode. This reduces the data cache misses which helps in improving the execution speed and ensures better performance in 64-bit mode.

5.4 parser

Parser is a syntactic parser of English. It is based on link grammar. Execution of parser consists of analysis of proposed input sentence from a dictionary of 60000 word forms. The analysis includes a set of links which capture the grammatical structure of the input sentence [1].

In the parser benchmark, it has been observed that there are more no. of branch instructions in 64-bit mode as against to that in 32-bit mode. This puts excessive load on memory subsystem and data cache misses are more in 64-bit as against to 32-bit mode because of this. This leads to degradation in performance in 64-bit mode as against to 32-bit mode.

5.5 vortex

vortex is a single-user object-oriented database transaction benchmark which exercises a system kernel coded in integer C.[1].

It has been observed that while executing in 64-bit mode there are less no. of indirect branch instructions as compared to that in 32-bit mode. This reduces the additional pressure on memory subsystem. The arithmetic operations are also reduced which can be observed from the instruction mix for this benchmark. Load-store instructions are also reduced in 64-bit mode. This reduces the cache memory access and helps in enhancement of speed of execution in 64-bit mode.

5.6 Summary

This chapter gives detailed analysis of the benchmarks which have shown considerable difference while moving to 64-bit from 32-bit mode. This analysis is useful for determining that, for which application areas 64-bit architecture is better and for some areas where 32-bit architecture is sufficient enough. This will help the developers in determining which architecture should be used for the specific purpose.

Chapter 6

Analysis Summary

In this chapter, first of all we figure out the major benefits and drawbacks of moving on to 64-bit mode from 32-bit mode environment. Then we sum up the analysis of five benchmarks as presented in chapter 5. After that we comment on the move of shifting from MC200 chip to MW300 for realising the serial to wi-fi converter IoT application.

6.1 ARM v8-A Architecture Analysis

The common attributes of SPEC 2000 integer benchmarks that suggest the benefits of moving to 64-bit mode environment are as follows.

1 64-bit integer arithmetic instructions are used

2 Loop bodies are present that require more than a few registers (Register pressure can also be increased because of loop unrolling which is a common compiler optimization).

3 Calls to the functions can be inlined.

4 Large number of registers available to accumulate neon and vfp instructions data.

5 Large Physical Address extension provides the mediaum for each of the software system environments to utilize expeditiously the available physical memory when handling enormous amounts of data. It helps in lowering the cache misses up to certain extent.

6 As the address space is more in the 64-bit mode, multiple OS systems can sustain easily and work efficiently in 64-bit mode.

The major traits of SPEC 2000 integer benchmarks that suggests potential pitfall for 64-bit mode environment are as follows.

1 Memory intensive applications; particularly those that have already experienced a high data cache miss rate in a 32-bit environment.

2 Rigorous use of long and pointer data types in terms of amount of memory allotted and the frequency of their access

3 Use of neon and vpf instructions can cause SP stalls which may introduce load on pipeline and cause higher cache access request increasing the load on memory subsystem.

Table 6.1 shows the key performance metrics (64-bit vs. 32-bit variations) of those five CPU2000 integer benchmarks and an outline of first order cause that ends up in these observed performance variations.

Table 6.1: Performance Difference Summary

Bench Marks	**Speed Up**	**Inst. Count**	**Load Store Inst.**	**Indirect Branches**	**Cause Of Performance Difference**
gzip	10.30%	-9.10%	-26.24%	-12.14%	Highest number of arithmetic operations is to be performed which can be performed using less no. of instructions in 64-bit mode. The load-store instruction count is reduced in 64-bit mode. Hence data cache misses are lower in 64-bit mode. this helps in obtaining the speed-up
mcf	-54.86%	-9.72%	0.08%	-24.57%	Use of long and pointer data types in 64-bit mode increases the code size, long data types adds pressure on memory subsystems, which increases the instructions and data cache misses, increasing the overall CPU cycles count that subsequently slows down the overall speed of operation in 64-bit mode
crafty	18.65%	-31.42%	-47.25%	-44.55%	Designed on 64-bit variable hence overall instruction count is reduced, the load-store and indirect branch instructions are drastically reduced in 64-bit mode, which reduces frequent memory access. Hence speed up is achieved.
parser	-15.99%	-6.18%	-10.95%	5.80%	More number of branch instructions and indirect branches and data cache misses are higher in 64-bit mode. Hence the slower performance.
vortex	6.38%	-21.05%	-19.73%	-26.38%	Less number of arithmetic operations and indirect branches in 64-bit mode reduces the load on memory subsystem. This enhances the performance

6.2 Cortex-M3 Analysis

From Section 4.2, it can be clearly observed that the current consumption of MW300 chip is much lesser than MC200 chip. This indicates that overall power consumed by MW300 is lesser than that of MC200. As low power is very critical thing in cases of IoT applications, the move to bring up the serial to wi-fi converter IoT application to a new MW300 chip is necessary.

Apart from low power consumption, MW300 is combination of Cortex-M3 micro-controller and Marvells 8801 wi-fi chip on a single die. This saves a lot of space on the chip and that space can be used for increasing the external flash memory. As increased memory is always very useful for some critical applications and can be utilised for providing additional features for the existing application and make the IoT application multi-functional. Hence it does make sense to bring up the existing serial to wi-fi application functional over MW300 chip.

6.3 Summary

This chapter gives overall summary and points on advantages and disadvantages of making the move to 64-bit architecture from 32-bit in case of ARM v8-A architecture and this chapter also covers the benefits of bringing up the serial to wi-fi application over a new MW300 chip.

Chapter 7

Test Scenarios

This chapter describes all the test scenarios of various features of serial to wi-fi converter like provisioning, network, http and https, sockets, websocket, async events that were being validated.

7.1 Test Cases

Serial to wi-fi converter module is tested over various features that it supports. All these test cases were written in node.js language. Node.js supports web-server creation which is useful in all test scenarios.

Following tables 7.1, 7.2, 7.3, 7.4, 7.5, 7.6 shows various test scenarios for provisioning, network, http and https, web-sockets, sockets and async events respectively

Table 7.1: Test Scenarios for Provisioning

Sr.No.	**Test Scenario**
1	Validate if device is in connected state then it can shift to provisioning state
2	Validate if device is in connecting state then it can shift to provisioning state
3	Validate if device is in disconnected state then it can shift to provisioning state

Table 7.2: Test Scenarios for Network

Sr.No.	**Test Scenario**
1	Validate 'Network not found'event if AP is down
2	Validate authentication failed event if passphrase is wrong
3	Validate if application goes in disconnected state after specific reconnect attempts
4	Validate if reconnect attempts equals the configured reconnect attempts available
5	WLAN configuration should work without rebooting the system
6	WLAN configuration should work even if executed after WLAN is started
7	Application should go to disconnected state after connect-failed state, if reconnect attempts are set to zero

Table 7.3: Test Scenarios for HTTP and HTTPS

Sr.No.	**Test Scenario**
1	Validate the connection with HTTP server, set header, post and get operations on the server and disconnect.
2	Validate the connection with HTTPS server, set header, post and get operations on the server and disconnect.
3	Find the maximum number of clients that can connect to HTTP/HTTPS server (Max. Clients = 8)
4	Find maximum number of headers that can be set.
5	Validate if the current header length is exceeding the maximum allowed header length (Max. allowed header length = 8K)

Table 7.4: Test Scenarios for Web-Sockets

Sr.No.	**Test Scenario**
1	Validate websocket send and receive the data
2	Validate websocket send/receive without upgrading the socket to websocket
3	Validate websocket send ping and receive pong
4	Validate websocket receive ping from the server

Table 7.5: Test Scenarios for Sockets

Sr.No.	**Test Scenario**
1	Validate TCP server bridge mode
2	Validate TCP client bridge mode
3	Validate UDP server bridge mode
4	Validate UDP client bridge mode

Table 7.6: Test Scenarios for Async Events

Sr.No.	**Test Scenario**
1	Validate async events for provisioning
2	Validate async events for network
3	Validate async events for HTTP and HTTPS
4	Validate async events for Sockets
5	Validate async events for Web-Sockets

7.2 Summary

This chapter covers various test scenarios of serial to wi-fi application that were being validated while working with MC200 chip and MW300 chip as well. These scenarios cover all the application framework area of serial to wi-fi converter IoT application.

Chapter 8

Conclusion And Future Scope

In this chapter, conclusion of the project work is described.

8.1 Conclusion

In this report, detailed analysis of performance variations observed between 64-bit and 32-bit modes of ARM v8-A architecture is done based on various performance metrics. The benchmarks from SPEC CPU2000 integer benchmarks suit are used for this study. We presented all the performance metrics which can affect the speed of operation and overall performance while migrating to 64-bit mode from 32-bit mode. We further analysed the benchmarks showing remarkable difference in performance to find the common program attributes that have shown advantage in one mode against the other.

It has been found that while migrating to 64-bit mode we can use wider registers and increased no. of general purpose registers for various operations more effectively. This will reduce the data cache misses and subsequently reduces the load on memory subsystem. Also the indirect branch instructions are lowered in case of 64-bit mode. For arithmetic operation, there are more general purpose registers are available with increased register width, which reduces the dynamic instruction count for 64-bit mode. Still the overall performance degrades by 5.85% in 64-bit mode against to 32-bit mode for the case of SPEC CPU2000 integer benchmarks because of performance of mcf benchmark. But still 64-bit mode can be beneficial for use on mobile and portable devices because of its other features at the cost of slightly degradation in speed-up for some applications.

Cortex-M3 micro-controller can be used for realising an IoT application serial to wi-fi converter because of all its features specially the operating speed of 300MHz which is quite faster as compared to other micro-controllers available. Also the various low power modes and MCU states of ARM cortex-M3 allows designers to enhance the efficiency in terms of power consumption of the device that is being used for serial to wi-fi converter IoT application. This serial to wi-fi converter application works perfectly over various features such as network, HTTP and HTTPS, websocket etc.

8.2 Future Scope

In the future, it would be beneficial to extend the analysis for the floating point benchmarks of SPEC CPU 2000 suit to find out how beneficial it is to move to 64-bit environment. Also some compiler optimization techniques can be investigated which can be helpful in mitigating the problems that encounter by these benchmarks while running on 64-bit mode.

References

[1] "SPEC Introduction, CPU2000 Integer Benchmarks", [Online]. Available: $http://www.spec.org/$ Last Updated: December 2014

[2] "Introduction to ARM v8 64-bit Architecture", [Online]. Available: $https://quequero.org/2014/04/introduction-to-arm-architecture/$, Last Updated: April 2014

[3] Richards Grisenthwaite, *ARMv8 Technology Preview*, ARM Tech Con, 2011

[4] *ARM Cortex A-57 MP Core Processor*, Revision r1p3, Technical Reference Manual 2013

[5] "Perf Basics and Command Options", [Online]. Available: $https://perf.wiki.kernel.org$ Last Updated: June 18, 2014

[6] "SPEC CPU2000 Integer Commands", [Online], Available: $http://kbarr.net/specint2000-commandlines$

[7] Shyam Sadasivan *An Introduction to the ARM Cortex-M3 Processor*, Oct 2006

[8] "Cortex-M3, Features, Technical Specificaion", [Online]. Available: $http://www.arm.com/products/processors/cortex-m/cortex-m3.php$

[9] *Marvell 88MC200 Microcontroller Datasheet*

[10] "88MC200 Chip, Features, Applications, Software ", [Online]. Available: $http://www.marvell.com/microcontrollers/wi-fi-microcontroller-platform/$