THE UNIVERSITY OF QUEENSLAND

AUSTRALIA

Title here:

Name

Candidate’s academic degrees

Candidate’s ORCID

A thesis submitted for the degree of Doctor {Master} of Philosophy at The University of Queensland in {year}

Name of the Enrolling Unit

Abstract

Start this section on a new page [this template will automatically handle this].

The abstract should outline the main approach and findings of the thesis and normally must be between 300 and 800 words.

Declaration by author

This thesis is composed of my original work, and contains no material previously published or written by another person except where due reference has been made in the text. I have clearly stated the contribution by others to jointly-authored works that I have included in my thesis.

I have clearly stated the contribution of others to my thesis as a whole, including statistical assistance, survey design, data analysis, significant technical procedures, professional editorial advice, financial support and any other original research work used or reported in my thesis. The content of my thesis is the result of work I have carried out since the commencement of my higher degree by research candidature and does not include a substantial part of work that has been submitted to qualify for the award of any other degree or diploma in any university or other tertiary institution. I have clearly stated which parts of my thesis, if any, have been submitted to qualify for another award.

I acknowledge that an electronic copy of my thesis must be lodged with the University Library and, subject to the policy and procedures of The University of Queensland, the thesis be made available for research and study in accordance with the Copyright Act 1968 unless a period of embargo has been approved by the Dean of the Graduate School.

I acknowledge that copyright of all material contained in my thesis resides with the copyright holder(s) of that material. Where appropriate I have obtained copyright permission from the copyright holder to reproduce material in this thesis and have sought permission from co-authors for any jointly authored works included in the thesis.

Acknowledgments

Contents

List of Figures

List of Tables

List of Abbreviations and Symbols

Abbreviations	
AC	Alternating Current
AFM	Atomic Force Microscopy/Microscope
etc.	*etc.*

Symbols	
$\hat{\rho}$	Density operator
etc.	*etc.*

List of todos

Chapter 1

Introduction

1.1 Your thesis topic

Introduce your topic.

Chapter 2

Research

2.1 Background

Introduce your topic.

Chapter 3

Stack

3.1 SCPNS Hardware Overview

Introduce your topic.

Chapter 4

Evaluation

4.1 Raw Performance Benchmarks

Here, the different configurations of VexRiscv: single-core, dual-core and quad core; will be compared in terms of performance alongside the Raspberry Pi Model 4B 1GB. For each one, we will run the performance benchmark, *stress-ng*, which profiles the IO overhead and memory usage of multiple cores, along with *Iperf3*, to gauge ethernet throughput capabilities.

4.1.1 Iperf3

As ethernet is a vital component of the application, it makes sense to evaluate the capabilities of the link, especially the average bitrate we can expect. After running `iperf3 -s`, which sets the device as a server and listens, another Raspberry Pi was chosen from the cluster to act as a client via running:

```
iperf3 -c 192.168.1.50 -t 30 -i 1 -w 8K -P 1 -R
```

This begins a single-threaded (`-P 1`), client that sends and receives TCP transmissions to *192.168.1.50*, for 30 seconds, sampling the bitrate every second (`-i 1`). Most notably, it constrains the TCP window size (`-w 8K`), to 8Kb, which matches the current size of the board's TX or RX ethernet buffers, more closely resembling the stop-start transfers in our software setup. Here are the results:

Add reference to software/hardware overview

Table 4.1: Ethernet Throughput Comparison of Configurations

	VexRiscvSMP, 100MHz			RPi
	Single	Dual	Quad	4B 1GB
Amount Transferred (MB)	31.5	35.5	42.0	1.13k
Amount Recieved (MB)	31.4	35.4	41.9	1.13k
Sending Bitrate (MBits/s)	8.78	9.90	11.7	323
Receiving Bitrate (MBits/s)	8.77	9.89	11.7	323
TCP Retransmissions	0	0	1	0

It is clear that the gigabit ethernet capabilities of the Raspberry Pi far outweigh the ethernet capabilities of the board, achieving 26x more throughput than that of the quad-core VexRiscvSMP.

Keep in mind as well, the Raspberry Pi is throttled because of the 8Kb window size we set. Additionally, the core amount does have an effect on the bitrate as the quad core is 1.8MBits/s faster than the dual core, which is 1.12MBits/s faster than the single core. However, an overall improvement of 32% across cores is basically negligible since the absolute performance is still far below what is required for what is essentially a high-speed ethernet application. It is clear from this benchmark alone that the bitrate will be a significant bottleneck for the design.

4.1.2 Stress NG

Stress-ng is a versatile benchmarking tool designed to stress test various components of a CPU. The command:

```
stress-ng --cpu $CORE_COUNT --io 2 --vm 1 --vm-bytes 128M --timeout 60s
--metrics-brief
```

Runs a set of tests in parallel. The rest of the arguments determine the amount of tests and the type: `--cpu`, creates CPU-intensive tasks equal to the core count; `--io`, creates two I/O-intensive tasks; and `--vm`, allocates and uses 128MB of virtual memory. This will evaluate for us how the system performs under combined CPU, I/O, and memory pressure, as well as how these metrics vary with the amount of cores. Additionally, the test will timeout after 60 seconds or until 6·`$CORE_COUNT` CPU bogo ops have been completed. Here are the results:

Table 4.2: Stress-ng Comparison of Configurations

	VexRiscvSMP, 100MHz			RPi
	Single	Dual	Quad	4B 1GB
CPU bogo ops	6	12	24	12,483
CPU real time (s)	125.34	119.09	121.30	60.03
CPU usr time (s)	78.32	164.00	378.16	146.85
CPU sys time (s)	0.01	0.12	0.09	0.03
CPU bogo ops/s (real time)	0.05	0.10	0.20	207.94
CPU bogo ops/s (usr+sys time)	0.08	0.07	0.06	84.99
IO bogo ops	21,794	31,190	27,819	440,066
IO real time (s)	60.00	60.01	60.00	60.00
IO usr time (s)	2.74	3.56	3.42	10.34
IO sys time (s)	25.71	44.44	66.44	48.56
IO bogo ops/s (real time)	363.22	519.76	463.65	7,334.31
IO bogo ops/s (usr+sys time)	766.05	649.79	398.21	7,472.11
VM bogo ops	2,280	3,053	2,464	617,668
VM real time (s)	61.92	62.54	61.54	60.16
VM usr time (s)	7.57	10.72	18.10	27.27
VM sys time (s)	7.60	14.93	18.32	6.78
VM bogo ops/s (real time)	36.82	48.82	40.04	10,266.29
VM bogo ops/s (usr+sys time)	150.30	119.03	67.66	18,143.58
Total time taken (s)	125.45	120.31	122.76	60.00

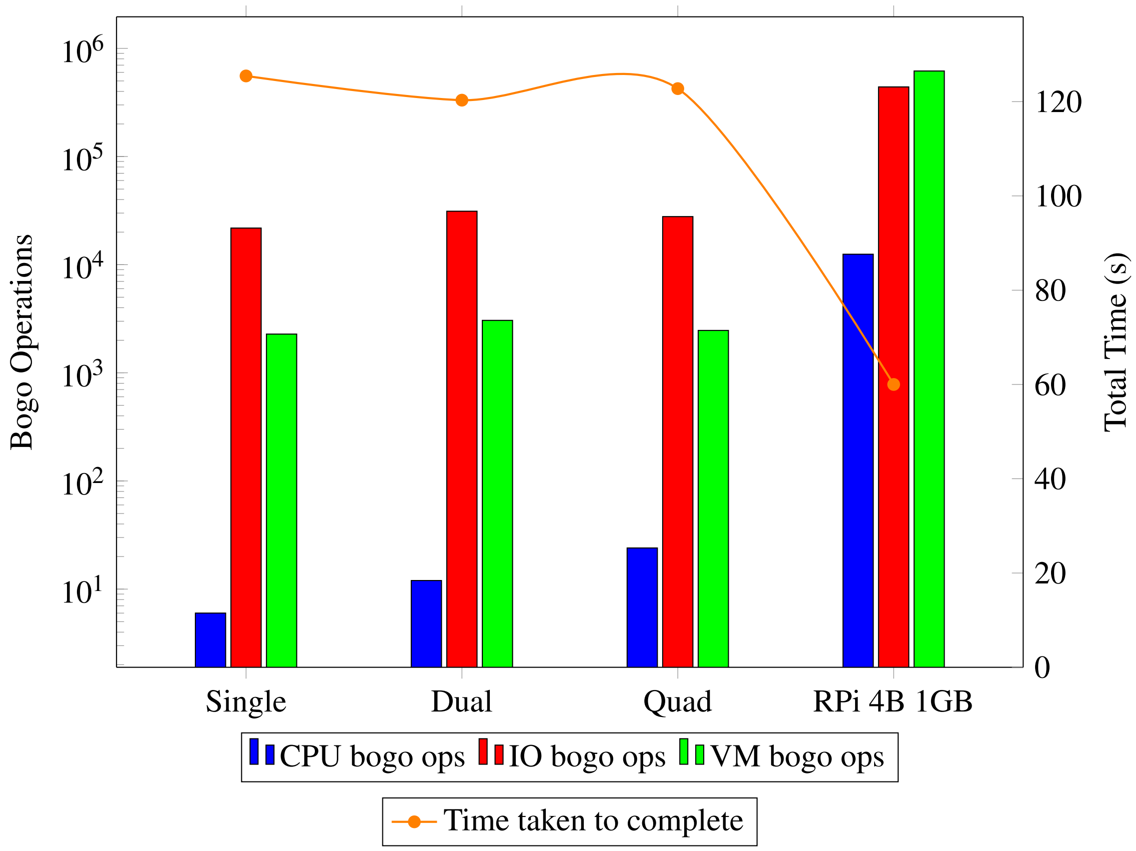

Figure 4.1: Stress-ng Performance Comparison Across Configurations

Immediately, we are presented with an extreme difference in performance scaling, which is to be expected when comparing an FPGA-based softcore processor to a Raspberry-Pi single-board computer. There is still nuance, however, between the different core configurations of the VexRiscvSMP. For instance, the amount of CPU bogo ops tend to scale linearly with the core count, maintaining a similar execution time (see ”CPU real-time” in table: 4.2). Other than this, there are no clear trends for the IO or VM bogo operations. It seems that once we extend to four cores, there are resource contentions with the IO bus and memory, leading to the quad core performing slightly less favourably than the dual core in these operations. Therefore, IO and memory have the potential to be another two bottlenecks, in addition to the low ethernet throughput, (although ethernet is still the dominant bottleneck). This can give us an estimation of how the dual core will perform relative to the quad core, *i.e.*, it may end up performing better since the quad core will not only have to contend with IO/memory constraints from the shared DRAM, but also the limited ethernet buffers.

make reference to hardware overview

Overall, out of the VexRiscvSMPs, dual core performed the best, but only if we exclude raw CPU power. As our application depends more on resource read/writes and management, dual core may prove to be the best configuration; besides the Raspberry Pi of course.

4.2 Results & Analysis

In this section, we will now compare how each configuration performed running the exact same full suite of tests on our application. We will then take the best performing out of the VexRiscv configurations and see if we can improve the design to achieve even more throughput.

Make reference to software setup

4.2.1 Test Suite Full Outline

The tests will focus only on encryption, since decryption and encryption times are the same, and we have proven that the device is encrypting properly. We are aiming to deduce how each configuration performs with single or multiple clients. Therefore, the test outline is as such:

add reference to proof of encryption

1. Basic test 1:
   1 client with 1 100mb file.
2. Basic test 2:
   10 clients with varying file sizes.
3. Basic test 3:
   20 clients with varying file sizes, in 10 client chunks
4. Large scale test:
   100 clients with varying files sizes, in 10 client chunks

While these tests are running, we are collecting a multitude of metrics regarding overall performance, but the primary result we are concerned with is the total bytes processed (or bytes encrypted) per second. This single metric is all we need to determine which amount of cores is superior for the application. It is a standard average, calculated from:

Make reference to database

$$Average\ Bytes\ Processed(Bytes/s) = \frac{Total\ Bytes\ Processed}{Test\ Duration}$$

Where *'Total Bytes Processed'* is the total amount of bytes processed during the test, and *'Test Duration'*, is how long the test took to execute. This is calculated from the server-side, as all the information is present there.

Make reference to schema

On the client-side, we are also keeping track of the client's total time for the transaction, time spent in the queue, time spent on network transfers and the time spent processing on the server, *i.e.*encrypting. Total time, network time and queuing time are collected in real time on client-side, leaving the processing time as the remainder, calculated via:

$$Processing\ Time(s) = Total\ Time - Network\ Time - Queue\ Time$$

Lastly, all timings were collected using the *'Time'*, Python standard library.

4.2.2 Single Core Results

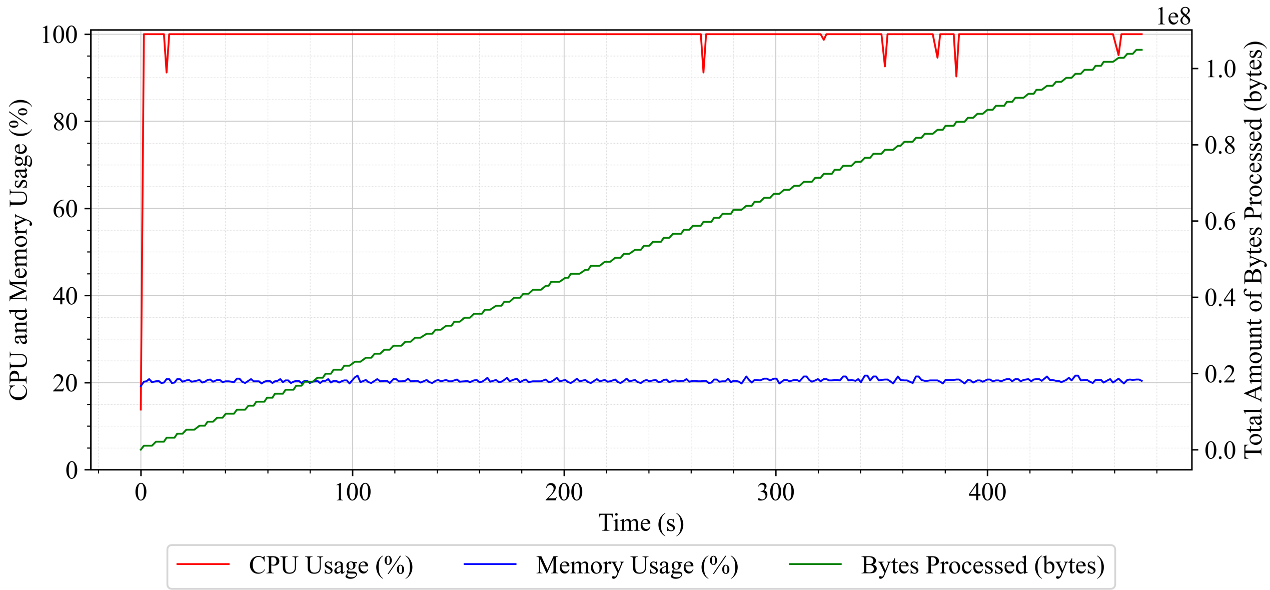

Figure 4.2: The University Of Queensland

Single Core Tests	Basic 1	Basic 2	Basic 3	Large Scale 100
Total Bytes Processed (Mb)	104.86	84.93	169.87	849.35
Avg Bytes per Second (Kb/s)	221.74	183.57	187.14	184.78
Test Duration (s)	472.89	462.69	907.70	4596.65
Avg Network Time (ms)	470.90	279.27	269.81	274.55
Avg Processing Time (ms)	2.55	0.17	0.14	0.13
Avg Queue Time (ms)	0.15	8.28	7.66	8.05

4.2.3 Dual Core Results

Dual Core Tests	Basic 1	Basic 2	Basic 3	Large Scale 100
Total Bytes Processed (Mb)	104.86	84.93	169.87	849.35
Avg Bytes per Second (Kb/s)	272.97	265.26	272.79	264.22
Test Duration (s)	384.13	320.19	622.70	3214.54
Avg Network Time (ms)	383.02	191.95	186.90	190.68
Avg Processing Time (ms)	1.07	0.13	0.13	0.13
Avg Queue Time (ms)	0.07	4.35	3.94	4.42

4.2.4 Quad Core Results

Quad Core Tests	Basic 1	Basic 2	Basic 3	Large Scale 100
Total Bytes Processed (Mb)	104.86	84.93	169.87	849.35
Avg Bytes per Second (Kb/s)	269.92	248.77	251.97	245.57
Test Duration (s)	388.48	341.42	674.16	3458.68
Avg Network Time (ms)	382.98	195.86	196.11	201.04
Avg Processing Time (ms)	0.88	0.13	0.13	0.13
Avg Queue Time (ms)	0.08	4.37	3.65	3.83

4.2.5 Raspberry Pi Results

4.3 Improved Dual-Core Design, Results and Analysis

Improved Dual Core Tests	Basic 1	Basic 2	Basic 3	Large Scale 100
Total Bytes Processed (Mb)	104.86	84.93	169.87	849.35
Avg Bytes per Second (Kb/s)	376.04	406.06	393.59	403.36
Test Duration (s)	278.85	209.17	431.59	2105.67
Avg Network Time (ms)	276.87	120.00	129.88	123.35
Avg Processing Time (ms)	1.05	0.14	0.14	0.14
Avg Queue Time (ms)	0.08	3.11	1.95	2.81

4.4 Utilisation, Resources and Timing

4.5 Power Usage

Chapter 5

Conclusion

Conclude your thesis.

Bibliography

Appendix A

Appendix

Write your appendix here. Following two are examples.

A.1 Name of Appendix-1

A.2 Name of Appendix-2

Endquote goes here.

Author of quote,
Source of quote