

UNIVERSITY OF THE
WITWATERSRAND,
JOHANNESBURG

Faculty of Science

1 Jan Smuts Avenue, Braamfontein 2000, Johannesburg, South Africa

---

Energy Reconstruction on the LHC ATLAS TileCal Upgraded Front End: Feasibility study for a sROD Co-Processing Unit

by

Mitchell Arij Cox

Dissertation presented in fulfilment of the requirements for the degree of:

Master of Science in Physics

Supervisors:

Prof. Bruce Mellado
School of Physics

Prof. Ivan Hofsajer
School of Electrical and Information Engineering

2016

Declaration

I, the undersigned, hereby declare that the work contained in this thesis is my own original work and that I have not previously in its entirety or in part submitted it at any university for a degree.



Mitchell Arij Cox

Braamfontein, Johannesburg
29 January 2016

Abstract

The Phase-II upgrade of the Large Hadron Collider at CERN in the early 2020s will enable an order of magnitude increase in the data produced, unlocking the potential for new physics discoveries. In the ATLAS detector, the upgraded Hadronic Tile Calorimeter (TileCal) Phase-II front end read out system is currently being prototyped to handle a total data throughput of 5.1 TB/s, from the current 20.4 GB/s. The FPGA based Super Read Out Driver (sROD) prototype must perform an energy reconstruction algorithm on 2.88 GB/s raw data, or 275 million events per second.

Due to the very high level of proficiency required and time consuming nature of FPGA firmware development, it may be more effective to implement certain complex energy reconstruction and monitoring algorithms on a general purpose, CPU based sROD co-processor. Hence, the feasibility of a general purpose ARM System on Chip based co-processing unit (PU) for the sROD is determined in this work.

A PCI-Express test platform was designed and constructed to link two ARM Cortex-A9 SoCs via their PCI-Express Gen-2 x1 interfaces. Test results indicate that the latency of the PCI-Express interface is sufficiently low and the data throughput is superior to that of alternative interfaces such as Ethernet, for use as an interconnect for the SoCs to the sROD. CPU performance benchmarks were performed on five ARM development platforms to determine the CPU integer, floating point and memory system performance as well as energy efficiency. To complement the benchmarks, Fast Fourier Transform and Optimal Filtering (OF) applications were also tested.

Based on the test results, in order for the PU to process 275 million events per second with OF, within the 6 μs timing budget of the ATLAS triggering system, a cluster of three Tegra-K1, Cortex-A15 SoCs connected to the sROD via a Gen-2 x8 PCI-Express interface would be suitable. A high level design for the PU is proposed which surpasses the requirements for the sROD co-processor and can also be used in a general purpose, high data throughput system, with 80 Gb/s Ethernet and 15 GB/s PCI-Express throughput, using four X-Gene SoCs.

Acknowledgements

The financial assistance of the South African National Research Foundation (NRF) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the author and are not necessarily to be attributed to the NRF.

I would also like to acknowledge the School of Physics as well as the School of Electrical and Information Engineering, the Faculty of Science and the Research Office at the University of the Witwatersrand, Johannesburg.

I would also like to acknowledge the "Massive Affordable Computing" project team - not only for the sharing of results and their interpretation, but as sounding boards and forums for ideas: Robert Reed, Matthew Spoor, Daniel O'Kwofie, Marc Sacks, Oscar Kureba and those at CERN and Valencia who have often proved invaluable sources of information and support: Carlos Solans and Alberto Valero.

I would like to thank both of my supervisors, Professors Bruce Mellado and Ivan Hofsajer for the funding and insight they provided to achieve this degree.

On a more personal note, I would like to sincerely acknowledge and thank the friends and family who have supported me over the last two years.

Most importantly, my girlfriend, Gillian Hulley who has put up with my long hours at the University and always kept the fridge stocked with good food and a home filled with love. She has been extremely supportive, never has any issues with my very early wake up times (and sometimes my early morning snoozing in winter while she gets up to go to work) and also my late homecomings after 18:00 on most days!

I doubt I would have got this far if it were not for my family. Of course, I am extremely grateful to my Mom and Dad for making sure I received a good education and loads of opportunities! Thank you Mom, Dad, Kirst, Omi and Opi, Granny and Grandpa (although you aren't around to read this), Aunt Corine, Uncle Marc, Bjorn, Emma and of course Rainer.

The office has been a pleasant place to get work done most of the time... For that I must thank my colleagues starting with my very knowledgeable friend Warren Carlson whom, for whatever reason, is *always* in the office. Getting to the office and grabbing a quality, strong cappuccino ("a cup of brown") at the Wits Art Museum at 7:30 with him is a good start to any day.

The office conversations and debates have been a most excellent source of distraction and would not have been possible without my colleagues in the office: Rob, Mbavhalelo, Matt, Marc, Steph and Chad. I must also thank Steve who is always keen to "hit the PiG" (the Postgraduate Club) for lunch.

Again, I would like to thank Bruce for the opportunity and Rob for introducing me to the project at a braai in 2013. Bruce has funded a lot of expensive equipment to make the project happen, as well as a trip to CERN (which let me tick off a personal bucket list item as well as feel proud to be an official member of the ATLAS experiment) and several oversees conferences. I think I have been very lucky!

Thanks again to Professor Hofsajer for the open door policy and the many interesting discussions about all sorts of things - usually at 6:15 in the morning after utilizing the EIE coffee machine.

Mitchell Cox

October 2015

Contents

Declaration i

Abstract ii

Acknowledgements iii

List of Figures vii

List of Tables xi

Nomenclature xii

List of Resulting Publications xvi

1 Introduction 1
- 1.1 Background . . . 2
- 1.2 Hypothesis . . . 3
- 1.3 Dissertation Outline . . . 4

2 The Large Hadron Collider and ATLAS 5
- 2.1 The ATLAS Experiment . . . 7
  - 2.1.1 Inner Detector . . . 8
  - 2.1.2 Liquid Argon Calorimeters . . . 10
  - 2.1.3 Hadronic Tile Calorimeter . . . 12
  - 2.1.4 Muon Spectrometer . . . 13
- 2.2 Summary . . . 15

3 The Hadronic Tile Calorimeter Read Out System 16
- 3.1 Online Energy Reconstruction Algorithm . . . 17
  - 3.1.1 The Optimal Filtering Algorithm . . . 19
- 3.2 Existing Front End Electronics . . . 22
- 3.3 Read Out Driver (ROD) . . . 23
- 3.4 Upgraded Phase-II Front-End . . . 24
  - 3.4.1 Mini Drawer . . . 25
  - 3.4.2 AdvancedTCA Chassis . . . 25
  - 3.4.3 Super Read Out Driver . . . 28

4 General Purpose sROD Co-Processing Unit 30
- 4.1 ARM System on Chips . . . 32
  - 4.1.1 ARM Test Platforms . . . 34

4.2 System on Chip External I/O Interfaces . . . 36
4.2.1 Ethernet . . . 39
4.2.2 USB . . . 41
4.2.3 PCI-Express . . . 42
4.3 PCI-Express Performance Tests . . . 47
4.3.1 Wandboard Test Hardware . . . 49
4.3.2 Throughput Tests . . . 49
4.3.3 Round Trip Latency . . . 53
4.3.4 Summary and Conclusion . . . 54
4.4 sROD - PU Interconnect Considerations . . . 55
4.5 Processing Unit Internal Design Considerations . . . 57

5 ARM Computing Benchmarks 59
5.1 CPU Performance . . . 62
5.1.1 CoreMark . . . 63
5.1.2 High Performance Linpack . . . 66
5.2 Memory Performance . . . 69
5.2.1 Memory Bandwidth . . . 70
5.2.2 Memory Latency . . . 76
5.3 Energy Efficiency . . . 78
5.3.1 Factors that Affect Energy Efficiency . . . 78
5.3.2 Measurement Methodology . . . 80
5.3.3 Results . . . 81

6 Application Benchmarks 84
6.1 Fast Fourier Transforms . . . 84
6.1.1 Single and Multi-Core Benchmarks . . . 85
6.1.2 Multi-Process Benchmarks . . . 89
6.1.3 Theoretical FFT Throughput . . . 90
6.2 Optimal Filtering . . . 92
6.2.1 Test Methodology . . . 92
6.2.2 Results . . . 93
6.2.3 Integration with sROD . . . 94

7 Conclusions 98
7.1 External I/O . . . 99
7.2 CPU Performance . . . 100
7.3 Memory Bandwidth . . . 101
7.4 sROD Co-Processor Feasibility . . . 102
7.5 Future Work . . . 103

References 104

Appendix A 113
Wandboard Adapter Circuit Diagram and Layout . . . 113

List of Figures

2.1 Proton - (Anti)Proton Standard Model cross sections for the LHC and Tevatron [13]. 6
2.2 ATLAS Online Luminosity plot showing mean interactions per crossing for 2011 and 2012 data [18]. 7
2.3 A cutaway 3D rendering of the LHC ATLAS detector [19]. 8
2.4 Sub-detector layers of ATLAS showing where different particles are detected. Dotted lines indicate no detection [19]. 9
2.5 A cutaway 3D rendering of the ATLAS inner detector [19]. 10
2.6 A cutaway 3D rendering of the ATLAS calorimeters [19]. 11
2.7 A diagram of an Electromagnetic (EM) calorimeter barrel module showing the accordion shaped layers of lead absorber and copper electrode. . . . 11
2.8 Mechanical assembly of a TileCal module [12]. 13
2.9 Segmentation in depth and η of cells on half of the TileCal [12]. 13
2.10 ATLAS Muon Spectrometer 3D rendering [19]. 14

3.1 Hard drive and Ethernet 802.3 data throughput and CPU performance on a log scale with time [24], [25]. 17
3.2 Block diagram of the ATLAS Trigger and Data AcQuisition system (TDAQ). 18
3.3 PMT pulse showing seven samples with one out of time pile up signal at +50 ns. 19
3.4 Block diagram of the existing TileCal Read Out Architecture. 22
3.5 Ideal PMT pulse showing seven samples with the pedestal, peak amplitude and phase indicated. 23
3.6 Block diagram of the planned Phase II TileCal Read Out Architecture. 24
3.7 Photo of the TileCal mini drawer prototype with Photo-Multiplier Tubes (PMTs), Daughter Board and Main Board visible [41]. 26
3.8 Photo of the completed mini drawer daughter board which is the interface to the sROD. 27
3.9 Photo of an AdvancedTCA or Advanced Telecommunications Computing Architecture (ATCA) chassis in the lab at the University of the Witwatersrand, Johannesburg, with a carrier board and CERN Gigabit Link Interface Board (GLIB) card partially inserted. 27
3.10 Block diagram of the sROD prototype I/O interfaces and FPGAs. . . . 29
3.11 Photo of the sROD prototype with QSFP+ ports on the left and AMC connector on the right. 29

4.1 ARM Cortex-A CPU roadmap from the Cortex-A8 in 2005 to Cortex-A57 in late 2013 [53]. 33

4.2 Photo of a Cubieboard A20 in the High Throughput Electronics Lab (HTEL) at the university. 34
4.3 Photo of a Wandboard Quad with heatsink removed in the HTEL at the university. 34
4.4 Photo of a NVIDIA Jetson-TK1 in the HTEL at the university. 35
4.5 Photo of an ODROID-XU+E in the HTEL at the university. 35
4.6 Photo of the X-Gene Mustang board in the HTEL at the university. . . 36
4.7 A rendering of the latest Mont-Blanc prototype blade [72]. 42
4.8 PCI-Express switching with a standard configuration to the left and a dual CPU failover configuration on the right. 44
4.9 Connection between two CPUs with an RC - EP connection on the left. To the right, the sROD can be configured as RC with multiple CPUs connected via EP or NT ports. 44
4.10 Structure and size of a PCIe packet with transaction layer parts highlighted in orange and data link layer in grey. The number of Bytes for each part is shown in brackets. For applicable acronyms see text. 45
4.11 PCIe bandwidth efficiency with a fixed 24 byte packet overhead and varying data payload size. 46
4.12 Simple flow of a PCIe Write and Read transaction showing the memory locations of device A and B and their contents. 47
4.13 Photo of the Wandboard PCI-Express Adapter with Wandboards populated. The PCB was manufactured in South Africa and hand soldered by the author. 49
4.14 Measurement of Linux kernel based PCIe transfer rate for Memory Copys (memcpys) from 128 kiB to 8 MiB without DMA. 50
4.15 PCIe 4 byte packet ping-pong latency histogram, normalised to unity, from 1.1 μs to 1.3 μs. 54
4.16 Diagram of connection between sROD and PU via ATCA Rear Transition Module (RTM). 56
4.17 Diagram showing four server grade System on Chips (SoCs) (X-Genes) interconnected with PCI-Express, with a total of 80 Gb/s Ethernet and 15.76 GB/s PCI-Express external connectivity. 57
4.18 Diagram showing a server grade gateway SoC for high speed external connectivity and using PCI-Express for internal connectivity to low cost worker SoCs with and without a PCIe switch. 58

5.1 CoreMark results for a single CPU core sweep of the available clock frequencies. 65
5.2 CoreMark results for a sweep of the available clock frequencies utilising all CPU cores. 65
5.3 Peak double precision HPL results versus clock frequency. The X-Gene is not shown because changing the CPU clock was not possible. 68
5.4 Peak double precision HPL results versus clock frequency, normalised to a single core. The X-Gene is not shown because changing the CPU clock was not possible. 68
5.5 Typical memory hierarchy showing CPU registers, L1, L2, optional L3 caches and RAM as well as their relative capacities. 70
5.6 X-Gene STREAM Copy results in GiB/s for array sizes from 4 kiB to 128 MiB with gcc 4.9 and different numbers of CPU cores to demonstrate varying levels of optimization “success” in comparison to Figure 5.7. . . 72

5.7 X-Gene STREAM copy results in GiB/s for array sizes from 4 kiB to 128 MiB using gcc 4.8 and different numbers of CPU cores to demonstrate varying levels of optimization "success" in comparison to Figure 5.6. . . 73
5.8 X-Gene STREAM results in MB/s for array sizes from 4 kiB to 128 MiB using 8 cores at 2.4 GHz. 73
5.9 Tegra-K1 STREAM results in GiB/s for array sizes from 4 kiB to 64 MiB using 4 cores at 2.3 GHz. 74
5.10 Exynos 5410 STREAM results in GiB/s for array sizes from 4 kiB to 64 MiB using 4 cores at 1.6 GHz. 74
5.11 i.MX6 STREAM results in GiB/s for array sizes from 4 kiB to 64 MiB using 4 cores at 996 MHz. 75
5.12 A20 (Cortex-A7) STREAM results in GiB/s for array sizes from 4 kiB to 64 MiB using 2 cores at 1.008 GHz. 75
5.13 Memory latencies for all platforms from 512 B to 64 MiB. 77
5.14 Simplified circuit diagram of the DC power measurement. 80
5.15 Load power of the platforms excluding the X-Gene (see text) for an all core High Performance LINPACK (HPL) benchmark at varying frequencies. 81
5.16 Power efficiency of the platforms at varying frequency with DP HPL. . . 82
5.17 Weighted energy efficiency for more performance oriented workloads as suggested in [100]. 83

6.1 FFTW single core results at maximum CPU frequency for single precision complex FFT's from 2 to 2^{21}. 86
6.2 FFTW single core performance at maximum CPU frequency for single precision complex FFT's with $N \in [2 : 2^{21}]$. The shaded areas show L1 cache in blue and L2 and L3 caches in orange. 87
6.3 FFTW dual core results at maximum CPU frequency for single precision complex FFT's from 2 to 2^{21}. 88
6.4 FFTW quad core results at maximum CPU frequency for single precision complex FFT's from 2 to 2^{21}. 88
6.5 FFTW eight core results at maximum CPU frequency for single precision complex FFT's from 2 to 2^{21}. 89
6.6 FFTW maximum results at maximum CPU frequency for single precision complex FFT's from 2 to 2^{21} with any number of cores. 89
6.7 FFTW maximum multi-process equivalent FLOPS at maximum CPU frequency for single precision complex FFT's from 2 to 2^{21}. 90
6.8 FFTW peak multi-core and multi-process equivalent FLOPS at maximum CPU frequency for single precision complex FFT's from 2 to 2^{21}. 91
6.9 FFTW peak multi-core and multi-process equivalent FLOPS at maximum CPU frequency for single precision complex FFT's from 2 to 2^{21}. 92
6.10 Arbitrary PMT pulse showing seven samples with the pedestal, peak amplitude and phase illustrated. 93
6.11 Optimal Filtering (OF) benchmark at maximum CPU frequency with one core. 94
6.12 OF benchmark normalised to 1 GHz CPU frequency with one core. . . . 94
6.13 OF benchmark at maximum CPU frequency scaled to all cores. Maximum theoretical PCIe throughputs are shown on the right. 95
6.14 Serial, parallel and combination queue methods for events. 96

1 PCB layout of the Wandboard Adapter. 114

2 Wandboard Adapter circuit diagram, Page 1. 115
3 Wandboard Adapter circuit diagram, Page 2. 116
4 Wandboard Adapter circuit diagram, Page 3. 117

List of Tables

1 Common binary and decimal prefixes used for units of data. xv

3.1 Summary of specifications for the sROD prototype [3]. 29

4.1 External I/O interface specifications for the available ARM platforms. . 38
4.2 Performance of the Open-MX networking stack compared to the default Linux stack [65]. 40
4.3 Specification comparison between PCI-Express generations for one lane [74]. 43
4.4 PCI-Express specifications for the available test platforms that support PCI-Express. 45
4.5 PCI-Express Memory Map (mmap) data throughput results. 52
4.6 PCI-Express Direct Memory Access (DMA) data throughput results with i.MX6 Image Processing Unit (IPU) 53

5.1 Specifications of the ARM development and evaluation platforms that were tested. 61
5.2 Optimal gcc flags for the CoreMark benchmark. 63
5.3 Comparison of CoreMark results for gcc 4.8 and 4.9.2. 64
5.4 CoreMarks per MHz (one thread) comparison between ARM and Intel CPUs. 66
5.5 Peak single (SP) and double precision (DP) HPL results for the ARM platforms. It was not possible to run the single precision benchmark on the X-Gene. 67
5.6 CPU calculated GFLOPS per MHz per core. 69
5.7 Operations performed and tested by the STREAM benchmark. 71
5.8 Calculated theoretical peak performance of the ARM CPUs. 76
5.9 Measured memory latencies of the ARM CPUs in nanoseconds using lmbench. Equivalent clock cycles are in brackets. Lower latencies are better. 77
5.10 Power consumption and GFLOPS/W at maximum frequency for all platforms. 82

6.1 Maximum block size for each SoC (single core) within the TileCal Trigger latency requirements assuming 4.8 μs available time. 97

Nomenclature

ADC Analog to Digital Converter

ALICE A Large Ion Collider Experiment

AMC Advanced Mezzanine Card (ATCA)

API Application Programming Interface

ARM A microprocessor architecture licensed by ARM Holdings

ASIC Application Specific Integrated Circuit

ATCA AdvancedTCA or Advanced Telecommunications Computing Architecture

ATLAS A Toroidal LHC Apparatus

bit (b) A boolean (binary) unit with two possible values: 0_2 and 1_2

BLAS Basic Linear Algebra Subprograms

BW Bandwidth

Byte (B) An 8 bit numerical unit in a computer: 00000000_2 (0_{10}) to 11111111_2 (255_{10})

CISC Complex Instruction Set Computing

CMS Compact Muon Solenoid

CoreMark A modern CPU integer benchmark utility where the score is given in CoreMarks

CPU Central Processing Unit

CSC Cathode Strip Chamber (ATLAS Muon Spectrometer)

DFT Discrete Fourier Transform

DLLP Data Link Layer Packet (PCI-Express)

DMA Direct Memory Access

DMIPS Dhrystone MIPS

DSP Digital Signal Processor (or Processing)

ECC Error Correcting Code

ECM Ethernet Control Model

EEM Ethernet Emulation Model

EMEC LAr Electromagnetic End-Cap

EM Electromagnetic

EP Endpoint

FATALIC Front end for ATLAS TilecAL Integrated Circuit

FCal Forward Calorimeter of the ATLAS Electromagnetic Calorimeter

FFT Fast Fourier Transform

FFT Fast Fourier Transform

FLOPS Floating Point Operations per Second

FPGA Field Programmable Gate Array

FWHM Full Width at Half Maximum

GLIB Gigabit Link Interface Board

GPGPU General Purpose Graphics Processing Unit

GPU Graphics Processing Unit

HEC LAr hadronic End Cap

HEP High Energy Physics

HL-LHC High Luminosity Large Hadron Collider (LHC)

HPC High Performance Computing

HPL High Performance LINPACK

HTEL High Throughput Electronics Lab (at the University of the Witwatersrand, Johannesburg)

HVC High Volume Computing

HV High Voltage

I/O Input/Output

IPU Image Processing Unit (on i.MX6Q SoC)

IP Intellectual Property or sometimes Internet Protocol, depending on the context

LAr Liquid Argon

LHC Run 2 LHC operation at p-p energies of 13 TeV (2015 - 2018)

LHCb Large Hadron Collider beauty

LHCf Large Hadron Collider forward

LHC Large Hadron Collider

LINPACK Benchmark A benchmark to measure how fast a computer solves a dense n by n system of equations of the form $Ax = b$

LS1 Long Shutdown 1 (2013 - 2014)

MAC Media Access Control

MDT Monitored Drift Tube (ATLAS Muon Spectrometer)

memcpy Memory Copy (a C or C++ API function)

MIPS Million Instructions per Second

mmap Memory Map (a Linux specific mechanism)

MoEDAL Monopole and Exotics Detector At the LHC

MPI Message Passing Interface

NCM Network Control Model

NDA Non Disclosure Agreement

NT Non-Transparent

OF Optimal Filtering

OOT Out Of Time (with respect to pile up)

PHY Physical Layer

PMT Photo-Multiplier Tube

PU Processing Unit

QF Quality Factor (with respect to online energy reconstruction)

QIE Charge (Q) Integrator and Encoder

RAM Random Access Memory

RC Root Complex

RDMA Remote Direct Memory Access

RISC Reduced Instruction Set Computing

ROB Read Out Board

ROD Read Out Driver

RPC Resistive Plate Chamber (ATLAS Muon Spectrometer)

RTM Rear Transition Module (ATCA)

SIMD Single Instruction Multiple Data

SKA Square Kilometre Array

SM Standard Model (Physics)

SoC System on Chip

sROD Super Read Out Driver

TCP Transmission Control Protocol

TDAQ Trigger and Data AcQuisition system

TDP Thermal Design Power

TGC Thin Gap Chamber (ATLAS Muon Spectrometer)

TileCal Hadronic Tile Calorimeter

TLP Transaction Layer Packet (PCI-Express)

TOTEM TOTal Elastic and diffractive cross section Measurement

UDP User Datagram Protocol

USB Universal Serial Bus

x86 A CPU architecture and instruction set originally designed by Intel

Note on Numerical Units in Computing

The prefixes used for decimal numbers are different from those used for binary numbers as shown in Table 1. The result of this is that in many cases the units used for specifying data sizes are confusing. For example, a 1 TB hard drive is 1000^4 Bytes which the operating system will report as approximately 976 GiB (or confusingly just 976 GB). Often, mistakenly, the "i" is omitted when multiples of 1024 are used. This potential confusion is undesirable and so in this dissertation the prefixes in Table 1 will be strictly adhered to.

Table 1: Common binary and decimal prefixes used for units of data.

Decimal		**Binary**	
Value_{10}	SI Prefix	Value_2	IEC Prefix
1000	k (kilo)	1024	Ki (kibi)
1000^2	M (mega)	1024^2	Mi (mibi)
1000^3	G (giga)	1024^3	Gi (gibi)
1000^4	T (tera)	1024^4	Ti (tebi)
1000^5	P (peta)	1024^5	Pi (pebi)

List of Resulting Publications

- M. A. Cox, R. Reed, and B. Mellado, "The development of a general purpose ARM-based processing unit for the ATLAS TileCal sROD," in *Journal of Instrumentation*, vol. 10, no. 1, 2015.
- M. A. Cox and B. Mellado, "The Massive Affordable Computing Project: Prototyping of a High Data Throughput Processing Unit," in *Journal of Physics: Conference Series*, vol. 608, no. 1, pp. 012002, 2015.
- M. A. Cox, "Towards Optimal Filtering on ARM for ATLAS Tile Calorimeter Front-End Processing," in *Journal of Physics: Conference Series*, vol. 645, no. 1, pp. 012018, 2015.
- M. A. Cox and B. Mellado, "The development of an ARM System on Chip based Processing Unit for Data Stream Computing," in *Computer Research and Modelling*, vol. 7, no. 3, pp. 499-503, 2015.
- R. Reed, M. A. Cox, T. Wrigley, and B. Mellado, " CPU Benchmarking Characterization of ARM Based Processors," in *Computer Research and Modelling*, vol. 7, no. 3, 2015.
- M. A. Cox and B. Mellado, "Online energy reconstruction on ARM for the ATLAS TileCal sROD co-processing unit," presented at *SAIP2015, the 60th Annual Conference of the South African Institute of Physics*, 2015.
- M. A. Cox and R. Reed, "Affordable and Power Efficient Computing for High Energy Physics: CPU and FFT benchmarks of ARM processors," in *Proceedings of SAIP2014, the 59th Annual Conference of the South African Institute of Physics*, pp. 180-184, 2014.
- M. A. Cox and B. Mellado, "The Development of a General Purpose Processing Unit for the Upgraded Electronics of the ATLAS Tile Calorimeter," in *Proceedings of SAIP2014, the 59th Annual Conference of the South African Institute of Physics*, pp. 175-179, 2014.

1

Introduction

The Large Hadron Collider (LHC) on the border of Geneva in Switzerland and France is currently the largest particle accelerator in the world. Inside the LHC, protons are collided at combined energies of 8 TeV in Run 1 and now 13 TeV in Run 2. This massive scientific and engineering effort has led to the discovery of the Standard Model (SM) Higgs boson in 2012 by the ATLAS and CMS experiments which are a part of the LHC [1].

The Standard Model of particle physics describes and classifies all of the known subatomic particles as well as the interactions between them in terms of the electromagnetic, weak and strong nuclear forces. The Standard Model initially had no explanation for particles that have non-zero mass and so the so-called Higgs boson was theorised in the 1960's as a scalar elementary particle which solved this problem. Unfortunately, the Standard Model is incomplete as it does not incorporate theories regarding gravity, dark matter and dark energy. Several Beyond the Standard Model (BSM) theories exist that may explain some of these phenomena but it is one of the aims of the LHC to one day help prove or disprove these theories.

The ATLAS detector at CERN is a 7000 ton instrument designed to probe the decay products of proton-proton collisions. The detector is general purpose and is able to detect photons, electrons, muons as well as neutrons and protons. Each type of particle requires different detection techniques and so the ATLAS detector is made up of layers of different sub-detectors. In total, hundreds of millions of sensor elements produce a vast quantity of data that must be distilled into useful physics information.

In the LHC, bunches of protons are accelerated in opposite directions to relativistic velocities. In Run 1 these bunches are steered to intersect inside the detectors every 50 ns or 20 MHz. With each bunch crossing, up to 20 protons collide at different locations, or vertices, inside the beam pipe. This so-called interaction rate is on the order of 20×20 MHz for Run 1. In Run 2, as of 2015, the bunch crossing rate will be increased to 25 ns or 40 MHz. The "term" in time pile up is used to describe the resulting superposition of measurements in the detectors resulting from multiple interactions per bunch crossing.

Some proton-proton collisions undergo hard scattering whereby many different types of interactions can take place with certain probabilities. The final products of this hard scattering process might be interesting particles such as Higgs bosons

or even exotic, as yet undiscovered particles. These final products may decay into different particles of which some undergo hadronisation or even further decays. This process eventually results in stable particles. When these stable particles pass through the detector they are detected and traced back through a computational process called event reconstruction which seeks the original state of the collision. Pile up multiplies the complexity of this process as it is sometimes difficult to discern which of the resulting particles belong to which vertices (collision points).

Petabytes per second of raw data are therefore produced by ATLAS. This quantity of data is impossible to store in it's entirety for analysis and so a high data throughput system is used whereby events are analysed in real time and only those that are interesting are stored. The ATLAS Trigger and Data AcQuisition system (TDAQ) is used for this process. For Run 2, the TDAQ must effectively reduce the number of events from the 40 MHz bunch crossing rate to a rate on the order of 200 Hz which is approximately 300 MB/s of data for offline storage.

Future upgrades to the TDAQ must be energy efficient, affordable and able to sustain a very high data throughput. The objective of this dissertation is to propose and study the feasibility of using ubiquitous, mobile ARM-based System on Chips (SoCs) for the Hadronic Tile Calorimeter (TileCal) read out system, which is a component of the ATLAS TDAQ.

1.1 Background

Hadrons such as protons and neutrons are detected by the TileCal, which is made up of alternating plates of steel absorber and plastic scintillator surrounding the interaction point at the centre of the ATLAS detector. Each tile is read out by two redundant Photo-Multiplier Tubes (PMTs) and the resulting analog pulses are digitised. An energy reconstruction algorithm called Optimal Filtering (OF) is used to determine the primary characteristics of each pulse based on the digitised samples, namely the peak amplitude, phase shift and pedestal (or noise floor). The particle energy and interaction time with the tile can be calculated from these parameters.

Since each scintillator tile has quite a large area, there is a high chance that several particles will interact with it from separate collisions and interactions and thus pile up is a concern. If the level of pile up is too high, causing too much distortion of the digitised pulse, the event data is rejected. If too many events are rejected then the overall efficiency of the ATLAS detector is lowered and hence less useful physics data is available for analysis. In extreme cases this may even affect the viability of the experiment as a whole. For new discoveries, large numbers of events must be collected in order to have adequate statistics for analysis. If the detector efficiency is too low then the time required to gather sufficient data to produce meaningful results may become unreasonable.

From 2020 the data production potential of the LHC will begin to plateau. The integrated luminosity, which is a measure of the total number of collisions and therefore the amount of data collected, will be increasing too slowly for new discoveries to be made. It is estimated that in order to halve the statistical error of measurements in 2020, the LHC will have to collect data for a further 10 years [2].

Because of this, there are plans to upgrade the LHC into what is called the High Luminosity LHC (HL-LHC). This upgrade has been named the Phase-II upgrade. The HL-LHC will have nearly double the number of protons per bunch as the current LHC and will have mechanisms to steer the beams more precisely in order to increase

the interaction rate and therefore the luminosity. The planned pile up of the HL-LHC will be on the order of 200 interactions per bunch crossing, which is ten times the current level. In terms of TileCal, the order of magnitude increase in pile up is an outstanding problem that must be solved and is the basis for the research presented in this dissertation.

The primary component of the TileCal Phase-II read out system is called the Super Read Out Driver (sROD) [3]. The sROD is based on Field Programmable Gate Arrays (FPGAs) which are very high performance, programmable devices that are capable of processing very high throughput data. Unfortunately, FPGAs are expensive and it is difficult and time consuming to write complex firmware for them.

The sROD will perform energy reconstruction on the raw data produced by the TileCal front end. It will also function as a buffer and Level-0 trigger, reducing the data rate from 40 MHz to approximately 500 kHz for the next stage of the TDAQ. The current Optimal Filtering (OF) algorithm is not suitable for a pile up of 200 and so new algorithms are in development by several research groups around the world [4], [5].

Implementing these new algorithms on FPGA may present a formidable challenge and as yet unknown future algorithms may indeed prove infeasible to implement on FPGA.

1.2 Hypothesis

ARM SoCs, as opposed to Intel x86 based SoCs, are found in the majority of mobile devices such as smart phones, tablets and even mini-PCs such as the Raspberry Pi [6], [7]. The reasons for this prevalence is low power consumption, low cost and also good CPU performance.

ARM is a company that develops CPU architectures and designs (known simply as IP) and then licenses these to other companies which in turn modify and combine the CPU IP with peripheral Intellectual Property (IP) obtained from other vendors to form SoCs. This model allows great variety in SoC features and performance and hence cost.

With every generation of mobile device, for example, consumers demand better battery life, higher performance and to some extent lower cost. These pressures lead to higher performance ARM SoCs which have recently attained a level of performance comparable to desktop and server machines except with lower power consumption. For this reason, several research groups and companies around the world are starting to look at ARM SoCs for server and cluster use.

Both HP and Dell have released servers based on ARM, specifically the HP Moonshot 1500 and the Dell Copper chassis [8], [9]. The Mont-Blanc project, based at the Barcelona Supercomputing Center, aims to develop an energy efficient supercomputer based on ARM SoCs [10] and Applied Micro has also recently released a high performance, server grade ARM SoC [11].

The energy efficiency and low cost of ARM SoCs make it an attractive candidate for a Processing Unit (PU) to be used in the ATLAS TileCal read out system as an sROD co-processor. This enables the sROD to offload tasks that may be challenging or inefficient to implement on FPGA.

This PU is in development at The University of the Witwatersrand High Throughput Electronics Lab (HTEL) by a team both local and at the Institute of Particle

Physics at the University of Valencia and CERN led by the author and this dissertation forms part of the preliminary research.

The primary research question addressed in this dissertation is therefore whether it is possible to satisfy the tight timing and data throughput requirements of the TileCal front end using ARM based SoCs, and if so, how well they are able to perform the task?

1.3 Dissertation Outline

A detailed overview of the LHC, and more importantly the ATLAS experiment, is given in Chapter 2. The TileCal read out system as well as its Phase-II upgrades are summarised in Chapter 3 with a specific emphasis on the optimal filtering algorithm and the required specifications of the system.

In order to meet these specifications using general purpose processors, Chapter 4 discusses the I/O interface requirements for the SoCs. First, an overview of what is typically available along with the advantages and disadvantages of the various interfaces are given. A test system for PCI-Express, which is theoretically the best available candidate for a high data throughput and low latency external I/O interface, is developed and promising results are presented as a validation of the interface for general purpose use. The method to interconnect the sROD and SoCs via PCI-Express is explored and proposed.

The assumption that ARM SoCs have adequate computing performance is tested through several benchmarks in Chapter 5. CPU integer as well as floating point performance in Floating Point Operations per Second (FLOPS) is measured and compared amongst the available platforms. A comparison to some Intel x86 based CPUs is also provided. The memory performance of the SoCs is also measured in terms of data bandwidth and latency. Finally, the energy efficiency of the various platforms is presented.

The synthetic performance benchmarks presented in Chapter 5 enable fair comparison to other computing systems but it is difficult to accurately calculate the performance of a particular algorithm based on these results. In Chapter 6, Fast Fourier Transform (FFT) and OF applications are benchmarked and the run time results for various data sizes are measured.

The FFT is a useful component of many mathematical, scientific and engineering applications. The OF implementation is based on that described in Chapter 3 and the results are used to propose a high level PU design using several ARM SoCs that satisfy the data throughput and latency requirements of the sROD.

Chapter 7 summarises the results obtained and concludes.

2

The Large Hadron Collider and ATLAS

The Large Hadron Collider (LHC) at The European Organisation for Nuclear Research or CERN (Conseil Européen pour la Recherche Nucléaire), which straddles the border of France and Switzerland at Geneva, is currently the worlds largest and highest energy particle accelerator with a circumference of 27 km [12].

In the LHC two beams of protons (or sometimes heavy ions) are circulated in the LHC beam pipes in opposite directions. The beams are normally isolated from each other but at interaction points inside the LHC experiments they cross, allowing the protons to collide with each other, producing a spray of different particles.

The resulting particles may have high mass and will usually decay after a very short time into other particles. Eventually, after potentially many different random interactions and decays, only stable particles remain. These stable particles such as photons, electrons, muons and even new protons and neutrons can be detected by the experiments built around the interaction points.

Neutrinos are produced often and are stable particles that are nearly impossible to detect, and therefore show up as "missing energy" which is determined by calorimeters in a detector.

Figure 2.1 shows the predicted production rates of various particles in proton - proton (p-p) collisions at varying energies [13]. It is clear that at higher energies, heavier particles such as the Higgs boson which was announced on 4 July 2012 and subsequently confirmed in 2013, for example, will be produced more often [1].

Some particles are produced at an extremely low rate and so accumulating enough measurements in order to make new discoveries and contributions to the field is difficult and time consuming, which is why higher energy particle accelerators are important. The detection efficiency of the experiments is also important so as not to miss any interesting events.

The designed maximum centre of mass p-p collision energy of the LHC is $\sqrt{s} =$ 14 TeV with bunch crossings at 40 MHz or 25 ns and a maximum instantaneous luminosity of 10^{34} cm^{-2}s^{-1}. Because in reality each bunch of protons has a three dimensional size, there are several p-p collisions per bunch crossing, resulting in an interaction rate which is a multiple of 40 MHz.

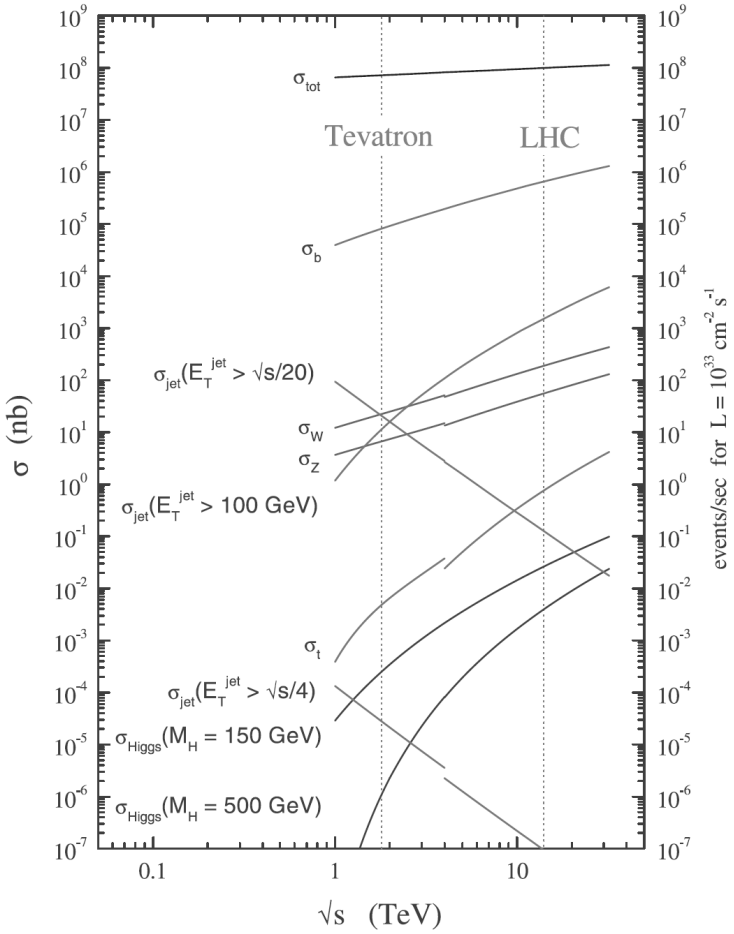


Figure 2.1: Proton - (Anti)Proton Standard Model cross sections for the LHC and Tevatron [13].

The LHC is also able to collide heavy ions. Lead nuclei can be collided at 5.5 TeV per nucleon pair with a design luminosity of 10^{27} cm^{-2}s^{-1}.

The LHC Run 1, as of 2012 had a maximum instantaneous luminosity of 7.7×10^{33} cm^{-2}s^{-1} with a centre of mass energy of $\sqrt{s} = 8$ TeV [14]. In continued operation from 2015 the centre of mass energy will be $\sqrt{s} = 13$ TeV with a peak luminosity in the order of 10^{34} cm^{-2}s^{-1}. Phase-I upgrades in 2019 will increase this to approximately 3×10^{34}cm^{-2}s^{-1} and finally the Phase-II upgrades in 2023 assume a maximum instantaneous luminosity of 7×10^{34} cm^{-2}s^{-1} [15], [16]. The Phase-II upgrade corresponds with a major LHC upgrade to what is known as the High Luminosity LHC (HL-LHC).

The average number of interactions per bunch crossing, $\langle\mu\rangle$, is known as "in time" pile up and is a function of the number of protons in a bunch. Fewer protons per bunch will result in statistically less interactions per bunch crossing, leading to a lower in time pile up. As of 2012, $\langle\mu\rangle = 20.7$, shown in Figure 2.2 [17], [18]. This will increase to $\langle\mu\rangle = 55$ for Phase-I and in Phase-II a maximum of $\langle\mu\rangle = 200$ is assumed.

The second type of pile up is known as Out Of Time (OOT) pile up. This arises when particles from bunch crossings before or after the current bunch crossing are present in the detector. OOT pile up is affected primarily by the bunch spacing.

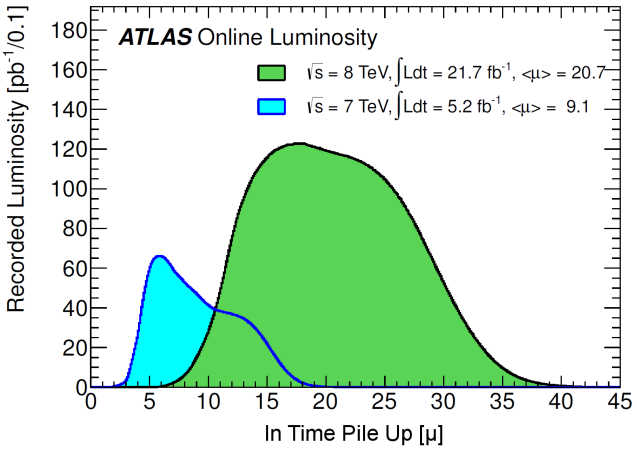


Figure 2.2: ATLAS Online Luminosity plot showing mean interactions per crossing for 2011 and 2012 data [18].

The shorter the time between bunch crossings, the higher the potential for out of time pile up. The LHC has been running with a 50 ns bunch spacing as of the start of Run 2, but this will be reduced to the planned 25 ns towards the end of 2015.

OOT pile up is obviously an issue to data quality because it is very difficult to isolate these spurious particles from the measurements. More detail on pile-up and its effects on energy reconstruction in Hadronic Tile Calorimeter (TileCal) are provided in Chapter 3, Section 3.1.

In Section 2.1, the ATLAS detector is described. The various sub-detectors of ATLAS are also described briefly in order to provide the reader with context to the Hadronic Tile Calorimeter, of which the read out system is the focus of this dissertation.

2.1 The ATLAS Experiment

A Toroidal LHC Apparatus (ATLAS) is one of seven detectors on the LHC, shown in Figure 2.3 [12]. The other detectors are ALICE, CMS, LHCb, LHCf, MoEDAL and TOTEM. ATLAS and CMS are both general purpose High Energy Physics (HEP) detectors which aim to improve human knowledge of the Standard Model and beyond.

The cylindrical ATLAS detector is 44 m long and 25 m high with a total mass of approximately 7000 tons, situated 100 m underground [12]. The detector is composed of several sub-detectors which typically form layers and end caps to surround the interaction point. A diagram of the different layers is shown in Figure 2.4.

Different particles are detected using pattern recognition which in essence performs a process of elimination. Charged particles such as electrons, protons and muons interact with the silicon inner detector and particles such as neutrons, photons and neutrinos do not.

The inner detector is made with silicon pixel and strip detectors which are used for precise charged particle tracking and is described further in Section 2.1.1. A large superconducting solenoid magnet surrounds the inner detector which bends the paths of the charged particles, enabling their momentum to be accurately measured.

The next layer is the electromagnetic calorimeter which uses Liquid Argon (LAr) as a scintillator material. Photons and electrons are absorbed in this calorimeter and

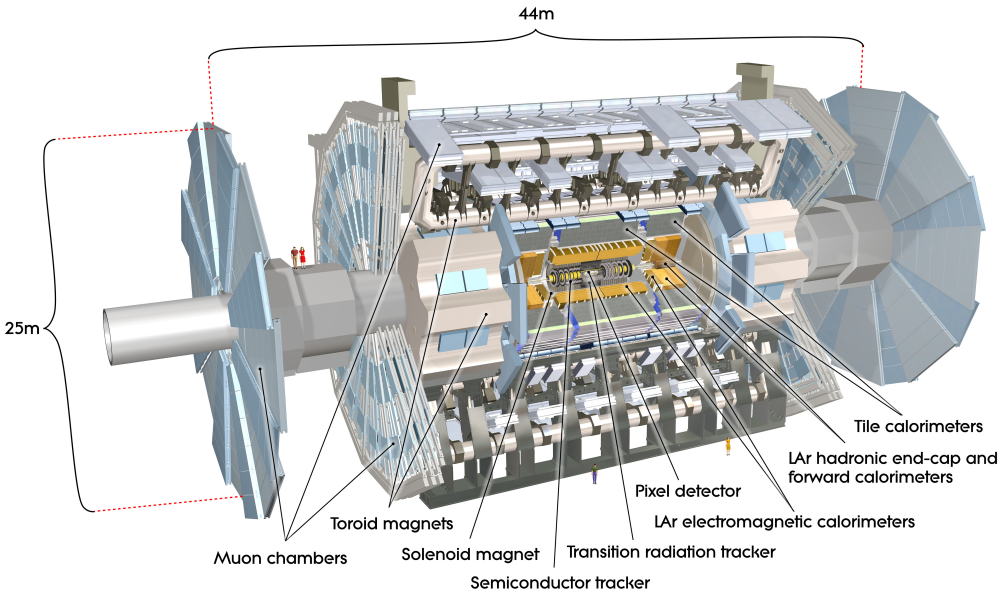


Figure 2.3: A cutaway 3D rendering of the LHC ATLAS detector [19].

in the process they are detected. It is possible to discern a photon from an electron during event reconstruction because the electron will have left a track in the inner detector whereas a photon is less likely to have do so. Some photons may decay to e^+ and e^- pairs inside the inner detector in which case they are detected and realised during energy reconstruction. More detail on the electromagnetic calorimeter is provided in Section 2.1.2.

Muons and protons are also detected in the LAr calorimeter but they pass through it into the TileCal which is described in Section 2.1.3. Hadrons are absorbed by TileCal and form what are known as jets.

It is possible to tell a proton from a neutron, or generally a charged from an uncharged hadron, in TileCal because an uncharged hadron will have left no track in the inner or electromagnetic calorimeters.

The final layer of ATLAS is the muon detection system. A set of large toroidal magnets which generate a strong magnetic field are used to bend the muon tracks to determine their energy, which are detected by the aptly named muon chambers. Muons are also picked up by all of the other sub-detectors in ATLAS, but all other particles except neutrinos and possibly other as-yet unknown exotic particles are absorbed before they reach the muon chambers.

2.1.1 Inner Detector

The inner detector, with a cavity outer radius of 1.15 m and a total length of 7 m, is made up of semiconductor silicon pixel and strip detectors as well as what is called the transition radiation tracker, covering a pseudo-rapidity region (η) of $|\eta| < 2.5$, shown in Figure 2.5 [20]. The inner detector has a very high resolution and can measure transverse momentum (p_T), $p_T > 0.5$ GeV.

A 2 Tesla magnetic field produced by a superconducting solenoid that surrounds the inner detector bends the paths of charged particles which allows their momentum

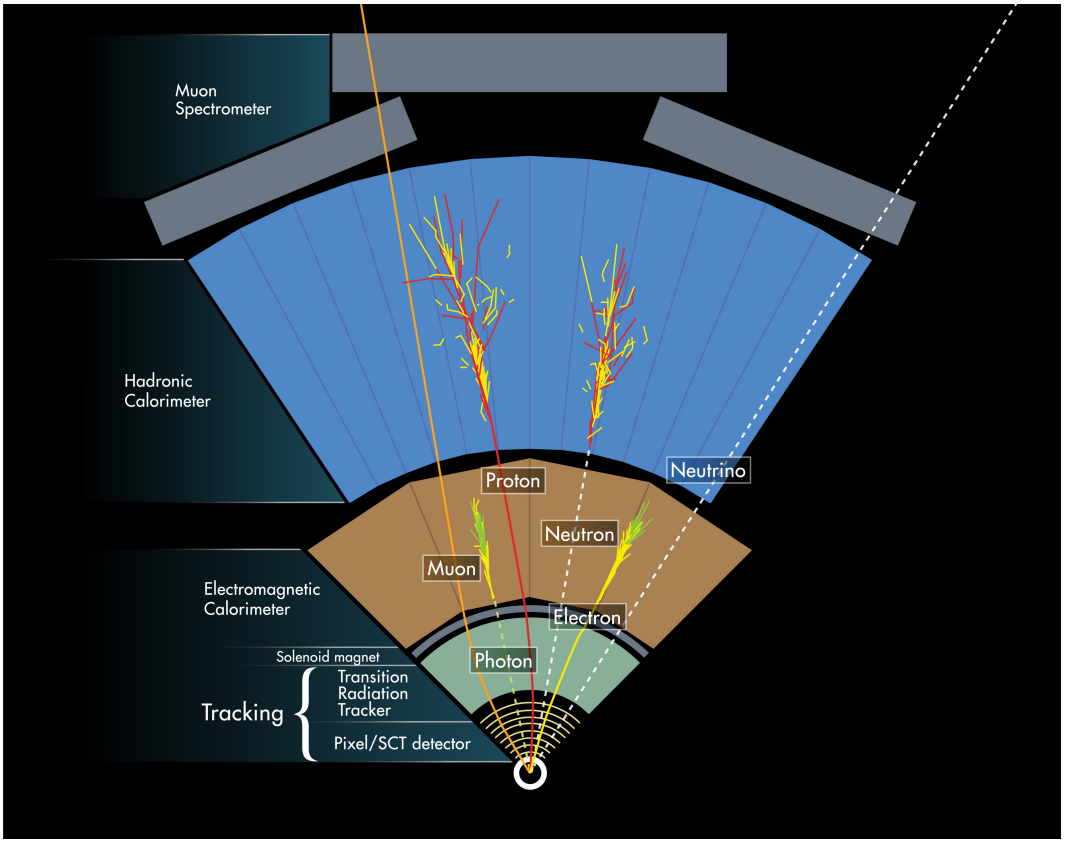


Figure 2.4: Sub-detector layers of ATLAS showing where different particles are detected. Dotted lines indicate no detection [19].

to be accurately calculated.

Ionizing radiation creates electron-hole pairs in the semiconductor material which allows a current to flow in the presence of an electric field. This electrical response is then measured and digitised.

The pixel detectors allow a very precise, two dimensional measurement of charged particles close to the beam interaction point. Three concentric cylindrical layers (called barrels) of pixel detectors are present around the beam axis and contribute 97 million channels of data. These layers are situated at radii 40, 110 and 140 mm from the beam axis. A further four end cap pixel disks are located perpendicular to the beam axis. These end caps provide an additional 43 million channels of high resolution tracking data.

Due to the large amount of silicon and its high cost, the number of pixel detector layers must be limited. The pixel detectors are surrounded by the silicon microstrip detectors, known as the semiconductor tracker, which allow fine granularity in one direction with a lower cost than the pixel detectors.

Four barrels of silicon microstrip detectors are located at radii 300, 373, 447 and 520 mm and there are also nine end-cap wheels on each side covering a total area of 61 m^2. The semiconductor tracker contributes 6.2 million channels of data.

Finally, the transition radiation tracker uses straw detectors which are 4 mm in diameter. Each straw is filled with xenon gas which allows charged particles to be detected. The straws are packed axially in a barrel around the silicon detectors and radially for the end-caps. In total there are 420 thousand straws contributing as

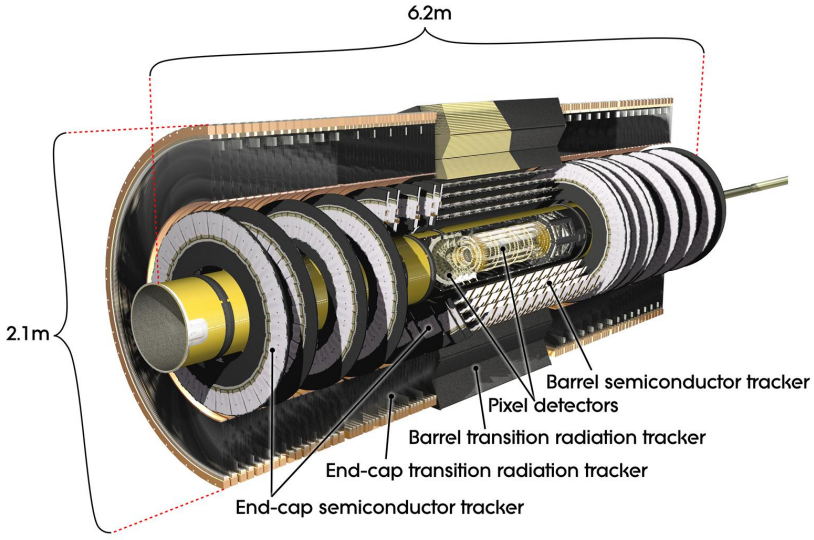


Figure 2.5: A cutaway 3D rendering of the ATLAS inner detector [19].

many channels of data.

The combination of these three sub-detectors allows precise measurement of the dense tracks, vertices and also displaced vertices for example from processes such as b-quark to b-meson production, produced in the ATLAS detector. Information obtained from the other sub-detectors allows the particles to be identified and tagged for later physics research.

2.1.2 Liquid Argon Calorimeters

The Liquid Argon (LAr) calorimeters are composed of several different modules for both Electromagnetic (EM) and hadronic calorimetry [21]. A 3D rendering of the ATLAS calorimeters is shown in Figure 2.6 showing the liquid argon calorimeters surrounded by the TileCal.

The EM calorimeter is made up of three main components and is symmetrical about $z = 0$. The EM calorimeter is uses liquid argon as a scintillating material and so it is located inside the ATLAS central cryostat which also contains the superconducting solenoid around the inner detector. The total thickness of the solenoid on the inner radius of the cryostat is 44 mm. The cryostat is 6.8 m long and has an outer radius of 2.25 m. The cryostat volume is made with aluminium alloy and uses vacuum insulation.

Two cylindrical EM barrel detectors, divided into 16 modules each, cover the pseudo-rapidity region $|\eta| < 1.4$. A 4 mm gap separates these two barrels at the midpoint, $z = 0$. The LAr EM barrel modules are composed of three regions with decreasing measurement granularity, depth wise, with a total radiation depth of approximately 24 X_0 where X_0 is the electromagnetic radiation length.

A diagram of a barrel module is shown in Figure 2.7. There are 1024 "accordion" shaped sheets of electrically grounded lead absorber covered with stainless steel. In between these sheets are 1024 copper electrode sheets. The absorber and electrode sheets are separated by a 2.1 mm gap filled with liquid argon. The gap is maintained by a honeycomb of non-conductive material.

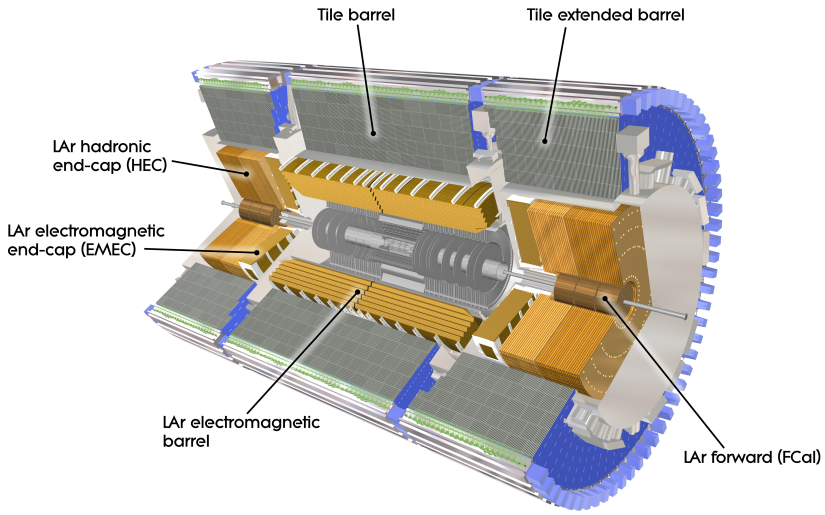


Figure 2.6: A cutaway 3D rendering of the ATLAS calorimeters [19].

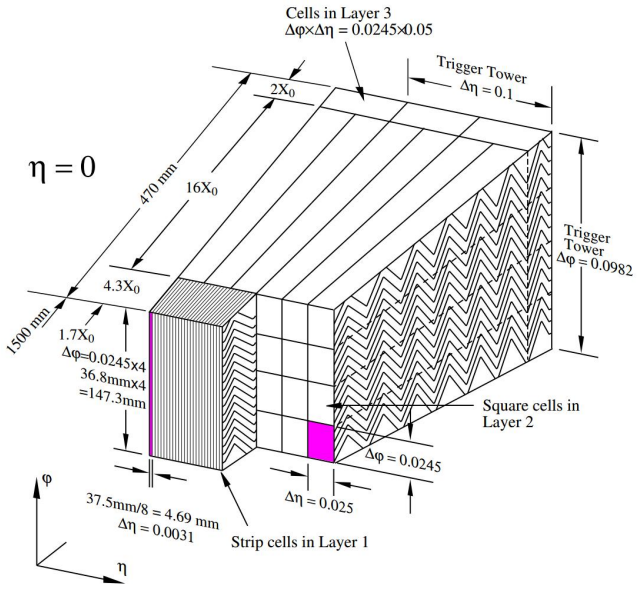


Figure 2.7: A diagram of an EM calorimeter barrel module showing the accordion shaped layers of lead absorber and copper electrode.

The electrode sheets are actually three layers of copper separated by kapton insulator. The two outer layers are charged to a 2000 V, High Voltage (HV) potential with respect to the grounded lead absorber sheets. This sets up an electric field which accelerates ions in the liquid argon which are created by electrons and photons passing through it.

The third copper electrode, which is located between the two HV electrodes, capacitively couples to the outer electrodes. Any pulse on the HV plates induced by ions in the liquid argon will be sensed by this inner electrode. Read out electronics digitise the pulses and use a similar methodology to the TileCal for energy reconstruction.

There are also three different end-cap "disk shaped" detectors perpendicular to the beam axis which are housed in the so-called end-cap cryostats. Their outer radius is 2.25 m and are 3.17 m long, with a total radiation depth of approximately 26 X_0.

The first is called the LAr Electromagnetic End-Cap (EMEC), which uses the same detection technique as the barrel detectors and covers the pseudorapidity region $1.375 < |\eta| < 3.2$ with a total of 1024 lead-stainless steel absorbers and 1024 copper detector "accordion" sheets.

Behind the EMEC is the LAr hadronic End Cap (HEC) which is a liquid argon based hadronic calorimeter which covers the region $1.5 < |\eta| < 3.2$. The HEC uses 25 and 50 mm thick parallel copper plates as absorbers with electrodes in between. The nuclear interaction length of the HEC is approximately 12 λ.

Finally, the Forward Calorimeter (FCal) is situated concentrically around the beam-pipe inside the EMEC and HEC. The FCal therefore covers the pseudorapidity region $3.2 < |\eta| < 4.9$. The FCal provides electromagnetic as well as hadronic shower measurements by using copper and tungsten absorbers with a total of 5644 channels on each side of the ATLAS detector.

In total, the EM calorimeter contributes approximately 170 thousand channels to the detector.

2.1.3 Hadronic Tile Calorimeter

A 3D rendering of the Hadronic Tile Calorimeter (TileCal) is shown in Figure 2.6 [12], [22]. TileCal is split into three main sections: a central barrel and two extended barrels with an overall pseudorapidity coverage of $|\eta| < 1.7$.

The central barrel is 5.8 m long and the extended barrels are each 2.9 m long. The inner radius of TileCal is 2.28 m from the beam pipe and the outer radius is 4.25 m which results in a nuclear interaction length of approximately 7.4 λ and a physical depth of 1.97 m.

The barrels are composed of 64 azimuthal wedges, also known as modules, with a coverage $\Delta\phi \approx 0.1$ rad each. A drawing of a single module is shown in Figure 2.8. Modules are built with layers of steel absorber and a plastic scintillator material. There are 440 thousand scintillator tiles in the TileCal.

When hadrons pass through the layers of steel absorber they interact with other atoms and cause a spray of particles which are detected by the layers of scintillator. The volume of this spray is used to determine the energy and direction of the original particles.

Wavelength shifting optical fibres link groups (called cells) of scintillator tiles to Photo-Multiplier Tubes (PMTs). The optical fibres collect light from both sides of each tile and are fed to two corresponding PMTs per cell. This is for redundancy as well as spacial uniformity. In total there are approximately ten thousand read out channels on TileCal.

At the outer radius of each module is a hollow steel beam which supports the module. The front-end read out electronics as well as PMTs are housed inside this long cavity. Access is possible by sliding the electronic "drawers" in and out from one side.

It is interesting to note that the TileCal is the main magnetic field return path for the inner detector solenoid. Approximately 70% of the magnetic flux returns through the support beam in which the front end electronics are installed. For this reason, the electronics and, more importantly, the PMTs must be magnetically shielded to prevent unexpected interference.

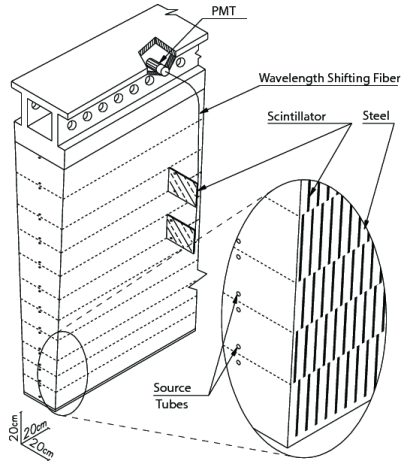


Figure 2.8: Mechanical assembly of a TileCal module [12].

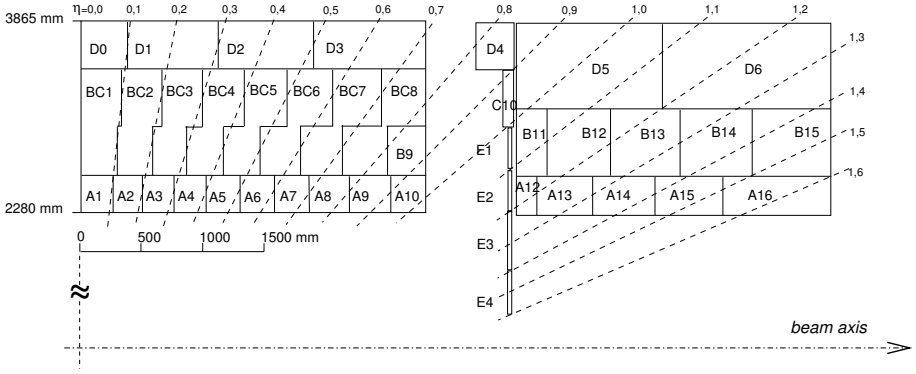


Figure 2.9: Segmentation in depth and η of cells on half of the TileCal [12].

Figure 2.9 shows the layout of the cells on half of the detector. The cells are designed to have pseudo-rapidity and azimuthal dimensions $\Delta\eta \times \Delta\phi = 0.1 \times 0.1$ in the layer closest to the beam and $\Delta\eta \times \Delta\phi = 0.2 \times 0.1$ in the outermost layer. The depth of the cells are designed for three different interaction lengths: 1.5λ, 4.1λ and 1.8λ.

2.1.4 Muon Spectrometer

The muon spectrometer is the outermost sub-detector of ATLAS [23]. It also plays an important part of the ATLAS Trigger and Data AcQuisition system (TDAQ) because of muon triggering. The muon spectrometer system is made up of four different detector chamber types and in total covers a pseudorapidity range of $|\eta| < 2.7$.

Ten large superconducting, air-core toroidal electromagnets provide a specifically shaped strong magnetic field which bends the muon tracks, enabling their momentum to be measured. The barrel toroid, made up of eight individual toroidal electromagnets, is a prominent feature of the ATLAS detector and is 25 m long and has an outer diameter of 20.1 m. Each end of the detector has an end cap toroid inserted into the barrel. They are 5 m long and have an outer diameter of 10.7 m.

The four different chamber types depend on the requirements for spacial positioning, timing resolution and radiation resistance contributing approximately 1.2 million

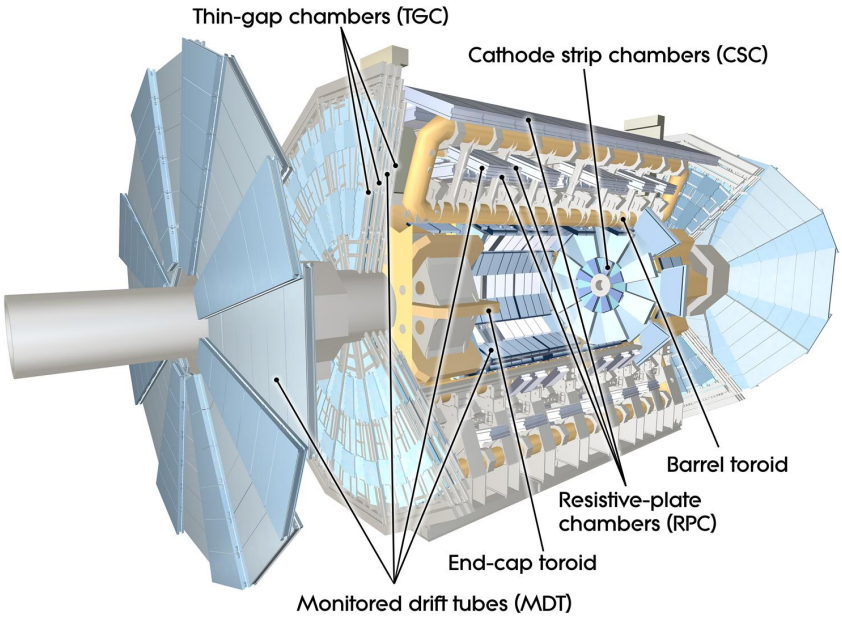


Figure 2.10: ATLAS Muon Spectrometer 3D rendering [19].

channels to the detector.

Monitored Drift Tubes (MDTs), which provide high resolution muon measurements, are 30 mm diameter aluminium tubes filled with an Ar - CH_4 - N_2 mixture pressurised to 300 kPa. A wire cathode of Tungsten - Rhenium passes down the center of the tubes and is at a potential of 3270 V. The MDTs are typically located in concentric cylinders around the beam pipe at radii 5, 7.5 and 10 m and also as end-caps located 7, 10, 14 and 21 - 23 m from the interaction point.

Cathode Strip Chambers (CSCs) use the same principle as the MDTs except they achieve high spacial resolution of 60 μm. Measurements are made by using multiple cathode strips and anode wires in a single chamber and interpolating the position of muons based on the charge proportion over several strips. The wires are charged to 2600 V. The gas mixture in the CSC is slightly different to that in the MDTs: Ar - CO_2 - CF_4.

The Resistive Plate Chambers (RPCs) offer high readout rate with lower spacial resolution to the ATLAS trigger system compared to the MDTs. As such, the RPCs are used by the TDAQ whereas the MDTs are used for offline energy reconstruction. Functionality is achieved by using two resistive parallel plates, 2 mm apart filled with a mixture of $C_2H_2F_4$ and iso-C_4H_{10}, with an operating potential of 8.9 kV between the plates. Readout strips of aluminium foil placed on both sides of the detector capacitively couple to the parallel plates and detect the avalanche effect between the plates when muons pass through.

The final component of the muon spectrometer are the Thin Gap Chambers (TGCs) and are located in the ATLAS end caps and provide data for the trigger system as well as an azimuthal coordinate to complement the MDTs. These two uses require good timing resolution as well as fine granularity for spacial measurements. The TGCs are constructed as parallel graphite cathodes which form a chamber filled with a mixture of CO_2 and n-C_5H_{12}. Many parallel wire anodes are situated between the two cathode plates at a potential difference of 3.2 kV. The distance between

the wires and the graphite plates is 1.4 mm and the distance between the wires is 1.8 mm. Read out electrode strips are placed outside the graphite layers, orthogonal to the wires, to provide a capacitively coupled readout.

2.2 Summary

The ATLAS experiment is highly sophisticated with several different sub-detectors resulting in different measurements. In total, hundreds of millions of channels of data must be amalgamated and processed in order to derive interesting physics events which may eventually lead to discoveries and enhance mankind's understanding of the universe.

In total, Petabytes of raw data is generated by ATLAS. It is the responsibility of each sub detector to only send meaningful and relevant data to higher levels in the ATLAS TDAQ to avoid overloading the system. Each sub detector is different and a discussion of the read out systems of each is out of scope.

That said, the TileCal read out system is described in Chapter 3 as it is relevant to the development of a new, general purpose Super Read Out Driver (sROD) co-processing unit which may be used to simplify the development and implementation of new energy reconstruction algorithms for TileCal that are more immune to the adverse effects of pileup in light of future upgrades and running conditions.

This processing unit may have use in other sub detector read out systems and even the ATLAS TDAQ, but these considerations are left for future research.

3

The Hadronic Tile Calorimeter Read Out System

As described in Chapter 2, the ATLAS experiment is composed of several sub-detectors, each of which has separate data processing requirements. Slightly over one hundred million channels produce a total raw data output of Petabytes per second. Current computing hardware and more importantly storage is not able to deal with these volumes of data.

A simple plot, shown in Figure 3.1, of the increase in CPU processing power in Million Instructions per Second (MIPS) and hard drive read-write speed in MB/s over many years. The trends clearly demonstrate the fact that hard drive I/O rates are insufficient, and will not become sufficient in the foreseeable future, to store the entirety of raw data from modern scientific experiments such as the LHC [24].

On the other hand, interconnects such as Ethernet, also shown in Figure 3.1, are increasing their maximum bandwidth at a rate similar to the increase in CPU performance. The cost of cutting edge variants of Ethernet and indeed other fast interconnects may be prohibitive but in theory it should remain possible to process data online, in real time, to minimise storage requirements. The power consumption of the computing systems required to deal with massive volumes of data is a very important factor.

The massive amount of raw data produced by ATLAS is reduced by a process called triggering performed by the ATLAS TDAQ. In the TileCal, which is the focal point of this chapter, there are currently three levels of triggering, shown in Figure 3.2.

The front end read-out system is based on analog circuitry as well as Field Programmable Gate Arrays (FPGAs) and Digital Signal Processors (DSPs) to form a part of the Level-1 trigger which serves to reduce the event rate from 40 MHz to about 100 kHz. The TileCal read out system also performs an initial energy reconstruction algorithm called Optimal Filtering (OF), which converts an electrical PMT pulse into a more usable equivalent hadronic or electromagnetic energy amplitude and time, described in Section 3.1.

The hardware components of the TileCal read out system are described in detail in Section 3.2 and 3.3. The TileCal read out system will be significantly modified

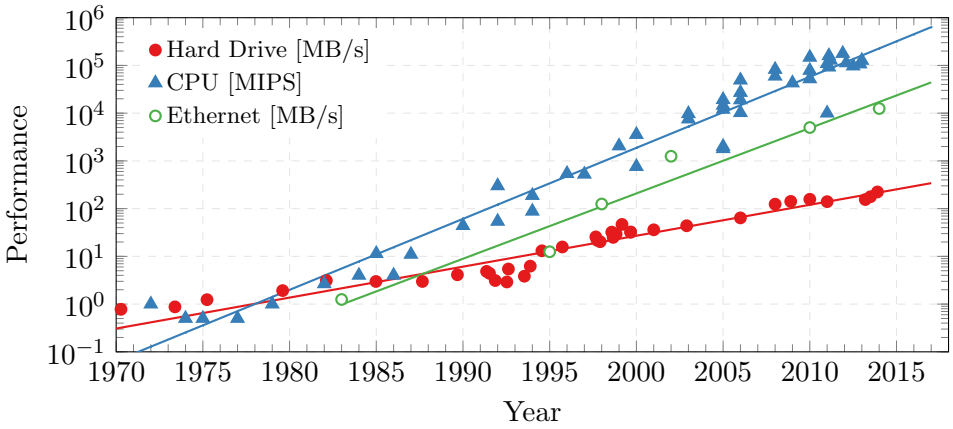


Figure 3.1: Hard drive and Ethernet 802.3 data throughput and CPU performance on a log scale with time [24], [25].

by the ATLAS Phase-II upgrades. Some of the pertinent changes are described in Section 3.4.

Once event data is accepted by the Level-1 trigger, it passes to the Level-2 and Level-3 trigger, which is known as the Event Filter. These systems are implemented with large computing clusters. Data from the level one trigger system is fed to the so-called Read Out Boards (ROBs) which pass the data to the Event Builder and Level-2 trigger at a rate of about 100 kHz or 120 GB/s [26].

The Level-2 filter only works on a portion of the data to determine whether it contains interesting physics data. If the portion is indeed interesting then all of the associated data is let through to the Event Builder.

The Level-2 compute cluster is built from approximately 500 machines, each of which has two 2.5 GHz quad-core Intel Harpertown CPUs, two 1 Gb/s Ethernet interfaces and 16 GB RAM. The Event Builder consists of 100 rack-mounted servers, each of which has two 2.6 GHz AMD Opteron 252 CPUs with 1 Gb/s Ethernet. [27]

Finally, the Event Filter is a much larger cluster of about 1900 machines which is the last level of triggering. The same specification machines as for the Level-2 filter are used. Each ATLAS event consists of about 1.5 MB data, of which a portion is made up from TileCal data. The total data output from the Event Filter is several hundred megabytes per second which is saved in long term storage [27].

3.1 Online Energy Reconstruction Algorithm

Online energy reconstruction translates the seven Analog to Digital Converter (ADC) samples provided by the front end electronics to three values: the pedestal, peak amplitude (A) and phase shift (τ) of the peak from the third sample. These parameters provide enough information to calculate the energy of the particle that caused the PMT pulse as well as the exact time it interacted with the scintillator material in the TileCal.

An OF algorithm is currently in use and is implemented on DSPs in the TileCal Read Out Drivers (RODs) [28], [29]. In summary, OF is a weighted sum of samples from the raw signal. The weights are pre-calculated on a per cell (within TileCal)

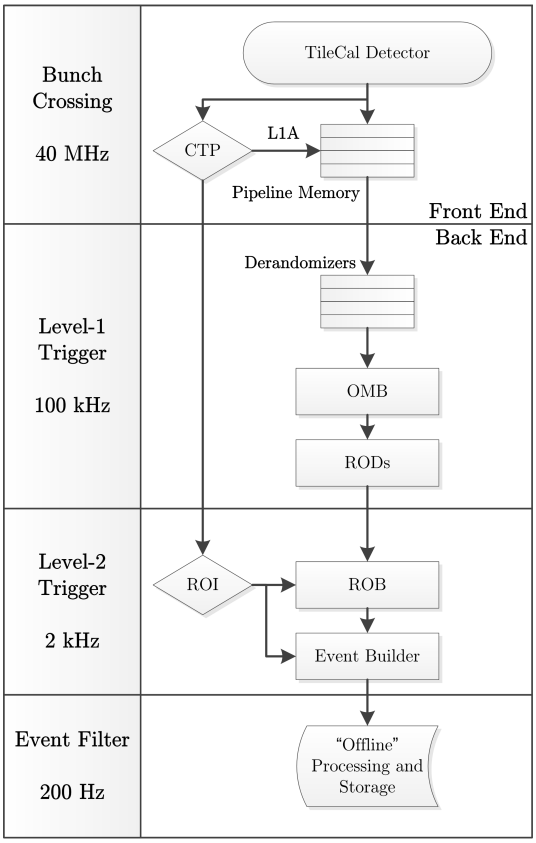


Figure 3.2: Block diagram of the ATLAS Trigger and Data AcQuisition system (TDAQ).

basis in order to determine a specific parameter of the signal, which, in the case of TileCal is A and τ.

OF can also be used to determine the noise floor (also known as the pedestal) of the signal, which must be subtracted from the samples before processing. In the existing implementation for the sake of reduced processing time, the value of the first sample is used as the pedestal [29].

A more in depth description and derivation of the OF algorithm is in Section 3.1.1.

Besides the pedestal, amplitude and phase of each pulse, a Quality Factor (QF) calculation is also important. The simplicity of the OF algorithm means that it can always calculate values for the amplitude and phase, even though they may be inaccurate if the shape of the raw data is significantly different from the expected pulse.

In the existing system the QF is implemented as a chi-square test but in higher pile up operation this calculation becomes an indicator of pile up and not of reconstruction quality and so new QF calculation methods are under investigation [30].

If the calculated quality factor is worse than some predefined value then the raw data samples are sent for storage along with the reconstructed values. This allows a more sophisticated processing algorithm to be applied offline to ensure accurate physics measurements [31].

Figure 3.3 shows an ideal PMT pulse with a single OOT signal 50 ns later. It is clear that the samples that are provided to the energy reconstruction algorithm are

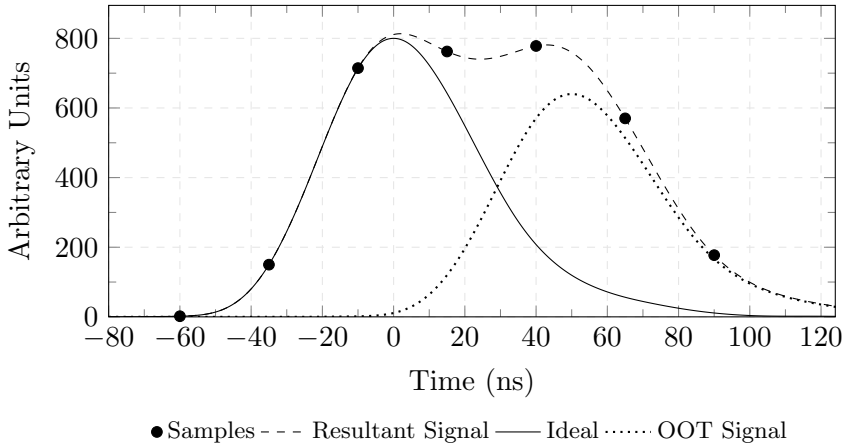


Figure 3.3: PMT pulse showing seven samples with one out of time pile up signal at +50 ns.

significantly different to what is expected in the ideal case. With in time pile up the superposition effect is similar but more subtle.

This pile up will clearly adversely affect the reconstructed signal quality unless the reconstruction algorithm is sophisticated enough to compensate for the effects.

From Chapter 2, the LHC Run-1 pile up was approximately $\langle\mu\rangle = 20.7$ interactions per bunch crossing. For the Phase-II upgrade this is an order of magnitude higher at $\langle\mu\rangle = 200$ [15].

Based on the LHC Run-1 operation and the theoretical derivation in Section 3.1.1, the OF algorithm is somewhat tolerant to pile up but unfortunately OF is not sophisticated enough to deal with higher levels of pile up in future and so several alternatives have been proposed and research is ongoing by other research groups.

A Matched Filter based technique appears to be slightly more tolerant to OOT pile up that OF but is not significantly better [5]. An algorithm called Signal Response Deconvolution shows promising results but is too complex to be implemented on the current front end DSPs [4].

3.1.1 The Optimal Filtering Algorithm

There are two variations of the OF algorithm. The variation described in this section assumes the signal pedestal has already been subtracted. The second version requires at least two iterations and is able to compute the pedestal as well as the amplitude and phase.

The OF algorithm described in this section is used in the TileCal and the derivation is repeated here for convenience and clarity [32]. It was originally derived from that which is used in the Liquid Argon Calorimeters [33].

Let $g(t)$ be defined as the ideal, normalised amplitude signal. The actual signal, $S(t)$ can therefore be written as:

$$S(t) = Ag(t) \tag{3.1}$$

where A is the true amplitude of the signal. A discrete version of this signal, with samples taken at regular intervals t_i is given by:

$$S_i = Ag(t_i) = Ag_i \tag{3.2}$$

We must introduce a phase shift, τ, which is introduced by the digitising and signal conditioning electronics and noise term, n_i, as well as the fact that particles interact at varying times with scintillator tiles in relation to the expected time:

$$S_i = Ag(t_i - \tau) + n_i \tag{3.3}$$

We want the output of the OF algorithm to be A and τ. A Taylor expansion is used to linearise the relationship between S and the two parameters. A first order expansion is sufficient:

$$S_i \backsimeq Ag(t_i) - A\tau g'(t_i) + n_i = Ag_i - A\tau g'_i + n_i \tag{3.4}$$

Clearly, if τ is zero then this expansion is exact and the quality of the reconstruction decreases with increasing τ. It is for this reason that different OF weights are used for each cell of TileCal to attempt to keep τ close to zero. Another useful side-effect of this is that the OF algorithm decreases the effect of out of time pileup which usually has a large phase shift with respect to the desired signal.

Let us define two new variables, u and v:

$$u = \sum_{i=1}^{n} a_i S_i \tag{3.5}$$

$$v = \sum_{i=1}^{n} b_i S_i \tag{3.6}$$

where n is the number of samples in the signal S and a and b are free parameters called the OF weights. If we require the expected values of u and v be equal to the two parameters we are searching for, A and τ (in the more convenient form of $A\tau$), we can eventually solve for the weights.

$$A = \langle u \rangle = \langle \sum_{i=1}^{n} a_i S_i \rangle = \sum_{i=1}^{n} a_i \langle S_i \rangle \tag{3.7}$$

$$A\tau = \langle v \rangle = \langle \sum_{i=1}^{n} b_i S_i \rangle = \sum_{i=1}^{n} b_i \langle S_i \rangle \tag{3.8}$$

By substituting in Equation 3.4 and assuming that all parameters are constant except for the noise, n_i, we get:

$$A = \langle u \rangle = \sum_{i=1}^{n} (Aa_i g_i - A\tau a_i g'_i + a_i \langle n_i \rangle) \tag{3.9}$$

$$A\tau = \langle v \rangle = \sum_{i=1}^{n} (Ab_i g_i - A\tau b_i g'_i + b_i \langle n_i \rangle) \tag{3.10}$$

The noise term, n_i, is assumed to be Gaussian which is a good approximation for the noise introduced by the electronics which is primarily thermal. The noise introduced by pile up is also assumed to be Gaussian but unfortunately this assumption fails under high pile up conditions [5]. We can therefore take the expected value of n_i to be zero. With this assumption, the equations can be written as:

$$A = \langle u \rangle = A \sum_{i=1}^{n} a_i g_i - A\tau \sum_{i=1}^{n} a_i g'_i \tag{3.11}$$

$$A\tau = \langle v \rangle = A\sum_{i=1}^{n} b_i g_i - A\tau \sum_{i=1}^{n} b_i g'_i \tag{3.12}$$

We therefore need the following constraints to solve for the weights in Equations 3.11 and 3.12:

$$\sum_{i=1}^{n} a_i g_i = 1, \quad \sum_{i=1}^{n} a_i g'_i = 0 \tag{3.13}$$

$$\sum_{i=1}^{n} b_i g_i = 0, \quad \sum_{i=1}^{n} b_i g'_i = -1 \tag{3.14}$$

The variances of the weight parameters are given by:

$$\mathrm{Var}(u) = \sum_{i,j=1}^{n} a_i a_j \langle n_i n_j \rangle \tag{3.15}$$

$$\mathrm{Var}(v) = \sum_{i,j=1}^{n} b_i b_j \langle n_i n_j \rangle \tag{3.16}$$

Since the goal is to find expressions and thus values for a and b, we use Lagrange multipliers to minimise the variance and also satisfy the constraints in Equations 3.13 and 3.14:

$$I_u = \sum_{i,j=1}^{n} a_i a_j \langle n_i n_j \rangle - \lambda(\sum_{i=1}^{n} a_i g_i - 1) - \kappa(\sum_{i=1}^{n} a_i g'_i) \tag{3.17}$$

$$I_v = \sum_{i,j=1}^{n} b_i b_j \langle n_i n_j \rangle - \mu(\sum_{i=1}^{n} b_i g_i) - \rho(\sum_{i=1}^{n} b_i g'_i + 1) \tag{3.18}$$

where λ, κ, μ and ρ are the Lagrange multipliers. If we take the partial derivative with respect to a_i and b_i and make them equal to zero we get:

$$\frac{\delta I_u}{\delta a_i} = 2\sum_{j=1}^{n} a_j \langle n_i n_j \rangle - \lambda g_i - \kappa g'_i = 0 \tag{3.19}$$

$$\frac{\delta I_v}{\delta b_i} = 2\sum_{j=1}^{n} b_j \langle n_i n_j \rangle - \mu g_i - \rho g'_i = 0 \tag{3.20}$$

The expected value of $\langle n_i n_j \rangle = R_{ij}$ is called the noise autocorrelation matrix which is defined as:

$$R_{ij} = \frac{\sum (n_i - \langle n_i \rangle)(n_j - \langle n_j \rangle)}{\sqrt{\sum (n_i - \langle n_i \rangle)^2 \sum (n_j - \langle n_j \rangle)^2}} \tag{3.21}$$

By using Equations 3.21, 3.13, 3.14, 3.19 and 3.20 it is possible to solve for the weights vectors a and b to reconstruct A and $A\tau$ as in Equation 3.7 and 3.8.

In summary, the online optimal filtering algorithm requires Equation 3.22 and 3.23 to be calculated for each sample vector S. The weights vectors a and b are computed offline and loaded into the ROD DSPs during initialisation.

A third calculation that is performed is the QF and is shown in Equation 3.24. The QF is a χ^2 calculation to indicate the quality of the reconstructed signal but is

has been shown to be highly sensitive to pile-up and so different methods are under investigation [30].

$$A = \sum_{i=1}^{n} a_i S_i \tag{3.22}$$

$$\tau = \frac{1}{A} \sum_{i=1}^{n} b_i S_i \tag{3.23}$$

$$QF = \sqrt{\sum_{i=1}^{n} (S_i - Ag_i + A\tau g'_i - ped)^2} \tag{3.24}$$

3.2 Existing Front End Electronics

The TileCal front end electronics are located in 3 m long "super drawers" which are located inside the box section of the outer steel support beam [34]. These drawers contain the PMTs, high voltage distribution system to bias the PMTs as well as the front end analog and digital electronics.

A block diagram of the existing ATLAS TileCal read out is shown in Figure 3.4 which is explained in this section [3].

The PMTs convert the light pulses received from the optical fibres into an electrical current proportional to the energy deposited into the scintillator. The resulting pulses are shaped by a 7-pole passive bessel filter and amplified by so-called 3-in-1 cards [35].

The output of the passive pulse shaping circuit is a 2 V per 800 pC pulse. The maximum output of 2 V equates to 1.5 TeV of hadronic energy or 1.3 TeV of electromagnetic energy. The "stretching" effect of the filter converts the initially fast PMT pulse to a slower pulse with a 52 ns Full Width at Half Maximum (FWHM) and an overall length of about 150 ns.

A low- (×1) and high-gain (×64) amplification then takes place and both resulting signals are sent to separate 10 bit 40 MSa/s (Mega Samples per Second) ADCs on the digitizer boards [36]. The result is 16 bits of dynamic range for the measurement of the PMTs with samples every 25 ns.

Figure 3.5 shows the "ideal" filtered PMT pulse shape with no noise or pile-up, seven samples and an arbitrary amplitude. The actual location of the samples with respect to the signal depends on the position in the detector of the cell being measured.

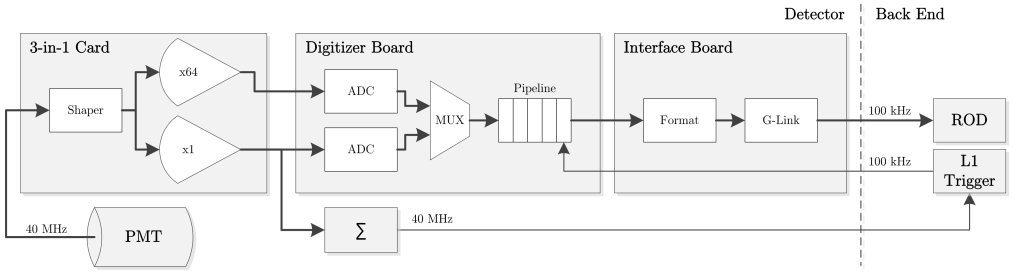


Figure 3.4: Block diagram of the existing TileCal Read Out Architecture.

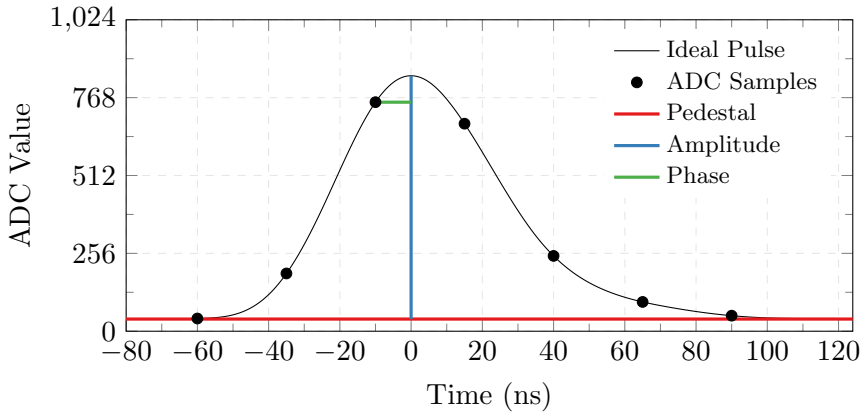


Figure 3.5: Ideal PMT pulse showing seven samples with the pedestal, peak amplitude and phase indicated.

Once the signals have been digitised they are placed in a pipeline memory buffer to create a 2.5 μs delay for the Level-1 Trigger. A Level-1 Accept signal causes the associated digitised pulse along with a specified number of pulses before and after (called a time frame) to be formatted and transmitted off the detector using the G-Link protocol over two redundant optical fibres to the ROD [37].

3.3 Read Out Driver (ROD)

The Read Out Driver (ROD) is the interface between the front end electronics and the back end [34], [38]. The ROD serves several purposes, some of which are listed below:

- Perform an initial, online energy reconstruction of the PMT pulses, detailed in Section 3.1.
- Apply calibration corrections.
- Combine data from several front end sources into a single high bandwidth link to the ROBs.
- Format the data and add metadata such as bunch crossing and trigger information.

The RODs use DSPs mounted on so-called processing unit cards of which four can be mounted to each ROD. In the current system, two processing units are mounted per ROD. There are 32 RODs installed in the current TileCal back end which are able to satisfy the data throughput requirements of the 10000 read out channels.

The ROD must perform the required tasks for all of the channels in less than 10 μs to satisfy the requirements of the Level-1 trigger. At a 100 kHz Level-1 trigger rate, an input data rate of 163 Gb/s (32 RODs $\times$ 8 Input Links/ROD $\times$ 16 bit $\times$ 40 MHz) is sustained with a resulting output data rate of 82 Gb/s [38].

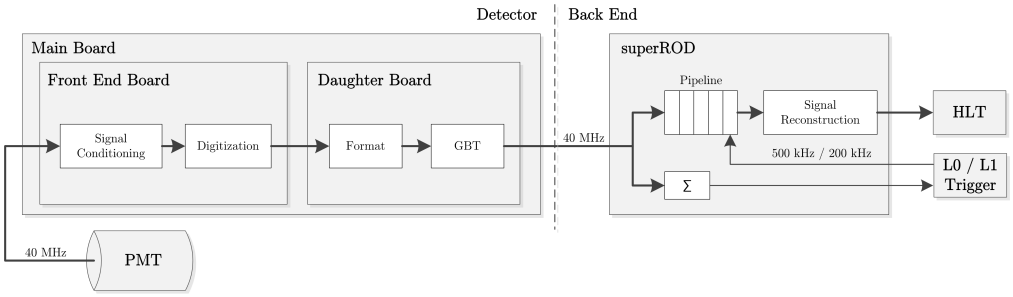


Figure 3.6: Block diagram of the planned Phase II TileCal Read Out Architecture.

3.4 Upgraded Phase-II Front-End

In the HL-LHC, the Phase-II upgrade of the TileCal front end read out system is focussed on improvements to its reliability and ease of maintenance and most importantly increasing the quality and efficiency of the online energy reconstruction. Higher luminosity collisions require more interactions per bunch crossing which results in higher levels of in time pile up.

In the Phase-II upgrade the existing super drawers will be replaced by separable mini-drawers for ease of maintenance, described in Section 3.4.1. The front end electronics will be replaced by high end FPGAs which forward the entirety of the digitised data to the back end sROD, described in Section 3.4.3, which replaces the existing ROD. The sROD will be housed in an AdvancedTCA or Advanced Telecommunications Computing Architecture (ATCA) chassis which will replace the existing VME crates. The VME crates will be replaced ATLAS-wide, with more details in Section 3.4.2.

The University of the Witwatersrand, Johannesburg is participating in the development of the sROD and has a working ATCA system which will be used to install and test the sROD prototypes in future.

Contrary to the existing system where the buffer pipelines are located on the front end electronics and only some data is sent to the back end on a Level-1 Accept signal, both the pipeline buffering and online energy reconstruction will take place on the sROD [3]. This avoids storing data on the front end where radiation may cause single event upsets and cause corruption. The total raw data that will be produced by the upgraded TileCal front end electronics and sent to the back-end will thus increase to approximately 41 Tb/s from the current 163 Gb/s.

A block diagram of the Phase-II system is shown in Figure 3.6 compared to that of the existing system in Figure 3.4 [3].

This new system is more flexible because changes to the operation of the read out system are possible with a firmware update as opposed to a hardware replacement. It will perform energy reconstruction and then reduce the data rate from the 40 MHz bunch crossing rate to 500 kHz with triggering [15].

It is envisaged that the sROD will become the new Level-0 trigger which will fulfil similar functionality to the existing Level-1 trigger. The sROD will also perform tower and clustering algorithms [3]. The upgraded Level-1 trigger will then reduce the data to 200 kHz as opposed to the current 100 kHz.

In summary, the upgraded TileCal read-out architecture will provide the following features [3]:

- All raw data from the detector front-end will be sent to the back-end for every bunch crossing.
- Fully digital Level-1 (and possibly a new Level-0) trigger.
- Higher radiation tolerance because sophisticated electronics are moved into the back-end (sROD) and data is not buffered on the front-end.
- Redundancy of data links and power supplies.

3.4.1 Mini Drawer

The mini drawer is the Phase-II upgrade replacement for the current super drawer with a photo in Figure 3.7 [39]. It is "mini" because it is half of the size of the existing drawers which fit into the support beams around the TileCal and therefore four mini-drawers are required to replace a single super drawer. Additionally, the chance of a single point of failure is reduced. The mini-drawer will also be mounted on chain-like tracks which, combined with the smaller size, allow dramatically improved accessibility for maintenance and replacements.

Each mini drawer is water cooled and houses 12 PMTs, a main board, daughter board and an HV board on the underside. Both the main and daughter board are fully redundant with a physical mirroring of board components lengthwise. On the demonstrator system an analog summing board is also present. This is for compatibility with the existing read out system.

The main board has 24 digitiser (ADC) channels which connect to 12 associated PMTs, which each have a high and low gain output from their integrated 3-in-1 board [40]. The existing system uses 40 MSa/s, 10 bit ADCs but the upgraded demonstrator main board uses 40 MSa/s, 12 bit ADCs. There are four control FPGAs on the main board which configure and adjust the ADCs for optimal operation. Finally, on one end of the main board is the power regulation circuitry. Each redundant side is able to power the other via a simple diode based logical or circuit.

The daughter board is shown in Figure 3.8 and is being developed by Stockholm University [42]. It makes use of two Xilinx Kintex-7 FPGAs connected to the main board via a 400 pin FMC connector and to two QSFP+ connectors for connection to the sROD via fibre optic cable. Data is acquired from the main board ADCs and forwarded immediately to the sROD without buffering to minimise the potential for single event upsets and hence data corruption caused by radiation. The CERN developed GBT protocol is used and data reception at 4.8 Gb/s is possible with transmission at either 4.8 or 10 Gb/s.

3.4.2 AdvancedTCA Chassis

There are many variations of ATCA chassis, governed by the PICMG 3.0 specifications [43]. Some of the goals of the specification are to provide very high levels of availability through monitoring, control and redundancy as well as reduced development time and cost for hardware that uses the specification.

The specification is agnostic of any specific protocol or interconnect technology so Ethernet, PCI-Express and many other modern I/O interfaces are compatible.

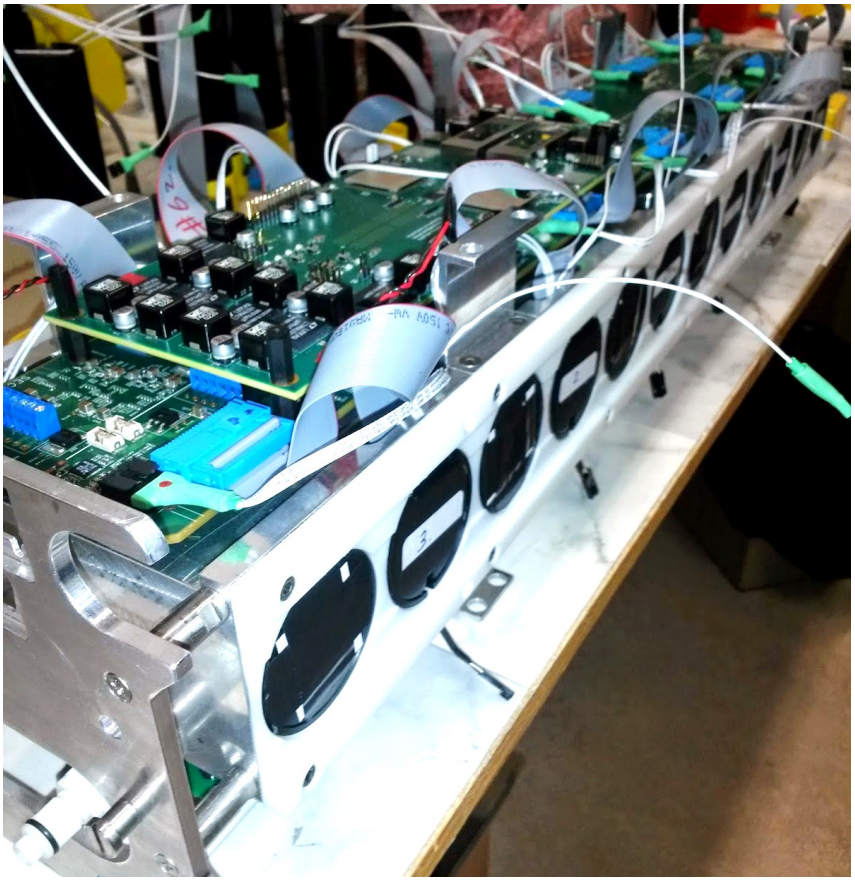


Figure 3.7: Photo of the TileCal mini drawer prototype with PMTs, Daughter Board and Main Board visible [41].

A populated ATCA chassis is made up by a front board, backplane and Rear Transition Module (RTM). Carrier boards are a type of front board which subdivide the large front board into four or eight smaller Advanced Mezzanine Card (AMC) slots.

Figure 3.9 shows a photo of an ATCA chassis with a carrier board (the front board) and CERN Gigabit Link Interface Board (GLIB) (double width AMC) half way into their slots [44]. The sROD prototype, described in Section 3.4.3 is also a double width AMC form factor.

It has been decided that for the Phase-II upgrade of ATLAS, ATCA chassis will replace the existing VME crates [16]. The VME standards which are in use are antiquated and do not support several features that are desired for the upgraded systems. The primary reasons for the use of ATCA are listed below [45]:

- ATCA conforms to a modern industry standard and is supported by many different companies.
- The chassis is modular which enables wide configuration flexibility. This is enhanced by the availability of components from many different vendors.
- There are sophisticated monitoring and control capabilities built into an ATCA chassis.

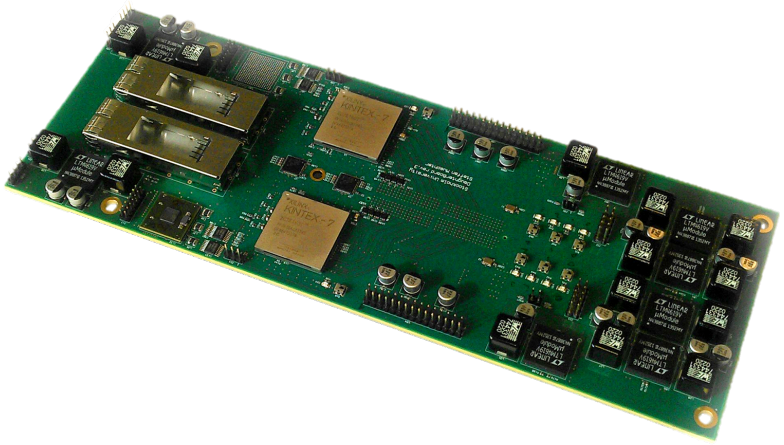


Figure 3.8: Photo of the completed mini drawer daughter board which is the interface to the sROD.



Figure 3.9: Photo of an ATCA chassis in the lab at the University of the Witwatersrand, Johannesburg, with a carrier board and CERN GLIB card partially inserted.

- An ATCA system provides very high reliability through redundant power supplies and components.
- ATCA supports "hot swapping" of components and as a result if an sROD, for example, needs to be replaced it can be done without powering down the rest of the chassis.
- The ATCA chassis is able to supply more electrical power than the VME system.

3.4.3 Super Read Out Driver

The Super Read Out Driver is the Phase-II upgraded version of the current ROD with several additional features [3]. The sROD prototype is a densely packed, 16 layer circuit board in an ATCA double width AMC form factor with two high-end FPGAs shown in Figure 3.11. The sROD prototype has a total of 290 Gb/s I/O capability divided over several different interface types to test different possible configurations. A block diagram of the sROD prototype is shown in Figure 3.10 and detailed specifications are shown in Table 3.1.

A Xilinx Virtex-7 FPGA interfaces to four QSFP+ connectors to provide 4 × 40 Gb/s fibre optic interfaces to four front end mini-drawers (or one super drawer). The Virtex-7 is more expensive but has better I/O capabilities than the Kintex-7 which is why it is used for interfacing to the front-end. There is a custom high data throughput interface between this Virtex-7 FPGA and the second FPGA, namely the Xilinx Kintex-7.

The Virtex-7 FPGA is responsible for managing all the data from the front-end and decoding the custom so-called GBT protocol in which the raw data is packaged. The pipeline buffer memories which are implemented in the front end in the existing system are implemented in this FPGA as well as the de-randomizer memories. Event selection from the Level-1 trigger accept signal is thus processed by the Virtex-7.

Most relevant to the research presented in this dissertation is the fact that the energy reconstruction algorithms will be implemented on this Virtex-7 FPGA. It may be feasible to offload these energy reconstruction algorithms to a high data throughput System on Chip (SoC) based processing unit external to the sROD. This will enable faster and easier prototyping and development of more sophisticated energy reconstruction algorithms. More details are provided in Chapter 4.

The Kintex-7 receives data from the Virtex-7 and performs some trigger pre-processing tasks before transmitting the processed data to the Level-1 Calorimeter trigger. In the sROD prototype, the Kintex-7 is connected to a separate, 10 Gb/s SFP+ port which links to the existing ROD for interoperability with the existing system for demonstration and testing purposes.

In the final version of the sROD for the Phase-II upgrade, there will be several Virtex-7 FPGAs to enable connection to multiple TileCal mini-drawers. The physical form factor for the final sROD will therefore be as a full ATCA front board. A single Kintex-7 will be connected each of the Virtex-7s and will amalgamate pre-processed data from them as well as perform higher level algorithms such as tower calculations and clustering.

This additional processing power with the availability of more data from the front end will facilitate a Level-0 trigger which will reduce the 40 MHz bunch crossing rate to 500 kHz within 6 μs [15]. The existing Level-1 trigger can then become far more sophisticated as it only needs to reduce the 500 kHz data to 200 kHz as opposed to dealing with the full 40 MHz workload as in the existing system.

Table 3.1: Summary of specifications for the sROD prototype [3].

Specification	Value
Channels	48
Sample Rate	40 MSa/s
Bits per Sample	12 bits
Data Rate per Channel	480 Mb/s - 60 MB/s
Total Data Rate	23.04 Gb/s - 2.88 GB/s
Maximum Latency	6 μs

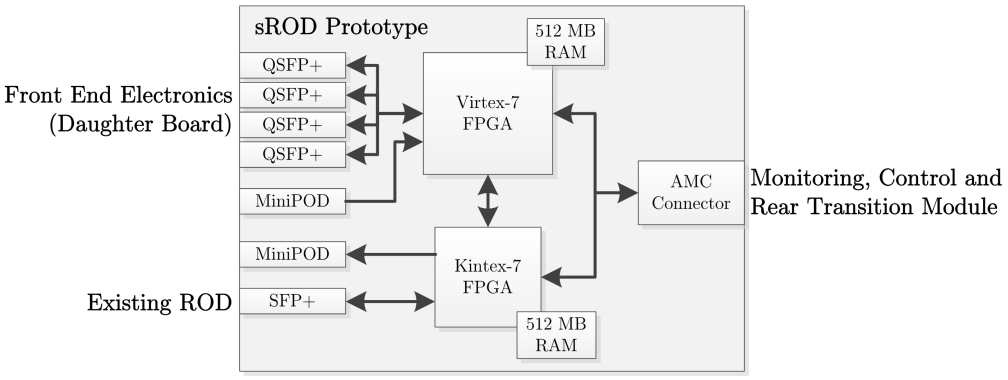


Figure 3.10: Block diagram of the sROD prototype I/O interfaces and FPGAs.

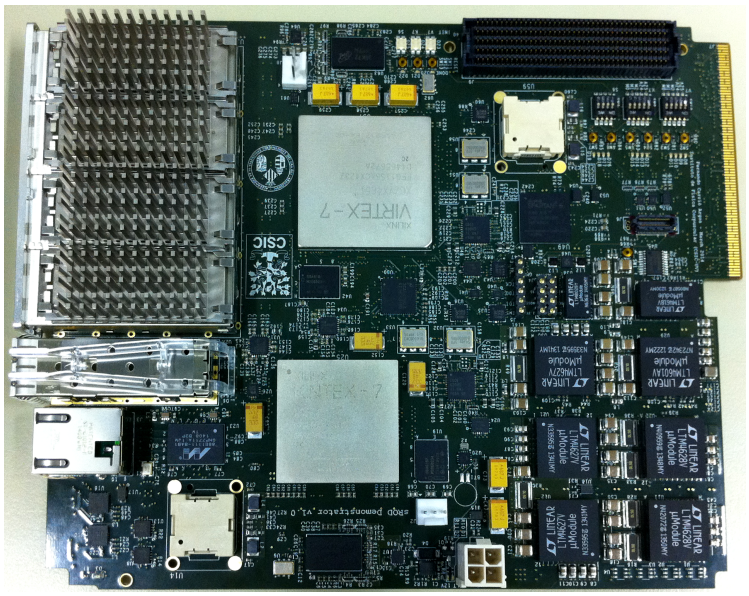


Figure 3.11: Photo of the sROD prototype with QSFP+ ports on the left and AMC connector on the right.

4

General Purpose sROD Co-Processing Unit

The sROD will link the front end read out electronics on the upgraded ATLAS TileCal to the back end data acquisition system. As discussed in Chapter 3, there are approximately ten thousand measurement channels on TileCal which must be processed, resulting in a very large quantity of raw data. In the upgraded system where the sROD will be present, it will function as a Level-0 filter by performing tower and clustering algorithms and also energy reconstruction [3].

The sROD is an FPGA based device and by nature has a significant amount of external I/O, as well as the potential to perform high performance processing tasks within very tight timing requirements. It is, however, a challenging and time consuming task to develop complex and dynamic firmware for an FPGA.

The TIOBE Index is an indicator of the popularity of many different programming languages [46]. The index is generated by counting the number of search results over many different search engines and normalising the number of hits for a specific language as a percentage of the total for all the programming languages on the TIOBE list. The "popularity" of languages such as Java, C and C++ are between 5% and 15%.

Developing and debugging firmware for FPGAs is a complex task which requires the use of specialised programming languages such as VHDL and Verilog. VHDL and Verilog are low on the TIOBE Index, being in the lower half of the 100 languages in the list and thus having less popularity than the 50^{th} language which is 0.2%, as of April 2015. It can, therefore, be assumed that the number of people with VHDL or Verilog skills is far less than the number of people with expertise in more common languages such as C and C++.

Developing FPGA firmware requires access to sometimes very expensive hardware and software whereas more conventional programming languages, such as C++, only require a standard computer with readily available software. FPGA development also requires intimate knowledge of the specific FPGA that is being developed for, which increases overall development time. Developing for general purpose CPUs requires no paradigm shift and very little intimate knowledge of the underlying hardware as the compiler abstracts this complexity away from the developer.

Thus, the challenge of implementing and tweaking more advanced signal processing algorithms for better quality physics data on an FPGA based device is not insignificant. If a general purpose CPU based device with suitable specifications were available, then various algorithms could be effectively offloaded from the sROD to a so-called Processing Unit (PU) and implemented in a more popular programming language such as C or C++.

Depending on the latency, throughput and processing power of this PU, it may be possible to dynamically adjust the energy reconstruction parameters of the sROD in real time using sophisticated decision logic that is difficult to implement on an FPGA. It is also a possibility for the actual energy reconstruction to take place on this PU, which would then act in a similar fashion to the DSPs in the existing system. In this case, the sROD would still be required as the interface to the front end electronics and would be used for data packaging and basic signal conditioning before and after it passes through the PU.

Depending on the specific use case of this general purpose PU, the system may be either CPU limited or I/O limited: In a CPU limited application, less I/O bandwidth may be adequate for keeping the CPU busy. Conversely, in a system where the expected algorithms are less computationally intensive, the I/O bandwidth is expected to be the limiting factor of the data throughput of the system and a less powerful CPU with more I/O bandwidth is preferable due to lower power consumption [47], [48]. This ratio of computing power to I/O bandwidth is called system balance.

The conventional "real time" computing paradigm is most suitable to the use case of the sROD co-processor or PU because of the strict timing constraints [49].

The computing paradigm that best describes that of the PU in a general purpose, non real time sense is High Volume Computing (HVC) [50]. HVC is a data center based computing paradigm where the focus is on loosely-coupled, throughput-oriented workloads in terms of either requests or processed data per unit time or energy. The use of the PU in the ATLAS high lever trigger is an example of an HVC application.

In comparison, the workload of a High Performance Computing (HPC) system is computationally intensive and the I/O bandwidth and latency are secondary to compute performance. Even so, the latency of the I/O interconnect plays a significant role in the overall performance of a system. In some cases, microsecond differences in latency affect the performance of some benchmarks by several percent [51].

In all three computing paradigms, the accurate characterisation of the bandwidth and latency of the available I/O interfaces is important.

Conventional, discrete computing systems have separate CPU and peripherals such as network interface cards. This increases the design complexity, cost and space requirements of the system. SoC based systems are typically more cost effective, compact and energy efficient than their discrete counterparts. For these reasons, SoCs have been widely adopted in mobile devices such as smart phones, tablets and some laptop computers.

Many different SoCs exist from several vendors with widely varying specifications such as CPU architectures and performance, memory interfaces and external I/O connectivity. This research is focussed on higher performance ARM Cortex-A based SoCs, which are introduced in Section 4.1. The section will also introduce the ARM SoC based hardware platforms that have been tested and used in this research.

In this chapter, a general purpose, SoC based PU will be proposed along with details of how it can be connected to the sROD. Various available SoC I/O interfaces are discussed and the most suitable interface, PCI-Express, is tested more thoroughly in a basic proof of concept system.

It should be noted that the processing performance of the PU will have a significant impact on the latency, or how long it takes to perform the required calculation which is critical to its success in a real time system.

In this chapter the algorithms that may be run on the PU are not considered and so the CPU performance of the various platforms is assumed to be sufficiently high so as to not cause a processing related bottleneck in data throughput or latency. The CPU performance in terms of the movement of data is indeed considered where applicable.

The CPU performance is discussed in detail in Chapters 5 and 6.

The PU should be able to sustain the data throughput and latency requirements of the sROD prototype described in Chapter 3, Section 3.4.3. In summary, a total data rate of at least 2.88 GB/s and a latency of less than 6 μs is required. The following questions must therefore be answered:

- Can a single, or a cluster of ARM SoCs sustain the required data bandwidth for the sROD prototype using a standard external I/O interface?
  - What is the latency and software overhead of the I/O interface?
  - What is the actual sustainable data throughput of the interface, assuming it is theoretically suitable?
- How will the sROD prototype connect to the PU, given the constraints of the existing ATCA system and sROD connectivity?

A discussion of the various I/O interfaces that are commonly found on SoCs can be found in Section 4.2. This discussion will also include some details of how these interfaces may be used, if at all, as well as some advantages and disadvantages of each interface.

PCI-Express is typically used for CPU to peripheral communications and not commonly used for inter CPU (or inter-SoC) communication, a proof of concept system whereby two ARM Cortex-A9 SoCs are linked via their PCI-Express interface is presented in Section 4.3 along with performance measurements.

The connection between the sROD and the PU, which will be located in an ATCA system its associated connectivity constraints, is discussed in Section 4.4. Finally, Section 4.5 provides a high level overview of how SoCs may be connected together on board the PU and some practical considerations.

4.1 ARM System on Chips

The various ARM architectures and instruction sets, named after the company, ARM Holdings plc, have been designed from conception with energy efficiency in mind. In the past, ARM was an acronym for Acorn and subsequently Advanced RISC Machine but is now simply a name.

ARM SoCs provide an energy efficient alternative to x86 and x86-64 instruction set based CPUs such as those commonly provided by Intel and AMD which are

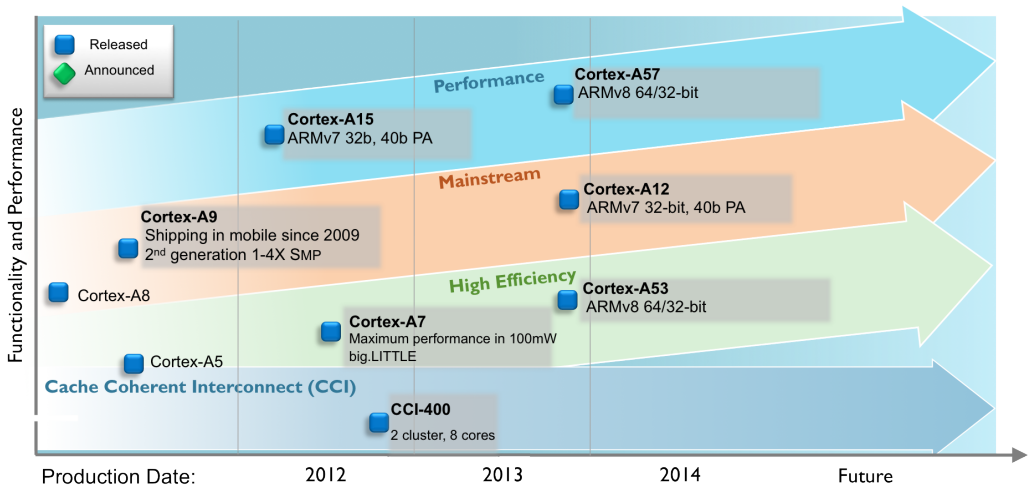


Figure 4.1: ARM Cortex-A CPU roadmap from the Cortex-A8 in 2005 to Cortex-A57 in late 2013 [53].

normally found in servers and desktop computers where low energy consumption is usually secondary to high performance [6]. This is primarily why the use of ARM SoCs is widespread in mobile devices where battery life is a primary concern. For similar reasons, the use of ARM SoCs in fixed installations is an active area of research. It has been calculated, for example, that a 10000 core data center based on SoCs with a 16 nm silicon process is estimated to be 26% cheaper overall than its discrete equivalent in terms of energy consumption, capital costs and other running costs [52].

ARM Holdings plc does not manufacture SoCs but rather licenses CPU designs to vendors which combine them with other ARM or third party peripherals such as PCI-Express, Graphics Processing Units (GPUs) and networking to create so-called SoCs. Because of this flexibility there is a huge variety of ARM-based SoCs on the market. The advent of server-grade ARM SoCs is relatively recent. The Applied Micro X-Gene is marketed as the first 64-bit (ARMv8 architecture) server-grade SoC as of 2014 [11].

There are several well known ARM SoC vendors such as Apple, Samsung and Texas Instruments. On the other hand, there are many low cost ARM SoC vendors such as AllWinner, MediaTek and Rockchip. This variety has dramatically increased the availability of ARM SoCs and has led to many open source community development boards and platforms.

Since the advent of the smart phone, ARM SoC performance has increased rapidly in line with consumer demand for faster mobile devices. Figure 4.1 shows a roadmap from ARM on their higher performance Cortex-A range of CPUs [53], [54].

For example, a single core ARM 1176JZ(F)-S v1.0, 620 MHz CPU (ARM11 architecture announced in 2002) was present in the original Apple iPhone released in 2007 and a single core ARM Cortex-A8 (architecture announced in 2005) was present in the Apple iPhone 3GS, released in 2009 [55], [56]. Smart phones and other mobile devices have since progressed through ARM Cortex-A9, A15 and recently Cortex-A57 based SoCs with up to 8 cores running at over 2 GHz.

Five different test platforms were purchased in order to gauge the performance, specifications and hence suitability for the sROD co-processing unit of a variety of

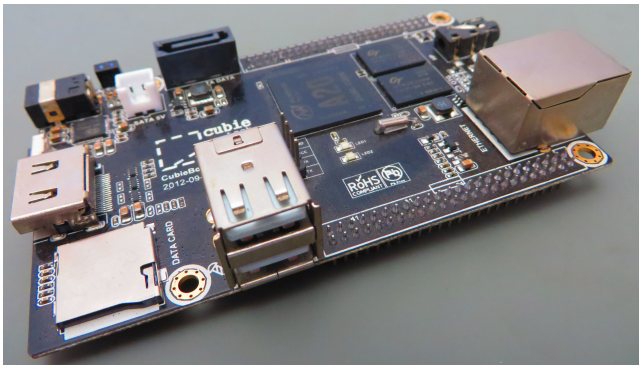


Figure 4.2: Photo of a Cubieboard A20 in the HTEL at the university.

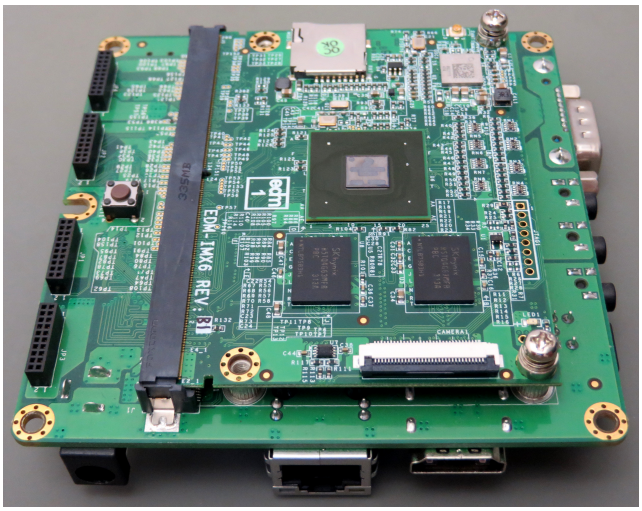


Figure 4.3: Photo of a Wandboard Quad with heatsink removed in the HTEL at the university.

ARM SoCs from the low end to the high end. A brief description and photo of each platform is provided in Section 4.1.1.

4.1.1 ARM Test Platforms

The Cubieboard A20 is a cost effective, open source board that uses a dual core ARM Cortex-A7 SoC running at 1 GHz with 1 GB RAM [57]. This is an example of a low power, very low price point SoC with the lowest theoretical performance of the five platforms. A photo of the Cubieboard is shown in Figure 4.2.

A quad core 1 GHz ARM Cortex-A9 SoC is present on the Wandboard Quad, which is also an open source board shown in Figure 4.3 [58]. The Wandboard has 2 GB RAM. The Cortex-A9 is the oldest ARM core that was tested, dating back to approximately the year 2009.

The NVIDIA Jetson TK1 is a commercial development platform provided by NVIDIA and the ODROID-XU+E is a community board but is not fully open source, most likely due to Non Disclosure Agreements (NDAs) with Samsung [59], [60]. Both of these platforms use high performance ARM Cortex-A15 based SoCs at 2.3 GHz and 1.6 GHz respectively. The SoC on the Jetson board is of particular interest

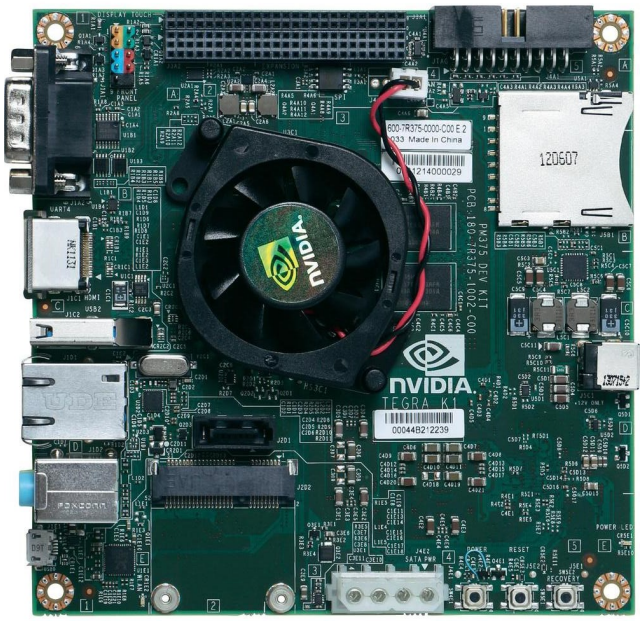


Figure 4.4: Photo of a NVIDIA Jetson-TK1 in the HTEL at the university.

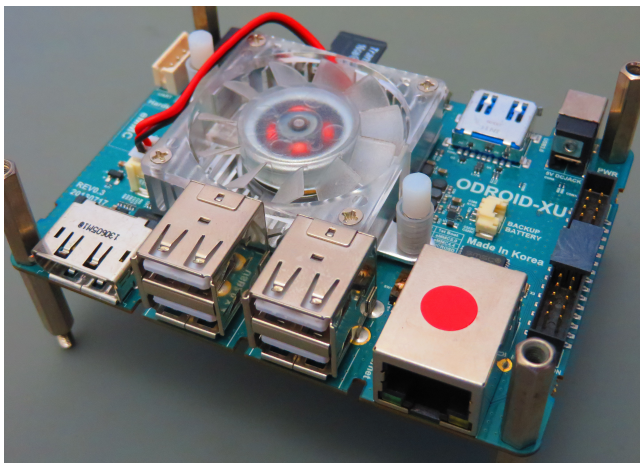


Figure 4.5: Photo of an ODROID-XU+E in the HTEL at the university.

because it contains a high performance general purpose programmable NVIDIA Kepler GPU which may be used for high performance processing. This research will not delve into the use of this GPU, however. The Cortex-A15 is currently the highest performing 32 bit ARMv7-A architecture core available. Figure 4.4 shows a photo of the Jetson board and Figure 4.5 shows the ODROID.

Finally, the X-Gene Mustang is a development platform provided by Applied Micro for their server grade X-Gene SoC [11]. This SoC has a high performance, 2.4 GHz eight core CPU whose architecture is closely based on the ARM Cortex-A57. The X-Gene Mustang supports up to 64 GB of RAM but by default has 16 GB installed.

The Cortex-A57 is an ARMv8 architecture, which supports native 64 bit computing. The Mustang is theoretically the highest performance platform that was tested, with the most advanced features. The X-Gene is also the most expensive SoC out of



Figure 4.6: Photo of the X-Gene Mustang board in the HTEL at the university.

the five. A photo of the Mustang system is shown in Figure 4.6.

4.2 System on Chip External I/O Interfaces

The SoCs found in mobile devices typically have many different I/O interfaces - some are high bandwidth such as the connection to the camera, and some are low bandwidth, such as a simple serial connection. For a high data throughput system, the SoCs (or CPUs more generically) must have some sort of high data throughput external I/O interface which can be used for inter SoC or sROD - SoC I/O.

High bandwidth external I/O on SoCs is somewhat rare because a mobile device such as a smart phone or a tablet has no requirement to connect to very high speed Ethernet, for example, when it has a WiFi connection and mobile data. In some cases a SoC is not only intended for mobile use: a smart phone SoC could also be used in a smart television, for example. These non-mobile applications typically demand additional I/O interfaces which may provide higher data throughput.

If the external I/O bandwidth available on a single SoC is insufficient to satisfy the bandwidth requirements of the sROD co-processor then several SoCs can be connected in parallel in order to amalgamate their I/O. They can also be connected to each other to form a "cluster on a board". By clustering several SoCs it allows more complex algorithms which can harness multiple CPUs and multiple I/O interfaces simultaneously.

All of the external I/O interfaces described in this section are packet based and in general this is indeed the case for most digital I/O interfaces. A general formula for maximum data throughput is shown in Equation 4.2. The packet size and latency must be realistic values for the particular interface.

The latency is the time taken from when a packet of data is sent by one device to when it is received by another. The time from when a packet is sent to when a response is received is called the round trip latency and is more common. Generally the latency of a communication link excludes time incurred by application processing but should include the time required by the drivers for the movement of that data.

A simple point to point link latency can be described by Equation 4.1.

$$\text{Link Latency(s)} = D + 2H + 2S \tag{4.1}$$

where D is the latency introduced by the physical distance between the two devices. The hardware latency, H, is incurred by the network adapters or controllers on both ends of the link, and the software latency, S, is a combination of the device drivers and possibly the operating system networking stack.

The data throughput of a link is proportional to its latency, as shown in Equation 4.2. Often the bandwidth of a link is quoted in specifications, but this a theoretical maximum with an ideal (typically large) packet size and minimum latency and is rarely reached in reality.

$$\text{Data Throughput (B/s)} = \frac{\text{Packet Size (B)}}{\text{Latency (s)}} \tag{4.2}$$

Several common external I/O interfaces are described in the following subsections. Their data throughput capabilities as well as latencies and CPU processing requirements are detailed. Table 4.1 presents the I/O interface specifications for the test platforms.

Table 4.1: External I/O interface specifications for the available ARM platforms.

	A20	**i.MX6Q**	**Tegra-K1**	**Exynos 5410**	**X-Gene**
Manufacturer	AllWinner	Freescale	NVIDIA	Samsung	Applied Micro
Platform	Cubieboard A20	Wandboard Quad	NVIDIA Jetson	ODROID-XU+E	X-Gene Mustang
ARM Core	Cortex-A7	Cortex-A9	Cortex-A15	Cortex-A15 + A7	X-Gene 1
Ethernet Ports	1	1	1	1	2
Ethernet Bandwidth (Gb/s)	1[1]	1[2]	1	0.1	10×2
USB Version	2.0	2.0	3.0	3.0	3.0
PCI-Express Version	-	Gen-2	Gen-2	-	Gen-3
PCI-Express Lanes	-	1	4[3]	-	16[4]

[1] Limited to 100 Mb/s by the board design.
[2] There is a bug in the silicon whereby the Ethernet bandwidth is limited to 470 Mb/s (ERR004512) [61].
[3] Only one lane is available externally via a Mini-PCIe connector.
[4] Only eight lanes are available via a PCIe x8 card connector.

4.2.1 Ethernet

Ethernet is the most common inter-system method of data transfer. It is based on a physical interconnect such as a copper or fibre optic cable and is governed by the IEEE 802.3 standard [62].

Most ARM-based SoCs have an Ethernet interface ranging from 100 Mb/s to 1 Gb/s. The SoC provides a Media Access Control (MAC) interface to which a Physical Layer (PHY) integrated circuit is connected to provide a RJ-45 or SFP connector where a standard Ethernet cable can be plugged into.

The bandwidth of the SoC Ethernet is limited by the SoC MAC interface, assuming a suitable PHY is used. If a 100 Mb/s PHY is connected to a 1 Gb/s MAC then the interface will run at 100 Mb/s.

Table 4.1 provides the Ethernet connectivity specifications on the SoCs that were tested. Most of the SoCs support 1 Gb/s Ethernet and only the server grade SoC, the X-Gene, supports faster Ethernet. 10 Gb/s Ethernet is still rare in a consumer type environment which is why it is not found on lower end SoCs.

Each packet or "frame" sent over an Ethernet link has a short, 38 byte header which contains information such as the destination and source address. The data payload of an Ethernet frame is typically 1500 bytes but can be increased to 9000 bytes in what is called a jumbo frame [62]. As a consequence, the data throughput efficiency is either $1500/1538 = 97.5\%$ or $9000/9038 = 99.6\%$.

The latency of an Ethernet interface is made up of several components. The software stack which includes drivers inside the Linux kernel as well as the application software make up a significant potion of the latency of a connection.

The hardware also has a latency that is dependent on the packet size and interface speed. Physical aspects of an Ethernet network such as the length of a link and number of switches also impacts the latency, but in the case of a sROD co-processor, switching is unlikely and the distance between components will be short.

The latency of application software will not be considered further as it is highly variable but the latency of the Linux kernel networking stack and drivers is worth looking into in more detail. Two very common network protocols are Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) [63], [64]. TCP is a reliable protocol which guarantees in order packet delivery with the possibility of increased latency. UDP, on the other hand, is an unreliable protocol where packets sent have a chance of not arriving, or arriving out of order.

TCP has a higher software complexity than UDP and, therefore, also has higher CPU processing requirements. If packets arrive out of order, TCP must spend time reordering them before presenting them to the end application. If packets do not arrive, or arrive corrupted, they must be retransmitted.

UDP has almost no processing requirements, as it presents packets to the application as soon as they arrive. UDP does include a checksum on the packets so it does discard them if they are corrupted, but it does not automatically request for that packet to be retransmitted as is the case with TCP.

There are alternative network stacks that can be used instead of or in conjunction with the built in Linux stack while still utilising standard Ethernet hardware. These offer performance and latency advantages through more optimised code or sometimes lightweight alternative protocols. An open source example called Open-MX implements the Myrinet Express network stack on a generic Ethernet network [65]. Open-MX reports significantly increased performance figures, shown in Table 4.2.

Table 4.2: Performance of the Open-MX networking stack compared to the default Linux stack [65].

	TCP MB/s - μs	**Open-MX** MB/s - μs
Open MPI (1 Gb/s)	117 - 25.1	117 - 17.9
Open MPI (10 Gb/s)	6462 - 15.1	9106 - 7.2
MPICH-MX (10 Gb/s)	-	9367 - 7.1

More advanced SoCs, such as the X-Gene, have network offloading built-in to the network interface hardware. This relieves the CPU from certain tasks, such as verifying the data integrity checksums, requesting retransmits and sending acknowledgements [66]. There is a network performance benefit from this but if UDP is used then the impact is less significant.

In the X-Gene 2 SoC, which has not been released yet, the network offload engine supports Remote Direct Memory Access (RDMA) [67]. RDMA is a feature that is harnessed by the iWARP protocol, which enables remote systems to write data directly to another systems memory without the intervention of the CPU [68]. This decreases the network latency and also frees the CPU from copying data from the network stack to the application leading to a further increase in throughput and overall performance.

Unfortunately, RDMA capable Ethernet interfaces are not available on commodity ARM SoCs and will not be in the foreseeable future because there is no requirement for it except in server environments.

4.2.1.1 Ethernet for the PU

As mentioned in Section 4.2, it is possible to connect multiple SoCs in parallel to amalgamate their I/O and processing power. Based on the required data throughput specifications for the sROD prototype in Table 3.1 and assuming a 1 Gb/s (125 MB/s) Ethernet interface, 24 interfaces and therefore 24 SoCs would be required to handle the total sROD data throughput of 2.88 GB/s.

10 Gb/s Ethernet is more suitable to the task as only three interfaces would be required. A single X-Gene SoC has two 10 Gb/s interfaces, and so two SoCs would be sufficient to satisfy the I/O requirements of the sROD prototype.

The physical attributes of the PU, which are described in detail in Section 4.4, limit the feasibility of an Ethernet solution for several reasons. First and foremost, the number of electrical links between the sROD prototype and PU is greater than the number available via the ATCA backplane. The physical area required by a large number of SoCs and associated components is also an issue.

Server grade SoCs with 10 Gb/s Ethernet provide a feasible solution but the cost is significantly higher than commodity SoCs with an alternative and yet suitable high bandwidth interface.

The latency of Ethernet is also a concern. The sROD is required to receive, process and retransmit data within a window of 6 μs. Table 4.2 shows that the latency of 10 Gb/s Ethernet with a low latency custom protocol stack still has a latency of at least 7 μs.

4.2.2 USB

Universal Serial Bus (USB) ports are ubiquitous and need little explanation [69]. A USB port must be either a host or a device and a single host can be connected to multiple devices. The latest USB standard is 3.0 which supports up to 5 Gb/s bandwidth. The more common USB 2.0 standard supports up to 480 Mb/s bandwidth. Several recent ARM SoCs support USB 3.0 but at minimum USB 2.0 is supported, as shown for the test platform in Table 4.1.

There are several ways in which to use USB to interconnect SoCs or to connect SoCs to external systems such as the sROD.

An obvious solution is to connect a USB to Ethernet adapter device (or controller chip) to one of the SoCs USB host ports. 1 Gb/s USB Ethernet adapters are readily available in both USB 2.0 and USB 3.0 versions. If USB 2.0 is used then the maximum Ethernet bandwidth is limited to that of the USB port, which is up to 480 Mb/s. USB 3.0 allows the full 1 Gb/s to be used, assuming that the USB to Ethernet controller indeed supports the full 1 Gb/s.

There are several disadvantages to this solution. The performance issues related to protocol stack overheads discussed in Section 4.2.1 apply but there is additional latency introduced by the USB driver stack in the Linux kernel as well. The USB 2.0 specification states a polling interval of 125 μs which indicates the approximate order of magnitude of possible latencies [70].

A further disadvantage is that USB Ethernet adapters are not able to use Direct Memory Access (DMA) and so the CPU must be used for all data transfers, potentially limiting maximum data throughput and wasting CPU resources that could otherwise be used for processing data.

It is interesting to note that this method is currently being used in the European Mont-Blanc project [71]. The project aims to develop an energy efficient HPC prototype system using commercially available SoCs and also to optimise associated software applications.

The latest Mont-Blanc prototype consists of Bull B505 blades with 15 dual-core Samsung Exynos Cortex-A15 SoCs on each which are interconnected via a USB 3.0 to Gigabit Ethernet network [72]. A picture of a blade is shown in Figure 4.7.

The Mont-Blanc USB to Ethernet interconnect solution is suitable for their design goal of HPC where the supercomputer will process compute intensive tasks but is not suitable for streaming high data throughput, which is required for the sROD PU.

Another method of using USB for data transfer between SoCs is by connecting the host port of one SoC to a device or On-The-Go (OTG) port of another. The Linux kernel includes several different USB "gadget" drivers that enable Ethernet over USB. This means that the host SoC "thinks" that it is connected to a USB-Ethernet adapter and proceeds to use the network as per usual when in fact it is connected directly to another SoC. These special drivers have some optimisations and do not require any special hardware besides the OTG port. Standards such as RNDIS from Microsoft, Ethernet Control Model (ECM), Ethernet Emulation Model (EEM) and Network Control Model (NCM) govern the operation of these special drivers [73].

An issue with these Ethernet-over-USB drivers is that the underlying USB protocol is not inherently optimised for sending Ethernet packets. By nature, many USB devices require low latency operation and so the USB protocol is designed for a balance of throughput and low latency. As mentioned in Section 4.2.1, an Ethernet data payload is typically 1500 Bytes whereas a packet in USB 2.0 is smaller. Multiple

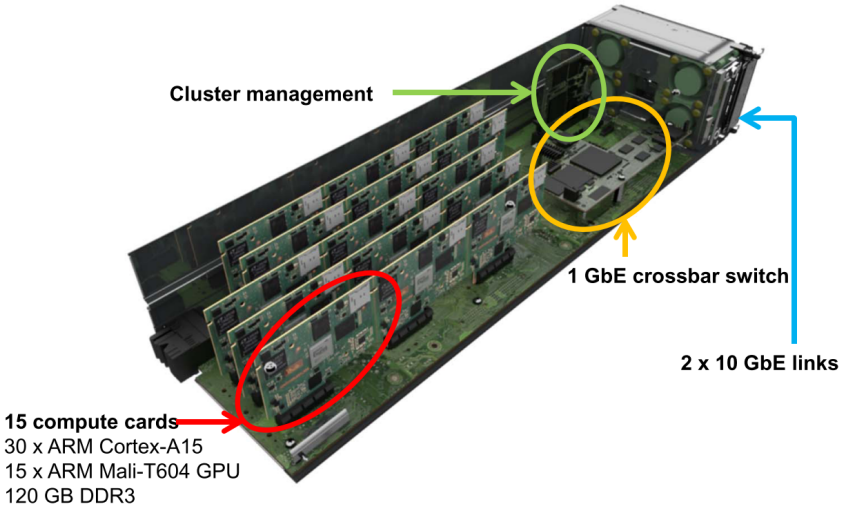


Figure 4.7: A rendering of the latest Mont-Blanc prototype blade [72].

USB packets are required per Ethernet packet, which introduces a significant latency to the Ethernet protocol stack.

Data throughputs of up to 28 MB/s have been reported, which is approximately 50% of the theoretical maximum of 60 MB/s for a USB 2.0 port [73]. USB 3.0 based Ethernet-over-USB may yield higher performance, up the theoretical maximum of 5 Gb/s, at the expense of high CPU usage because of the Ethernet emulation and lack of DMA.

4.2.2.1 USB for the PU

The high latency and CPU usage incurred by both USB-Ethernet adapters and also Ethernet-over-USB drivers, as well as the limited data throughput, means that USB is not suitable for high data throughput communications even though the specifications appear promising. That said, Ethernet-over-USB would work well as a simple and low cost out of band SoC interconnect for a control and administration network.

A solution that may be feasible is if the sROD FPGA were to implement a custom USB device. A device driver could be written for the PU Linux kernel to read and write data to and from the FPGA. The total bandwidth and latency of the USB interface may still be an issue and the effort required for this solution is significant.

4.2.3 PCI-Express

PCI-Express or PCIe is less common than Ethernet and USB but it is found on higher end SoCs. PCIe is typically used to connect a CPU to peripherals such as graphics cards, video capture and sound cards and even network adapters. PCIe is a high bandwidth I/O interface with very low latency, but it is designed for use on a circuit board and not for long distance cables.

A PCIe link consists of one up to 32 parallel lanes, where each lane is bidirectional. Each lane consists of two controlled impedance differential pair signals for transmitting and receiving data.

Table 4.3: Specification comparison between PCI-Express generations for one lane [74].

	Encoding (Data/Total)	**Data Rate** (GT/s)	**Data Bandwidth** (MB/s)
Gen-1	8b/10b	2.5	250
Gen-2	8b/10b	5	500
Gen-3	128b/130b	8	985
Gen-4	128b/130b	16	1969

The use of differential signals is important because electrical interference from other nearby signals or components will typically couple to both wires in the pair. By measuring the difference between each wire in the pair this "common mode" interference is nullified. When the voltage difference between the pair of signals is above a certain threshold it is interpreted as a logical one, and when the voltage is less than a certain threshold it is a logical zero.

PCI-Express has several generations, abbreviated as Gen-1, Gen-2, Gen-3 and Gen-4. The bandwidth increase of each successive generation of PCIe is typically double. This is accomplished with a combination of more efficient encoding of the data and a faster link frequency. Table 4.3 shows the details of different PCIe generations for one lane. The number of lanes multiplied by the bandwidth of a single lane is the bandwidth of the link.

PCIe, like USB, has two modes: Root Complex (RC), which is usually on the CPU side of a link, and Endpoint (EP) which is on the peripheral side.

Unlike the older PCI standard, which used a bus where the CPU is connected to several peripherals with the same wires, a PCIe link is point-to-point. Several EPs can be connected to a single PCIe RC by using a PCIe switch chip which works much like an Ethernet switch, as shown on the left of Figure 4.8. When a switch receives packets from an EP or RC it reads their destination field and forwards them to the appropriate port. The typical switching latency for PLX Technology switches, which is one of the major PCIe switch manufacturers, is 150 ns [75].

Connecting two CPUs via PCI-Express is not common. Some high-end PCIe switches provide what is called a Non-Transparent (NT) port. The NT port allows a RC to be connected to another RC by making the RC behind the NT port appear as an endpoint. This is used in server environments for failover and redundancy [76], as illustrated to the right of Figure 4.8 and also in Figure 4.9. A larger switch with more than one NT port would allow several CPUs to be connected to each other.

Some SoCs have a PCIe controller which supports EP mode as well as RC mode by setting a specific register and writing an appropriate Linux device driver. Hence, the use of PCIe between SoCs as an interconnect is possible. This direct connection is illustrated to the right of Figure 4.9.

Based on the data throughput specifications from Gen-2 and up, PCIe appears to be a promising candidate for high data throughput I/O on ARM SoCs. The low latency of PCIe is also an excellent attribute. Unfortunately, in some SoCs, certain features such as PCIe DMA are not available. The impact of not having PCIe DMA, as well as a description of how it may be used is provided in Section 4.2.3.2.

To the best of the authors knowledge there is no academic literature confirming the performance and feasibility of a PCI-Express ARM SoC interconnect, however,

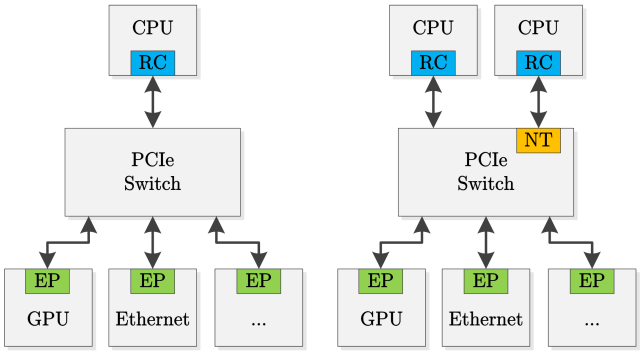


Figure 4.8: PCI-Express switching with a standard configuration to the left and a dual CPU failover configuration on the right.

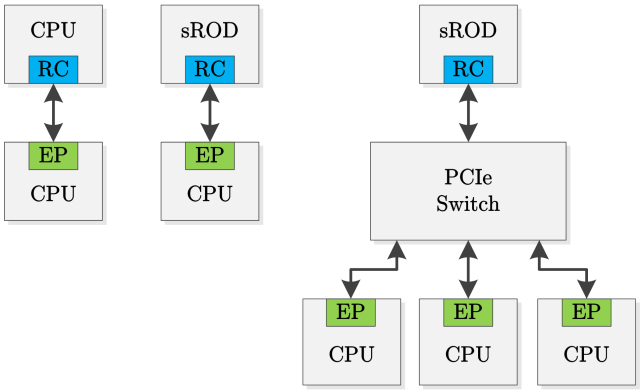


Figure 4.9: Connection between two CPUs with an RC - EP connection on the left. To the right, the sROD can be configured as RC with multiple CPUs connected via EP or NT ports.

one unofficial study has been done by a Freescale employee with results available on their web forum [77]. The test conditions in the Freescale i.MX6Q EP-RC system appear to be sub-optimal with little consideration given to signal integrity which is of utmost importance in high speed signalling to reduce errors and wasteful retransmits. The results presented on the forum are also not thorough.

To this end, although it is known from specifications that the configuration is possible, it is necessary to characterise the true data throughput and latency of the PCI-Express controllers on SoCs of interest and to determine their suitability for use in a sROD co-processing unit. In depth testing of a pair of Wandboards is presented in Section 4.3.

4.2.3.1 PCI-Express for the PU

A PCIe connection between the sROD FPGA, as the RC, and an ARM SoC, as EP, or vice versa is a feasible solution. PCIe provides high bandwidth capabilities as well as very low latency with custom device driver and if DMA is available.

With PCIe Gen-2, eight lanes are sufficient to transport the entirety of the 2.88 GB/s sROD prototype raw data requirement to one or more SoCs and with Gen-3 PCIe, four lanes are theoretically sufficient. The theoretical latency of PCIe hardware is on the order of 150 ns which leaves the majority of the 6 μs window for

Table 4.4: PCI-Express specifications for the available test platforms that support PCI-Express.

	SoC	**Mode**	**Lanes**	**Bandwidth** (GT/s - GB/s)
Wandboard	i.MX6Q	Gen-2 EP,RC	1	5 - 0.5
Jetson TK1	Tegra K1	Gen-2 RC +DMA	4	20 - 2
X-Gene Mustang	X-Gene 1	Gen-3 EP,RC +DMA	16	128 - 15.8

processing [75].

A direct connection between CPUs using PCIe is not the only way that PCIe can be used to achieve high data throughput. As mentioned in Section 4.2.1, 10 Gb/s Ethernet may be suitable for the high data throughput requirements of the PU.

A standard, high end RDMA capable 10 Gb/s Ethernet controller chip could simply be connected to the PCIe port of the SoC. This could then be used as a standard 10 Gb/s Ethernet connection to the sROD. This is a viable solution but would introduce network stack latency which is an issue that has been discussed in Section 4.2.1 as well as increasing the overall cost and space requirement of the solution.

Table 4.4 lists the ARM platforms that were studied and their PCI-Express specifications and theoretical PCIe performance.

4.2.3.2 Operation Overview

PCI-Express is a packet based, point-to-point interface whereby multiple devices can be connected using special switch chips [74]. A brief overview of how PCI-Express transfers data is important for interpreting the performance results. This section describes the basic protocol, where relevant to this dissertation, as well as performance implications of the various aspects of the protocol and controller hardware.

Like Ethernet, each packet contains several pieces of information as well as the data payload itself. The sender and receiver address is required as well as the type of packet, intended memory location to be written or read, if applicable, data integrity checksums and also sequence numbers to ensure in-order delivery of packets. Figure 4.10 shows a typical PCI-Express packet with the various data fields labelled.

These additional fields in both the transaction layer and data link layer required for the basic operation of the PCI-Express link add additional data that must be transmitted besides the data payload and therefore impact the efficiency of the link. The maximum Transaction Layer Packet (TLP) payload is limited by the PCIe

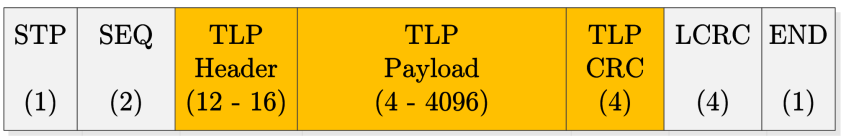


Figure 4.10: Structure and size of a PCIe packet with transaction layer parts highlighted in orange and data link layer in grey. The number of Bytes for each part is shown in brackets. For applicable acronyms see text.

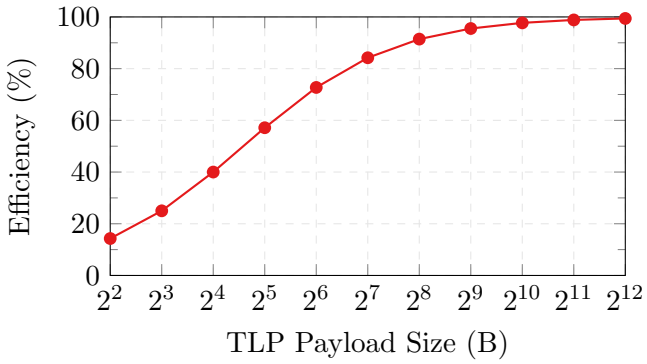


Figure 4.11: PCIe bandwidth efficiency with a fixed 24 byte packet overhead and varying data payload size.

controller hardware. This payload size can be configured but must be the same for all devices on a PCIe network.

The supplementary data in a PCIe packet is a constant size of either 24 or 28 bytes, depending on whether greater than 4 GiB memory must be addressed. This data is primarily made up of a sequence number (SEQ), the data payload and Cyclic Redundancy Checks (CRCs).

The data payload can be from 4 to 4096 bytes but is typically between 128 and 512 bytes for compatibility with legacy devices and drivers. Figure 4.11 shows the effect on a links data bandwidth efficiency for different sized payloads and is calculated according to Equation 4.3. A higher bandwidth efficiency implies that more data is sent per packet and should therefore approach the maximum bandwidth of the link, without taking latency into account. The i.MX6Q SoC, which was used in all tests, supports a maximum data payload (for the TLP) size of 128 bytes which corresponds to an efficiency of 84.2%.

$$\text{Link Efficiency } (\%) = 100 \times \frac{P}{P+S} \tag{4.3}$$

where P is the number of bytes in the TLP payload and S is the number of supplementary bytes. S is assumed to be 24 in Figure 4.11.

The PCI-Express controller handles all of the data link layer operations such as retransmits and error checking without CPU intervention to ensure the best case efficiency.

The RC CPU or EP CPU can place data onto the PCIe link by copying it to a special memory location which is mapped by the PCI-Express controller. The controller detects when data is written to this memory location, buffers and sends it over the PCIe link. When this mapped memory location is read by the CPU, the PCIe controller issues a read request to the other device which subsequently responds with the data at the requested memory location. The PCIe controller automatically splits and recombines the data that is sent or recieved in the case that it is longer that the configured maximum payload.

Figure 4.12 shows a simple example case where the EP writes an arbitrary number "DECAFBAD_{16}" which is the 32-bit hexadecimal representation of the number 3737844653_{10} and the RC reads this number and subsequently writes the number "01010101_{16}" which is transmitted back to the EP.

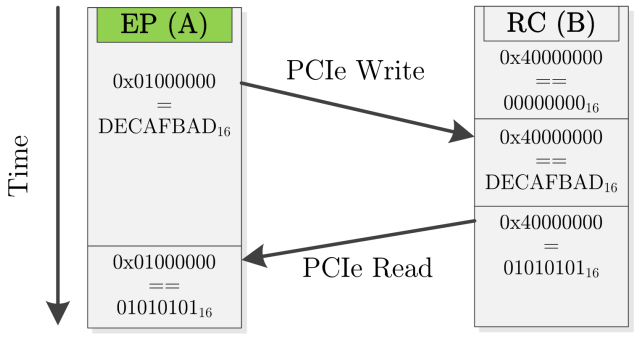


Figure 4.12: Simple flow of a PCIe Write and Read transaction showing the memory locations of device A and B and their contents. Note: '=' means 'set equal to' and '==' means 'is equal to'.

The use of this special memory location is not the standard way to transfer "bulk" data from one device to another. It is normally only used to store configuration data and memory locations to be used by another component of the PCIe controller: the PCIe DMA controller. The PCIe DMA controller is optional and it is the choice of the SoC manufacturer to include it or not based on cost, intended SoC usage and other business and design factors.

If there is no DMA capability on the PCIe controller then the CPU must typically wait for responses to read requests or must continually feed the PCIe controller with data for large writes. This leads to inefficiencies because the CPU is kept busy with tasks other than data processing. A DMA capable PCIe controller frees the CPU for tasks other than moving data from one memory location to another because it takes care of fetching or placing data in the specified memory locations.

A DMA controller typically has access to a large memory space which would normally include application (user-space) memory. This enables an application with a suitable device driver to write data directly to a peripheral device without CPU intervention. In the case of a CPU-CPU PCIe connection, one application could write data directly into the other CPUs memory where it can be processed immediately.

For example, a receiving device may be continuously running a data processing application where the input data is in the form of an array. This application would provide the address of this array to the sending device. The sender can then write data directly to this memory location for immediate processing. The write may be facilitated by the senders DMA by simply providing the controller with the memory location and length of the outgoing data. Without DMA, the sender can only write data to a special memory location on the receiver after which the Linux kernel must use the CPU to copy the data to the location provided by the application.

It is clear that DMA controller removes a step from the process of moving data and would increase data throughput, latency and processing efficiency. Test results in Section 4.3.2.3 show this to be the case.

4.3 PCI-Express Performance Tests

Only the Wandboard (Freescale i.MX6Q SoC) was thoroughly tested in a RC - EP system. On the NVIDIA Jetson-TK1 platform it is only possible to access a single PCIe lane and the SoC does not support EP mode and so an expensive

Non-Transparent switch would be required for testing. The X-Gene supports both RC and EP mode but it was only purchased later in the course of the research and so there was no time for the development of a test rig. The results obtained and experience gained with the Wandboards are sufficient to determine guideline specifications and requirements for a sROD co-processing unit.

The Wandboard Quad is an open source platform based on the Freescale i.MX6Q ARM SoC shown in Figure 4.3 [58]. The i.MX6Q is designed for use in mobile devices and is found in several consumer products such as the Amazon Kindle ebook reader. The i.MX6Q features a quad core ARM Cortex-A9 CPU with various connectivity options including 1 Gb/s Ethernet, USB 2.0 and most importantly, a single lane PCI-Express Gen-2 port which is theoretically capable of 500 MB/s data bandwidth. More in depth details of the Wandboard can be found in Section 4.1.1 and Chapter 5.

It is assumed that the Wandboard is designed well and to the various electrical specifications set out by the different components of the board. This assumption implies that the PCI-Express performance of the Freescale i.MX6Q SoC is being tested fairly. The so-called "PCI-Express Adapter Board" that is described in Section 4.3.1 has also been designed carefully to ensure minimum errors due to signal quality.

As discussed in Section 4.2, both throughput and latency specifications are important to the success of the sROD co-processor. The results obtained in testing the Wandboard should in most cases be comparable to other SoCs of similar performance, specification and price range due to standardisation with the PCI-Express standards.

The first throughput test, described in Section 4.3.2.1, is a modification of the PCI-Express driver in the Linux kernel where different sized blocks of data up to 8 MiB are written and then read back. The time taken to write and read the data is used to calculate the data throughput.

Since the I/O performance purely inside the kernel is of limited interest, a standard user-space C++ application was written which performs a Memory Map (mmap) to the PCIe controller, described in Section 4.3.2.2. This application was used measure the data throughput from a simple application, and is an example of best case "real world" performance of the Wandboard PCI-Express in an EP - RC configuration without DMA or special drivers.

Finally, a DMA throughput test was performed in Section 4.3.2.3. Since the PCI-Express controller on the i.MX6 SoC does not support DMA, the Image Processing Unit DMA unit was used instead. This "hack" re-purposes the Image Processing Unit to perform a DMA capable memory copy from a specified location in memory to the PCIe controller. Although this solution is not feasible for normal use, it serves as a proof of concept for the data throughput that can be expected with a PCI-Express controller that does indeed have a DMA controller.

The round trip latency of the pair of i.MX6 SoCs was tested with another custom application. A similar memory mapped approach was taken except very small packets were used to minimise the impact of data bandwidth. The time taken for a packet to be sent and a response to be received was recorded and the test was repeated one million times. The results and analysis of the round trip latency measurements are provided in Section 4.3.3.

Section 4.3.4 summarises the results obtained and draws conclusions on the use of PCI-Express as an interconnect for the sROD co-processing unit.

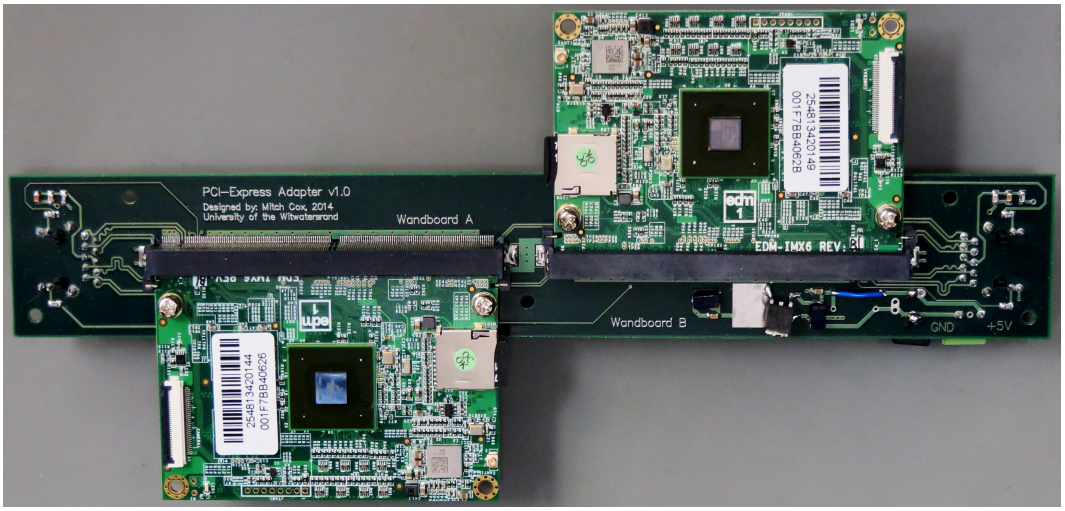


Figure 4.13: Photo of the Wandboard PCI-Express Adapter with Wandboards populated. The PCB was manufactured in South Africa and hand soldered by the author.

4.3.1 Wandboard Test Hardware

The Wandboard is designed to the EDM standard [78]. As such, the SoC as well as RAM and Ethernet PHY are on a small module which can be connected to any EDM compatible base-board which provides connectivity.

In order to test the PCI-Express performance of the i.MX6Q SoC, an adapter board was designed and manufactured to the relevant aspects of the EDM as well as PCI-Express Gen-2 standards. A photo of this adapter board is shown in Figure 4.13. The adapter board provides power and allows the Ethernet and Serial debug ports of the SoC to be accessed. The PCI-Express ports of each board were also linked to create an RC - EP system.

Care was taken with the layout of the PCI-Express traces on the custom adapter board. The PCI-Express standard requires length matching to within 0.127 mm between the traces in each pair as well as controlled impedances of 85 Ω ±10% differential and 60 Ω ±15% single ended [74]. As mentioned in Section 4.2.3, one pair is required for transmission and another for reception.

A sophisticated software package called HyperLynx from Mentor Graphics was used to simulate the PCB trace impedances [79]. The license was kindly provided by the School of Electrical and Information Engineering at the university. HyperLynx is an electromagnetic field solver and is used to verify and tweak high-end PCB designs before manufacture. The final circuit diagrams and PCB layout artwork can be found in Appendix A.

4.3.2 Throughput Tests

Data throughput measurements were done in several ways to ascertain the performance with different software configurations. The following subsections describe the implementation, justifications and results for the different tests.

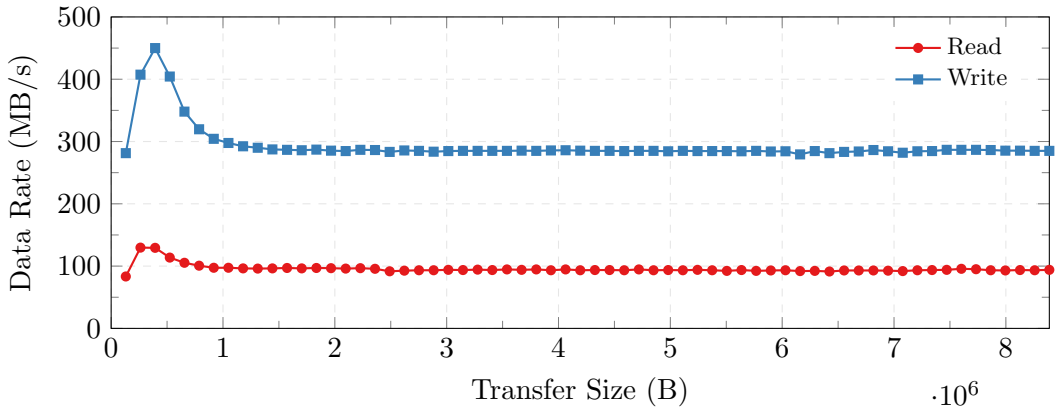


Figure 4.14: Measurement of Linux kernel based PCIe transfer rate for memcpys from 128 kiB to 8 MiB without DMA.

4.3.2.1 Kernel Memory Copy

Initial kernel throughput tests were based on those presented by the i.MX6 SoC manufacturer on an online forum for an i.MX6 EP - RC system where a questionable hardware setup was used [77]. In the test, a specific size block of data is copied over the PCIe link from the EP to the RC and then copied back. The time taken is used to calculate the data throughput in bytes per second.

The EP PCI-Express driver inside the Linux kernel was modified slightly to include a simple Memory Copy (memcpy) command before the device initialization completed. The RC driver on the second device waits until initialization is complete before continuing with the boot process. As a consequence, the system not busy with any other tasks except for the PCIe copy, ensuring an accurate result.

A range of measurements were taken for block sizes from 128 kiB to 8 MiB with results shown in Figure 4.14. 8 MiB is the maximum amount of data that can be copied at a time and is a hardware limitation.

The write throughput converges on 284.5 ± 1.3 MB/s and the read throughput converges to 93.3 ± 0.8 MB/s. The spikes in data throughput at smaller transfer sizes are artefacts due to cache effects.

The write throughput is higher because less transactions must take place - the CPU writes a block of data to the special memory location mapped by the PCI-Express controller and it is “posted” to the other device. The CPU is free to continue with other work such as copying the next block of data to the PCI-Express controller. The process of reading is more complex and requires two or more transactions. The first is a read request which specifies the address of data to read as well as how many bytes are required. The second transaction is the response which contains the data.

The write performance does not appear to correspond with the theoretical maximum of 500 MB/s for the Gen-2 x1 link, however, the performance can indeed be explained with a more in depth view by taking the link efficiency calculated in Section 4.2.3.2 into account.

Based on a maximum payload of 128 bytes, which is a specification of the i.MX6 PCI-Express controller, the resulting efficiency of 84.2% would result in 421 MB/s. This still does not correspond with the measured value of 284.5 MB/s even with an error of several percent for protocol overheads other than the packet header.

Upon further investigation, the implementation of the memcpy function in the Linux kernel is only able to copy 32 bytes at a time which corresponds to the L2 cache line size of the Cortex-A9 CPU in the i.MX6 SoC. The efficiency of the PCI-Express link with a 32 byte payload is only 57.1%, resulting in a theoretical maximum of 285.5 MB/s which is almost identical to the measured value of 284.5 MB/s.

4.3.2.2 Memory Mapped Copy

In Linux it is possible for a normal application to map a specific region of memory and read and write to it as if it is a file. The kernel handles translation between user space and kernel space memory but there is a small overhead. This is called a mmap and it typically enables higher performance I/O operations that going through standard Application Programming Interface (API) based channels.

A simple C++ application was created to mmap the special memory location that is used by the PCI-Express controller (described in Section 4.2.3.2). Blocks of data were copied to and from this memory mapped file and the time taken to do so was recorded. A sample of the code to perform the block copy is shown below. Volatile pointers must be used, otherwise the compiler may cache the data and it will not be written to the PCI-Express controller immediately.

Listing 4.1: Memory mapped copy loop code.

```
// Begin timer
for (int i = 0; i < dataPtrSize; ++i)
{
  *(volatile int*)(mmapPtr + i) = *(volatile int*)(dataPtr + i);
}
// End timer
```

The results of the mmap throughput tests are shown in Table 4.5. The results shown are for a 2 MB block of data but other block sizes were tested and all yielded similar results.

A benchmark of the maximum CPU copy throughput was run to verify the code was performed and is also shown in Table 4.5. This benchmark simply copied the data between two locations in RAM as opposed to the PCI-Express controller. It is clear that this benchmark memory copy is indeed faster than a copy to the PCI-Express controller. This result is on the same order of magnitude as the STREAM memory benchmarks in Chapter 5.

The data type used has an effect on the data throughput. This is due to the packet overhead described in Section 4.2.3.2 and shown graphically in Figure 4.11. In Listing 4.1, an *int* data type is used. An *int* occupies four bytes and so only four bytes of data payload are transmitted across the PCI-Express link per transaction.

The theoretical efficiency of the link with a four byte payload is only approximately 15%. If the *int* is changed to a *long long* data type which has a size of eight bytes, the efficiency is approximately 23% and the resulting throughput increase is clearly visible in the results. The results could probably be improved further with caching, up to the kernel memory copy maximum of 285 MB/s. Unfortunately, memory maps do not support caching in a suitable manner.

There is not an exact correspondence between the theoretical efficiency and the measured throughput. Automatic vectorisation by using the CPU Single Instruction Multiple Data (SIMD) NEON unit may have been performed by the compiler. As a

Table 4.5: PCI-Express mmap data throughput results. The theoretical maximum bandwidth is also shown, calculated with the bandwidth efficiency.

Data Type	**Size** (Bytes)	**Theoretical** (MB/s)	**CPU Maximum** (MB/s)	**Read** (MB/s)	**Write** (MB/s)
int	4	75	460	49	36
long long	8	115	1089	182	123

result, it is possible that the copy loop has been optimised to some extent and more than one *int* or *long long* is being copied at a time.

Interestingly, the read results are better than the write, contrary to what is expected based on the tests in Section 4.3.2.1. As discussed, the writes are being limited by the small data payload size. The reads, on the other hand, may be benefiting from a larger read request that is built in to the PCI-Express driver and is being partially taken advantage of.

4.3.2.3 DMA Copy with i.MX6 Image Processing Unit

The kernel memory copy described in Section 4.3.2.1 used the CPU to copy blocks of bytes in a loop. This naturally occupies CPU time which could be used for processing data.

Although the concept of DMA has already been described, a short discussion of how a DMA controller is typically used will be useful. The Linux kernel has a driver for the DMA controller which configures it and allows other drivers to make use of it in a relatively simple manner.

In the simplest case, a driver may specify that n blocks of memory should be copied from one location to another, specified by address pointers. The main CPU is used to write the appropriate configuration registers and to issue the "start" command. The CPU is then free to continue with other tasks. When the DMA controller has completed the copy, it issues an interrupt which indicates to the CPU that the operation is complete.

This functionality has been extended to peripherals on a SoC and is often more sophisticated. For example, the DMA controller may be used to read and write data to and from a hard drive, audio controller, GPU and even a PCI-Express controller.

In Section 4.2.3.2 the operation of a simple PCI-Express link is described. In summary, a special memory location can be mapped and when a CPU writes data to this location it is transmitted over the link. When a CPU reads data from this location it is requested. In a similar fashion, a DMA controller can be used to monitor this location and copy any received data to an arbitrary memory location for direct use by an application with very little CPU intervention. The converse is true for transmitting data.

In order to use DMA for PCI-Express, the DMA controller must be built-in as a component of the PCI-Express controller. Unfortunately on the i.MX6Q SoC, which was used for testing, there are no DMA capabilities on the PCI-Express controller. This is due to the very low price point of the SoC as high performance I/O was not likely to have been a critical design criterion. The Tegra-K1 and X-Gene SoCs do have DMA capable PCI-Express controllers.

Table 4.6: PCI-Express DMA data throughput results with i.MX6 IPU. The theoretical maximum bandwidth is also shown, calculated with the bandwidth efficiency.

TLP Size (Bytes)	**Theoretical** (MB/s)	**RC Read** (MB/s)	**RC Write** (MB/s)	**EP Read** (MB/s)	**EP Write** (MB/s)
64	372	248	375	182	369

As a work around, there is indeed a form of DMA available to the i.MX6Q PCI-Express controller. The Image Processing Unit (IPU) of the i.MX6Q SoC is able to access the mapped memory space of the PCIe controller as well as main memory. The IPU has a DMA controller typically used for fetching images from RAM or from a camera device, modifying them and then writing them to the GPU for display on screen. This functionality can be reconfigured to moving data from RAM and writing it to the PCIe memory instead.

In order to test the maximum throughput of this system, 4 MiB of random data was generated and then written to the PCIe mapped memory with the IPU. The data was then read back, also using the IPU. The time taken for these operations was recorded. Tests were run on both the Endpoint (EP) and Root Complex (RC). Six runs were performed in each direction, with the average results shown in Table 4.6.

The size of data transmitted per packet is 64 Bytes and this is governed by the specifications of the IPU. No memcpys take place and so the 32 byte limit described in 4.3.2.1 does not apply. The write throughput initiated by the EP and RC are almost exactly what is predicted by the theoretical link efficiency.

The read performance of both devices is lower than the write performance as is expected because of the additional read request packet overhead. The EP has significantly worse performance than the RC. An exact reason for this is unknown but it is likely due to the RC hardware and drivers being more optimised than that of the EP. This is because the expected usage mode of the SoC is RC with an attached peripheral and not necessarily EP mode which is only supported in a relatively recent Linux kernel release by the manufacturer.

4.3.3 Round Trip Latency

The round trip latency is defined as the time taken from when a message is sent to when a response is received. Typically, a small packet size is used to mitigate the effects of bandwidth since a large packet will take more time to traverse the link. This type of test is aptly called a “ping-pong” test.

A similar methodology to the memory mapped copy throughput test in Section 4.3.2.2 was used. A program was written in C++ called “PCIe Echo” and it memory maps the PCI-Express controller memory region. The program is run on both the transmitter and receiver Wandboards. The receiver waits in an infinite loop to receive a four byte number from the transmitter after which it immediately responds with an incremented number. The transmitter records the time from when a number is sent across the PCIe connection to when a response is received.

The expected latencies are in the order of a microsecond and so a very precise timer is required on the SoC. The ping-pong test is repeated 100 times inside a loop which is timed. The elapsed time is then divided by the number of runs in the loop. This compensates for the fact that the high resolution Linux timer call,

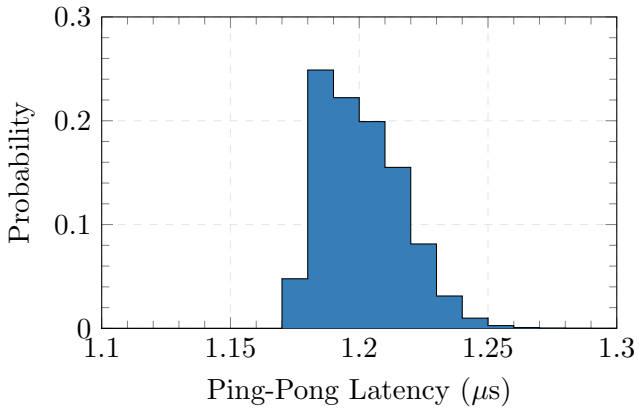


Figure 4.15: PCIe 4 byte packet ping-pong latency histogram, normalised to unity, from 1.1 μs to 1.3 μs.

gettimeofday(), takes a certain amount of time to execute and also is limited to a best case 1 μs resolution.

A histogram was constructed by running the ping-pong test one million times with 200 bins spaced evenly between 1 μs and 1.8 μs. Figure 4.15 shows the results. The mean round trip latency is 1.20 μs with median 1.19 μs. The approximately Gaussian distribution has a positive skew of 0.36. This tail makes sense as there is a physical limit to the minimum latency whereas the upper bound of the latency is more dependent on CPU processing delays and other unexpected processes.

The jitter of a network connection is defined as the variation of the latency. The standard deviation can be calculated from the histogram values shown in Figure 4.15 and is equal to $\sigma = 0.02\ \mu s$. The average jitter of the PCIe link is, therefore, 0.02 μs.

The PCI-Express specification allows for sub-microsecond latency but in practise this will be difficult to achieve because of CPU processing delays [74]. The memory mapped PCIe Echo test software will measure a slightly higher latency that the minimum because of the additional latency introduced by the mmap driver. If DMA was available then it is likely that the latency would be lower due to fewer memory copies than the current implementation.

4.3.4 Summary and Conclusion

The data throughput and round trip latency of a PCI-Express Gen-2 x1 link has been tested. Two Wandboard single board computers were linked via their PCI-Express interfaces using a custom designed and manufactured adapter board. The Wandboard is an open source design and is based on the Freescale i.MX6 quad core ARM Cortex-A9 SoC. One Wandboard was configured as a PCI-Express End Point and the other as a Root Complex to enable PCI-Express communications without the requirement of special drivers as would be required with so-called multi root connections described in Section 4.2.3.

The i.MX6 SoC's PCI-Express controller is relatively low-end compared to the controllers found on high-end server grade SoCs and as a result does not contain a DMA controller. The performance results obtained are comparable to similar specification devices and will be useful for specifying what is indeed required for a sROD co-processing unit.

The Linux kernel based memory copy between the two Wandboards using a modified PCI-Express device driver during the boot up process achieved read and write data throughputs of 93 and 284 MB/s respectively. The write throughput is less than the theoretical maximum 500 MB/s because of the bandwidth efficiency due to PCI-Express protocol overheads. The PCI-Express packet header requires approximately 24 bytes and the data payload is limited to 128 bytes by the PCI-Express controller. Unfortunately, the data payload is limited further by the memory copy function to only 32 bytes which is the L2 cache line size of the CPU.

The read data throughput is lower than the write throughput because of the underlying communication mechanism. A so-called PCI-Express memory read requires a read request packet before data can be received. The latency overhead of these packets results in a lower throughput.

A memory mapped data throughput test was performed in order to test the possible data throughput over a PCI-Express connection from a normal application without the requirement of a special device driver. It is expected that the throughput results be similar to the in kernel tests but with slightly higher overheads and hence lower throughputs.

The data type used dramatically affected the results. A loop was used to copy data from a location in RAM to the memory mapped location. The read and write data throughput for an *int* data type (4 bytes) was 49 and 36 MB/s respectively. A *long long* data type (8 bytes) achieved 182 and 123 MB/s read and write data throughput respectively. The read result for the *long long* data type is better than the in kernel test. This is likely due to better compiler optimisation than that attained inside the kernel and hence a larger read request size in the PCI-Express driver. The write performance is lower but this is due to the lack of caching on the memory map for write operations. With write caching enabled, the data was not regularly flushed to the PCI-Express controller and so performance was degraded.

A DMA test was performed by using the i.MX6 IPU as a DMA controller. This "hack" enabled the performance of a DMA style data copy to be measured without CPU interaction and is indicative of achievable performance with a proper DMA controller. Read and write data throughputs of 248 and 375 MB/s were attained when initiated by the Root Complex. Results for operations initiated by the End Point were slightly worse, probably due to less driver optimisation in the kernel.

A ping-pong round trip latency test was performed with one million round trips. A histogram of the time was plotted and is visible in Figure 4.15. The average round trip time is 1.20 μs with a standard deviation of 0.02 μs. This latency is superior to that achievable by gigabit and even 10 Gb/s Ethernet which is capable of approximately 17.9 and 7.1 μs respectively, as discussed in Section 4.2.1.

4.4 sROD - PU Interconnect Considerations

Based on the requirements and specifications of the sROD as discussed in Chapter 3, PCI-Express is indeed suitable for a sROD - PU interconnect in terms of latency. To achieve the required throughput, however, a high end PCI-Express link is required. Alternatively, several more basic links, such as the one that was tested, could be used in parallel. The actual implementation must be determined and in this section several possibilities are discussed. An alternative methodology to use 10 Gb/s Ethernet in a more generic sense is presented in Section 4.5, but this is not suitable in terms of latency for the sROD - PU interconnection.

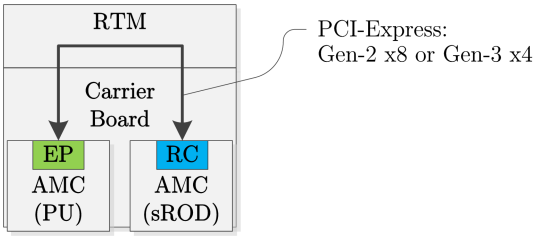


Figure 4.16: Diagram of connection between sROD and PU via ATCA RTM.

One of the specifications for the CERN ATLAS Phase-II upgrade is that front-end hardware such as the sROD will be housed in industry standard ATCA chassis [3]. An sROD co-processing unit must conform to this specification as well. A means must be found to interconnect the sROD and PU within the confines of a standard ATCA chassis, described in Section 3.4.2.

The PU could take the form of an Advanced Mezzanine Card (AMC) card similar to the sROD prototype, a Rear Transition Module (RTM) module or as a dedicated front board.

An issue with commonly available ATCA carrier boards is the lack of very high bandwidth I/O between AMC slots and between the AMC slots and the backplane fabric interconnect. The PICMG specifications, which govern the ATCA architecture, have provision for high bandwidth connectivity. A survey of the available carrier boards from two different manufacturers indicated, however, that the most common I/O between AMCs and from AMCs to the backplane is four or five 1 Gb/s Ethernet connections and up to four PCI-Express connections [80]–[82].

On the surface this seems suitable because of the four PCIe lanes which enable up to 2 GB/s bandwidth with Gen-2. Upon further investigation of the block diagrams, there is typically only one lane of PCIe between all four AMC slots and three lanes between neighbouring slots. This is unsuitable for a sROD - PU interconnection as more than one PCIe lane is required, even with higher bandwidth Gen-3 PCIe.

Each AMC slot has between four and seven dedicated connections to the RTM. Each of these connections can be used for a lane of PCI-Express. If a custom RTM loop-back card were to be designed then the AMC slot with the sROD prototype could be connected to the AMC slot with the PU. Figure 4.16 shows this graphically.

The sROD already has four connections to the RTM which can be used for PCIe [83]. From communications with the designer of the sROD, Fernando Carrió Argos, at CERN, there is space on the PCB to route more if required, assuming compatibility with the ATCA carrier board.

The PU could be implemented on the RTM board itself but then it would be difficult to use the standard ATCA monitoring and control functionality and it would lose the ability to function in a generic system without the sROD. A future front board implementation of the sROD can simply use the existing ATCA backplane for high data throughput communications between the sROD and PU. Both of these solutions would required additional board design time for standards compliance compared to an AMC card.

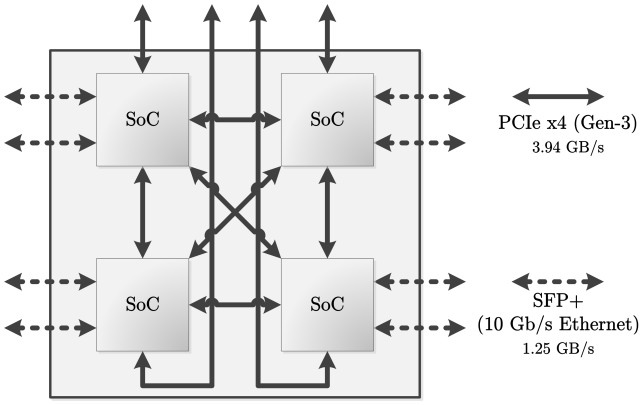


Figure 4.17: Diagram showing four server grade SoCs (X-Genes) interconnected with PCI-Express, with a total of 80 Gb/s Ethernet and 15.76 GB/s PCI-Express external connectivity.

4.5 Processing Unit Internal Design Considerations

The interconnection between the sROD and the PU is described in Section 4.4 and it is concluded that PCI-Express is the most suitable external I/O interface. There are several use cases where a dedicated PCI-Express link is not possible, however. It may transpire that the PU is located on a different ATCA front board or that the PU is used without the sROD in the high level trigger, for example.

In these cases, a more generic and industry standard external I/O interface may be required. 10 Gb/s or faster Ethernet is an option, along with other proprietary external I/O interfaces such as Infiniband [84]. It is assumed that very low latency operation will not be a primary concern in general purpose operation.

Depending on the performance of the SoCs, several may be required in a cluster configuration to increase the total processing power for complex algorithms. Multiple SoCs on a single board also increases the density of the system, lowering costs due to rack space requirements. Some components of the system are also common such as the power supplies and so costs may be reduced due to less duplication. The relatively low power requirements of ARM SoCs also allows for lower cooling capacity than conventional systems, making densely packed SoCs feasible.

On higher end, server grade ARM SoCs, 10 Gb/s Ethernet is available. In this case PCI-Express will be used as a low latency, high data throughput interconnect between SoCs on the PU and external I/O can take place via the available 10 Gb/s Ethernet connections. Figure 4.17 illustrates a configuration where both external Ethernet and PCI-Express is available, resulting in a PU that can be used as an sROD co-processor or in a standalone, generic application. The X-Gene is particularly suited for the configuration in Figure 4.18 because it has four Gen-3 x4 PCI-Express and two 10 Gb/s Ethernet interfaces.

If low cost mobile SoCs are used in the PU then the chance of the SoCs having built in 10 Gb/s Ethernet is low. A possible solution to this issue is to have a single high end SoC with 10 Gb/s or faster external I/O for external connectivity and an internal PCI-Express network between this “gateway” and the low cost “worker” SoCs. This single high end SoC fills the role of a gateway or router in networking terminology while maintaining the benefits of high parallelism, energy efficiency and low cost by using multiple mobile SoCs. The use of this solution strongly depends

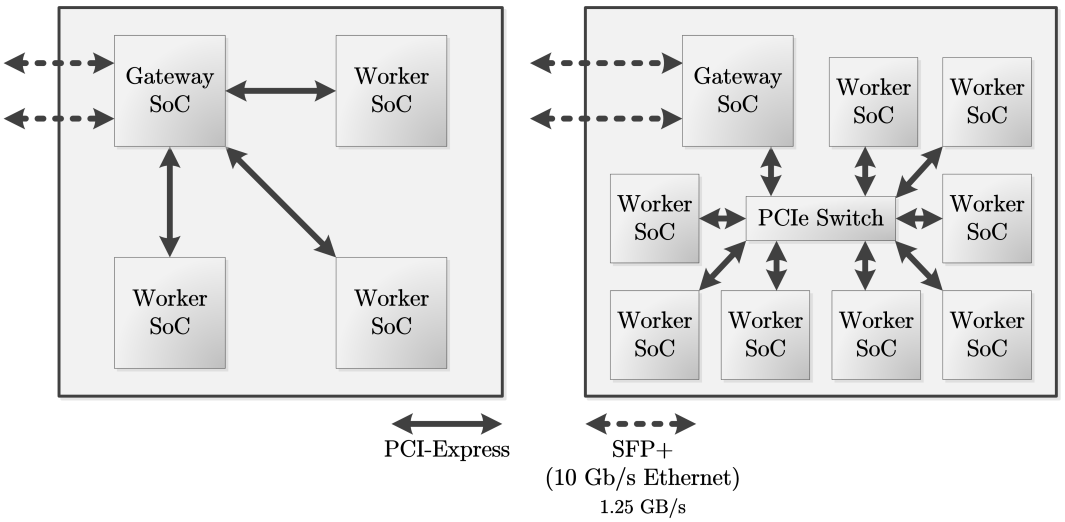


Figure 4.18: Diagram showing a server grade gateway SoC for high speed external connectivity and using PCI-Express for internal connectivity to low cost worker SoCs with and without a PCIe switch.

on the relative performance to cost ratio of low cost mobile versus higher cost server grade ARM SoCs.

Given that PCI-Express does not allow for inter-CPU communication in a standard manner, a software layer is required to facilitate this. It is assumed that the worker SoCs do indeed support EP mode operation for connection to the gateway SoC which will be the RC. In the case shown in Figure 4.17, one of the SoCs is arbitrarily chosen as the primary "node" and will be the RC. If the SoCs do not support EP operation, a non transparent PCI-Express switch can be used at the expense of a slightly more complex software configuration. It is henceforth assumed that the SoCs will support EP mode.

The most portable solution is to develop a custom Linux device driver which emulates a standard Ethernet interface to applications. Internally, each Ethernet packet will be reformatted with non-essential headers stripped out before the data is sent across the PCI-Express link to the appropriate SoC. The "Ethernet emulation" driver on the receiving system will then receive the PCI-Express packet, reformat it back to standard Ethernet and present it to the receiving application. With this solution, standard applications which support Ethernet such as Message Passing Interface (MPI) will be able to operate with little or no modification for the custom PCI-Express interconnect between CPUs.

If DMA is present on the PCI-Express interface then the implementation of this solution is relatively straight forward as the sender can write the data directly into the receiving Ethernet ring buffers as if it were a standard network interface card, for example. The data throughput of this solution is expected to be high. The custom driver can be extended to support standard Ethernet RDMA protocols allowing data to be written directly to the appropriate application memory space. Without DMA the driver becomes significantly more complex as the ability to move data between devices is more limited and requires multiple memory copies by the sending and receiving CPUs, harming performance.

5

ARM Computing Benchmarks

CPU compute performance, I/O throughput and latency (discussed in Chapter 4) as well as energy efficiency are all critical to the success of a sROD co-processing unit. This co-processing unit may also be used as a general purpose processing unit in a high data throughput computing system such as the ATLAS high level trigger or even other big science projects such as the Square Kilometre Array (SKA) in South Africa and Australia. In line with these requirements, this chapter is focussed on the CPU computational performance of several different ARM SoCs.

In order to determine the relative performance of ARM SoCs to other CPUs and systems, a process called benchmarking is used. There is a wide variety of benchmark software available, all suited to particular workloads and tasks. The output of a benchmark is typically a single number which can be compared between systems. In many cases this number is an actual measurement such as memory bandwidth or CPU Floating Point Operations per Second (FLOPS). In other cases the benchmark output is a dimensionless number which is calculated using an algorithm specific to that benchmark, but most importantly is still comparable to other runs of the same benchmark on other systems or configurations.

A synthetic benchmark is used to test a specific aspect of a system, as opposed to an application benchmark which tests an actual application's performance in a way that is comparable between systems and configurations. This chapter is focussed on synthetic benchmarks of memory and CPU performance. Chapter 6 is focussed on specific application benchmarks.

The Green500 list, which ranks supercomputers based on power efficiency (rather than pure performance like the Top500), indicates that computer performance has increased by a factor of 10 000 since 1992, while performance per Watt has increased by a factor of only 300 in the same period [85], [86]. As such, there is increasing interest in using ARM SoCs in server and cluster environments because of their energy efficiency and cost-effectiveness. A hypothetical 10 000 core datacenter based on SoCs with a 16 nm silicon process is estimated to be 26% cheaper than its conventional, discreet equivalent in terms of capital and running costs [52].

In a comparison between a cluster of ARM Cortex-A9 SoCs and an Intel Core2-Q9400 it was found that the ARM cluster had between 1.21 and 9.5 times the energy efficiency of the Intel system for web server, in-memory database and video transcoding workloads [87].

One of the early papers produced by the European Mont-Blanc project presents results for NVIDIA Tegra 2 and 3 (ARM Cortex-A9) SoCs compared to one of the first Cortex-A15 SoCs by Samsung, the Exynos 5250 and an Intel Core-i7 2760QM CPU [10]. In single core benchmarks with CPUs all set to 1 GHz, the Intel i7 was three times faster then the Tegra 2 Cortex-A9 and only twice as fast as the Cortex-A15.

The latest Mont-Blanc prototype will be a full cluster of 810 compute cards [72]. It is theoretically capable of 26 TFLOPS with 18 kW of power consumption. The majority of the processing performance comes from the ARM Mali-T604 General Purpose Graphics Processing Unit (GPGPU) embedded on the SoCs. This equates to approximately 1.4 GFLOPS/W.

Based on the literature it appears that ARM SoCs are a promising candidate for the sROD co-processing unit in terms of energy efficiency. However, due to the rapid improvement of ARM SoC performance the existing literature has become out dated and so the performance of recent ARM SoCs such as the Cortex-A7, Cortex-A15 and newer ARMv8 architecture cores such as the Cortex-A57 must be investigated. The aim of this chapter is, therefore, to answer the following questions:

- What are the synthetic benchmark results for a range of recent ARM SoCs to allow CPU and memory comparison with other systems?
- What are the energy efficiencies of recent ARM SoCs and how do they compare to other existing systems?

Chapter 4 provides overall details and photos of the different platforms that were tested but Table 5.1 details their CPU and memory specifications. Section 5.1 goes into detail on the synthetic CPU benchmarks that have been performed and Section 5.2 is similar except is dedicated to memory performance such as bandwidth and latency. Energy efficiency is presented in detail in Section 5.3.

Table 5.1: Specifications of the ARM development and evaluation platforms that were tested.

	A20	**i.MX6Q**	**Tegra-K1**	**Exynos 5410**	**X-Gene**
Manufacturer	AllWinner	Freescale	NVIDIA	Samsung	Applied Micro
Platform	Cubieboard A20	Wandboard Quad	NVIDIA Jetson	ODROID-XU+E	X-Gene Mustang
ARM Core	Cortex-A7	Cortex-A9	Cortex-A15	Cortex-A15 + A7[4]	X-Gene 1[2]
ARM Core Revision	r0p4	r2p2	r3p3	r3p2	r1
Feature Size (nm)	40	40	28	28	40
CPU Cores	2	4	4 + 1[4]	4 + 4[4]	8
Max. CPU Clock (MHz)	1008	996	2300	1600	2400
RAM (GiB)	1	2	2	2	16
RAM Capacity (GiB)	2	4	4	4	256
RAM Type	DDR3 1x32b	DDR3 2x32b	DDR3L 1x64b	DDR3 1x64b	DDR3 2x64b[3]
RAM Clock (MHz)	480	532	933	800	800
L1 Cache (kiB)[1]	32 / 32	32 / 32	32 / 32	32 / 32[3]	32 / 32[3]
L2 Cache (kiB)	256	1024	2048	2048[3]	1024[3]
L3 Cache (kiB)	0	0	0	0	8192[3]
FPU	VFPv4-D32	VFPv3-D32	VFPv4-D32	VFPv4-D32	VFPv4
SIMD (NEON) Registers[5]	16 x 128b	32 x 64b	16 x 128b	16 x 128b	32 x 128b
2014 Platform Retail (US$)	65	129	192	169	1495

[1] Instruction / Data cache.
[2] The X-Gene 1 is a custom core based on the ARM Cortex-A57 instruction set.
[3] Error Correcting Code (ECC) memory.
[4] Low power core(s) for light load conditions. ARM calls this big.LITTLE™ technology and NVIDIA 4-PLUS-1™.
[5] Registers x Total bit Width. 16 x 128b is 16 registers of 4 single precision numbers, for example.

5.1 CPU Performance

The performance of a CPU must be tested because it is difficult to accurately predict the true performance based solely on specifications. Factors such as the compiler used and the version thereof, the human effort expended to optimize the code, the algorithm used and also the size of the data to be processed all influence the performance of the CPU in question.

In the HPC field, the de-facto performance measure is High Performance LINPACK (HPL) [88]. Supercomputers are rated according to their FLOPS performance using HPL and the results are submitted to the TOP500 list [86]. As of November 2014, the highest performing computer on the list is Tianhe-2 which is developed by The National University of Defense Technology in China with a rating of 33.86 PFLOPS. Tianhe-2 consists of 3.12 million Intel Xeon E5-2692 and Xeon Phi 31S1P cores and consumes 17.808 MW of electricity which results in an efficiency of approximately 1.9 GFLOPS/W.

The HPL benchmark is CPU oriented and intentionally does not stress the memory and other subsystems of a computing system. This is accomplished with finely tuned algorithms for effective caching, to minimize memory related bottlenecks.

The relationship between floating point performance is not always proportional to the integer, or general performance of a CPU which is measured in MIPS [89]. The integer performance of a CPU is typically found using the Dhrystone benchmark which was developed in 1984 and is still in widespread use [90].

Dhrystone tests a broad range of program execution tasks such as pointer operations, assignments and procedure calls, however, there are several issues with the benchmark [91]. Modern compilers are able to optimise away many of the code paths in the benchmark which can skew the results from one machine to another. The CPU architecture also affects the result as the number of instructions per second on a Reduced Instruction Set Computing (RISC) CPU will be different to that of a Complex Instruction Set Computing (CISC) machine. This is why the result is usually reported in Dhrystone MIPS (DMIPS) per MHz as opposed to simply MIPS per MHz. Another issue with the Dhrystone benchmark is due to its small size: the code often fits into the execution cache of a CPU and so fetching and branch prediction operations may not be well tested.

To overcome these issues, a new benchmark has been created by EEMBC called CoreMark [92]. CoreMark is also recommended over Dhrystone by ARM [93]. CoreMark tests several common algorithmic workloads such as list processing, state machines, matrix manipulation and cyclic redundancy checks. The benchmark is also written in a way that prevents the compiler from optimizing away work by pre-calculating results as could sometimes be the case. Like Dhrystone, CoreMark reports the benchmark result as a single number enabling simple comparisons between systems but the parameters of the run such as compiler version, flags, threading and CPU clock must be specified for results to be published on the online database making fair comparison possible. The unit for the CoreMark benchmark is CoreMark per MHz or CoreMark per core.

Both HPL and CoreMark were run on the platforms in Table 5.1. The test methodology and results for CoreMark are described first in Section 5.1.1 and then the same is done for HPL in Section 5.1.2.

Table 5.2: Optimal gcc flags for the CoreMark benchmark.

	Flags
Cortex-A7	-O3 -mfpu=neon-vfpv4 -ffast-math
Cortex-A9	-O3 -mfpu=vfpv3 -ffast-math
Cortex-A15p2	-O3 -mfpu=neon-vfpv4 -mtune=cortex-a15 -ffast-math
Cortex-A15p3	-O3 -mfpu=neon-vfpv4 -mtune=cortex-a15 -ffast-math
X-Gene	-O3 -flto -fwhole-program -funroll-all-loops -floop-strip-mine -ftree-loop-if-convert -ftree-loop-if-convert-stores -mcpu=cortex-a57

5.1.1 CoreMark

CoreMark is a well known CPU performance benchmark which measures a CPUs integer and general purpose processing performance [92]. CoreMark outputs a single number that can be compared across platforms. It supersedes the older Dhrystone benchmark, which is still in use but is no longer recommended by ARM [91], [93]. This section will document the CoreMark results for single and multi-core tests of the platforms in Table 5.1.

It was found that the compiler version used to build CoreMark affected the results significantly. From the change logs of gcc, basic support for the ARM Cortex-A15 was only officially added in gcc 4.6. The gcc 4.7 compiler added support for Cortex-A7 and in gcc 4.8 there were optimisations added for the Cortex-A15. In gcc 4.8, AArch64 (ARMv8) support was first added. In gcc 4.9 there were further performance improvements to Cortex-A15 as well as the ARMv8 architecture CPUs.

Table 5.3 shows that, when normalised to 1 GHz, the difference between the gcc 4.8 and gcc 4.9 compilers is several percent. On the X-Gene the compiler version made a more significant difference than the other platforms. The ARM Cortex-A57 or ARMv8 architecture is very recent with relatively immature compiler support.

The open source community has not had much time with Cortex-A57 development boards in order to optimise the compilers. Applied Micro has their own compiler, APM 6.0.4, which produced higher performance executables than the standard gcc 4.8 compiler which it was based on. The latest version is APM 8.0.3, which is based on gcc 4.9.1. It was found that standard gcc 4.9.2 performed better than this customized version even though the APM 8.0.3 compiler provides a custom `-mcpu=xgene1` optimisation flag, which is not present yet in the mainline gcc compiler.

There was, unfortunately, no gcc 4.9 compiler available for the i.MX6Q SoC. Based on the results shown in Table 5.3, it may be assumed that given a gcc 4.9 compiler, the results may be several percent higher. It is, however, not likely that the results would be very different given the maturity of the ARM Cortex-A9 CPU.

The compiler flags used also affect the results. Many different flag combinations were tested and only the best were used for final testing. Table 5.2 also shows the optimal flags that were used with the gcc 4.9.2 and for the Cortex-A9 the gcc 4.8.3 compiler.

Benchmarks were run at each of the available CPU clock frequencies. A single core benchmark was used to ascertain the relative performance of the various ARM cores. A multi core test was used to find maximum performance of the SoC with results in Table 5.3. Figure 5.1 plots the results of the single core tests.

It is clear that both Cortex-A15s have identical performance even though they

Table 5.3: Comparison of CoreMark results, normalised to 1 GHz from the closest available CPU clock frequency with two versions of the gcc compiler. All CPU cores are used.

	gcc 4.8	**gcc 4.9.2**	**% Increase**
Cortex-A7	4819	4925	2.2
Cortex-A9	11555	-	-
Cortex-A15p2	14994	15692	4.5
Cortex-A15p3	15311	15614	2.0
X-Gene	24870	26254	5.6

are different core revisions. The Cortex-A15p2 appears to deviate from the other Cortex-A15 at lower clock frequencies. This is because the Samsung Exynos SoC utilises the ARM big.LITTLE architecture where there are two separate CPUs. The main CPU is a quad core Cortex-A15 for maximum performance and the secondary CPU is a quad core Cortex-A7 for power saving when lower performance is acceptable.

The switch-over mechanism in the Exynos SoC is simplistic and appears to be based on clock frequency. The Linux CPU frequency governors will automatically reduce the CPU clock under lower load and so this mechanism is effective. Unfortunately, it is not possible to disable the Cortex-A7 CPU in the Exynos 5410 and thus lower frequency Cortex-A15 benchmarks on the Exynos were not possible.

It is interesting to note that the Cortex-A7 of the Exynos outperforms both the Cortex-A15 of the Tegra SoC and also the Cortex-A7 of the A20 SoC. It is unknown why this is the case but the difference in performance between the two Cortex-A7s is likely due to the fact that the Exynos Cortex-A7 has 512 kiB of L2 cache memory whereas the A20 SoC only has 256 kiB of L2 cache. The RAM bandwidth of the Exynos, which is discussed in Section 5.2 is superior to that of the A20 which may also be a contributing factor but the nature of the CoreMark benchmark means that it's results are not highly reliant on RAM. Finally, the A20 is a very cheap SoC and is intended for extremely low cost devices. It is therefore possible that the SoC silicon is of suboptimal design.

The relative performance between the A20 and the i.MX6Q, Cortex-A9 SoC is as expected from the specifications.

Figure 5.2 shows the CoreMark results for when all cores are utilised. The scaling between single core and multi-core benchmarks appears to be linear. This is expected as there is no algorithmic interdependence between cores. The gradient of the curves represent the CoreMarks per MHz of each SoC.

CoreMark has an official online database which can be used to compare results between systems [92]. Figure 5.1 demonstrates that the CoreMark results are extremely linear with frequency and so a comparison of CoreMarks per MHz is fair. Table 5.4 shows this comparison between the ARM SoCs as well as some Intel x86 CPUs whose results are taken from the online database. The multi-threaded results were normalised to a single thread (or single core).

The X-Gene results are lower than the Cortex-A15 CPUs. This is surprising as it should have higher performance. It is possible that the compiler optimisation is still inferior. A pre-compiled version of CoreMark was supplied by Applied Micro which produced higher results, but since the source code was not provided the results cannot be trusted.

The CoreMark differences per core per MHz between the ARM CPUs and a

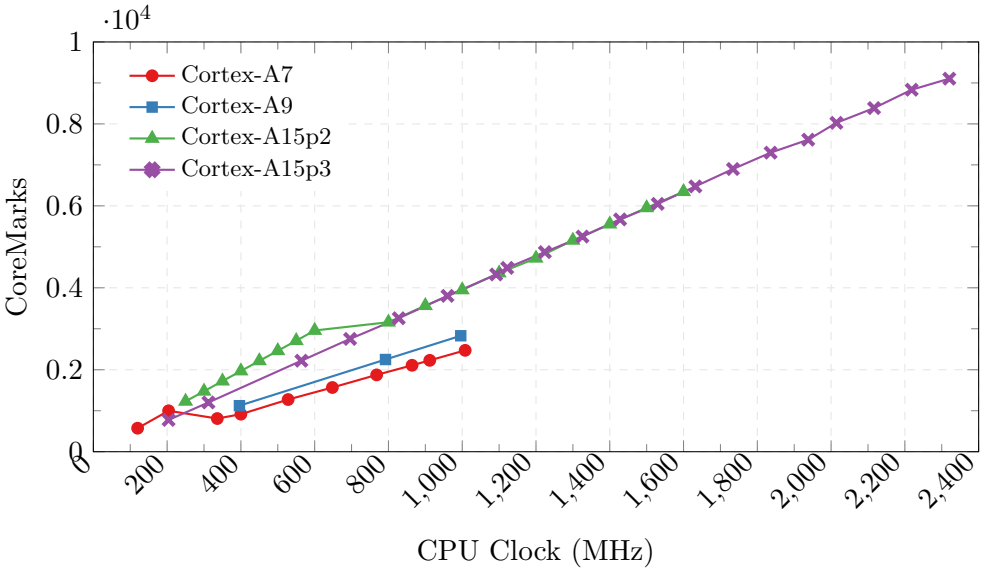


Figure 5.1: CoreMark results for a single CPU core sweep of the available clock frequencies.

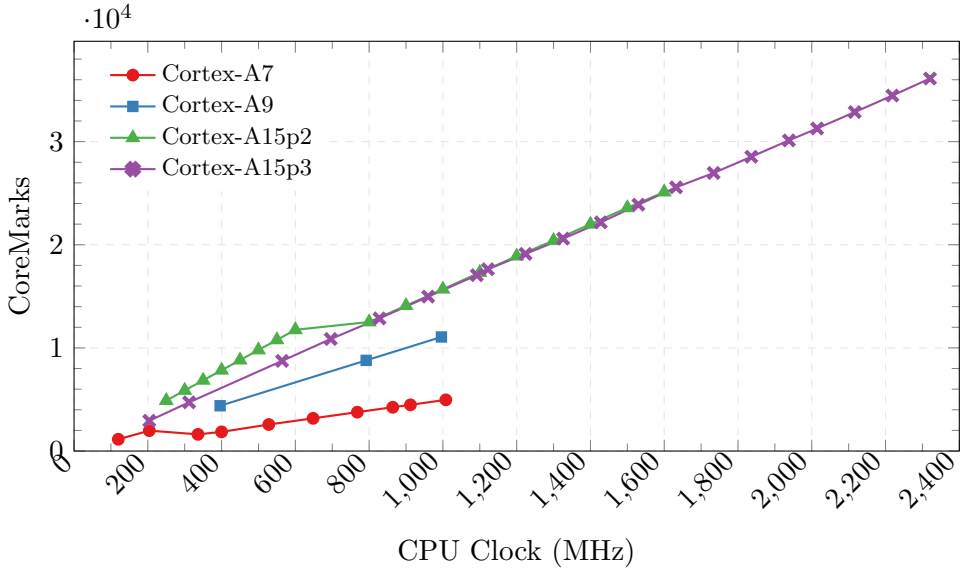


Figure 5.2: CoreMark results for a sweep of the available clock frequencies utilising all CPU cores.

variety of high performance Intel CPUs is not very large. Older ARM cores such as the Cortex-A9 are similar to older Intel Atom CPUs. Interestingly, the Cortex-A15 and 3^{rd} generation Intel i7 CPU have almost exactly the same integer performance based on these results. The Intel i7-4770 is a 4^{th} generation Intel i7 CPU and has superior performance to the ARM CPUs.

Intel CPUs typically have higher clock frequencies and more cores which is why their overall performance is higher than that of modern ARM SoCs in terms of integer performance. It is also likely that in real-world applications, the more sophisticated branch prediction, pipelining and other features of x86 CPUs leads to higher overall

Table 5.4: CoreMarks per MHz (one thread) comparison between ARM and Intel CPUs.

	CoreMark / MHz	**gcc**
Intel Atom N2800	2.28	4.7.2
Cortex-A7	2.46	4.9.2
Intel i7-860	2.49	4.9.2
Cortex-A9	2.89	4.8.3
X-Gene	3.28	4.9.2
Intel Core2 Q6600	3.81	4.5.2
Cortex-A15p3	3.89	4.9.2
Cortex-A15p2	3.92	4.9.2
Intel i7-3930K	3.93	4.4.6
Intel i7-4770	4.38	4.9.2

performance but the CoreMark benchmark is unable to provide evidence for this.

In Section 5.3, these CoreMark results will be compared in terms of power consumption.

5.1.2 High Performance Linpack

HPL stems from the older LINPACK Benchmark which was developed in the 1970s in Fortran [94]. The original LINPACK Benchmark does not support parallel processing which is why HPL was developed.

The benchmark solves a large, $n \times n$ system of linear equations in the form $Ax = B$ and records the time it takes to do so. LU decomposition with partial row pivoting is used. The algorithmic operation count of the calculation is taken to be $\frac{2}{3}n^3 + 2n^2$ and so it is possible to calculate the number of FLOPS [88].

HPL is dependant on a Basic Linear Algebra Subprograms (BLAS) library to run the underlying calculations. Several open and closed source BLAS libraries are available with widely varying performance and compatibilities. ATLAS (Automatically Tuned Linear Algebra Software, not to be confused with the LHC ATLAS detector) was chosen as the BLAS package for the benchmarks in this section [95].

ATLAS is one of the highest performance open source BLAS packages due to its auto-tuning nature and it also officially supports the ARM architecture contrary to many other high performance BLAS implementations which are typically architecture specific.

Both single and double precision floating point benchmarks were performed on the ARM platforms described in Section 4.1.1. Table 5.5 presents the results of the tests at the maximum frequency and utilising all of the available cores, which is the maximum attained performance of the SoCs given the test conditions.

The single precision HPL results are approximately two times the double precision results. This makes sense from a data point of view: a double precision floating point number requires eight bytes and a single requires four bytes. Twice as many single precision floating point numbers could fit into CPU cache or floating point unit registers as could double precision, so an increase in performance is expected.

On the contrary, the ARM SoCs have a NEON SIMD unit, which theoretically allows up to four single precision floating point results to be calculated simultaneously.

Table 5.5: Peak single (SP) and double precision (DP) HPL results for the ARM platforms. It was not possible to run the single precision benchmark on the X-Gene.

	CPU (MHz × Cores)	**SP HPL** (GFLOPS)	**DP HPL** (GFLOPS)	**Ratio** (SP ÷ DP)
Cortex-A7	1008 × 2	1.76	0.67	2.62
Cortex-A9	996 × 4	5.12	2.40	2.13
Cortex-A15p2	1600 × 4	15.90	8.34	1.91
Cortex-A15p3	2320 × 4	23.10	12.15	1.90
X-Gene	2400 × 8	-	26.31	-

Double precision floating point is not supported by NEON or NEONv2, found in the Cortex-A7 and A15 SoCs.

The difference in performance between single and double precision should, therefore, be on the order of four times and not only two times, as shown in the results. From Table 5.5, the Cortex-A7 achieved the most optimal SIMD optimisation with a ratio of 2.62 between the single and double precision results. The Cortex-A9 also achieved higher than two times SIMD optimisation but both Cortex-A15s were on the order of 1.9 times.

A possible explanation of this requires a short diversion to a discussion on processor micro-architecture. The Cortex-A7 is a relatively simple CPU which is "in order, partial dual issue" with an eight to ten stage pipeline, depending on the manufacturer. This roughly means that the CPU processes instructions (and naturally data) linearly but is sometimes able to decode, issue and execute two instructions simultaneously.

The Cortex-A15 is more complex than the Cortex-A7 and is an "out of order, sustained triple issue" CPU with a pipeline of 15 to 24 stages. This means that not only can the CPU decode, issue and execute three different instructions per clock cycle, but it can do so out of order. Sophisticated logic in the CPU decides whether it is faster to execute certain instructions before others, contrary to what may be in the machine code generated by the compiler, and then recombine the results later.

It is logical to assume that it is "easier" for the compiler to optimise for a simpler CPU micro-architecture than a complex one. Given time, these optimisations will improve. Based on this assumption, the gcc compiler is doing a better job at compiling SIMD code for the Cortex-A7 than for the more complex Cortex-A15 processor.

Figure 5.3 shows the double precision HPL results for all platforms except the X-Gene over a range of frequencies from the minimum to the maximum supported by each SoC respectively. A comparison between the HPL results in Figure 5.3 and the CoreMark results shown in Figure 5.2 show similar relative performance between the SoCs with some notable exceptions.

In CoreMark, at higher frequencies the performance of both Cortex-A15 SoCs converges to an identical value whereas with HPL the results are of the same order of magnitude but not identical. It is likely that the HPL performance differences between the Cortex-A15p2 and Cortex-A15p3 CPUs is simply due to the compiler producing slightly different executables. The difference in cache and memory performance as well as between the two revisions cannot be discounted, however.

The last data-point for the Cortex-A15p2 is lower than expected based on the

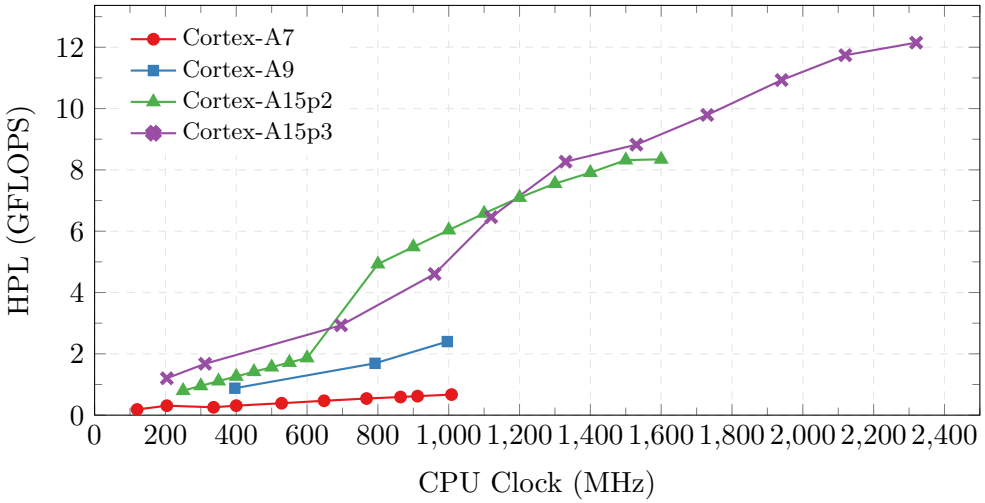


Figure 5.3: Peak double precision HPL results versus clock frequency. The X-Gene is not shown because changing the CPU clock was not possible.

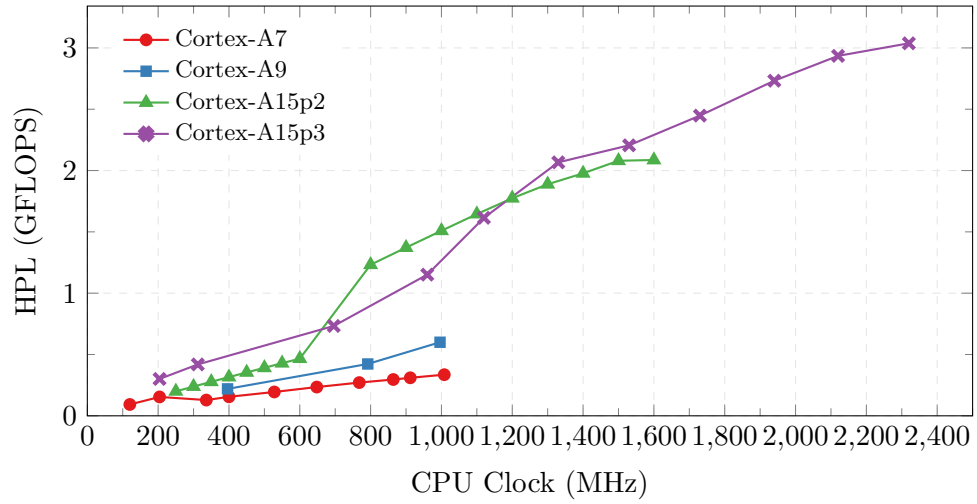


Figure 5.4: Peak double precision HPL results versus clock frequency, normalised to a single core. The X-Gene is not shown because changing the CPU clock was not possible.

approximately linear trend of the preceding measurements. It was found that the Exynos SoC overheats at 1.6 GHz when running HPL and automatically throttles the CPU clock down as a protection mechanism. Additional cooling was installed in an attempt to prevent this but the overheating issue persisted.

A significant difference is visible between the Cortex-A7 CPU on the Exynos SoC (the Cortex-A15p2) and the A20 SoC (Cortex-A7). If the results are normalised to a single core, shown in Figure 5.4, then there is still a clear yet unexpected separation of performance. It is interesting to note that even the i.MX6Q Cortex-A9 performance is inferior to the Cortex-A7 built-in to the Exynos SoC. This unexpected difference is also visible in the CoreMark benchmark described in Section 5.1.1, implying that it is not due to the benchmark or compiler optimisation.

In contrast to the anomalous CoreMark results, the Cortex-A7 performance

Table 5.6: CPU calculated GFLOPS per MHz per core.

	GFLOPS/MHz	**R^2**
Cortex-A7	0.0004	0.84
Cortex-A9	0.0006	0.97
Cortex-A15p2	0.0015	0.95
Cortex-A15p3	0.0014	0.98
X-Gene	0.0014	-

is indeed lower than the Cortex-A15 results. The Cortex-A7 is expected to be less efficient than the Cortex-A15 which has a more sophisticated processor micro-architecture.

A possible explanation for the strangely high performance of the Exynos Cortex-A7 lies in the memory performance of the respective SoCs, discussed in detail in Section 5.2. The Exynos SoC has approximately five times the peak cache and memory bandwidth of the A20. The Exynos also has five times the peak RAM bandwidth of the i.MX6Q SoC and more the double the cache bandwidth. Even though HPL is not a memory intensive benchmark, the total array size used by HPL is approximately 80% of the total RAM on the development platforms so there will be an impact.

The performance scaling of the measured CPU GFLOPS to CPU clock frequency appears to be linear, which is expected. A GFLOPS per MHz "rating" for each CPU would allow the results to be extrapolated to other SoCs. Table 5.6 shows the calculated GFLOPS per MHz based on the gradient of a line of best fit with the y-intercept forced to zero.

Multiple data points are not available for the X-Gene but is a y intercept of zero is used then the GFLOPS per MHz can be guessed. The R^2 "coefficient of determination" value of the Cortex-A7 is not as good as the other cores. If the first two data points of the available ten are removed then the R^2 value improves to 0.98 and the GFLOPS/MHz becomes 0.0003. Finally, the last data point of the Cortex-A15r2 was not used in the calculation because it is known to be inaccurate because of overheating.

In comparison, an Intel i7-4770 CPU which has four cores operating at 3.5 GHz achieves 0.013 GFLOPS/MHz per core, which is significantly more than the X-Gene or Cortex-A15 [96]. The high performance is due to advanced floating point SIMD capabilities such as SSE2, SSE3, SSE4, AVX and AVX-512 which are superior to ARM NEON, which is equivalent to SSE1.

5.2 Memory Performance

The memory subsystem of a SoC has several layered components for maximum performance, cost effectiveness and manufacturability. Figure 5.5 shows a typical memory hierarchy that is found on both ARM and Intel x86 CPUs.

The CPU itself has registers which are a form of memory but can only store several bytes at a time. These are used to store the data that a CPU is currently working on, or frequently accesses. One level up is the L1 cache. The L1 cache is very fast and can be accessed within a few clock cycles but due to it's fast nature it

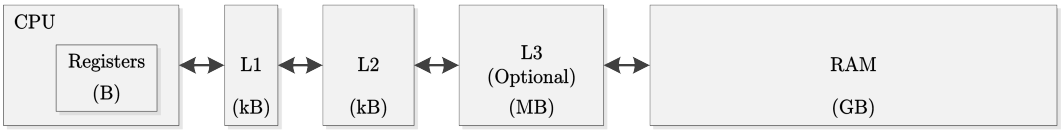


Figure 5.5: Typical memory hierarchy showing CPU registers, L1, L2, optional L3 caches and RAM as well as their relative capacities.

is difficult to install more than a few kilobytes per CPU. In high performance ARM and x86 architecture CPUs, the L1 cache is typically 32 kiB and sometimes 48 kiB. In most cases there are two L1 caches: a data cache and an instruction cache.

If the CPU requires data that is not present in its registers or L1 cache, it requests data from L2 cache. The L2 cache is larger than the L1 cache but takes more time to access. This access time is called latency. Typical sizes for the L2 cache are between 256 kiB and 2 MiB.

Finally, if the data that is required by the CPU is not in L2 cache, the memory controller requests the data from RAM which has relatively high access latency and lower bandwidth than the caches. In some higher-end CPUs an L3 cache is present. A typical compromise when an L3 cache is present is that the L2 cache is smaller. The L3 cache is usually several megabytes and also has a low access latency in comparison to RAM.

Larger caches lead to higher performance due to overall lower memory latency and high data throughput. With lower memory latency, the CPU will spend less time on average waiting for data and potentially not processing anything while it waits. Compilers do a very good job at optimising and reordering code in an attempt to keep a CPU busy and working efficiently. A CPU also has features to predict and pre-emptively pre-fetch data from higher cache levels or RAM, but sometimes these do not work as effectively as one might hope, resulting in lower performance.

The memory subsystems of the ARM platforms have been rigorously characterised. The benchmarks used as well as the results and analysis are presented in this section.

Two benchmark programs were used for memory testing: STREAM and lmbench. STREAM in an industry standard benchmark used to test peak sustainable memory data throughput [97]. It does not test random memory access or latencies. In light of this, a different benchmark must be used. The lmbench benchmark is a suite of benchmarks that have a wide range of objectives [98]. Many are used to test various aspect of the operating system but there are several benchmarks which test hardware.

The lmbench benchmark provides a platform for simple implementation of micro-benchmarks, which are known as kernels which usually test a very small and specific aspect of a system. The lmbench platform consists of a “timing harness” which is a comprehensive and accurate system used for measuring the length of time a kernel takes to run. Knowing the time and the number of runs, it’s possible to calculate a meaningful result.

5.2.1 Memory Bandwidth

The latest version of STREAM (5.10) was used to test all of the platforms. gcc 4.8 as well as gcc 4.9 was used to compile in order to ascertain any performance differences such as those seen in CoreMark in Section 5.1.1. The benchmark was run 100 times

Table 5.7: Operations performed and tested by the STREAM benchmark.

STREAM Test	Operation
Copy	`a[i] = b[i]`
Scale	`a[i] = scalar * b[i]`
Add	`a[i] = b[i] + c[i]`
Triad	`a[i] = b[i] + scalar * c[i]`

for each test and the best result was used. STREAM performs averaging within each run and so using the peak result can be used.

The documentation for STREAM recommends that an array size at east 4 times the size of the CPU L2 or L3 cache is used for testing. This ensures that the RAM and not the cache is tested. Array sizes from 4 kiB, which corresponds to 512 double precision floating point numbers which are 8 bytes each, to 64 MiB were tested.

It is possible to use STREAM to get an idea of the data throughput of the CPU caches as well, although the benchmark was not specifically designed for this purpose.

STREAM runs four different tests: Copy, Scale, Add and Triad. The Scale, Add and Triad use the CPU to perform some arithmetic on the data before assigning it to a new memory location. The code listing below shows the simplicity of the STREAM benchmark kernel which is run numerous times to increase the precision of the timing. Table 5.7 lists the operations performed by STREAM with pseudo-code.

Listing 5.1: STREAM kernel pseudo-code.

```
// Begin timer
for (int i = 0; i < STREAM_ARRAY_SIZE; i++)
{
        // Operation (Copy, Scale, Add or Triad)
}
// End timer
```

Figure 5.6 shows a STREAM copy test using a specified number of CPU cores, from 1 to the maximum available using the gcc 4.9 compiler. Figure 5.7 shows the same test except gcc 4.8 was used. The performance differences due to varying levels of compiler optimisation effectiveness are clearly visible for smaller array sizes. When the array sizes are sufficiently large, the RAM bandwidth becomes a bottleneck regardless of compiler optimisation.

The range of throughputs for the X-Gene Copy tests with large array sizes and different numbers of CPU cores is 4.9 GiB/s which is less than that of the Triad which is 9 GiB/s. A graph of Triad results is not shown. This result makes sense because the Triad requires more CPU work than the Copy. With more cores working on the data, more processing power is available and so the difference between a single core result and an 8 core result is more pronounced.

For the remaining STREAM tests of the other platforms, gcc 4.9 was used. The i.MX6 tests were performed with gcc 4.8 because gcc 4.9 was not available but the results would probably not have been very different due to the maturity of the Cortex-A9 CPU.

Figures 5.8, 5.9, 5.10, 5.11 and 5.12 show the summarised Copy, Scale, Add and Triad results when using the maximum number of CPU cores at maximum clock

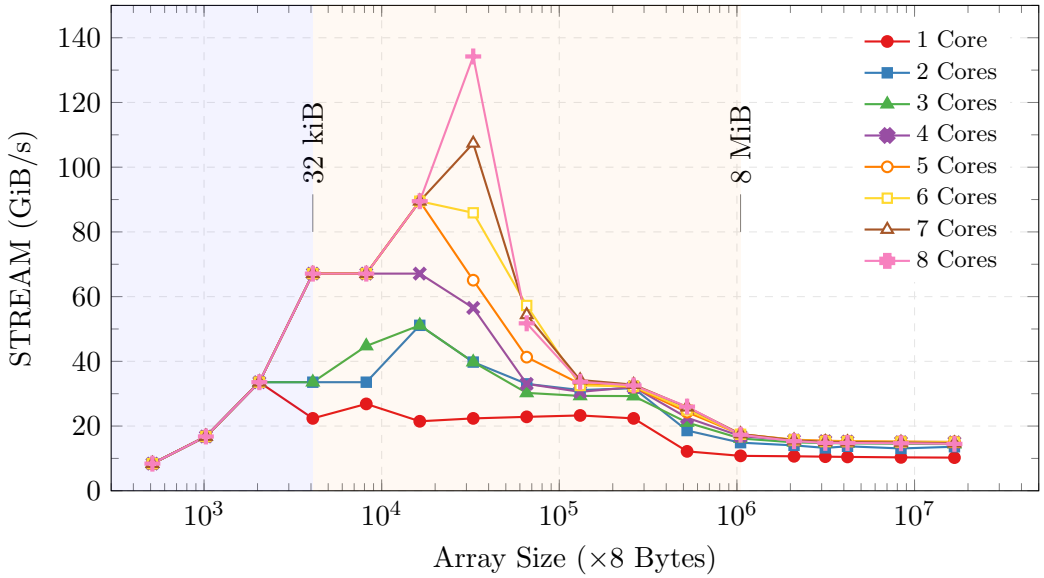


Figure 5.6: X-Gene STREAM Copy results in GiB/s for array sizes from 4 kiB to 128 MiB with gcc 4.9 and different numbers of CPU cores to demonstrate varying levels of optimization "success" in comparison to Figure 5.7.

frequency to minimise the effects of possible CPU bottlenecks when computing the Scale, Add and Triad. Table 5.8 summarises the RAM bandwidth for each platform.

It is clear that all the results converge to the maximum RAM bandwidth by 8 MiB. The L2 cache sizes of the various platforms is visible in Table 5.1. None of the platforms have more than 2 MiB of L2 cache. The X-Gene has an L3 cache which is 8 MiB and it's influence is clearly visible up to array sizes of about 4 MiB.

All of the platforms have 32 kiB of L1 cache. The influence of this cache is visible on all of the platforms but typically the L2 cache is where the very high data rates occur. This may be because of timing inaccuracies for very fast copies in light of the fact that STREAM is not specifically designed to measure such small array sizes.

In each results figure, the cache and RAM regions have been shaded for clarity. The increased performance shows up to half the size of the cache because at least two arrays are always used. For example, in the case of Copy, there is a source and a destination array.

The Copy results should be, and are indeed highest in most tests. The Copy test is the closest representation of the peak RAM bandwidth since no CPU work needs to be done on the data, besides moving it.

On average the memory bandwidth of the Cortex-A15 based SoCs (Exynos and Tegra-K1) is superior to that of the Cortex-A7 (A20) and Cortex-A9 (i.MX6). ARM has documented that the memory controller in the Cortex-A15 was enhanced and so this is expected [99].

The theoretical maximum RAM bandwidth (BW), known as the burst rate, can be calculated based on the specifications in Table 5.1 using Equation 5.1, below. The ratio between the actual bandwidth (or throughput) as tested by STREAM and the theoretical peak is known as the memory bandwidth efficiency and is a measure of how good the memory controller is. Higher bandwidth efficiencies are better. This efficiency can be used to compare the SoCs fairly, with results presented in Table 5.8.

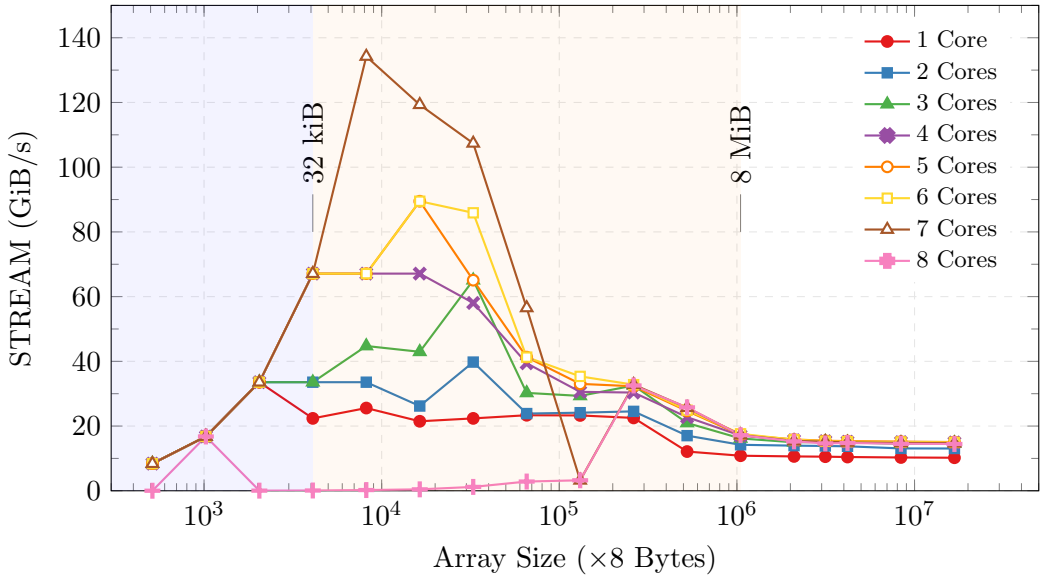


Figure 5.7: X-Gene STREAM copy results in GiB/s for array sizes from 4 kiB to 128 MiB using gcc 4.8 and different numbers of CPU cores to demonstrate varying levels of optimization "success" in comparison to Figure 5.6.

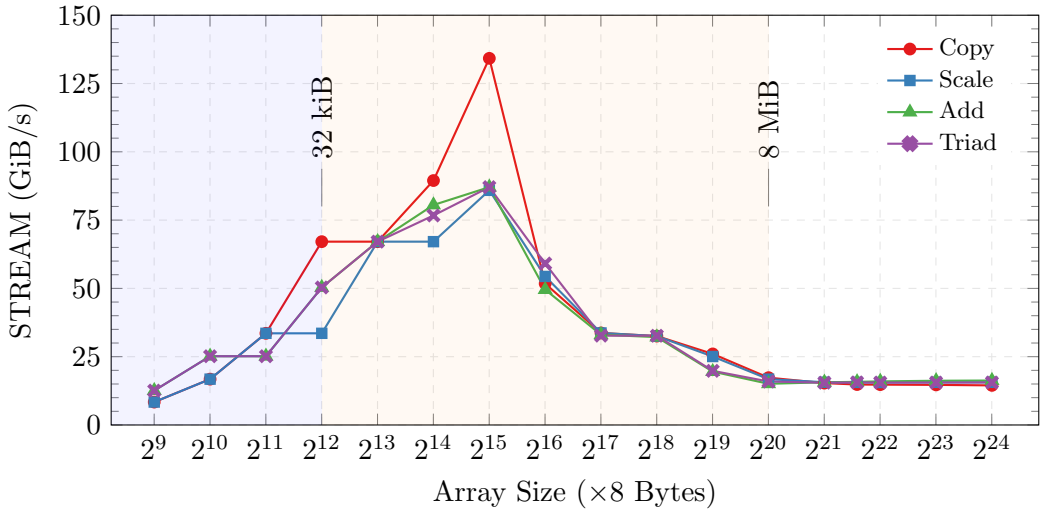


Figure 5.8: X-Gene STREAM results in MB/s for array sizes from 4 kiB to 128 MiB using 8 cores at 2.4 GHz.

$$BW_{peak} = d \times f \times w \times n \tag{5.1}$$

where f is the clock frequency of the memory interface in Hz, w is the bus width of the memory interface in bits, n is the number of channels and d is the number of data transfers per clock. For DDR (Double Data Rate) RAM, d is always two.

The version of DDR RAM, for example DDR, DDR2, DDR3 and even DDR4 does not affect this d value. The version is an indication of the physical technology and electrical specifications as opposed to the number of transfers per clock. Typically, higher DDR versions have lower cost, higher memory density per chip and lower

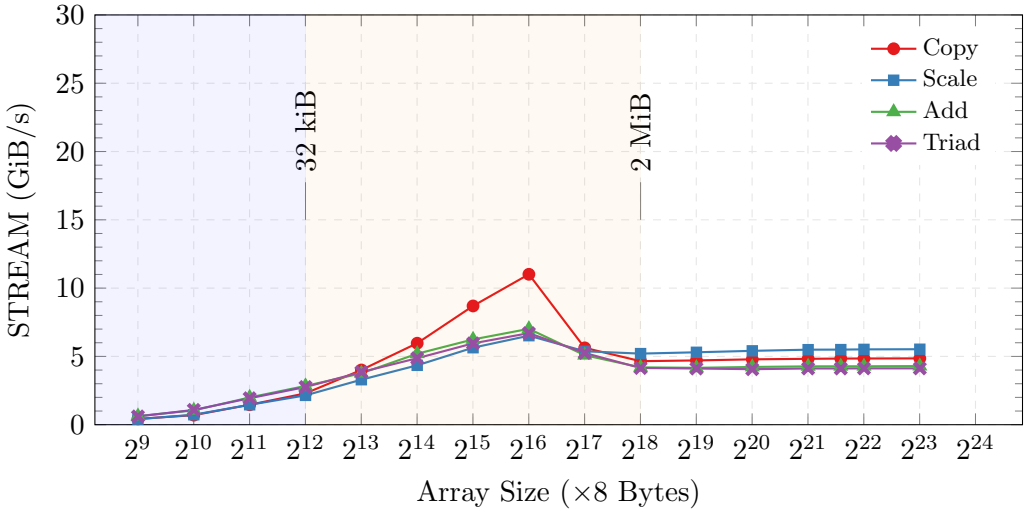


Figure 5.9: Tegra-K1 STREAM results in GiB/s for array sizes from 4 kiB to 64 MiB using 4 cores at 2.3 GHz.

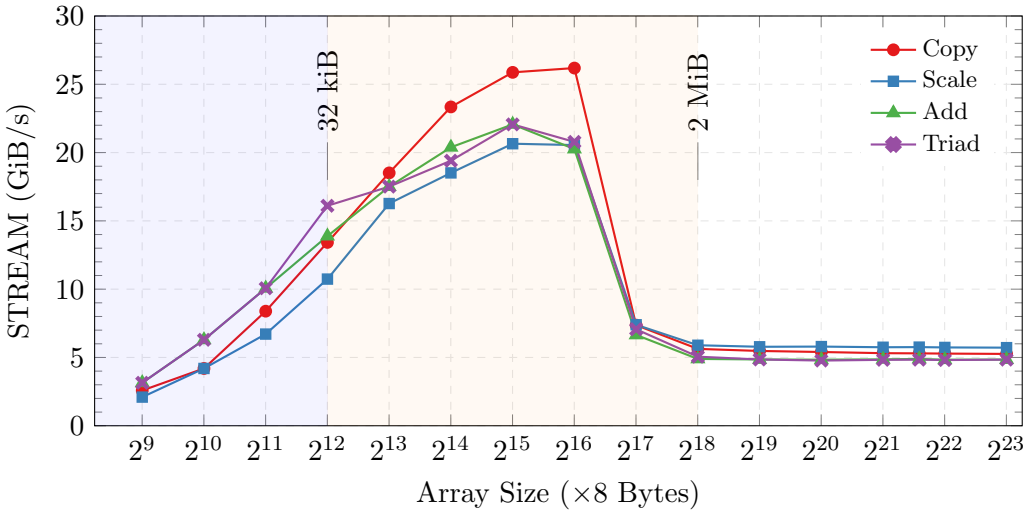


Figure 5.10: Exynos 5410 STREAM results in GiB/s for array sizes from 4 kiB to 64 MiB using 4 cores at 1.6 GHz.

energy consumption.

The unit for BW_{peak} is bits per second, which is normally converted into MB/s or MiB/s. Typically RAM bandwidth is reported in MB/s and not MiB/s. The reader is referred to the nomenclature for details on the difference between these two units but in summary *MB* indicates a megabyte that is 1000×1000 Bytes and with *MiB* the colloquial "megabyte" is 1024×1024 Bytes. The results from STREAM have been converted to MB/s before being inserted into Table 5.8.

Clearly the X-Gene's (Cortex-A57 based) memory controller is the best with an efficiency of 68.3%. This is expected as the X-Gene is a server grade SoC where good performance is expected. The Exynos SoC (Cortex-A15) is the second best with a memory bandwidth efficiency of 48%.

The Tegra-K1 (Cortex-A15) SoC appears to have a significantly lower efficiency

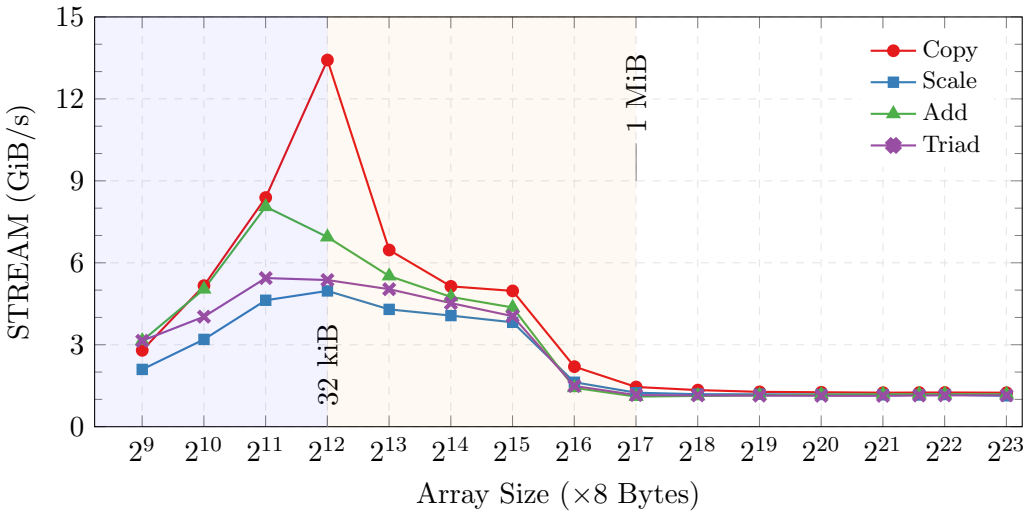


Figure 5.11: i.MX6 STREAM results in GiB/s for array sizes from 4 kiB to 64 MiB using 4 cores at 996 MHz.

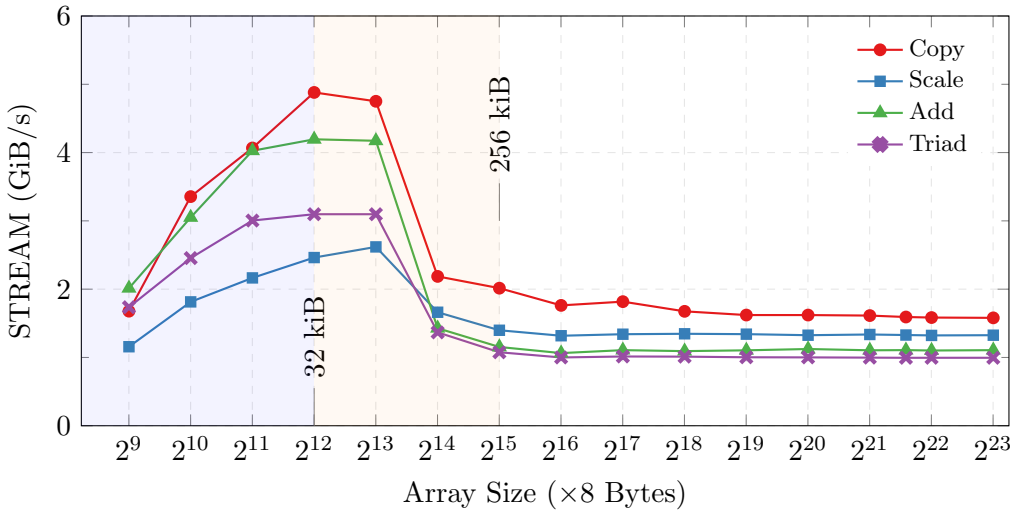


Figure 5.12: A20 (Cortex-A7) STREAM results in GiB/s for array sizes from 4 kiB to 64 MiB using 2 cores at 1.008 GHz.

than that of the Exynos, however the absolute bandwidth is only 3.5% lower. It is possible that although the Jetson-TK1 development board that is used to test the Tegra-K1 SoC is specified to use a 933 MHz memory interface, it may in fact be using an 800 MHz one which puts the efficiency on par with that of the Exynos SoC.

The efficiency of the A20 (Cortex-A7) is surprisingly good, at 44.3%, for the lowest cost SoC in this comparison. The i.MX6 has the worst efficiency at only 15.8%. This is surprisingly bad and may be due to incorrect operating system configuration. Several versions of Linux and bootloader have been installed and all lead to a similar result.

The Cortex-A7 and A15 processor cores have a more advanced memory controller than the Cortex-A9 and this may explain much of the difference. The differences between the memory controllers are clearer when comparing the latency and other

Table 5.8: Calculated theoretical peak performance of the ARM CPUs.

	Theoretical Max. (MB/s)	**Actual** (MB/s)	**BW Efficiency** (%)
A20	3840	1700	44.3
i.MX6	8512	1344	15.8
Exynos 5410	12800	6139	48.0
Tegra-K1	14928	5931	39.7
X-Gene	25600	17472	68.3

performance parameters.

5.2.2 Memory Latency

Memory latency has a large impact on processing performance. As described in Section 5.2, there are several layers or levels in what is knows as the memory hierarchy. Levels "closer" to the CPU have higher data throughput as shown in Section 5.2.1 and they also have lower latency, as presented in this section.

When a CPU requests data from a memory location, the memory controller fetches a chunk of memory which contains the requested memory address in the hope that surrounding data will also be used. If there are several computations that can be performed on a single chunk of data then the impact of the memory latency is decreased because fetching data can be done in parallel to processing it.

If only one byte of data is required per chunk, the latency will be the same as if all bytes inside the chunk are processed. Naturally, the overall efficiency of the processor increases if more data per chunk is processed.

The reordering of instructions to increase processing efficiency is one of the jobs of the compiler. Some CPUs also contain complex logic that attempts to execute different parts of the program out of order in what are called pipelines in the hope that there will be more efficient utilisation of cache as well as the benefits of parallel processing at the instruction level.

ARM CPUs typically have inferior branch prediction, out of order execution and shorter and fewer pipelines than modern x86 CPUs. The branch predictor is a sophisticated component of a CPU which attempts to guess which path a program will take, in the case of an "if-else" statement, for example, in order to pre-emptively fetch the next instructions or data. A correct branch predication leads to increases in processing performance and efficiency whereas an incorrect branch prediction carries a performance penalty. It is for this reason that ARM CPUs consume less power and require less silicon and are therefore cheaper. The downside of this is lower processing performance in some situations.

Contrary to STREAM, which uses the maximum result out of a number of runs, lmbench uses the median of 11 or more measurements [98]. The reason why a median is used is because typically the tests done produce many measurements clustered around a specific region with a few outliers which would skew the results if an average was used.

The latency results reported by lmbench version 3.0a9 are shown in Figure 5.13. The increasing latencies of L1, L2, L3 in the X-Gene and RAM are clearly visible as distinct plateaus. The average latency for each cache level and RAM are summarised in Table 5.9.

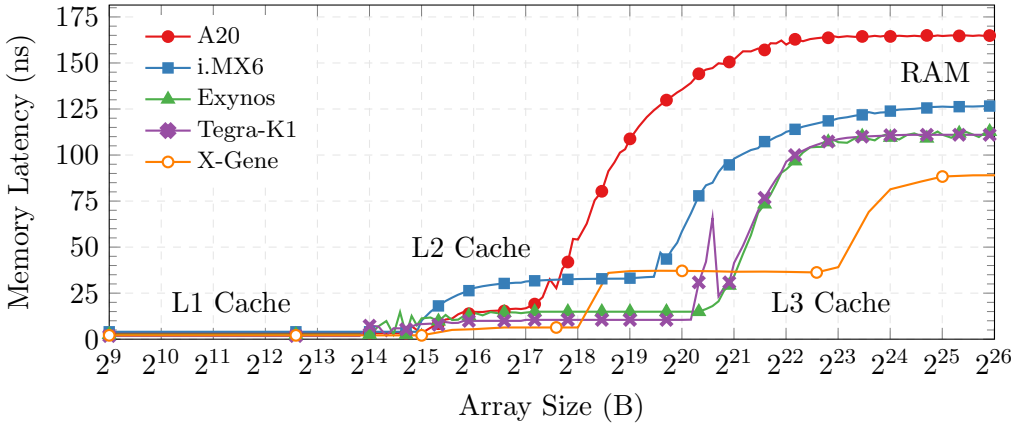


Figure 5.13: Memory latencies for all platforms from 512 B to 64 MiB. Differences between cache levels and RAM are clearly visible.

Table 5.9: Measured memory latencies of the ARM CPUs in nanoseconds using lmbench. Equivalent clock cycles are in brackets. Lower latencies are better.

	L1	**L2**	**L3**	**RAM**
A20	3.3 (3)	15.0 (15)	-	167 (167)
i.MX6	4.0 (4)	32.0 (32)	-	126 (126)
Exynos	2.5 (4)	14.9 (24)	-	111 (178)
Tegra-K1	1.7 (4)	10.5 (24)	-	111 (255)
X-Gene	2.1 (5)	6.4 (15)	37 (89)	89 (213)

The L1 cache latency is typically four clock cycles. It is important for the L1 cache to be quickly accessible by the CPU. The L2 cache results are more interesting. The Cortex-A9 (i.MX6) has nearly double the cache latency of the other platforms. Although they are all ARMv7-A architecture, the Cortex-A9 core is the oldest design and clearly significant improvements have been made in newer ARM processors.

The Tegra-K1 has the expected latencies for L1 and L2 cache, exactly like the Exynos. NVIDIA has most likely customized the memory controller of the Tegra-K1 SoC because it's performance is quite different to that of the Exynos. The RAM latency is significantly worse than that of the Exynos and other SoCs, at 255 CPU clock cycles. The X-Gene also has a relatively high RAM access latency at 213 CPU clock cycles.

In both cases, the increased latency is probably due to the increased complexity of the respective memory controllers. In the case of the Tegra-K1, the CPU and onboard general purpose Keplar GPU share the RAM. It may be the case that the logic required for this RAM sharing is adding a delay and hence increasing the RAM latency. The GPU has its own caches which is why no difference is seen on the L1 or L2 caches.

The X-Genes memory controller is also more complex than that of the other SoCs. Logic is required inside the memory controller to decide where the data is situated physically. Since the X-Gene memory controller is dual channel, supports very large amounts of RAM and has advanced virtualisation capabilities, it is possible that a delay is being introduced.

5.3 Energy Efficiency

The energy efficiency of a CPU is defined as the amount of processing that can be done per unit energy or more conveniently per Watt of power. A common measurement for a CPUs energy efficiency is GFLOPS per Watt but DMIPS per Watt is also sometimes used. As discussed in Section 5.1, the Dhrystone benchmark to measure DMIPS is no longer prevalent and so HPL, which measures a CPUs FLOPS is used.

It has been suggested that the standard GFLOPS/W metric is not suitable for HPC workloads because energy efficiency is of lesser concern in comparison to absolute performance [100]. Instead, where CPU performance should be weighted more that pure efficiency which is that case in most server type environments, $GFLOPS^2/W$ or even $GFLOPS^3/W$ are more appropriate. In terms of the sROD co-processor, GFLOPS/W is suitable with $GFLOPS^2/W$ becoming relevant in an HVC environment.

Historically, the widespread use of ARM SoCs in mobile devices is due to their low absolute power requirements, which are in the order of Watts as opposed to tens or even hundreds of Watts, which is vital to ensure longer battery life. Consumer demand for higher performance smart phones and other mobile devices is driving the performance of ARM CPUs to increase while the demand for longer battery life is forcing the power consumption of the SoCs to decrease or at least stay constant.

These pressures are leading to an increase in ARM CPU energy efficiency which makes them a possible alternative to conventional x86 based platforms. The massive market for smart mobile devices is also leading to more energy efficient x86 based CPUs and SoCs such as the Intel Atom and AMD Fusion but the focus of this section is on ARM based SoCs.

Several factors affect the energy efficiency of a CPU and these are discussed in Section 5.3.1. The measurement methodology applied to the ARM based platforms initially described in Section 4.1 is presented in Section 5.3.2 and results are presented and discussed in Section 5.3.3.

5.3.1 Factors that Affect Energy Efficiency

Typical factors that affect a CPUs absolute power consumption and energy efficiency are listed below. Further details on each item are subsequently discussed.

- The CPU architecture (or instruction set) such as ARMv7, ARMv8, x86-64, etc. plays a large part in the energy efficiency due to engineered design goals.
- The silicon feature size and process. Reducing the silicon feature size, for example from 40 nm to 28 nm, will reduce the power consumption at similar frequencies.
- Whether peripherals are built in to the SoC or the system is discrete.
- The minimum and maximum CPU clock frequencies and the ability to dynamically scale between them.

It is argued in [101] that the instruction set energy efficiency between ARM and x86, being examples of RISC and CISC instruction sets is in fact negligible and that the differences in energy efficiency are simply due to design decisions and implementations. Advances in compilers have effectively negated the performance

differences between RISC and CISC. The complexity and performance of the memory controllers, instruction decoders, branch prediction and floating point units, as well as the depth of processor pipelines make a large impact on the power consumption of a CPU, primarily due to the requirement of additional transistors and therefore a larger silicon area.

For these reasons, ARM CPUs consume less power than conventional x86 CPUs. The design goals behind x86 based CPUs have historically been high performance with little consideration to power consumption. ARM CPUs on the other hand have always been used in low power and often portable applications where performance is secondary to low power consumption.

The Intel Atom range of CPUs and recently SoCs is a competitor to ARM SoCs. Previous generations of Intel Atom CPU are inferior in both performance and energy efficiency to their ARM counterparts such as the Cortex-A9 [87], [101]–[103]. To the best of the authors knowledge, no benchmark literature is available on newer generation Intel Atom SoCs.

The use of System on Chips instead of discreet systems where the CPU and chipset are separate chips on a motherboard has led to significant savings in both cost and energy [103]. Large energy saving gains by SoCs are due to the ability to easily turn unused peripherals off when they are not required and having shorter high speed interconnects [104]. A 10000 core datacenter based on SoCs with a 16 nm silicon process is estimated to be 26% cheaper overall than its discreet equivalent [52].

$$P \propto cfV^2 \tag{5.2}$$

A common method to reduce a CPUs power consumption is by frequency scaling. Inherently, more energy is required to switch transistors at a higher frequency and can be characterised by Equation 5.2 where P is the power consumption, c is the gate capacitance of the transistor, f is the switching frequency and V is the threshold or switching voltage of the transistor [105]. By scaling the CPU clock frequency down when high performance is not required, power savings are achieved.

This technique is not always as effective as expected. For example under partial CPU loading, Intel SpeedStep technology is only able to provide a 20% energy saving at less than 40% CPU utilization [87].

Smaller silicon feature sizes and different processes affect the capacitance of the transistor gates and allow lower threshold voltages to be used. Higher clock frequencies are, therefore, attainable in a similar power dissipation envelope. Similarly the same clock frequencies can be used at reduced power consumption.

The step from a 40 nm process to a 28 nm approximately enables a 49% higher clock frequency, with a 44% decrease in energy per transistor switch and a 25 % reduction in transistor leakage current which primarily affects the power consumption of a CPU when it is idle [106].

Because of the significant advantages of more advanced processes, the silicon feature size must be considered when comparing different SoCs. In comparisons between ARM and Intel SoCs this is especially pertinent. Intel manufactures its own SoCs and is at the cutting edge of silicon technology, whereas ARM licenses CPU designs to other manufacturers, which may not have access to similar cutting edge manufacturing processes.

Because of this fact, it may indeed appear that the latest Intel Atom SoC is more energy efficient than the latest ARM based SoC. One may naturally assume,

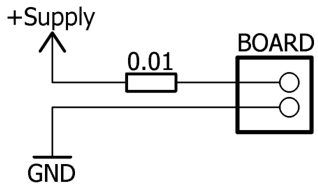


Figure 5.14: Simplified circuit diagram of the DC power measurement.

therefore, that the Intel architecture in question is more energy efficient than that of the ARM. The actual difference, however, may be that the Intel SoC uses a 22 nm process and the ARM SoC is using a 28 nm or even 40 nm process which would affect the relative power consumptions in a way that makes the SoCs difficult to compare fairly.

5.3.2 Measurement Methodology

In order to determine the GFLOPS per Watt rating of each platform, the electrical power input is measured while the HPL benchmark is running. The benchmark result is then simply divided by the power measurement.

The input power measurement is, unfortunately, not always straight forward. It is often impossible to gain access to the circuit board tracks to perform a direct measurement of the CPU power supply, which would have been ideal. Fortunately, the ODROID-XU+E (Exynos) platform was designed with this constraint in mind and has power measurement components located on the board for accurate measurements. None of the other platforms had this facility.

The DC power supply input was measured as opposed to the incoming AC to the power supply to avoid incorrect measurements due to the unknown conversion efficiency of the AC-DC power supply. In a further attempt to measure the CPU power as accurately as possible, peripherals such as Ethernet, audio and display were disconnected or disabled where possible.

Figure 5.14 shows a simplified circuit diagram of the measurement circuit. An oscilloscope or voltmeter is used to measure both the supply voltage as well as the voltage across a resistor in series with the power supply. Ohms Law is applied to the measured voltage across the resistor to find the current flowing through it and this is multiplied by the supply voltage to find the power, as shown in Equation 5.3, below:

$$P = \frac{V_{Supply} V_R}{R} \tag{5.3}$$

where V_{Supply} is typically 5 V, V_R is the voltage across the series resistor and R is the value of the resistor which in this case was 0.01 $\Omega \pm 1\%$.

The error introduced by the resistor tolerance is very small and in the typical worst case of a 15 W consumption with a 5 V supply, an error of 0.09 W is introduced. This is shown by Equation 5.4, below. In other cases where the power is less than 15 W, the error is less than 0.6 %.

$$P_R = I^2 R = (\frac{15 \text{ W}}{5 \text{ V}})^2 \times 0.01 = 0.09 \text{ W} \tag{5.4}$$

In the case of the X-Gene platform, a series resistor is impractical due to the multiple power supply voltages present on the ATX power supply. In this case, a

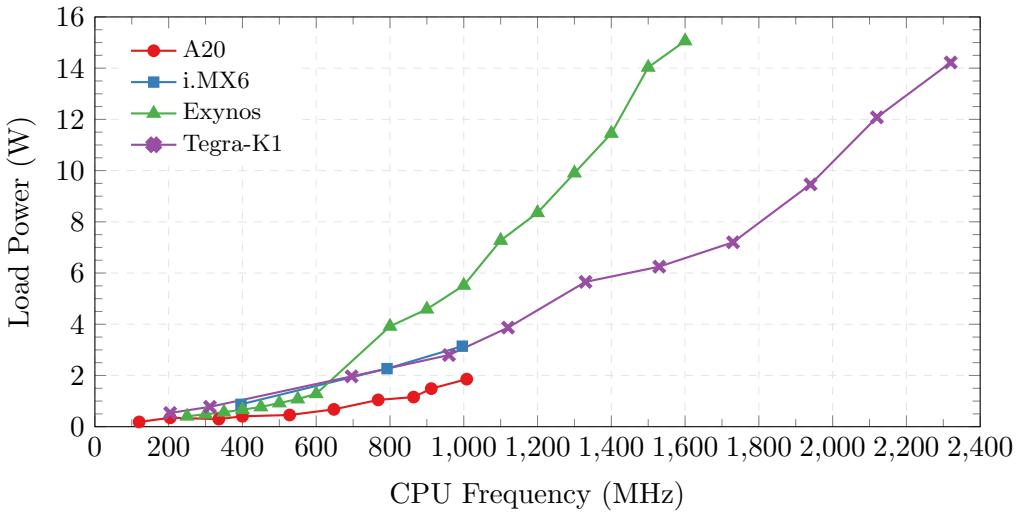


Figure 5.15: Load power of the platforms excluding the X-Gene (see text) for an all core HPL benchmark at varying frequencies.

DC current clamp meter with a 0.1 A resolution was used to measure the relevant power supply wires.

In the case of platforms where benchmark results are available but not power measurements, it is possible to use the SoC or CPU Thermal Design Power (TDP) as an indication of the power consumption under heavy operating conditions such as running a stressful benchmark. The TDP is used by designers when choosing appropriately sized cooling and power supplies but the TDP may be exceeded under synthetic conditions such as those induced by a "power virus", which is a tailored assembly code for maximum CPU utilisation [107].

5.3.3 Results

While the double precision HPL benchmarks from Section 5.1.2 were run, the power consumption by the boards was measured using the methodology described in Section 5.3.2.

Figure 5.15 shows the power consumption of all of the platforms, except the X-Gene, versus clock frequency. The operating system and Kernel installed on the X-Gene system does not support frequency scaling and so only a single measurement is available at the maximum frequency of 2.4 GHz. Table 5.10 summarises the power consumption of all of the platforms at maximum clock frequency as well as the GFLOPS/W.

From Figure 5.16 it is clear that at lower clock frequencies the energy efficiency of the SoCs is higher. The benefits of frequency scaling is clear, assuming the intended workload does not require maximum clock frequency.

This trend is why multi-core CPUs are now ubiquitous. As described in Section 5.3.1, increasing the CPU clock frequency requires a corresponding increase in voltage, which increases the power dissipation of the CPU. At a certain point, it is more energy efficient to add additional processors for increased performance as opposed to increasing the clock frequency further. Unfortunately this scaling does not always result in higher performance because applications must be carefully written to take advantage of the parallel processing abilities of a multi-core CPU.

Table 5.10: Power consumption and GFLOPS/W at maximum frequency for all platforms.

	ARM Core	Process	Power (W)	GFLOPS/W
A20	Cortex-A7	40 nm	1.9	0.35
i.MX6	Cortex-A9	40 nm	3.1	0.77
Exynos 5410	Cortex-A15r2	28 nm	15.1	0.55
Tegra-K1	Cortex-A15r3	28 nm	14.2	0.86
X-Gene	X-Gene	40 nm	63.9	0.41

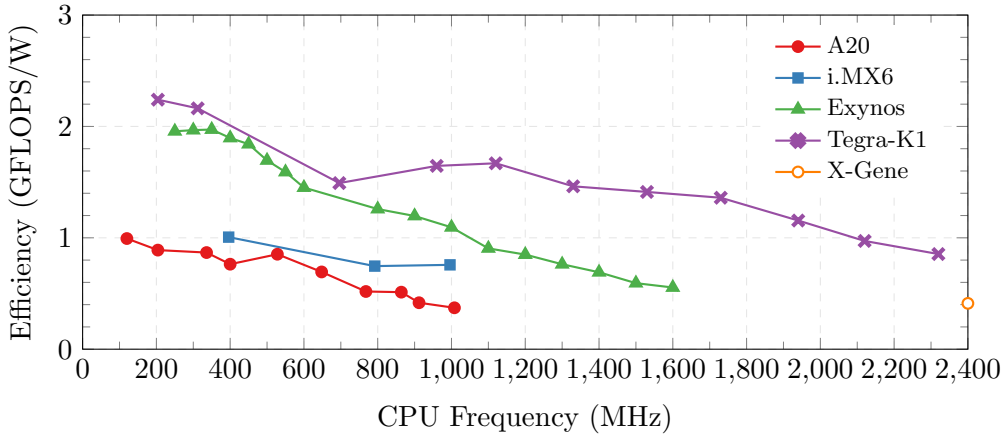


Figure 5.16: Power efficiency of the platforms at varying frequency with DP HPL.

In Figure 5.17, the results have been weighted towards processing performance versus energy efficiency ($GFLOPS^2/W$) which is sometimes more appropriate for compute intensive workloads [100]. At maximum operating frequency the efficiency is not at a maximum.

For comparison, the Intel i7-4770 CPU, which achieves 178.5 GFLOPS with four cores at 3.5 GHz has a TDP of 84 W. This results in an energy efficiency of 2.1 GFLOPS/W. This is almost double the energy efficiecny of the ARM SoCs.

The energy efficiency of integer performance based on CoreMark is higher on ARM [108]. The energy efficiency of the Exynos SoC is approximately 3600 CoreMarks/W where the Intel i7-3930K is only 1000 CoreMarks/W.

Based on these results and as a guideline when choosing a suitable SoC for a particular workload, the optimal energy efficiency will typically be achieved at a slightly lower frequency than maximum. If the performance of the SoC at this lower frequency is suitable for the application then optimal energy efficiency will be achieved.

Based on the presented results, the Tegra-K1 platform is best for computationally intensive workloads in terms of energy efficiency. Interestingly, the energy efficiency of the Exynos is quite different to that of the Tegra-K1 even though their silicon feature sizes are the same and both have quad core ARM Cortex-A15 CPUs.

In order to explain this discrepancy, the circuit diagrams and data-sheets of the Exynos and Tegra-K1 platforms were inspected. Both platforms have similar efficiency power regulators as well as similar external peripheral components. The

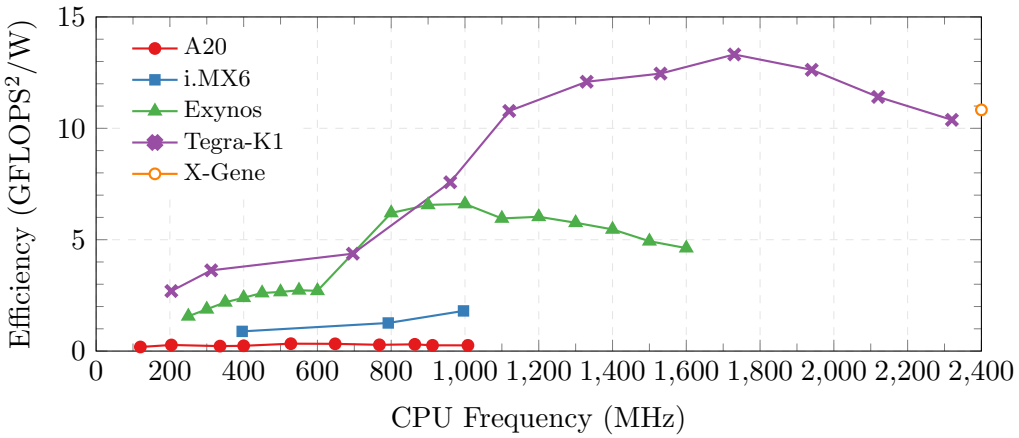


Figure 5.17: Weighted energy efficiency for more performance oriented workloads as suggested in [100].

resulting differences to the energy efficiency should, therefore, be negligible unless the difference lies within the SoC.

Both CPUs are Cortex-A15 but the Exynos 5410 is revision two (r2) and the Tegra-K1 is revision three (r3). According to the ARM Cortex-A15 technical reference manual, the r3 has numerous enhancements to clock gating and power management [99]. The ability to selectively turn off idle parts of the CPU with a higher granularity increases the potential for power savings. This is the most likely explanation for the difference in energy efficiency.

6

Application Benchmarks

In Chapter 5 the synthetic performance of the ARM platforms, initially introduced in Chapter 4, is presented. The memory latencies and bandwidth as well as the integer and floating point performance of the SoCs are presented.

These results are useful as a comparison between SoCs and other CPUs but it is difficult to use the results in order to predict actual application performance. The PU may be used as a sROD co-processor or as a stand alone general purpose computer. Two important applications are described and tested in this chapter: Fast Fourier Transforms (FFTs) and Optimal Filtering (OF).

The FFT finds use in many scientific processing algorithms such as "channelisation" in radio astronomy. The FFT benchmark results and interpretation in terms of data throughput is in Section 6.1.

The OF algorithm, described in detail in Chapter 3, is currently used by the TileCal for processing of raw data and will be implemented on the sROD. An implementation of the OF algorithm is described and tested in Section 6.2.

6.1 Fast Fourier Transforms

Fourier transforms have numerous uses in science, mathematics and engineering and are typically used to convert a function in the time domain into a function in the frequency domain. The Discrete Fourier Transform (DFT) is used when only discreet samples of a function are available, as is the case with digitised data on computers.

The DFT is defined by Equation 6.1.

$$X_k \stackrel{def}{=} \sum_{n=0}^{N-1} x_n . e^{-i2\pi kn/N}, k \in \mathbb{Z} \tag{6.1}$$

where $X_k \in \mathbb{C}$ is the DFT of the function x_n with N samples. The domains of k and n are $[0, 1, ..., N-1]$. The complex number X_k describes the amplitude and phase of a sinusoidal component of x_n whose frequency is k cycles per N samples.

The computational intensity of the DFT is $O(n^2)$ which is quite high. An algorithm for faster computation of the DFT called the FFT was first published in 1965 by Cooley and Tukey with $5N \log_2(N)$ operations [109]. There are no known

implementations of a size N FFT that require fewer than $O(N \log_2(N))$ operations but several are better than the Cooley-Tukey implementation [110].

Computational intensity aside, the FFT is also able to achieve higher numerical accuracy on a computer than a DFT because a lower operation count results in less numerical errors introduced by the use of finite precision floating point numbers.

There are numerous software implementations of various FFT algorithms available. FFTW is an open source, high performance FFT library which implements many different FFT algorithms in a unique way [111].

Before a FFT computation, FFTW tests many variations of different FFT algorithms in order to determine the fastest implementation on the computer that it is being run on. This "brute-force" approach ensures near-optimal performance on almost any CPU. This discovered configuration, called a "recipe", can be saved for use in future FFT computations.

FFTW has a benchmark facility which reports the time in microseconds, t, and the estimated FLOPS of a specific length N, FFT computation. The FLOPS reported by FFTW are estimated by assuming $5N \log_2(N)$ operations for the FFT, shown in Equation 6.2 [111].

$$\text{FLOPS} = \frac{5N \log_2(N)}{t} \tag{6.2}$$

One dimensional, complex number FFTs were tested. The tests were run for both single and double precision floating point. FFTW offers both an "in place" and "out of place" algorithm whereby the input data is either overwritten by the output data or a new output data array is created. The "out of place" algorithm was used in all test cases as this is the expected mode of operation assuming PCI-Express based data input and output using DMA.

The CPUs were set to their maximum operating frequency for testing. FFTW supports multi-threaded operation. Single core tests were performed as a base-line. Multi-core tests utilising up to the total available cores was also done. The test results and other pertinent information is given in Section 6.1.1.

It is feasible that in production, if smaller FFTs are required that do not require the full processing power of the available CPU, independent parallel data streams can be processed simultaneously. This case is tested with combinations of multi-threaded and multi-process runs with results and further details in Section 6.1.2.

In Section 6.1.3, the results obtained in prior sections are used to calculate a theoretical FFT data throughput.

6.1.1 Single and Multi-Core Benchmarks

FFTW is very flexible and can be configured in various ways. The configure script used for building accepts several options. In all cases, FFTW was built in as standard a way as possible. The configure command used is:

```
./configure --enable-float --enable-threads --with-slow-timer
```

Where `--enable-float` enables a single precision floating point version of FFTW, `--enable-threads` enables the use of multiple threads to utilise multiple CPU cores for the FFT caluclation and finally `--with-slow-timer` was used because of the lack of support in the Linux kernel for ARMs cycle counter.

The cycle counter is a timer that is accurate to a single CPU clock tick, which is excellent when testing the various FFTW recipes against each other. In the absence

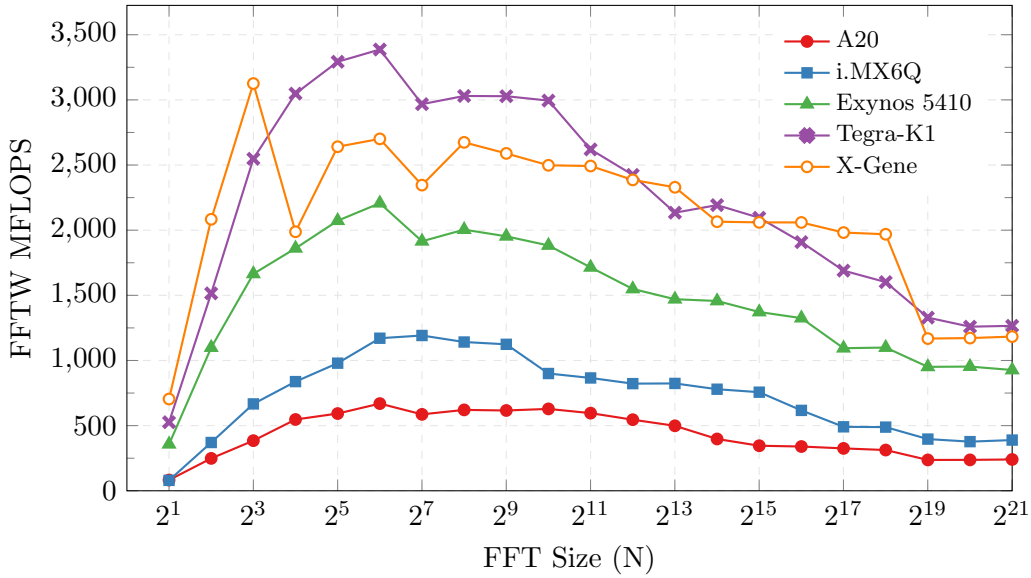


Figure 6.1: FFTW single core results at maximum CPU frequency for single precision complex FFT's from 2 to 2^{21}.

of the cycle counter, a normal timer is used which has microsecond accuracy. This means that the once-off setup time of an FFT is longer, but has very little adverse impact on performance.

FFTW officially supports NEON for SIMD on ARM processors by using the configure option `--enable-neon`. Unfortunately there were segmentation faults on all of the SoCs except for the Cortex-A9 which has an older floating point unit (VFPv3 versus VFPv4). This is assumed to be a bug in FFTW or gcc. It was encountered in both FFTW version 3.3.3 and 3.3.4. There may be a memory alignment issue or some hard-coded assembly that is not compatible.

Based on Figure 6.1, the relative performance between the i.MX6Q (Cortex-A9) SoC which had the `--enable-neon` flag set and the other SoCs which did not have the flag set is similar to that seen in the synthetic benchmarks. It is therefore assumed that the gcc vectorizer on the other SoCs did its job correctly without explicit FFTW support.

The performance of FFTW for smaller FFT sizes is lower than that of medium FFT sizes. This is because the overheads required to do the calculation such as memory latency are larger than the CPU time required to do the actual caluclation. FFT sizes for 8 up to approximately 1024 have the highest performance overall with a peak at 64. There appears to be a relatively sharp dropoff in performance between 2^{16} and 2^{18}. This can be explained in terms of the memory architecture of the SoCs.

The memory requirement of the FFT is described by Equation 6.3:

$$\text{Memory(Bytes)} = N \times 8 \times 2 = 16N \tag{6.3}$$

Where N is the size of the FFT. A single precision floating point number requires four bytes to represent, and since complex numbers are used, two floats must be used. Finally the factor of two is because an "out of place" algorithm is used: the output data is separate to the input data.

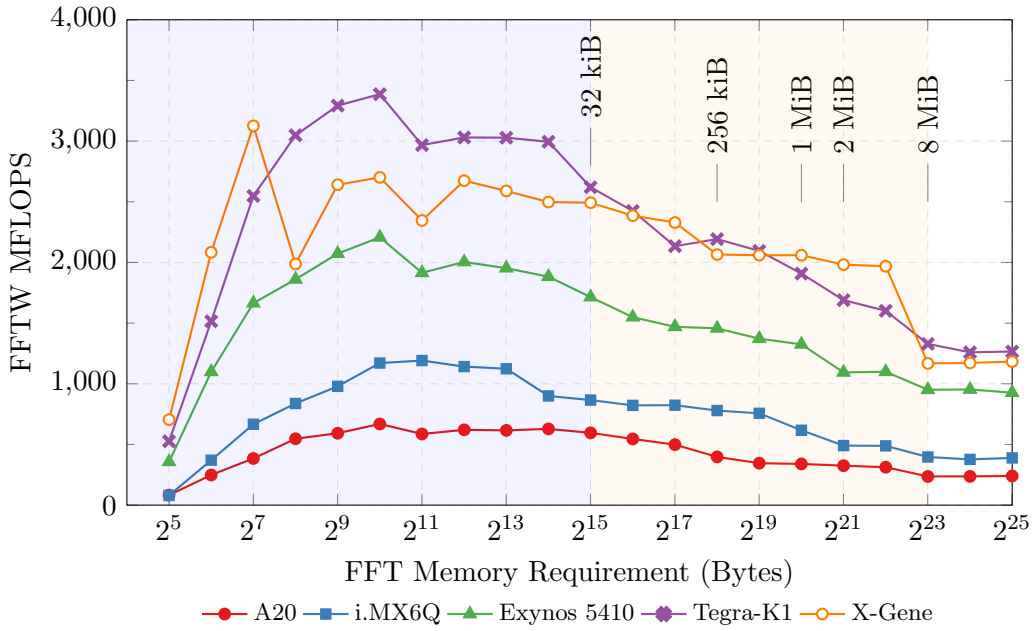


Figure 6.2: FFTW single core performance at maximum CPU frequency for single precision complex FFT's with $N \in [2 : 2^{21}]$. The shaded areas show L1 cache in blue and L2 and L3 caches in orange.

The x-axis of Figure 6.1 has been recalculated according to Equation 6.3 and is shown in Figure 6.2. The 32 kiB L1 cache of all the SoCs is shaded in blue. The performance drop when the required memory is greater than this cache is clear.

Orange shading has been used to cover the domain of the L2 and in the case of the X-Gene, L3 cache which is 8 MiB. The A20 and X-Gene have a 256 kiB L2 cache shared between pairs of CPUs and the i.MX6Q has a 1 MiB L2 cache. The Tegra-K1 and Exynos 5410 have 2 MiB L2 caches. In the X-Gene, the L3 cache is used for data that must be shared between CPUs that are not already linked by their L2 cache. The performance decrease is clearly visible when the FFT size approaches and surpasses the L2 and L3 cache sizes.

Figures 6.3, 6.4 and 6.5 show multi-threaded tests of FFTW, utilising multiple cores. In all cases there is a dramatic dip in performance at an FFT sizes from 2^6 to about 2^{11}. The reason for this is similar to the reason for the lower performance on very small FFTs. The overhead required to pass data between multiple cores is greater than the processing gain of having multiple cores.

For larger FFTs, the multi-core performance is better than that of a single core by a factor of three on average. The L3 cache of the X-Gene is clearly advantageous. The two, four and eight core results shown indicate dramatically higher performance than the other SoCs in the range of FFT sizes that fall into the 256 kiB to 8 MiB region.

The low performance of the Tegra-K1 for larger FFT sizes in comparison to the Exynos SoC which is also a Cortex-A15 is interesting. The Exynos has a clock frequency of 1.6 GHz and the Tegra-K1 is 2.3 GHz. The STREAM results presented in Section 5.2.1 indicate that the RAM throughput is similar but the cache performance of the Exynos is superior to that of the Tegra-K1.

The memory latency results in Section 5.2.2 also indicate that the RAM latency

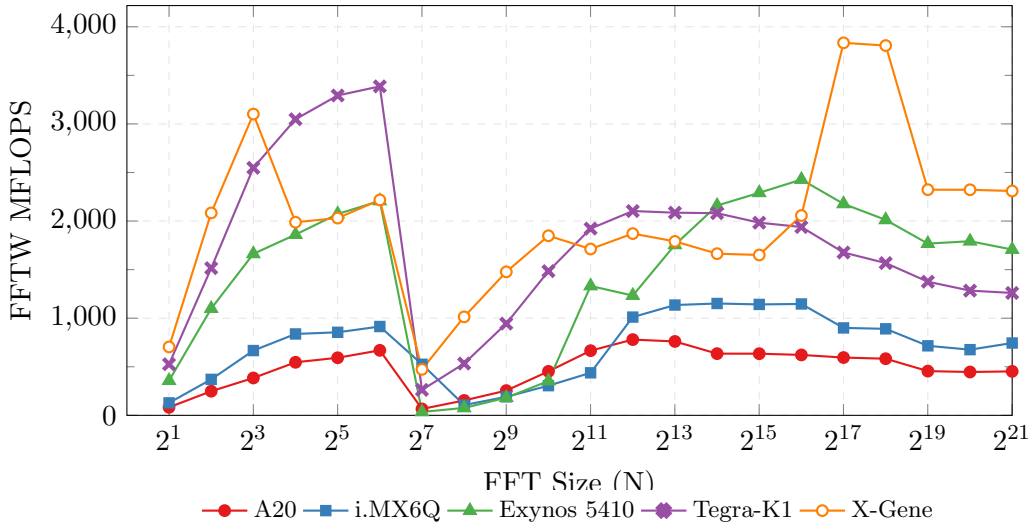


Figure 6.3: FFTW dual core results at maximum CPU frequency for single precision complex FFT's from 2 to 2^{21}.

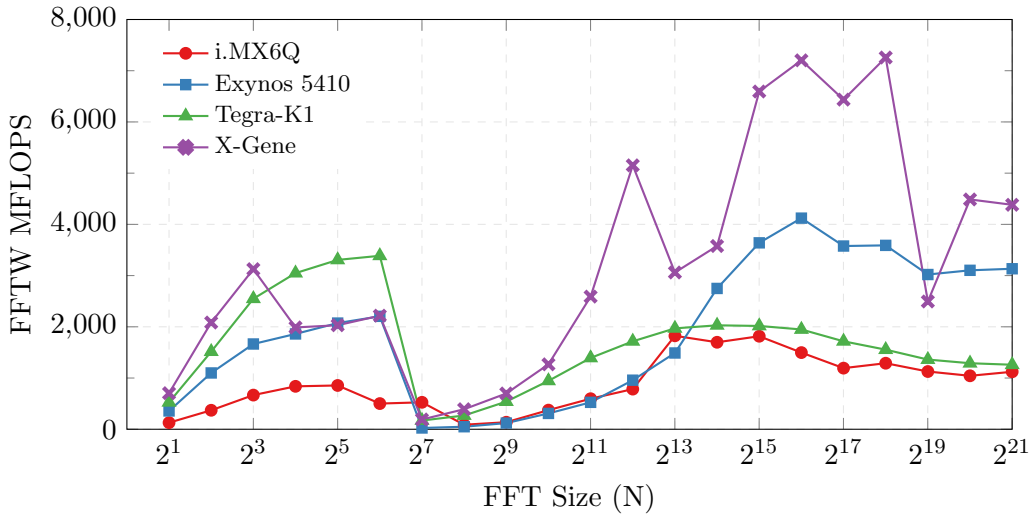


Figure 6.4: FFTW quad core results at maximum CPU frequency for single precision complex FFT's from 2 to 2^{21}.

of the Tegra-K1 is also inferior ot that of the Exynos. The FFT algorithm is very sensitive to memory latency due to the high interdependence of the data chunks within the algorithm. The DFT shown in Equation 6.1 demonstrates that at worst case, the first piece of data (x_0) is dependent on the last piece of data (x_{N-1}) and in the FFT this situation is similar. For a large FFT of several megabytes, the entire array cannot be stored in cache and so the latency of fetching pieces of the array will significantly impact performance.

In real use, the best case configuration for a specific FFT will be used. To this end, the maximum performance for each SoC has been determined from one to the total available cores and is shown in Figure 6.6 where the additional CPU cores and the sophisticated memory architecture of the X-Gene are clearly advantageous to larger FFTs.

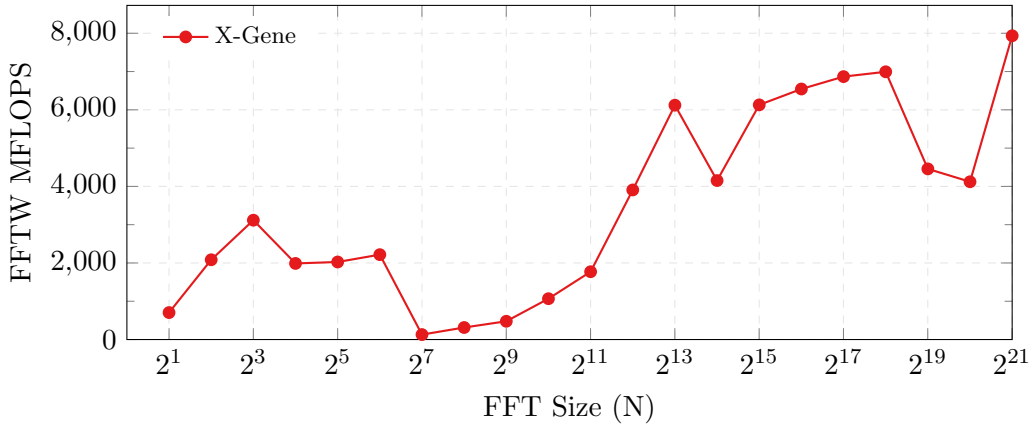


Figure 6.5: FFTW eight core results at maximum CPU frequency for single precision complex FFT's from 2 to 2^{21}.

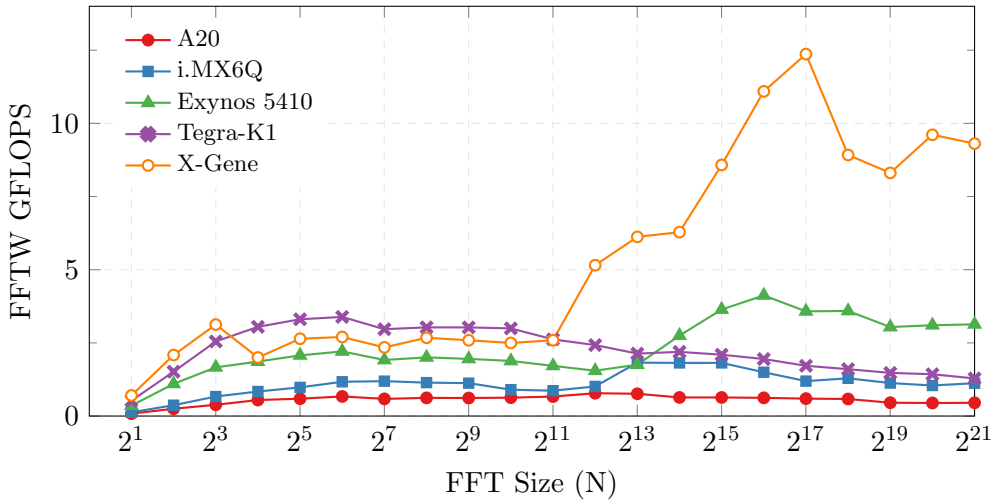


Figure 6.6: FFTW maximum results at maximum CPU frequency for single precision complex FFT's from 2 to 2^{21} with any number of cores.

6.1.2 Multi-Process Benchmarks

As mentioned in Section 6.1.1, in production use, the configuration of FFTW will be optimised. If large FFT calculations are required then it is likely that a single multi-threaded FFT calculation will yield optimal performance. In the case of smaller FFTs, it is possible to run multiple FFT calculations in parallel in the form of multiple processes in Linux.

The command line that was used to execute multiple processes of the FFTW benchmark program in parallel is:

```
./bench oc1024 & ./bench oc1024; wait;
```

The "&" character indicates to the terminal that the following executable command must be run simultaneously. It is possible to chain as many processes in this way as is desired, but if there are less processes than the number of CPU cores then the Linux scheduler will place processes on separate cores. A "wait" command is used at the end which indicates to the system that it must wait for both processes to finish

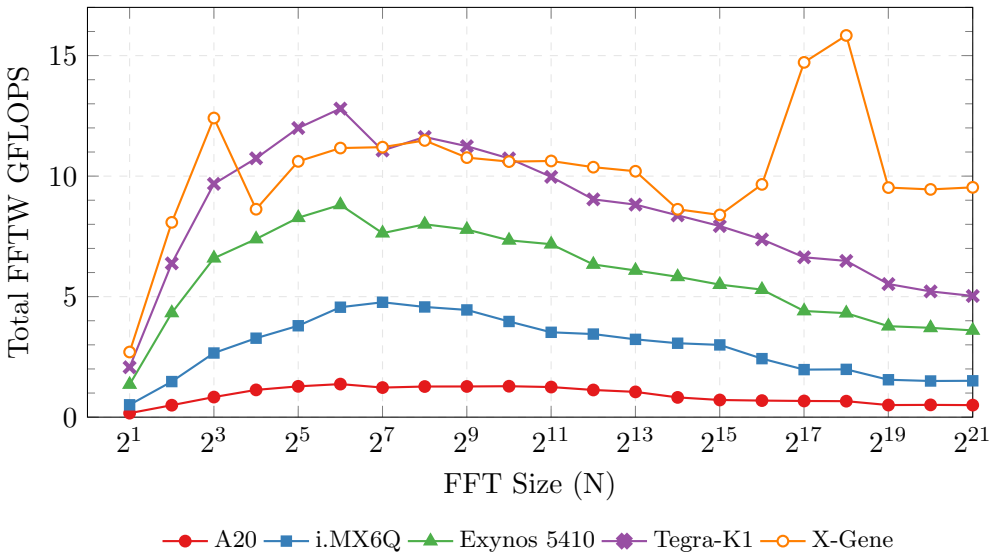


Figure 6.7: FFTW maximum multi-process equivalent FLOPS at maximum CPU frequency for single precision complex FFT's from 2 to 2^{21}.

before proceeding with other commands. In the above case, two simultaneous out of place complex FFTs with size 1024 will be computed.

All permutations of multi-process runs were tested. On the A20 SoC, which is dual core, the only multi-process run possible is two single core FFTs simultaneously. On the quad core SoCs there are more options: four single core processes or two dual core processes. On the X-Gene, which has eight cores, the third possible permutation is two quad core processes.

The sum of the FLOPS for each process is taken for each run as an indication of the total work done. Since FLOPS are calculated from the benchmark duration, the total FLOPS are calculated as shown in Equation 6.4. In Figure 6.7, the maximum of the FLOPS sums for the different permutations are shown. The increased utilisation of the SoC using this method is clear in comparison to Figure 6.6.

$$\text{FLOPS} = \frac{5N\log_2(N)}{\frac{t}{k}} = \frac{5kN\log_2(N)}{t} \tag{6.4}$$

Where N is the size of the FFT, t is the time duration of the calculation and k is the number of simultaneous processes.

The results shown in Figure 6.7 have been combined with Figure 6.6 to find the peak possible FFTW FLOPS for both multi-threaded and multi-process operation, shown in Figure 6.8. The combined results are almost identical to the pure multi-process results except for some data points at larger FFT sizes as is expected. For larger FFT sizes that those tested this trend is expected to continue, with a multi-threaded approach being better than a multi-process approach.

6.1.3 Theoretical FFT Throughput

In the sROD cop-processing unit, FFTs may be used to determine spectral characteristics of the raw data which may aid in determining levels of pile up and algorithm performance. Similarly, in an application such as channelisation in radio astronomy, if an FFT is used, it will be required to compute the FFT on a stream of data [112].

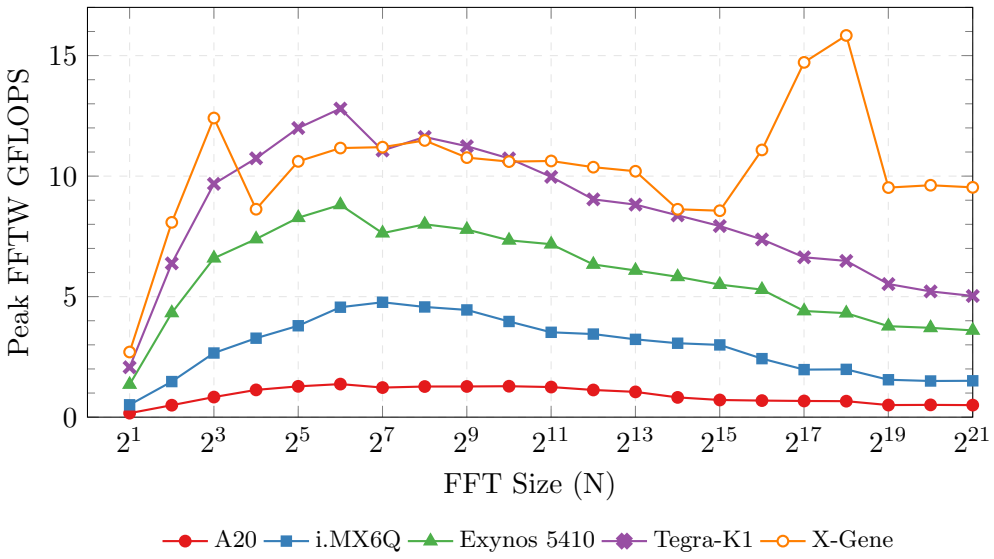


Figure 6.8: FFTW peak multi-core and multi-process equivalent FLOPS at maximum CPU frequency for single precision complex FFT's from 2 to 2^{21}.

In the FFTW benchmarks performed, the size and run time of the various FFTs in known. If it is assumed that there are no bottlenecks in a system such as external I/O then the theoretical maximum FFT data throughput can be calculated based on the FFTW results.

$$\text{Throughput(Bytes/s)} = \frac{8N}{t} \tag{6.5}$$

Where N is the size of the FFT, t is the calculation time and the constant, 16, is the number of bytes per N, described by Equation 6.3.

Figure 6.9 shows the results of this calculation for half of the total data-rate (for example for receive only). For smaller FFTs, the possible data throughput is very large. In reality, the FFT sizes required may be larger and so the results for larger FFT sizes are still interesting.

The X-Gene performs similarly to the Tegra-K1 for most FFT sizes. This is odd because the X-Gene (similar to Cortex-A57) is supposed to be a higher performance architecture than the Cortex-A15. As is the case in the synthetic benchmarks presented in Chapter 5, the lower than expected performance is likely due to gcc compiler immaturity. At larger N, however, the differences become clear and this is simply due to the fact that the X-Gene has eight cores and the other CPUs have four.

There is a peak in performance at $N = 2^{18}$ which corresponds to the 256 kiB point. The X-Gene architecture ties the L2 caches together via the L3 cache and pairs of CPUs via L2 which is 256 kiB. It is likely that the inherent way that the FFT algorithm works is more efficient with this topology. The data throughput at higher N is almost double the other CPUs which is expected, but is still only approximately 1 GB/s at $N = 2^{21}$.

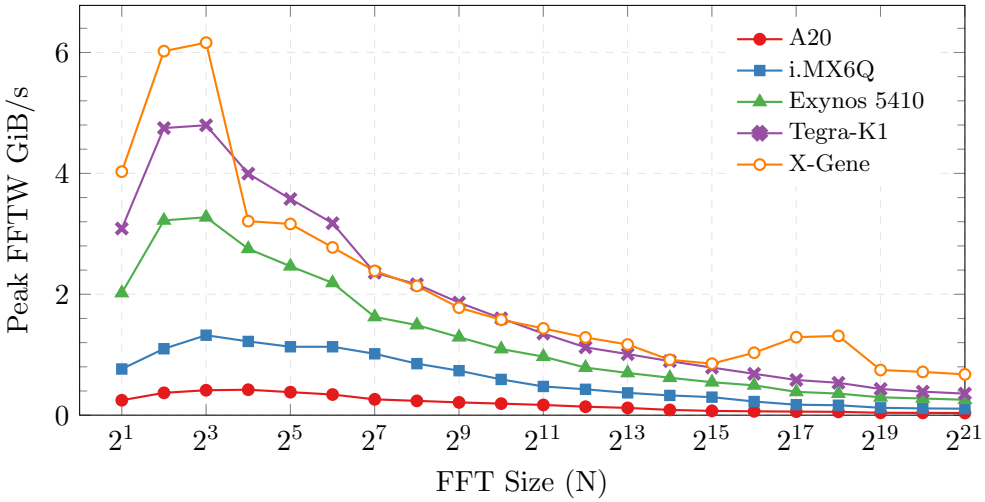


Figure 6.9: FFTW peak multi-core and multi-process equivalent FLOPS at maximum CPU frequency for single precision complex FFT's from 2 to 2^{21}.

6.2 Optimal Filtering

The Optimal Filtering (OF) algorithm described in Section 3.1 in Chapter 3 is used to "translate" the seven noisy ADC samples of each filtered TileCal PMT pulse into three numbers. The amplitude (A) and phase shift (τ) of the pulse peak from the third sample is calculated by using two separate weighted sums, effectively dot products, repeated in Equation 6.6 and 6.7, below. These three numbers and their relationship to a filtered PMT pulse are illustrated in Figure 6.10.

$$A = \sum_{i=1}^{n} a_i S_i \tag{6.6}$$

$$\tau = \frac{1}{A} \sum_{i=1}^{n} b_i S_i \tag{6.7}$$

where S is the ADC sample vector with $n = 7$ samples and a and b are precomputed weights vectors.

The computation of the required multiply-add operations on a processor can be made more efficient by using SIMD operations. The NEON SIMD unit on ARM CPUs is able to process four 32 bit floating point or integer numbers simultaneously. The gcc compiler does not always notice that code can be "vectorised" to run on the SIMD unit. By padding the vectors S, a and b with a zero to make $n = 8$, the compiler is given a hint and the code tends to be optimised better.

6.2.1 Test Methodology

A program in C++ was written to do the calculations shown in Equations 6.6 and 6.7 using vectors populated with random numbers. The time taken to perform the calculation one thousand times and averaged, to compensate for limited timed resolution, was used to calculate an effective data throughput.

Since dealing with small quantities of data can be inefficient in comparison to blocks of data that fit well into cache, several OF sample vectors can be grouped

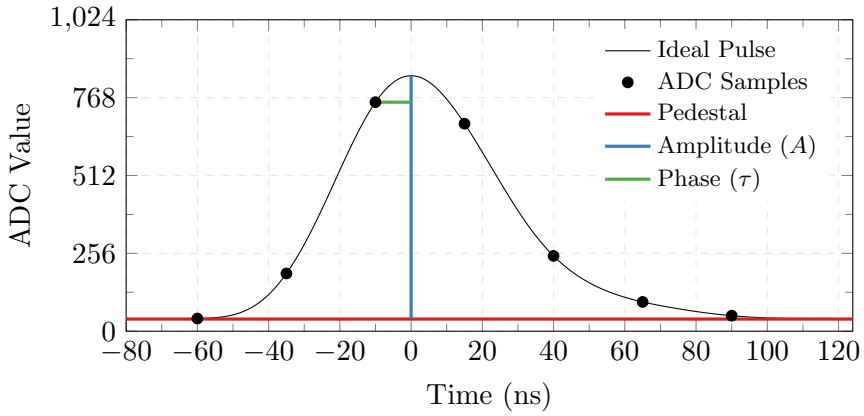


Figure 6.10: Arbitrary PMT pulse showing seven samples with the pedestal, peak amplitude and phase illustrated.

into a matrices as shown in Equation 6.9. Blocks from two sample vectors up to 4096 vectors were measured. The amount of data processed by each calculation is therefore:

$$\text{Size(Bytes)} = 4 \times 2 \times n + 4 \times B \times n \tag{6.8}$$

where $n = 7$ or 8 for better processor efficiency and B is the block size which is from 2 to 4096. The factor of four is because a single precision floating point number or a standard 32 bit integer is four bytes long.

$$\begin{bmatrix} A_1 & A\tau_1 \\ A_2 & A\tau_2 \\ \vdots & \vdots \\ A_B & A\tau_B \end{bmatrix} = \begin{bmatrix} \vec{S_1} \\ \vec{S_2} \\ \vdots \\ \vec{S_B} \end{bmatrix} \times \begin{bmatrix} \vec{a} \\ \vec{b} \end{bmatrix}^T \tag{6.9}$$

There are numerous linear algebra libraries which have been optimised for different CPU architectures. These libraries are able to make optimal use of a CPUs floating point unit, SIMD and general mathematical capabilities. ATLAS is used for the HPL benchmark but a more lightweight and in some cases higher performance library called Eigen was used for the Optimal Filtering implementation [113].

Only platforms with PCI-Express connectivity were tested.

6.2.2 Results

A graph of the OF implementation performance converted to MB/s is shown in Figure 6.11. If these results are compared to the HPL benchmark results in Section 5.1.2, the relative difference between the i.MX6 and Tegra-K1 SoCs is similar with the exception of the X-Gene. This discrepancy of the X-Gene is explained later.

Since each CPU runs with a different CPU frequency making it difficult to compare the relative performance between the ARM cores and architectures, Figure 6.12 shows the results that have been normalised to 1 GHz.

An Intel i7-4770 was also tested to give an idea of a high performance CPU. The ARM SoCs all perform relatively well in comparison with the Tegra-K1 SoC having approximately half the performance of the Intel i7. The i.MX6 has the

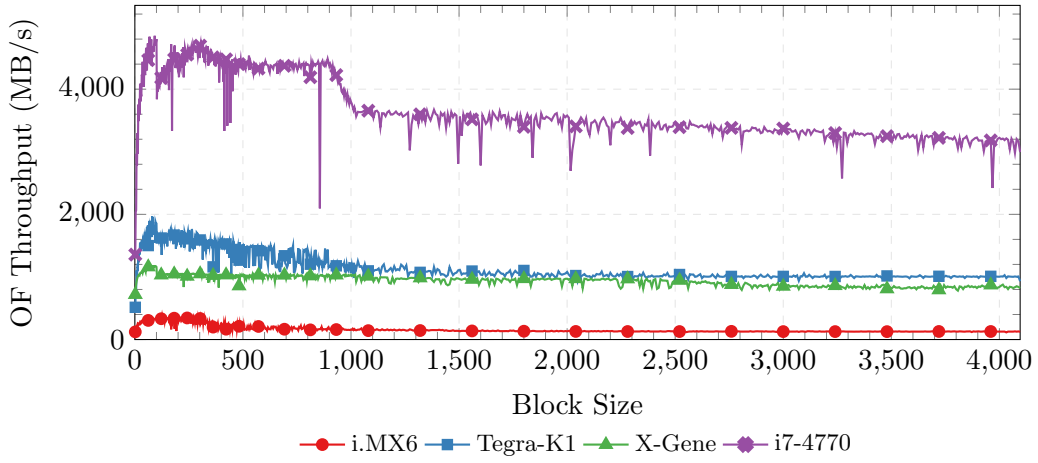


Figure 6.11: OF benchmark at maximum CPU frequency with one core.

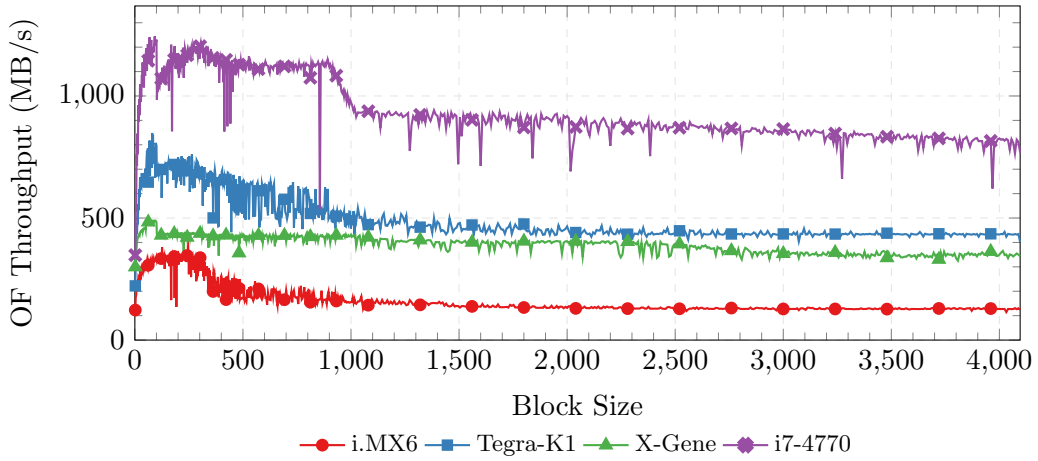


Figure 6.12: OF benchmark normalised to 1 GHz CPU frequency with one core.

lowest performance which is expected. With block sizes between 100 and 300, which correspond to approximately 6 kB to 20 kB (the L1 cache is 32 kB), the theoretical throughput of the OF algorithm is over 350 MB/s.

Table 5.6 in Section 5.1.2 indicates that the relative performance between the Tegra-K1 and i.MX6 in GFLOPS per MHz is 230%. In the range of block sizes between 100 and 300 the difference in performance of the normalised results is 215% which lends confidence that the results are indeed accurate.

Interestingly, the X-Gene has lower performance than the Tegra-K1 even though it has a more sophisticated architecture and performs higher in the CPU and memory benchmarks presented in Chapter 5. This is most likely because the X-Gene, which is ARMv8, has a different SIMD unit to the ARMv7 architecture and the gcc compiler and more importantly the Eigen library is not optimised yet for the changes.

6.2.3 Integration with sROD

Since the sROD co-processing unit may require an algorithm such as OF to be run on the data, it is important to compare the results obtained with theoretical PCI-Express throughput and latencies. Only the range of peak results will be considered since

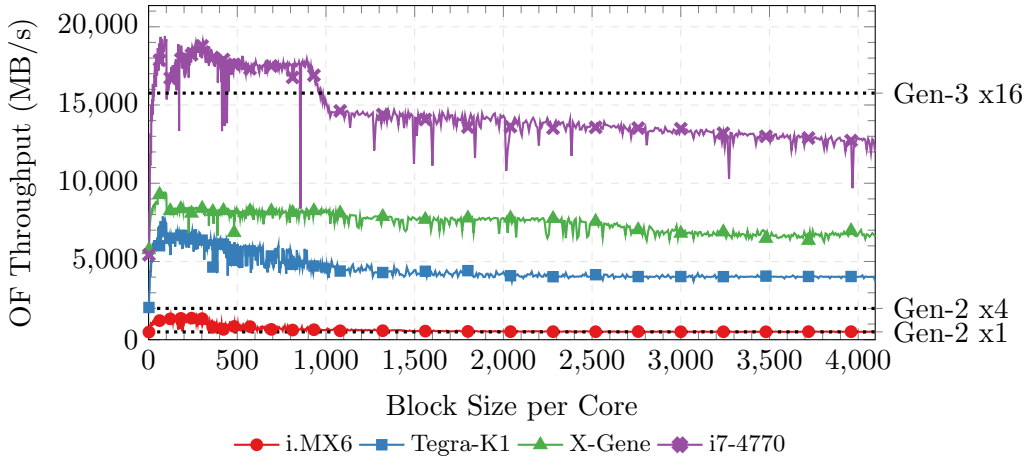


Figure 6.13: OF benchmark at maximum CPU frequency scaled to all cores. Maximum theoretical PCIe throughputs are shown on the right.

operating at block sizes with lower performance makes little sense. In Figure 6.13, the results at maximum CPU frequency have been multiplied by the number of cores. This is accurate because the different blocks of ADC samples from the front end are not correlated and can therefore be computed in isolation as an "embarrassingly parallel" process.

The PCI-Express throughput results from Section 4.3.2 have been plotted as horizontal lines on Figure 6.13. In reality, as discussed in Section 4.3.2, the actual throughput will be slightly lower depending on the capabilities of the SoC. In this case, only the i.MX6 (Cortex-A9) which does not support DMA or larger PCI-Express packet sizes would be significantly different from the 500 MB/s theoretical line on the graph.

The i.MX6 SoC is able to sustain almost 1 GB/s of OF throughput. If the lack of DMA on the PCI-Express controller is factored in then one CPU may potentially be required for data movement and the other three for data processing. At 75% OF performance, there is still an I/O bottleneck. A more complex algorithm is potentially viable as there is theoretically a small excess of processing power.

A similar situation applies to the Tegra-K1 SoC where OF throughputs of over 6 GB/s are possible. The PCI-Express controller supports DMA and other required features for efficient operation and so close to the theoretical 2 GB/s throughput is feasible. There is potential on the Tegra-K1 for a significantly more complex algorithm to be used in comparison to OF while still fully utilising the available I/O.

On the X-Gene SoC there appears to be an excess of I/O in comparison to the OF throughput. The theoretical performance of the ARM Cortex-A57 core on which the X-Gene is based is supposedly on the order of 50% faster than the Cortex-A15 [114]. Unfortunately no official references were found but based on ARM press releases and slides, the DMIPS/MHz of the Cortex-A15 is approximately 3.5 and the Cortex-A57 is 4.0 with the X-Gene being 4.2 DMIPS/MHz. This is an approximately 20% improvement. The difference in HPL results indicates an 8% improvement when the X-Gene is normalised to four cores.

If it is assumed that the X-Gene is indeed 20% faster than the Cortex-A15 then the results shown in Figure 6.13 for the X-Gene should be in the region of 14 GB/s when extrapolated from the Cortex-A15 results. There is a small excess of I/O in

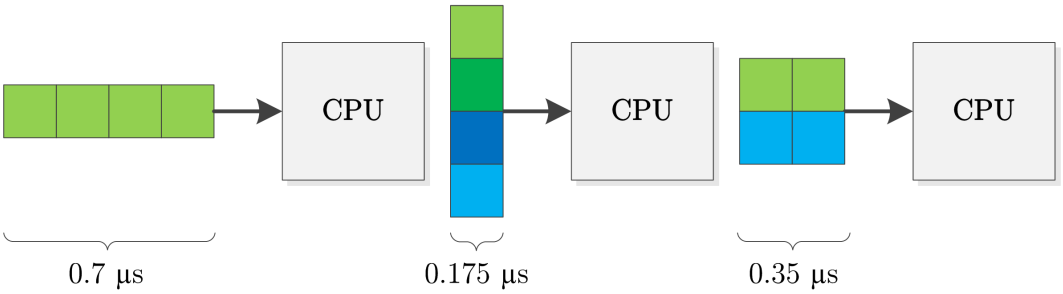


Figure 6.14: Serial, parallel and combination queue methods for events.

this case but when potential overheads are taken into account, an X-Gene based system would be the most well balanced.

Detailed information and benchmarks of the Intel i7-4770 CPU has not been considered, but like the X-Gene it is likely that it would also result in a well balanced system. The i7 CPU is not a SoC so in practise the system could not be as compact. The TDP of the i7-4770 is 84 W and so the power consumption of the system is close to double that of an X-Gene system discussed in Section 5.3.

A further consideration to sROD integration is that of latency. From Section 4.3 the round trip latency is 1.2 μs for the i.MX6. Subsequent packets of a PCI-Express request are governed by the bandwidth of the link. The latency may be lower with higher end PCI-Express hardware but this can be treated as a worst case.

If we subtract this latency from the TileCal Level-1 Trigger budget of 6 μs, and not taking into account overhead in the sROD itself we are left with 4.8 μs to perform the OF computation. In order to maximise efficiency by minimising the effect of latency, the maximum block size should be provided to the SoC for processing.

In the following examples and calculations, it is assumed that the sROD will act as an efficient scheduler for the SoCs. In reality there will be additional overheads and complexity and so the calculations should be viewed as an ideal case and as an uncomplicated illustration of the requirements.

There are in principal three main ways in which blocks can be constructed from samples: series, parallel and a combination, shown in Figure 6.14.

The time budget for the optimal filtering calculation includes the time required to accumulate the samples for each event as well as to process them. In series, each event requires at minimum 7×25 ns which is 0.175 μs. In parallel, samples from each of the 48 channels processed by the sROD prototype can be collected simultaneously and so in 0.175 μs, a block size of 48 is possible. The time taken to compute a block size of 48 on the Tegra-K1, for example, is 0.98 μs. The total time in this case is a little over 1 μs and so there is potential for a larger block size.

A block size of 192, which is made up of 48 channels in parallel with four sets of event samples results in a buffering time of 0.7 μs and a computation time of 3.61 μs which is just under the 4.8 μs limit. A similar calculation is required for the other SoCs. Table 6.1 shows the optimal block sizes, computation times and effective throughput for the different platforms which is calculated by dividing the number of events processed (block size) per 6 μs.

The total number of events per second that the sROD must process can be calculated. Every seven samples for each channel is one event which equates to a maximum of 275 million events per second. Continuing with the Tegra-K1 for

Table 6.1: Maximum block size for each SoC (single core) within the TileCal Trigger latency requirements assuming 4.8 μs available time.

	Block Size	**Buffer Time** (μs)	**Compute Time** (μs)	**Throughput** (MEvents/s)
i.MX6Q	40×1	0.175	4.61	6.6
Tegra-K1	48×4	0.7	3.61	32
X-Gene	48×2	0.35	2.67	16
i7-4770	48×9	1.58	3.08	72

example purposes, each core is able to process 32 million event per second, from Table 6.1, and so 9 cores would be required. Each Tegra-K1 SoC has four cores and so three SoCs would be suitable with some room for more sophisticated algorithms. A suitable streaming data scheduler would be required to split the incoming data streams between cores.

7

Conclusions

The first three chapters of this dissertation are dedicated to describing the CERN Large Hadron Collider and ATLAS Hadronic Tile Calorimeter (TileCal) in detail. The description of the ATLAS detector emphasises the extremely large amount of raw data that is produced by the various sub detectors which must be processed before useful physics data is available.

In particular, this research is centered on the TileCal which has over ten thousand read-out channels resulting in many gigabytes per second of raw data which must be processed by a computing system that is energy efficient and easy to maintain.

Pulses of light produced by scintillation inside TileCal are captured by PMTs, filtered and then digitised with 40 MSa/s ADCs situated on the TileCal front end electronics. The resulting raw data is very noisy due to what is known as in time and out of time pile up. A digital filtering algorithm called Optimal Filtering (OF), described in Chapter 3, is utilised to extract the actual energy and time information from the noisy raw data.

In future, the LHC will be upgraded to the High Luminosity LHC (HL-LHC), which presents several challenges with respect to the TileCal read out system and associated data processing. In preparation, the TileCal front end electronics have been redesigned and the read out processing will be migrated to the back end and be performed on the so-called sROD.

This new scheme enables an increased volume of raw data to be transmitted to the back end for processing by more sophisticated algorithms, eventually leading to better quality physics data. In the existing system, the ten thousand read-out channels on the TileCal produce approximately 163 Gb/s raw data. The volume in the upgraded system will be increased to approximately 41 Tb/s.

The sROD uses high end FGPA chips to perform the required triggering and energy reconstruction of the data. In Chapter 4 it is argued that the development of complex FPGA firmware for the sROD, or any high end FPGA based device, is a significant challenge and may not be a flexible enough solution to suit as yet unknown future energy reconstruction algorithms for the HL-LHC.

It is indeed possible to implement the current OF algorithm and other future algorithms on the sROD. However, because of the difficulties associated in doing so, a general purpose and high data throughput, SoC based co-processing unit for the sROD is proposed in this work. The skills required for developing algorithms to run

on general purpose CPUs found in the SoCs used by the PU are widespread because well known programming languages such as C or C++ can be used effectively.

ARM System on Chips (SoCs), which are the focus of this research, are hypothesised to have the appropriate characteristics for use in this PU. They are ubiquitous in modern mobile devices such as tablets and smart phones due to their low power consumption, low cost and good performance. Consumer demand is driving rapid improvements to the performance and energy efficiency of these SoCs. For these reasons, ARM SoCs are being investigated by several companies and research groups for their usability and effectiveness in fixed computing environments such as web servers and computing clusters. Their performance for real time and high data throughput computing is not well studied.

The question of whether this so-called PU is a feasible solution to the high data throughput problem, or more specifically the sROD co-processing unit, is answered in several stages, with details in Chapters 4, 5 and 6. The sub-issues addressed in order to determine the feasibility of the PU are available external I/O bandwidth and latency, CPU integer and floating point processing performance as well as energy efficiency and CPU memory bandwidth. These factors are summarised in Sections 7.1, 7.2 and 7.3, respectively.

In Section 7.4, actual application performance of the current energy reconstruction algorithm, OF, is summarised and a high level PU design is proposed which is indeed feasible for use as a sROD co-processing unit.

Five ARM SoC development boards were available for testing: the Cubieboard A20, Wandboard Quad, ODROID-XU, NVIDIA Jetson and X-Gene Mustang which are based on ARM Cortex-A7, A9, A15r2, A15r3 and X-Gene (similar to the Cortex-A57) CPUs, respectively.

7.1 External I/O

The first question asked is whether these SoCs, which are primarily intended for use in mobile devices, have the necessary external I/O to satisfy the requirements of a high data throughput sROD co-processing unit?

In Chapter 4, Ethernet, USB and PCI-Express interfaces are considered and it is concluded that PCI-Express is the most appropriate interface because it is both very low latency and high bandwidth. The CPU processing requirements of the other interfaces in terms of drivers and network stacks are also typically higher than those of PCI-Express, which would lower the overall processing efficiency of the system.

PCI-Express is not typically used for inter-CPU communication as it would be in the PU and so it is important to test and validate the theoretical performance. A test system was designed and constructed to connect two Freescale i.MX6Q SoC based development boards via their PCI-Express interfaces. Several test programs were implemented and run to determine the true performance of a basic PCI-Express link and of the i.MX6Q SoCs.

It was found that the PCI-Express Gen-2 x1 link used is able to sustain 123 MB/s write and 182 MB/s read performance with a simple memory mapped user space application. Higher performance is possible by using a custom Linux kernel driver. A write throughput of 284 MB/s and a read throughput of 93 MB/s was achieved by modifying the existing Linux PCI-Express driver to run a test during the system boot process.

Unfortunately, DMA functionality is not present on the i.MX6Q SoC because of its low cost nature. As a workaround, the Image Processing Unit (IPU) DMA was used to copy data between the SoCs and the resulting throughput was 248 MB/s read and 375 MB/s write. Unfortunately, the IPU cannot be used to move data in a generic way for a user space application but the improved results serve to demonstrate the possible gains provided by DMA.

Several other SoCs on the market have better PCIe specifications. The Tegra-K1 and X-Gene both have DMA capabilities with the former SoC supporting PCIe Gen-2 x4 and the latter supporting PCIe Gen-3 x16, allowing much higher data rates. Based on the theoretical specifications as well as the validation of performance on the i.MX6 SoC, PCI-Express appears to be a viable I/O interface for connecting the sROD to the co-processing unit and also for interconnecting SoCs on the PU itself.

In Chapter 4, the connection between the sROD prototype and PU is discussed. There is sufficient available I/O between the AMC cards and RTM on a typical ATCA chassis for a high bandwidth PCI-Express connection between the two.

7.2 CPU Performance

The question of ARM SoCs CPU processing performance is considered in Chapter 5. Industry standard benchmarks are used to measure the integer, floating point and memory subsystem performance of the available ARM platforms. For a clearer interpretation of the CPU performance results, it is interesting to include some benchmarks of typical high end x86 CPUs which are commonly used in server environments.

It is found that in integer performance, measured by CoreMark, ARM SoCs are not far behind high end Intel x86 based CPUs. Section 5.1.1 reports that, for example, the ARM Cortex-A15 CPU achieves approximately the same number of CoreMarks per MHz as an Intel i7-3930K CPU. The X-Gene CoreMark performance attained is between the Cortex-A9 and Cortex-A15 which is unexpected. It should be higher than that of the Cortex-A15 but the difference is likely due to immature compiler optimizations. This is motivated by the fact that the difference between gcc 4.8 and gcc 4.9 with CoreMark on the X-Gene is 5.6% which is significant compared to the 2.0% difference seen with the Cortex-A15.

The Intel i7 and other similar CPUs have a higher absolute integer performance because they typically have more CPU cores and operate at higher clock frequencies. The branch prediction and other micro-architectural features within modern x86 CPUs are also more sophisticated than those found in ARM CPUs which results in higher overall performance at the expense of increased power consumption.

HPL is used to measure the floating point performance of the SoCs. The double precision floating point performance of the Cortex-A15s and the X-Gene are approximately 0.0014 GFLOPS/MHz. For comparison, lower performance ARM CPUs such as the Cortex-A7 and Cortex-A9 achieved 0.0004 and 0.0006 GFLOPS/MHz. These are fairly dramatic performance increases from one generation to the next. For comparison, the Intel i7-4770 CPU achieves 0.013 GFLOPS/MHz which is significantly higher than the ARM CPUs that were tested. This is due to the numerous advanced floating point SIMD capabilities of modern x86 CPUs such as SSE2, SSE3, SSE4, AVX and AVX-512, compared to ARM's NEON which is equivalent to SSE1. The single precision performance of the ARM SoCs is approximately twice the double precision performance.

The energy efficiency in GFLOPS per Watt for ARM SoCs is approaching and in some cases breaches 1 GFLOP/W. This is an important metric in HPC environments but is also relevant in high data throughput computing where massive CPU parallelism is also necessary.

The Cortex-A7 SoC had the lowest energy efficiency at 0.35 GFLOPS/W but this is due to the very low cost nature of the chip, where it is likely that little effort was placed in optimising the energy efficiency. The Cortex-A9 and Cortex-A15 SoCs achieved 0.77 and 0.86 GFLOPS/W respectively. The reason for the relatively consistent energy efficiency is because of the smaller silicon feature size on the Cortex-A15 which inherently results in higher possible clock frequencies and lower power consumption. If the Cortex-A15 clock frequency is set to 1 GHz to match the Cortex-A9, the efficiency increases to approximately 1.6 GFLOPS/W. The X-Gene, based on an older, 40 nm silicon process, had low energy efficiency at 0.41 GFLOPS/W. This is likely to be dramatically improved in the X-Gene 2 which is based on a 28 nm process.

The integer (CoreMark) energy efficiency of the ARM Cortex-A15 is more than triple that of an Intel i7-3930K CPU. Energy efficiency results for 4^{th} generation Intel CPUs such as the i7-4770 are not available, but it is likely that ARM is still superior.

If the PU is required to do intensive double precision floating point computations, ARMv-7 and ARM-v8 are not good CPU architectures compared to modern Intel x86 architectures. If single precision floating point or integer computation is required then ARM should be considered.

7.3 Memory Bandwidth

Memory bandwidth is an important benchmark and is important when considering the external I/O bandwidth capabilities of the SoCs. If the memory bandwidth is not higher than the PCI-Express theoretical maximum, then it is impossible to achieve that maximum.

Memory bandwidth tests of the various ARM SoCs was performed using the STREAM benchmark. The X-Gene has the highest memory bandwidth and the Cortex-A7 has the lowest. This is expected as the X-Gene is a server-grade SoC with comprehensive memory controller features such as ECC, dual channel, high clock frequency and wide bit width. The Cortex-A7 on the other hand, which is designed for low end consumer devices, has a relatively limited memory controller, and is adequate performance for typical mobile device workloads such as web browsing.

The memory bandwidth efficiency, which is a measure of the sustained memory bandwidth (or throughput) compared to the theoretical maximum, is a good indication of the overall performance of the SoC memory system. The A20, Exynos 5410 and Tegra-K1 SoCs all achieve between 40% and 50% efficiency. The i.MX6 SoC only achieves 16% efficiency which was verified several times in case the operating system or compiler was malfunctioning. The X-Gene has a 68% memory bandwidth efficiency.

7.4 sROD Co-Processor Feasibility

Although the synthetic benchmark results obtained can be used to effectively and fairly compare the tested ARM platforms against other compute hardware, their performance for sROD coprocessor applications is difficult to extrapolate.

Two application benchmarks were tested in Chapter 6. FFTW, which is a high performance, open source implementation of a FFT library, was thoroughly tested as well as a custom OF implementation to simulate a more likely workload for the PU when it is used as a sROD co-processor.

Taking into account the PCI-Express capabilities and the FFTW results of the Tegra-K1, the 2 GB/s maximum bandwidth allows for FFTs up to 256. The X-Gene is able to sustain FFT throughputs approximately double that of the Cortex-A15 at FFT sizes above 32768. Below that point the calculated throughput is almost identical to that of the Tegra-K1 which has a similar clock frequency. The reason for the performance difference is that the X-Gene has eight cores and the Tegra-K1 has four. Clearly the use of ARM for FFT calculations results in a CPU bottleneck, but the amalgamated I/O and performance of a cluster of ARM SoCs should nevertheless be considered for energy efficiency purposes in a large installation.

The OF results can be interpreted in terms of data throughput but it is more meaningful to compare the SoCs in terms of events per second. The strict latency requirements of the ATLAS TileCal front end dictate that only 6 μs is available to the sROD for energy reconstruction.

The PCI-Express results in Chapter 4 indicate that on average 1.2 μs is required for the PCI-Express data transfer, leaving 4.8 μs for data processing with the assumption of negligible latency on the sROD. Each core of the Tegra-K1 SoC is able to perform the OF algorithm at approximately 32 million events per second. The sROD prototype must process 275 million events per second in total. Nine Tegra-K1 CPU cores would be required to satisfy this requirement and since the Tegra-K1 is a quad core device, three SoCs would be sufficient with processing power to spare for more sophisticated energy reconstruction algorithms in future.

The available I/O bandwidth on the X-Gene enables much higher data throughput, in theory. If the apparent performance issues of the compilers are solved in later versions then the X-Gene should be able to sustain more than double the number of events per second than the Tegra-K1. This is because the X-Gene has eight cores and the CPU architecture is also allegedly up to 50% faster per MHz.

An ATCA, AMC based PU, with two X-Gene SoCs connected via the ATCA RTM or backplane to the sROD prototype by a PCI-Express Gen-3 x4 or x8 interface will satisfy the requirements of the sROD co-processor. This PU will have approximately double the I/O bandwidth of the current sROD prototype and low enough latency to satisfy the strict timing requirements and data throughput of the TileCal front-end readout system. The amalgamation of two X-Gene SoCs, interconnected by a second PCI-Express Gen-3 x8 interface in EP-RC mode will also enable ample CPU processing power to perform the existing OF energy reconstruction algorithm.

Future algorithms such as matched filters and deconvolution can be implemented in a well known programming language such as C++ and easily “installed” on the PU in order to test and prototype solutions to the problem of pile up in the future HL-LHC.

7.5 Future Work

The PU will be a very complex and compact piece of hardware. The hardware design and manufacture of the PU will be a time consuming endeavour but could be achieved with a team of qualified designers. The successful demonstration of the PU on the ATLAS TileCal test beam or detector itself will be high impact.

The use of the PU in other applications should also be investigated. For example, the SKA has a back end correlator system which may be based on FPGAs, GPUs or a CPU farm. The high data throughput and good CPU performance of some ARM SoCs, with high energy efficiency in general, make them a good alternative candidate for this task. The "programmer friendly" nature of a CPU based system over FPGA and to some extent GPU systems is also an advantage.

Several commercial products are becoming available that use ARM and Intel SoCs because of their energy efficiency in non HPC tasks. For example, HP has their ProLiant Moonshot server chassis which supports up to 45 so-called cartridges, connected to an Ethernet backplane. Currently available cartridges make use of low power Intel SoCs as well as Texas Instruments ARM based SoCs with DSP co-processors.

There appears to be support for a two dimensional torus interconnect between the cartridges which would enable similar functionality to the PU if the interconnect is able to use PCI-Express. This may allow for a high data throughput system if the I/O is amalgamated.

Ever increasing global requirements for higher data throughput web, audio and video streaming servers as well as big science projects such as the LHC and SKA are driving the need for energy efficient and cost effective computing with high bandwidth I/O capabilities.

It is, therefore, likely that industry will continue to pursue the use of SoCs in their server offerings. What is yet to be seen is whether PCI-Express becomes a commonly used interconnect for SoCs in order to facilitate the amalgamation of their performance and I/O in a cost effective way, as proposed in this dissertation.

GPU processing has not been discussed at length in this dissertation. There are at least two ways in which GPUs may be used with ARM SoCs to create a high data throughput or indeed a HPC system. The first is currently being pursued by the Mont-Blanc project, where the integrated GPUs on ARM SoCs are being used for processing and the ARM CPUs for managing data and providing an interface to the rest of the system. Very high performance could be achieved by using the NVIDIA Tegra-K1 SoCs which host powerful, 192 core Keplar GPUs.

The second manner in which GPUs may be used are as co-processors. Rather than using the GPU for all processing, the CPU could simply offload simple but computationally intensive calculations as required. The difference between these two methodologies is subtle, but research could be done to investigate the effectiveness of these strategies for future energy efficient, high data throughput computing systems.

References

[1] The ATLAS Collaboration, "Observation of a new particle in the search for the standard model higgs boson with the atlas detector at the lhc", *Physics Letters B*, vol. 716, no. 1, pp. 1–29, Sep. 2012. DOI: 10.1016/j.physletb.2012.08.020.

[2] L Rossi and O Brüning, "High luminosity large hadron collider a description for the european strategy preparatory group", CERN, Geneva, Tech. Rep., Aug. 2012.

[3] F Carrió, V Castillo, A Ferrer, *et al.*, "The srod module for the atlas tile calorimeter phase-ii upgrade demonstrator", *Journal of Instrumentation*, vol. 9, no. 02, pp. C02019–C02019, Feb. 2014. DOI: 10.1088/1748-0221/9/02/C02019.

[4] L. M. A. Filho, D. O. Damazio, and A. S. Cerqueira, "Calorimeter signal response deconvolution for online energy estimation in presence of pile-up", *ATLAS Internal Note*, Sep. 2012.

[5] B. S. Peralva, "The tilecal energy reconstruction for collision data using the matched filter", in *Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC), 2013 IEEE*, Seoul: IEEE, 2013, pp. 1–6. DOI: 10.1109/NSSMIC.2013.6829472.

[6] T. Krazit, *Armed for the living room*, 2006. [Online]. Available: http://news.cnet.com/ARMed-for-the-living-room/2100-1006_3-6056729.html (visited on 04/03/2014).

[7] Raspberry Pi Foundation, *Raspberry pi*, 2015. [Online]. Available: https://www.raspberrypi.org/ (visited on 06/29/2015).

[8] HP, *Hp moonshot 1500 chassis*, 2015. [Online]. Available: http://www8.hp.com/za/en/products/moonshot-systems/product-detail.html?oid=5365577 (visited on 06/29/2015).

[9] Dell, *Copper enables the arm server ecosystem*, 2014. [Online]. Available: http://www.dell.com/armserver (visited on 03/31/2014).

[10] N. Rajovic, P. M. Carpenter, I. Gelado, *et al.*, "Supercomputing with commodity cpus", in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis on - SC '13*, New York, New York, USA: ACM Press, Nov. 2013, pp. 1–12. DOI: 10.1145/2503210.2503281.

[11] AppliedMicro, *X-gene*, 2014. [Online]. Available: http://www.apm.com/products/data-center/x-gene-family/x-gene/ (visited on 04/22/2014).

[12] The ATLAS Collaboration, "The atlas experiment at the cern large hadron collider", *Journal of Instrumentation*, vol. 3, no. 08, S08003–S08003, Aug. 2008. DOI: 10.1088/1748-0221/3/08/S08003.

[13] J. M. Campbell, J. W. Huston, and W. J. Stirling, "Hard interactions of quarks and gluons: a primer for lhc physics", en, *Reports on Progress in Physics*, vol. 70, no. 1, pp. 89–193, Jan. 2007. DOI: 10.1088/0034-4885/70/1/R02.

[14] CERN Press Office, *Lhc to run at 4 tev per beam in 2012*, 2012. [Online]. Available: http://press.web.cern.ch/press-releases/2012/02/lhc-run-4-tev-beam-2012 (visited on 11/26/2014).

[15] The ATLAS Collaboration, "Letter of intent for the phase-ii upgrade of the atlas experiment", CERN, Geneva, Tech. Rep. CERN-LHCC-2012-022. LHCC-I-023, Dec. 2012.

[16] The ATLAS Collaboration, "Letter of intent for the phase-i upgrade of the atlas experiment", CERN, Geneva, Tech. Rep. CERN-LHCC-2011-012. LHCC-I-020, Nov. 2011.

[17] The ATLAS Collaboration, "Luminosity determination in pp collisions at $\Gamma"026e30fsqrt\{s\}= 7$tevusingtheatlasdetectoratthelhc", *The European Physical Journal C*, vol. 71, no. 4, p. 1630, Apr. 2011. DOI: 10.1140/epjc/s10052-011-1630-5. arXiv: 1101.2185.

[18] ATLAS Collaboration, *Atlas luminosity public results*. [Online]. Available: https://twiki.cern.ch/twiki/bin/view/AtlasPublic/LuminosityPublicResults (visited on 06/01/2015).

[19] ATLAS Experiment, *Multimedia*, Geneva, 2014.

[20] ATLAS Collaboration, *ATLAS inner detector: Technical Design Report, 1*, Technical. Geneva: CERN, 1997.

[21] ATLAS Collaboration, *ATLAS liquid-argon calorimeter: Technical Design Report*, Technical. Geneva: CERN, 1996.

[22] The ATLAS Collaboration, "Readiness of the atlas tile calorimeter for lhc collisions", *The European Physical Journal C*, vol. 70, no. 4, pp. 1193–1236, Dec. 2010. DOI: 10.1140/epjc/s10052-010-1508-y.

[23] ATLAS Collaboration, *ATLAS muon spectrometer*. Geneva: CERN, 1997.

[24] J. Dursi, "Parallel i/o doesn't have to be so hard: the adios library", SciNet, Tech. Rep., 2012.

[25] Wikipedia, *Ieee 802.3 — wikipedia, the free encyclopedia*, 2014. [Online]. Available: http://en.wikipedia.org/wiki/IEEE_802.3.

[26] H. P. Beck, M. Abolins, A. Battaglia, *et al.*, "Performance of the final event builder for the atlas experiment", *IEEE Transactions on Nuclear Science*, vol. 55, no. 1, pp. 176–181, 2008.

[27] F. Winklmeier, "The atlas high level trigger infrastructure, performance and future developments", in *2009 16th IEEE-NPSS Real Time Conference*, IEEE, May 2009, pp. 183–188. DOI: 10.1109/RTC.2009.5321918.

[28] E. Fullana, J. Castelo, V. Castillo, *et al.*, "Digital signal reconstruction in the atlas hadronic tile calorimeter", *IEEE Transactions on Nuclear Science*, vol. 53, no. 4, pp. 2139–2143, Aug. 2006. DOI: 10.1109/TNS.2006.877267.

[29] A. Valero, J. Abdallah, V. Castillo, *et al.*, "Dsp online algorithms for the atlas tilecal read-out drivers", *IEEE Transactions on Nuclear Science*, vol. 55, no. 1, pp. 158–164, Feb. 2008. DOI: 10.1109/TNS.2007.914019.

[30] J. M. Seixas, "Quality factor for the hadronic calorimeter in high luminosity conditions", *Journal of Physics: Conference Series*, vol. 608, no. 1, p. 012 044, May 2015. DOI: 10.1088/1742-6596/608/1/012044.

[31] J Poveda, E Fullana, B Salvachúa, *et al.*, "Implementation of the optimal filtering reconstruction algorithm in the tilecal offline software", *ATLAS Note*, no. ATL-TILECAL-INT-2008-003. ATL-COM-TILECAL-2008-005, May 2008.

[32] E Fullana, J Castelo, V Castillo, *et al.*, "Optimal filtering in the atlas hadronic tile calorimeter", *ATLAS Internal Note*, 2005.

[33] W. Cleland and E. Stern, "Signal processing considerations for liquid ionization calorimeters in a high rate environment", *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 338, no. 2-3, pp. 467–497, Jan. 1994. DOI: 10.1016/0168-9002(94)91332-3.

[34] ATLAS Collaboration, *Tilecal technical design report*, 1996. [Online]. Available: http://atlas.web.cern.ch/Atlas/SUB_DETECTORS/TILE/TDR/TDR.html (visited on 06/05/2015).

[35] K. Anderson, A. Gupta, F. Merritt, *et al.*, "Design of the front-end analog electronics for the atlas tile calorimeter", *Nuclear Instruments and Methods in Physics Research, Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 551, pp. 469–476, 2005. DOI: 10.1016/j.nima.2005.06.048.

[36] S. Berglund, C. Bohm, M. Engstrom, *et al.*, "The atlas tile calorimeter digitizer", *1999 IEEE Nuclear Science Symposium. Conference Record. 1999 Nuclear Science Symposium and Medical Imaging Conference (Cat. No.99CH37019)*, vol. 2, 1999. DOI: 10.1109/NSSMIC.1999.845779.

[37] K Anderson, R Teuscher, H Sanders, *et al.*, "Atlas tile calorimeter interface", in *8th Workshop on Electronics for LHC Experiments*, Colmar, France, 2002.

[38] J. Castelo, V. Castillo, and Others, *Tilecal rod hardware and software requirements*, 2005. [Online]. Available: http://cds.cern.ch/record/821482/files/tilecal-2005-003.pdf (visited on 07/07/2014).

[39] F. Tang, H. Akerstedt, K. Anderson, *et al.*, "Upgrade analog readout and digitizing system for atlas tilecal demonstrator", *IEEE Transactions on Nuclear Science*, vol. PP, no. 99, pp. 1–1, 2015. DOI: 10.1109/TNS.2015.2411993.

[40] F. Tang, K. Anderson, G. Drake, *et al.*, "Design of the front-end readout electronics for atlas tile calorimeter at the slhc", English, in *2010 17th IEEE-NPSS Real Time Conference*, IEEE, May 2010, pp. 1–5. DOI: 10.1109/RTC.2010.5750382.

[41] M. Spoor, *Hep group blog: helping out with atlas tile calorimeter demonstrator*, 2014. [Online]. Available: http://wits-hep.blogspot.com/2014/06/helping-out-with-atlas-tile-calorimeter.html (visited on 08/25/2015).

[42] S Muschter, K Anderson, C Bohm, *et al.*, "Development of a readout link board for the demonstrator of the atlas tile calorimeter upgrade", *Journal of Instrumentation*, vol. 8, no. 03, pp. C03025–C03025, Mar. 2013. DOI: 10.1088/1748-0221/8/03/C03025.

[43] PICMG, *Advanced TCA base specification: advanced TCA*. Wakefield, MA: PICMG, 2008.

[44] P Vichoudis, S Baron, V Bobillier, *et al.*, "The gigabit link interface board (glib), a flexible system for the evaluation and use of gbt-based optical links", *Journal of Instrumentation*, vol. 5, no. 11, pp. C11007–C11007, Nov. 2010. DOI: 10.1088/1748-0221/5/11/C11007.

[45] R. G. Reed, "An integration framework tool for atca chassis in the atlas detector control system", in *Proceedings of SAIP2014, the 59th Annual Conference of the South African Institute of Physics*, C. Engelbrecht and S. Karataglidis, Eds., Johannesburg, 2014.

[46] TIOBE Software, *Tiobe index*, 2015. [Online]. Available: http://www.tiobe.com (visited on 05/04/2015).

[47] A. S. Szalay, G. C. Bell, H. H. Huang, *et al.*, "Low-power amdahl-balanced blades for data intensive computing", *ACM SIGOPS Operating Systems Review*, vol. 44, no. 1, p. 71, Mar. 2010. DOI: 10.1145/1740390.1740407.

[48] K. Lim, P. Ranganathan, J. Chang, *et al.*, "Understanding and designing new server architectures for emerging warehouse-computing environments", in *2008 International Symposium on Computer Architecture*, IEEE, Jun. 2008, pp. 315–326. DOI: 10.1109/ISCA.2008.37.

[49] J. A. Stankovic, "Real-time computing", University of Massachusetts, Amherst, MA01003, Tech. Rep., 1992, p. 4.

[50] J. Zhan, L. Zhang, N. Sun, *et al.*, "High volume throughput computing: identifying and characterizing throughput oriented workloads in data centers", in *2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum*, IEEE, May 2012, pp. 1712–1721. DOI: 10.1109/IPDPSW.2012.213.

[51] K. P. Saravanan, P. M. Carpenter, and A. Ramirez, "Power/performance evaluation of energy efficient ethernet (eee) for high performance computing", in *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, IEEE, Apr. 2013, pp. 205–214. DOI: 10.1109/ISPASS.2013.6557171.

[52] S. Li, K. Lim, P. Faraboschi, *et al.*, "System-level integrated server architectures for scale-out datacenters", in *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture - MICRO-44 '11*, New York, New York, USA: ACM Press, Dec. 2011, p. 260. DOI: 10.1145/2155620.2155651.

[53] B. Jeff, *Arm processors: a walk through the cortex-a mobile roadmap*. [Online]. Available: http://community.arm.com/groups/processors/blog/2013/11/19/a-walk-through-the-cortex-a-mobile-roadmap (visited on 03/05/2015).

[54] ARM, *Arm company milestones*. [Online]. Available: http://www.arm.com/about/company-profile/milestones.php (visited on 03/03/2015).

[55] B. Patterson, *Under the hood: the iphone's gaming mettle — touch arcade*, 2008. [Online]. Available: http://toucharcade.com/2008/07/07/under-the-hood-the-iphones-gaming-mettle/ (visited on 03/06/2015).

[56] C. Sorrel, *Iphone teardown reveals underclocked 833mhz cpu — wired*, 2009. [Online]. Available: http://www.wired.com/2009/06/iphone-teardown-reveals-underclocked-833mhz-cpu/ (visited on 03/06/2015).

[57] Cubieboard, *Cubieboard*, 2013. [Online]. Available: http://docs.cubieboard.org/ (visited on 02/18/2014).

[58] Wandboard.org, *Wandboard - freescale i.mx6 arm cortex-a9 opensource community development board*, 2012. [Online]. Available: http://www.wandboard.org/ (visited on 02/18/2014).

[59] NVIDIA, *Jetson tk1 embedded development kit*. [Online]. Available: http://www.nvidia.com/object/jetson-tk1-embedded-dev-kit.html (visited on 07/10/2015).

[60] Hardkernel, *Odroid-xu*, 2013. [Online]. Available: http://hardkernel.com (visited on 02/18/2014).

[61] Freescale Semiconductor, *Chip errata for the i.mx6*, 2013.

[62] IEEE 802.3 Working Group, *Ieee 802.3 ethernet*. [Online]. Available: http://www.ieee802.org/3/ (visited on 05/05/2015).

[63] J. Postel, "Rfc768: user datagram protocol", Tech. Rep., 1980.

[64] Information Sciences Institute, "Rfc793: transmission control protocol", University of Southern California, Tech. Rep., 1981.

[65] B. Goglin, "High-performance message-passing over generic ethernet hardware with open-mx", *Parallel Computing*, vol. 37, no. 2, pp. 85–100, Feb. 2011. DOI: 10.1016/j.parco.2010.11.001.

[66] P. Shivam and J. S. Chase, "On the elusive benefits of protocol offload", in *Proceedings of the ACM SIGCOMM workshop on Network-I/O convergence experience, lessons, implications - NICELI '03*, New York, New York, USA: ACM Press, Aug. 2003, p. 179. DOI: 10.1145/944747.944750.

[67] R. Zenkert, *X-gene 2 features*, 2014.

[68] M. J. Rashti and A. Afsahi, "10-gigabit iwarp ethernet: comparative performance analysis with infiniband and myrinet-10g", in *2007 IEEE International Parallel and Distributed Processing Symposium*, IEEE, 2007, pp. 1–8. DOI: 10.1109/IPDPS.2007.370480.

[69] USB Implementers Forum, *Usb.org - welcome*, 2015. [Online]. Available: http://www.usb.org/home (visited on 05/11/2015).

[70] USB Implementers Forum, *Usb.org - usb 2.0 specification*. [Online]. Available: http://www.usb.org/developers/docs/usb20_docs/ (visited on 09/08/2015).

[71] Barcelona Supercomputing Center, *Mont-blanc project*, 2014. [Online]. Available: https://www.montblanc-project.eu/ (visited on 03/31/2014).

[72] A. Ramirez, *Building supercomputers from embedded technologies*, 2014. [Online]. Available: http://www.montblanc-project.eu/publications/building-supercomputers-embedded-technologies.

[73] Belcarra Technologies, *Belcarra usblan: protocols*. [Online]. Available: http://usblan.belcarra.com/p/protocols.html (visited on 04/29/2015).

[74] PCI-SIG, *Pci express base specification revision 3.0*, 2010.

[75] PLX Technology, *Pcie expresslane i/o interconnect*, 2015. [Online]. Available: http://www.plxtech.com/products/expresslane/switches (visited on 04/30/2015).

[76] PLX Technology, "Express app 91: failover system using multi-root", Tech. Rep., 2013.

[77] H. Zhu, *I.mx6q pcie ep/rc validation system*, 2013. [Online]. Available: https://community.freescale.com/docs/DOC-95014 (visited on 02/21/2014).

[78] EDM Consortium, "Edm standard - specifications", Tech. Rep., 2013.

[79] Mentor Graphics, *Pcb analysis with hyperlynx*, 2014. [Online]. Available: http://www.mentor.com/pcb/hyperlynx/ (visited on 04/22/2014).

[80] PICMG, *Open standards for embedded computing applications*, 2015. [Online]. Available: https://www.picmg.org/ (visited on 09/14/2015).

[81] Radisys, "Quad amc carrier reference", Tech. Rep., 2008, p. 13.

[82] Kontron, *At8404 atca carrier*. [Online]. Available: http://www.kontron.com/products/boards-and-standard-form-factors/advancedtca/carrier/at8404.html (visited on 05/28/2015).

[83] F Carrió, *Srod rtm connectivity*, 2014.

[84] InfiniBand Trade Association, *Infiniband trade association: home*, 2015. [Online]. Available: http://www.infinibandta.org/ (visited on 08/18/2015).

[85] W.-c. Feng and K. Cameron, "The green500 list: encouraging sustainable supercomputing", *Computer*, vol. 40, no. 12, pp. 50–55, Dec. 2007. DOI: 10.1109/MC.2007.445.

[86] E. Strohmaier, J. Dongarra, H. Simon, *et al.*, *The top500 list*. [Online]. Available: www.top500.org.

[87] Z. Ou, B. Pang, Y. Deng, *et al.*, "Energy- and cost-efficiency analysis of arm-based clusters", in *2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012)*, IEEE, May 2012, pp. 115–123. DOI: 10.1109/CCGrid.2012.84.

[88] A Petitet, R. C. Whaley, J Dongarra, *et al.*, *Hpl - a portable implementation of the high-performance linpack benchmark for distributed-memory computers*, 2008. [Online]. Available: http://www.netlib.org/benchmark/hpl/ (visited on 04/22/2014).

[89] R. Giladi, "Evaluating the mflops measure", *IEEE Micro*, vol. 16, no. 4, pp. 69–75, 1996. DOI: 10.1109/40.526927.

[90] R. P. Weicker, "Dhrystone: a synthetic systems programming benchmark", *Communications of the ACM*, vol. 27, no. 10, pp. 1013–1030, Oct. 1984. DOI: 10.1145/358274.358283.

[91] A. R. Weiss, "Dhrystone benchmark: history, analysis, "scores" and recommendations", ECL, Tech. Rep., 2002.

[92] Embedded Microprocessor Benchmark Consortium, *Coremark*, 2014. [Online]. Available: https://www.eembc.org/coremark/ (visited on 04/22/2014).

[93] ARM, "Application note 273: dhrystone benchmarking for arm cortex processors", 2011.

[94] J. J. Dongarra, P. Luszczek, and A. Petitet, "The linpack benchmark: past, present and future", *Concurrency and Computation: Practice and Experience*, vol. 15, no. 9, pp. 803–820, Aug. 2003. DOI: 10.1002/cpe.728.

[95] R. Clint Whaley, A. Petitet, and J. J. Dongarra, "Automated empirical optimizations of software and the atlas project", *Parallel Computing*, vol. 27, no. 1-2, pp. 3–35, Jan. 2001. DOI: 10.1016/S0167-8191(00)00087-9.

[96] C. Vilsbeck, *Cpu test: intel core i7-4770k brings more power*, 2013. [Online]. Available: http://www.tecchannel.de/pc_mobile/prozessoren/2040710/cpu_test_intel_core_i7_4770k_haswell/index4.html (visited on 10/09/2015).

[97] J. D. McCalpin, *Stream: sustainable memory bandwidth in high performance computers*, 1995. [Online]. Available: http://www.cs.virginia.edu/stream/ (visited on 04/22/2014).

[98] C. Staelin, "Lmbench: an extensible micro-benchmark suite", *Software - Practice and Experience*, vol. 35, no. 11, pp. 1079–1105, 2005. DOI: 10.1002/spe.665.

[99] ARM, *Cortex-a15r3 technical reference manual*, 2014. [Online]. Available: `http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.set.cortexa/index.html` (visited on 06/19/2015).

[100] P. Cook, M. Gupta, V. Zyuban, *et al.*, "Power-aware microarchitecture: design and modeling challenges for next-generation microprocessors", *IEEE Micro*, vol. 20, no. 6, pp. 26–44, 2000. DOI: `10.1109/40.888701`.

[101] E. Blem, J. Menon, and K. Sankaralingam, "Power struggles: revisiting the risc vs. cisc debate on contemporary arm and x86 architectures", in *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*, IEEE, Feb. 2013, pp. 1–12. DOI: `10.1109/HPCA.2013.6522302`.

[102] M. Jarus, S. Varrette, A. Oleksiak, *et al.*, "Performance evaluation and energy efficiency of high-density hpc platforms based on intel, amd and arm processors", in *Energy Efficiency in Large Scale Distributed Systems*, Springer, 2013, pp. 182–200.

[103] R. V. Aroca and L. M. G. Gonçalves, "Towards green data centers: a comparison of x86 and arm architectures power efficiency", *Journal of Parallel and Distributed Computing*, vol. 72, no. 12, pp. 1770–1780, Dec. 2012. DOI: `10.1016/j.jpdc.2012.08.005`.

[104] T. S. Rosing, K. Mihic, and G. De Micheli, "Power and reliability management of socs", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 4, pp. 391–403, Apr. 2007. DOI: `10.1109/TVLSI.2007.895245`.

[105] T. Mudge, "Power: a first-class architectural design constraint", English, *Computer*, vol. 34, no. 4, pp. 52–58, Apr. 2001. DOI: `10.1109/2.917539`.

[106] Global Foundries, "Low power high-k metal gate 28nm cmos solutions for mobile high performance applications", Tech. Rep., 2011, p. 6.

[107] K. Ganesan, J. Jo, W. L. Bircher, *et al.*, "System-level max power (sympo)", in *Proceedings of the 19th international conference on Parallel architectures and compilation techniques - PACT '10*, New York, New York, USA: ACM Press, Sep. 2010, p. 19. DOI: `10.1145/1854273.1854282`.

[108] R. Reed, M. A. Cox, T. Wrigley, *et al.*, "A cpu benchmarking characterization of arm based processors", *Computer Research and Modeling*, vol. 7, no. 3, 2015.

[109] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex fourier series", *Mathematics of Computation*, vol. 19, no. 90, pp. 297–301, 1965.

[110] S. G. Johnson and M. Frigo, "A modified split-radix fft with fewer arithmetic operations", *IEEE Transactions on Signal Processing*, vol. 55, no. 1, pp. 111–119, Jan. 2007. DOI: `10.1109/TSP.2006.882087`.

[111] M. Frigo and S. Johnson, "The design and implementation of {fftw3}", *Proceedings of the IEEE*, vol. 93, no. 2, pp. 216–231, 2005.

[112] A. Parsons, D. Backer, D. Werthimer, *et al.*, "A new approach to radio astronomy signal processing: packet switched, fpga-based, upgradeable, modular hardware and reusable, platform-independent signal processing libraries", *National Radio Science Meeting*, vol. 3, no. c, 2006.

[113] B. Jacob and G. Guennebaud, *Eigen c++ library*, 2014. [Online]. Available: `http://eigen.tuxfamily.org` (visited on 02/24/2015).

[114] ARM, *Cortex-a57 processor*, 2014. [Online]. Available: `http://www.arm.com/products/processors/cortex-a/cortex-a57-processor.php` (visited on 06/23/2015).

Appendix A

Wandboard Adapter Circuit Diagram and Layout

The Wandboard adapter was designed using Altium Designer. The design itself is relatively straight forward with primary concern being signal integrity. A four layer PCB was used with the two internal planes being ground and 5 V power.

The traces for the PCI-Express as well as Ethernet are length matched as well as impedance controlled so satisfy the respective standards and requirements to ensure signal integrity. In Figure 1, the length matching of the traces is visible as small deviations along the length of the parallel traces.

The circuit diagrams are also presented in Figure 2, 3 and 4.

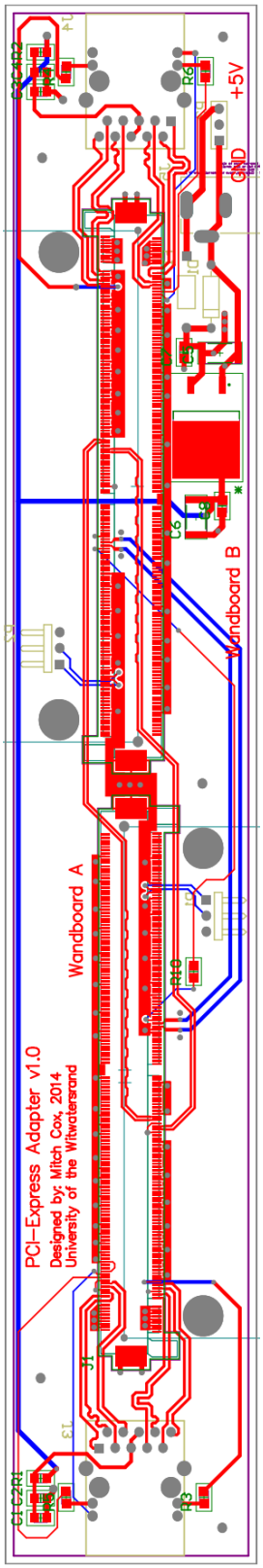


Figure 1: PCB layout of the Wandboard Adapter.

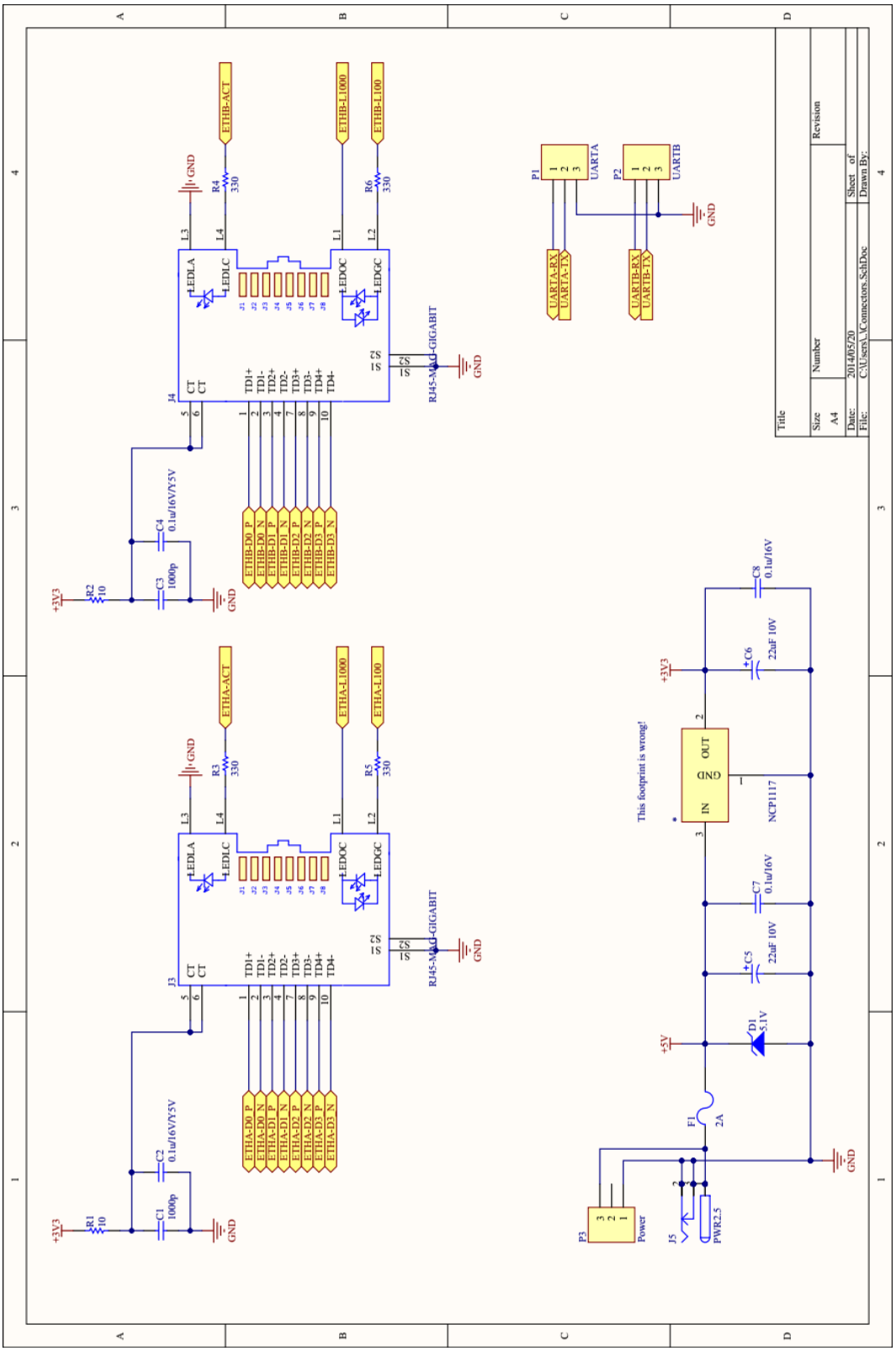


Figure 2: Wandboard Adapter circuit diagram, Page 1.

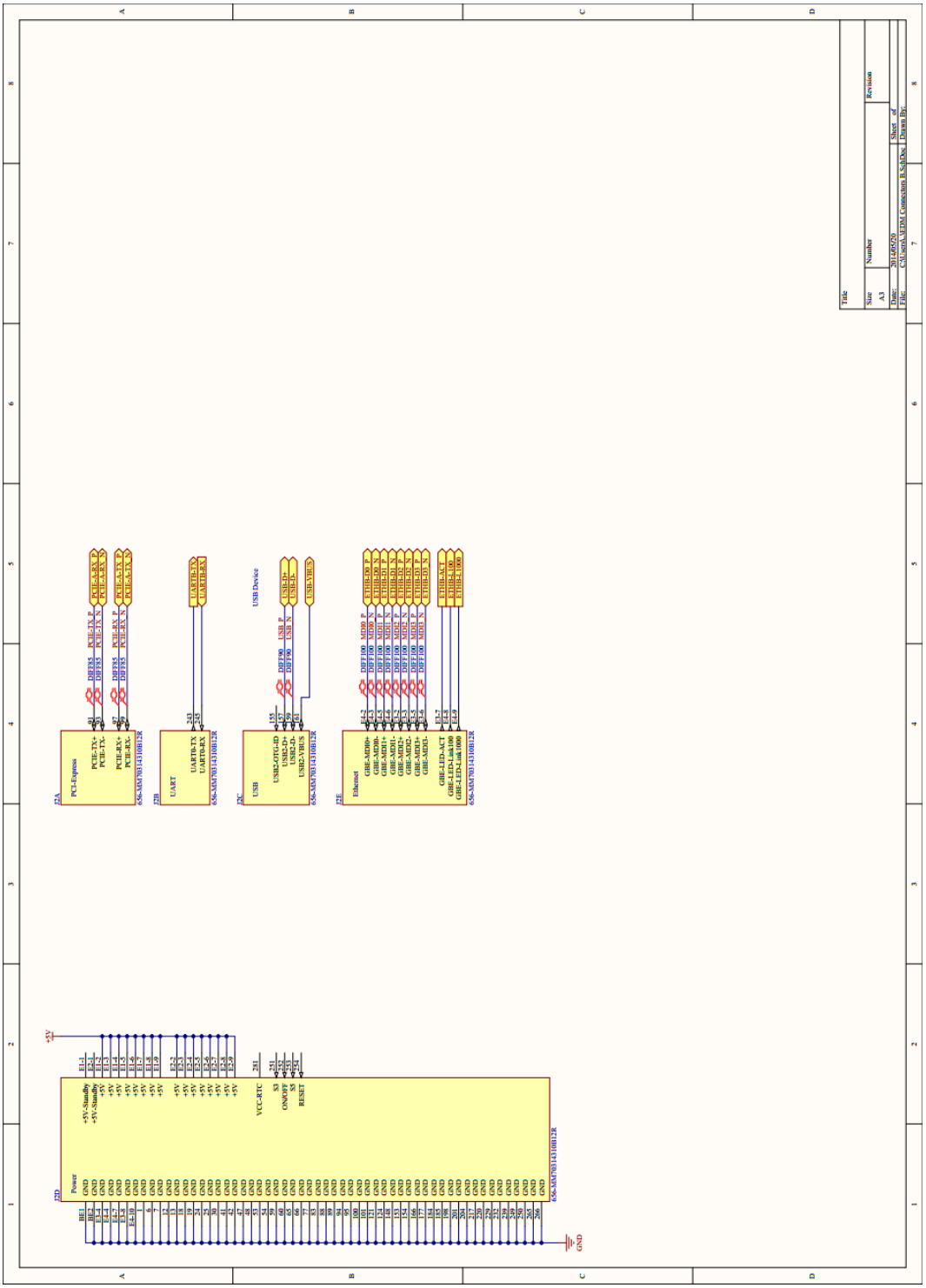


Figure 3: Wandboard Adapter circuit diagram, Page 2.

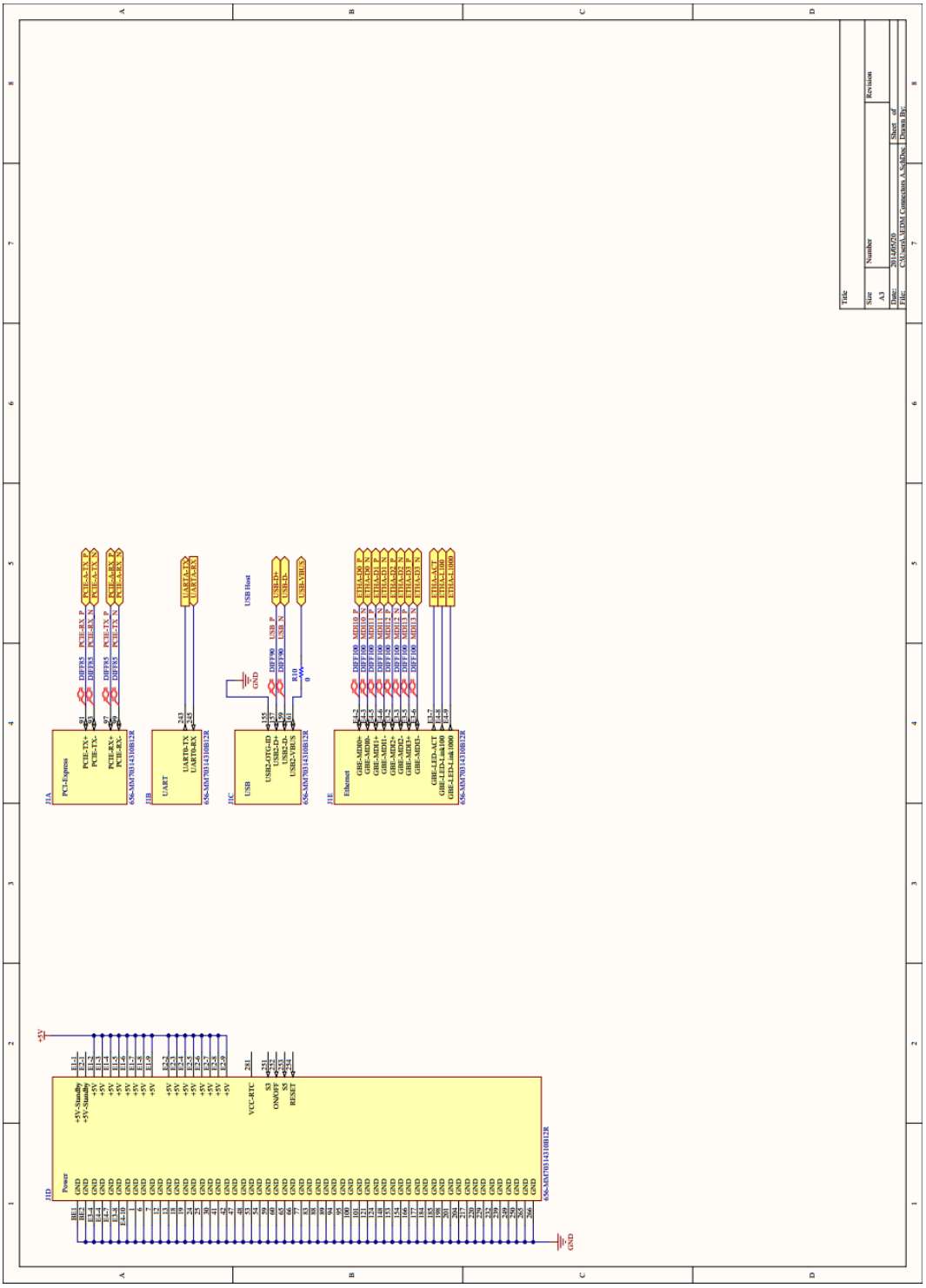


Figure 4: Wandboard Adapter circuit diagram, Page 3.