



# THE UNIVERSITY OF QUEENSLAND

A U S T R A L I A

## Optimising a RISC-V Task Scheduler for Dark Silicon Parameters

Thomas Steel - 43573136

June 11, 2020

Thomas Steel

thomas.steel@uqconnect.edu.au

22/06/2020

Prof Amin Abbosh  
Acting Head of School  
School of Information Technology and Electrical Engineering  
The University of Queensland  
St Lucia QLD 4072

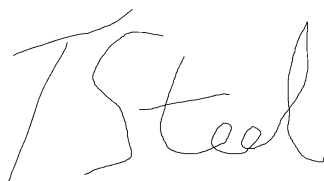
Dear Professor Abbosh,

In accordance with the requirements of the Degree of Bachelor of Engineering (Honours) in the School of Information Technology and Electrical Engineering, I submit the following thesis entitled

”Optimising a RISC-V Task Scheduler for Dark Silicon Parameters”

The thesis was performed under the supervision of Dr Matthew D’Souza. I declare that the work submitted in the thesis is my own, except as acknowledged in the text and footnotes, and that it has not previously been submitted for a degree at the University of Queensland or any other institution.

Yours sincerely

A handwritten signature in black ink, appearing to read 'T Steel', with a stylized, cursive script.

Thomas Steel

# Contents

<b>1</b>	<b>Thesis Definition and Scope</b>	<b>4</b>
1.1	Definition . . . . .	4
1.2	Scope . . . . .	4
<b>2</b>	<b>Background Theory</b>	<b>6</b>
2.1	Dark Silicon . . . . .	6
2.1.1	Obituary for Moore’s Law and Dennard Scaling . . . . .	6
2.1.2	The minimum energy for computing . . . . .	6
2.1.3	The Four Horsemen of Dark Silicon . . . . .	7
2.1.4	Memory Driven Computing . . . . .	8
2.1.5	The Dark Side of Silicon . . . . .	8
2.2	Task Scheduling . . . . .	9
2.2.1	Relationship with Dark Silicon . . . . .	9
2.2.2	Algorithms . . . . .	9
2.2.3	Methodologies . . . . .	9
2.3	Thermal Modelling . . . . .	9
2.3.1	Methods . . . . .	9
2.3.2	Limits . . . . .	10
2.3.3	Relationship with Dark Silicon . . . . .	10
2.3.4	Impact on Chip Design . . . . .	10
2.4	RISC-V . . . . .	10
2.4.1	The RISC-V ISA . . . . .	10
2.4.2	RISC-V Cores . . . . .	10
2.4.3	RISC-V Literature and Documentation . . . . .	10
2.4.4	Impact on Chip Design . . . . .	10
<b>3</b>	<b>Approach and Execution</b>	<b>11</b>
3.1	The Initial Plan . . . . .	11
3.1.1	Overview . . . . .	11
3.1.2	Software Packages Used . . . . .	11
3.1.3	Hardware Packages Used . . . . .	11

3.1.4	Project Plan . . . . .	11
3.1.5	Technology Readiness Level . . . . .	12
3.1.6	Algorithm Schemes . . . . .	12
3.1.7	Performance Indicators . . . . .	13
3.2	The Outcome . . . . .	13
3.2.1	Implemented . . . . .	13
3.2.2	Dead ends . . . . .	13
3.2.3	Unimplemented . . . . .	13
3.2.4	Discarded . . . . .	13
3.3	What Went Wrong . . . . .	13
3.3.1	Assumptions Made and their effect on the Project . . . . .	14
3.3.2	The Scope . . . . .	14
3.3.3	Key Events . . . . .	14
3.3.4	Forced Errors . . . . .	14
3.3.5	Unforced Errors . . . . .	14
3.3.6	Sunk Cost Fallacy . . . . .	15
3.4	A Revised Approach . . . . .	15
3.4.1	Possible Hardware . . . . .	15
3.4.2	Possible Cores . . . . .	15
3.4.3	Revised Timeline . . . . .	15
3.4.4	TLR . . . . .	15
3.4.5	Zephyr - an OS alternative . . . . .	16
3.4.6	Key Approach . . . . .	16
3.4.7	Refined Scope . . . . .	16
<b>4</b>	<b>Conclusion</b>	<b>17</b>
<b>A</b>	<b>Appendix</b>	<b>18</b>

# Chapter 1

## Thesis Definition and Scope

### 1.1 Definition

In modern computing, Moore’s Law has been an important design consideration.

Dark Silicon is the issue in modern computing where the assumptions of Moore’s Law and Dennard scaling break down. As the size of transistors reduces, we outpace our ability to cool the chips, leading to faster degradation when run for maximum performance. There is an additional energy-performance trade off, where chips cannot be powered at their maximum clock speed without exponential increases in energy requirements each generation. Alternatively, chips can use 40% less energy, with each generation but they will no longer be able to scale performance. This has led to an era constrained by Post-Dennard scaling, where doubling the number of transistors in a chip decreases the percentage of these transistors that are powered on at any given time.

The clearest description available that gathers the collective issues is found in the Dark Side of Silicon, by Rahmani et al., shown in Figure 1. While it is a simple model, it covers the collection of parameters that cause the dark silicon issue. The increasing power density of modern chips, without advancement in cooling technology, slowly leads to an increasing amount of silicon that cannot be powered. The term “dark silicon” refers to this silicon that is unpowered or being clocked significantly below its maximum frequency.

### 1.2 Scope

The research space for dark silicon stretches from material scientists designing 3D silicon architectures or new transistor designs, to firmware and software engineers optimising resource allocation on the chip for these new parameters that must now be considered. There are also hardware approaches to of dark silicon, that focus on harnessing the

principles to develop different styles of chips that steers into the metaphorical curve, using techniques such as heterogeneous cores, running cores at near threshold voltages and using computational sprinting to produce high throughput while conserving energy. The scope of this project is being strictly limited to what can be implemented on an FPGA. This obviously precludes all approaches related to material science. Reducing the size of the cores to combat dim silicon will not be considered as it doesn't harness the principles of dark silicon. Dim silicon methodologies such as computational sprinting and underclocking are within the scope, but dynamic voltage scaling is not, as it requires additional modules that add a significant amount of complexity to the design and testing. Heterogeneous multicores are in scope as they represent an important way to scale up computational efficiency. CPU scheduler design methodologies and resource allocation techniques are in scope as they will be required to capitalise on the optimisation at the hardware level. Finally, an approach to computing that weaves between several of the above methods known as memory driven computing is within the scope, as it represents an important possible future in the era of dark silicon.

# Chapter 2

## Background Theory

### 2.1 Dark Silicon

#### 2.1.1 Obituary for Moore's Law and Dennard Scaling

In his seminal document on the subject, Gordon E. Moore predicted an exponential increase in the number of transistors in an integrated circuit [1]. As shown in Figure 1, this exponential law has become less relevant in the past 20 years, in a way that is largely explained by factors that represent the downfall of Dennard scaling [2].

The companion of Moore's Law is Dennard Scaling. In 1974, Robert H. Dennard released a paper describing the scaling of MOSFET devices [3], a scaling which held alongside Moore's Law until decreasing transistor size resulted in an increase in the significance of physical constraints on the scaling of circuits. In a 33 year retrospective in 2007, assumptions that underlay Dennards scaling failed. For example, Dennard assumed that there would be a continual increase in channel doping concentration that would allow continuously shorter channel lengths. However, as the doping concentration increases, impurity scattering causes degradation of carrier mobility and performance [4].

#### 2.1.2 The minimum energy for computing

In the last century, it was proven by Charles H Bennet and Ralph Landauer that for any irreversible computation, there is a minimum energy required required per switching equal to  $kBT\ln 2$ , approximately 10-21J at room temperature. This is called the Shannon-von Nuemann-Landauer limit, and it describes the physical limits of switching elements, such as a CMOS transistor. This is relevant because it helps to explain the issue of dark silicon: that the transistors are not, in modern microprocessors, the primary source of energy consumption. Energy consumption in electronics is primarily caused by electrical capacitances. In modern microprocessors in the 22-65nm node range, 20-30% of the total energy consumed is attributed to transistors, a percentage that is expected to reduce as

transistors become smaller. This is caused by an inverse relationship between leakage currents and transistors size. The other significant losses are due to interconnects (e.g. wires) [5]. The wire interconnects can be modelled as capacitive lumps in the case of short wires, lossy RC transmission lines in the case of medium length wires, and long global wires need to be modelled as lossy RLC transmission lines. As scaling increases, these wire interconnects become the dominant limiting factor in terms of clocking frequency, delay, power and area [5, 6].

### 2.1.3 The Four Horsemen of Dark Silicon

As Dennardian scaling breaks down, the period of Post-Dennardian scaling arises, shown in Table 1. There is an exponential reduction in the percentage of silicon in a chip that can switch at full frequency that decreases with every generation. An important result of this is that multicore scaling becomes prohibitive as it produces large amounts of dark silicon. Taylor, in an early survey of the dark silicon problem, defined 4 “horsemen” as solutions for the challenge of dark silicon. Ordered from most to least relevant, these are

- The Specialised Horseman defined by a new age of Heterogeneous core design
- The Dim Horseman focusses on spreading the same power budget over more transistors
- And the Shrinking Horseman, which removes dark silicon by reducing the size of the die
- The Deus Ex Machina Horseman where technological advancement nullifies the issue.

The Specialised Horseman relies on the principle that ASIC circuitry is usually 100 to 1000x faster than general processing units, and that general processing units typically use 157 to 707x the energy of an ASIC circuit for a similar task [7]. Heterogeneous cores and computational accelerator are both possible answers to the inefficiencies of modern CPU’s [8, 9].

The Dim Horseman, focussed on dim silicon methodology, relies on underpowering or underclocking portions of the chip. Near Threshold Voltage processors and Dynamic Voltage and Frequency Scaling are both successful methods for this. Near Threshold Voltages, the voltage across the transistors is reduced to a point that is optimal for energy delay. This point is the lowest voltage that can be used without requiring a steep drop in frequency. Although intriguing, it requires additional hardware that increases the complexity of this project [10]. Whilst Dynamic Voltage Scaling is precluded for the same reason, Dynamic Frequency Scaling can be implemented with appropriate use of prescalers and will be pursued as a possible source of optimisation.

Another popular method is computational sprinting, where the chip spends most of the time in a low-performance state but moved into a high-performance state for brief periods, or a second core is activated for a brief period. This brief period is a sprint and can be implemented to increase the energy efficiency as the resource of the extra CPU can be



allocated as needed, and short operations can be sprinted through allowing the chip to go idle afterwards. This helps to manage the energy and thermal issues of dark silicon [11]. The Shrinking Horseman refers to the design approach of simply designing smaller processors using less chip space to not go above the power budget on the chip.

The Deus Ex Machina horseman represent new technologies, such as the QFET, the GAA transistor model, or perhaps the transistors that can be achieved through using diamond as the conducting material. Future projects on optimising a CPU scheduler for their parameters would be compelling. It stands to reason, these things being the focus of data science, that this is entirely out of scope [9].

### 2.1.4 Memory Driven Computing

Memory driven computing occurs as a solution to the intense power usage of interconnects and memory blocks. In standard computing, memory is stored at varying distances and sizes with the focus being on the computing unit. In memory driven computing, the memory is the focus and the processing is on the periphery. This is an intuitive answer to the problems presented by an increased prominence of wire and memory costs. As computing move into a period where there are diminishing returns on computational performance due to externalities, switching the design focus to these externalities could prove to be a bountiful approach [12].

### 2.1.5 The Dark Side of Silicon

In the Dark side of Silicon, Rahmani et al. describe a suite of methodologies for managing dark silicon. The heterogeneous methodology gives more direction than that of Taylor, but the section of interest is the management in the OS layer. Variation-Aware Core Selection and Scheduling is a methodology for efficient resource allocation within the power budget constraints of dark silicon [8, 13]. The Dark Side of Silicon suggests the SiLago methodology for implementing and exploiting heterogeneity. In a problem analysis of the industry, 4 issues became apparent:

- Most design focussed on the cores, ignoring the memory and interconnects
- Software abstraction relies on naïve resource allocation, resulting in inefficiencies at runtime
- The lack of dynamic runtime customization means that non-deterministic concurrency and communication patterns result in inefficiency power usage and parallelism
- Customisation has a large engineering cost that make it prohibitive.

The SiLago platform attempts to solve these problems by implementing a Distributed Memory Architecture. This is a form of memory driven computing and has been shown to have significant reductions in power and energy expenditure if done properly [14].

Similar methodologies have found that by arbitrarily spacing memory throughout a chip, the power cost of on-chip interconnects can be severely reduced [15].

## 2.2 Task Scheduling

### 2.2.1 Relationship with Dark Silicon

### 2.2.2 Algorithms

CPU scheduling is concerned with a number of things

- CPU utilisation, keeping the CPU running as often as possible
- Data throughput, a measure of CPU utilisation
- Turnaround time, the period it takes for a process to be completed
- Waiting time, how long the process spends waiting to use the CPU,
- And response time, the time between when a process is put onto the queue and when the first response is produced

Algorithms for scheduling tend to focus on the priority of tasks first, and the allocation of resources second. An example of this is a Priority Scheduler, which weighs the priority of incoming tasks and parcels out operating time based on precedence of priority. A more complex and relevant method however is a Multilevel Feedback Queue, which prevents any single process from taking up too much time [16].

### 2.2.3 Methodologies

## 2.3 Thermal Modelling

### 2.3.1 Methods

Given the nature of dark silicon, it seems intuitive to use heat as a metric for progress. This approach is hobbled by the stumbling block that is the difference between the physical model of a microprocessor and an FPGA. While it is possible to gain considerable improvements to efficiency with thermal aware design designing, this increases the complexity and requires optimisation and tuning that is device specific. Gains from thermal aware design would not necessarily translate between devices, and so more translatable measures and improvements will be the focus of this project [17, 18].

### **2.3.2 Limits**

### **2.3.3 Relationship with Dark Silicon**

### **2.3.4 Impact on Chip Design**

## **2.4 RISC-V**

### **2.4.1 The RISC-V ISA**

Placeholder for a description of the RISC-V ISA and its qualities.

### **2.4.2 RISC-V Cores**

Placeholder for a outline of available RISC-V cores.

### **2.4.3 RISC-V Literature and Documentation**

Placeholder for a section on the theory behind implementing a RISC-V system.

### **2.4.4 Impact on Chip Design**

# Chapter 3

## Approach and Execution

### 3.1 The Initial Plan

#### 3.1.1 Overview

The initial plan was to upload the RPU core onto an FPGA, confirm a basic working model, then upload an OS and start running test units.

#### 3.1.2 Software Packages Used

**VMWare** - Used to handle the virtual machine

**Ubuntu 20.4** - Used as the OS for RISC-V compilation.

**RISC-V GNU Toolchain** - To compile RISC-V code.

**Python** - For scripting and to transpile RISC-V ASM into 32 bit machine code for RPU.

#### 3.1.3 Hardware Packages Used

**Computer** - This project required a Vivado setup and a virtual machine to handle the RISC-V installation.

**Rice University WARP2** - Initial FPGA

**Nexys 4 DDR** - A FPGA board was used to implement the core.

#### 3.1.4 Project Plan

The project milestones will mostly be marked in hardware, with hardware and firmware development being done in tandem. The goal is to develop several separate approaches (computational sprinting, heterogeneous cores and memory driven computing focusses) alone and then to test combined syntheses of these approaches, with the most promising design being using to implemented and tested on complex datasets.

## Project Schedule and Timeline

August 2019	RISC-V softcore implemented on FPGA, with running Linux Kernel.
September 2019	Control and sensitivity testing of metrics Control data collected from Linux.
October 2019	Begin optimisation for memory driven computing.
November 2019	Testing basic datasets for memory driven and computational sprinting.
December 2019	Begin work on heterogeneous cores.
February 2020	Test basic datasets with heterogeneous cores and computational sprinting.
March 2020	Test basic datasets on heterogeneous cores with memory driven computing.
April 2020	Implement project as general computer with TRL of 6.
May 2020	Poster and demonstration prepared.
June 2020	Thesis completed and submitted.

### 3.1.5 Technology Readiness Level

The goal is to achieve TLR 6 on the following scale. TRL 1 – Task scheduler optimised, metrics show improvement over naïve resource allocation.

TRL 2 – Task scheduler showing clear links between target optimizations and metrics and can process basic datasets.

TRL 3 – Task scheduler can be used to process moderately complex, sanitized datasets.

TRL 4 – Task scheduler running Linux with evidence of optimisation

TRL 5 – Task scheduler, under controlled circumstance and settings, can be used as a basic general-purpose computing system, shown to work on range robust datasets.

TRL 6 – Task scheduler performing optimal resource allocation for a variety of datasets. Can be used as general use computing system.

### 3.1.6 Algorithm Schemes

The initial plan was to implement a round robin scheduler, a priority queue scheduler and a hybridised scheduler, to produce diversity of data. The testing was to consist first of a Fast Fourier Transform (FFT) during the proof of concept stage, the results of which would be used as a heuristic to judge the quality of iterative improvements. After modifications had been made to the algorithms and the core, the goal was to use Dhrystone, Geekbench and CoreMark to test the quality of the different dark silicon methodologies in use.

While some of these programs have received criticism for the quality and usefulness as benchmarks, for the use of a before and after dataset, they are adequate.

The Variation-Aware Core Selection and Scheduling will also be an important methodology for reducing the amount of naïve scheduling caused by abstraction away from the hardware layer [13].

### 3.1.7 Performance Indicators

A basic control dataset will be produced using the RISC-V core with a basic Linux system run on the softcore. This will test for:

- Data Throughput
- Power and energy requirements
- Area of chip used
- Clocking Frequency
- Cycles per operation
- Quality of Resource allocation

These metrics will be used as a baseline to contrast the CPU core against as it is developed.

## 3.2 The Outcome

To do.

### 3.2.1 Implemented

Placeholder to outline implemented sections of the project.

### 3.2.2 Dead ends

Placeholder for things that were attempted but abandoned.

### 3.2.3 Unimplemented

Placeholder for things that were not implemented

### 3.2.4 Discarded

Placeholder for things that were discarded - may be folded together with dead end section.

## 3.3 What Went Wrong

This thesis was supposed to generate a working model of a soft core processor running an OS. This has not come to pass, and so it is important to evaluate the combination of

forced and unforced errors that accumulated during this project as well as events outside of the authors control, to contextualise the timeline.

### **3.3.1 Assumptions Made and their effect on the Project**

Placeholder for list of assumptions.

### **3.3.2 The Scope**

Placeholder for a review of things in the scope of the thesis that should have been added or removed.

### **3.3.3 Key Events**

A trio of large events impacted time management and productivity during this project. The first was a medical event which negatively impacted the first semester, the second was house move that impacted the end of year holiday period and the final event was the global pandemic of COVID-19 which impacted the second semester.

### **3.3.4 Forced Errors**

A large drawback in this project was the unexpected complexity of setting up the environment. The initial hardware I was given required a specialised but depreciated environment, which took a considerable amount of time to set up first on Windows 10, and then again on a Windows 7 Virtual Machine, only to discover that the full installation would take a debilitating time to set up. My supervisor suggested a moved to an alternate piece of hardware, which resolved the issues, but by this stage several weeks were already casualties to fruitless debugging and forum based research. The new hardware had an easy environment to set up and at the end of the semester the project had a working environment, but behind schedule on additional research and implementation.

### **3.3.5 Unforced Errors**

The holiday period was extremely dense on work and preparations for moving houses, so an assumption was made that the thesis work could be caught up with during the second semester. The unforced error here was not pushing through additional work during the holidays. While it was impossible to predict the COVID-19 pandemic, further work done during this period would have offset issues faced a later junctures. It also would have helped to discover dead ends earlier in the project, allowing for appropriate replanning. Beginning the second semester, the plan was to catch up on additional research and spend

the first several weeks streamlining the rest of the semester so a majority of the semester could be spent on thesis work, as would be required. Some progress was made in teasing out the workings of the RPU core, but a majority of work was put into preparing for the mid-late period of the semester.

The other unforced error was the choice of RPU due to its low level nature and readiness for adaptation. Choosing a harder to modify, but easy to set up processor would have been preferable, as the difficulty of implementing an OS on RPU ultimately was a massive detraction from the project.

### 3.3.6 Sunk Cost Fallacy

During this project there were many dead ends and junctures where a choice had to be made to continue with an approach that wasn't working or abandoning lost time and finding a new approach. During the first half of the project many dead ends were hit and methods abandoned in the hope of greener pastures that never appeared. During the second half of the project, an attempt was made to stick to a single approach, based on the seemingly reasonable assumption that that method would pay off.

## 3.4 A Revised Approach

### 3.4.1 Possible Hardware

**SiFive Board** - One of the more advanced RISC-V capable boards, these have a lot of support and a lot of pre-done implementations. **ArtyS7** - This board is used and support by many RISC-V projects. Uses the same Artix-7 FPGA chip as the Nexys 4 DDR.

### 3.4.2 Possible Cores

**VexRISC-V** - Initially dismissed because it is written in SpinalHDL, a Scala based HDL, this core seems very capable.

**AndesCore** - Initially dismissed because it is written in verilog and listed as having no OS capabilities.

### 3.4.3 Revised Timeline

### 3.4.4 TLR

Unchanged.



### **3.4.5 Zephyr - an OS alternative**

Zephyr is not listed on the RISC-V site, but is a viable OS for several RISC-V cores.

### **3.4.6 Key Approach**

Instead of trying to build upwards from a basic core up to an core running a kernel, it is advisable to start with an already working system. There is a lot of complexity to work out in this approach but it means that single core methodologies could be easily implemented and there is model to compare multicore adaptations against.

### **3.4.7 Refined Scope**

Placeholder for a refined scope for the project. Best scope would be in 2 flavours: split into a multicore approach that focusses on priority task scheduler, and a single core approach that varies the task scheduling and implements computational sprinting.

# Chapter 4

## Conclusion

Placeholder.

**Appendix A**

**Appendix**