Low Power Synthesis Techniques

Henry Lovett

***Abstract*—This paper explores a number of low power techniques used in digital systems. Each technique is explained, giving the underlying theory of operation, and the synthesis techniques used to realise the technique. This report investigates power gating, power scaling, frequency domains, frequency scaling and clock gating,**

***Index Terms*—Low Power, Clock gating, power gating, frequency domains, frequency scaling, power scaling.**

I. Introduction

The desire for low power devices has been driven by the mobile age. Companies are competing on battery life of portable devices, such as smartphones and tablets. This drive for low power has resulted in different synthesis techniques. This paper will review and explore some of the techniques used to help reduce the power consumption of a circuit, with particular interest on the synthesis techniques used to do so.

There are two main categories of power consumption - dynamic and leakage. Dynamic power is the power consumed when the circuit is enabled and functioning. Power is used here through the charging and discharging of the internal capacitors of the module. Leakage power is due to the non-ideal characteristics of sub-micron CMOS transistors. Leakage power mainly occurs when the module is in a idle, non-switching state [1] and can contribute around 22% of the total power consumption of a $90nm$ FPGA [2]

The report begins with a review of different techniques in turn. This includes a brief introduction to the theory and a discussion of synthesis problems and solutions. The report concludes by reviewing all techniques and their relevant advantages and disadvantages.

II. Power Techniques

A. Power Gating

Leakage currents begin to occur in sub-micron technologies, typically lower than 100nm [1], [3]. With each technology generation, gate leakage increases 30 fold [4]. Leakage currents can are identified as two sources - sub-threshold currents and gate tunnelling. Sub-threshold currents occur when the voltage applied to the gate is lower than the transistor threshold voltage. The channel of the transistor during weak inversion still allows some current to flow from source to drain. This current occurs when the transistor is off and is also exponentially dependant on V_{th} [5]. Gate tunnelling currents increase with thinner oxide layer [6]. A thin oxide results in electrons being able to pass through the gate into the substrate of the transistor. Gate tunnelling occurs when the transistor is active [3]. Figure 1 shows where these leakage currents occur on an NMOS transistor.

1) Theory: Power gating, in principle, is where the power to a module is switched off when not in use. By doing this, the module does not consume any power. It can be achieved by using either a header or a footer switch to disable either the supply connection or the ground connection. Figure 2 shows the realisation of the power gating circuits.

Although the theory of operation is simple, the technique poses many issues in the implementation. Firstly,

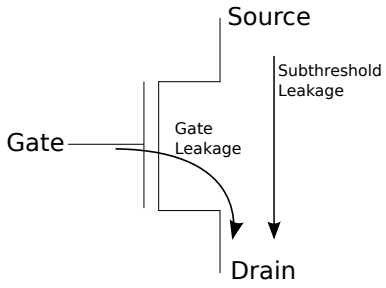

Fig. 1. The locations of gate and subthreshold leakage on an NMOS transistor

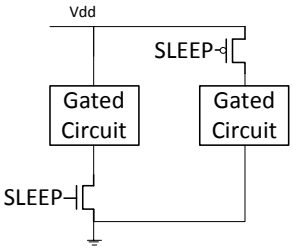

Fig. 2. Circuit diagrams showing the use of footer switches (left) or header switches (right).

Break this into two subfigures

as the module is floating, the outputs are undefined. This can cause the gates in another module to short circuit - at an input voltage of $V_{dd}/2$, both the NMOS and PMOS transistors will be on, resulting in a direct line from supply to ground. The solution to this is is to add logic gates with a 'clamp' signal. This could be either an `AND` gate, of which the clamp signal is active low, or an `OR` gate with an active high clamp. This added logic is needed per output signal of the power gated module.

The second issue that power gating raises is when the gated module contains sequential logic. By removing the power to the sequential logic, the state is lost. This can then cause issues as state retention may be necessary, as well as low power. The solution is to use a state retention register. There is a timing overhead involved to store the register before putting the device to sleep, disabling the majority of the circuit. State retention registers also require an ungated power supply, meaning all power gated modules require two supplies; one which is gated and one constant supply.

The final aspect of power gating that needs addressing is the power management. The power manager is responsible for when to gate the module.As well as the control of the sleep transistors, the power manager is also responsible for the stopping the clock, clamping the outputs and saving the state to the state retention mechanism.

There is a large overhead in the additional circuitry needed for power gating. The main increase are the state retention registers as these can require 20% more area in silicon [7]. Other techniques exist to avoid this, such as the use of the scan path to quickly clock the state in and out to/from memory. Digital signal processing, however, would not need this as it is data dependent. Processors with cache have a large amount of state data, so state retention registers are needed here.

When footer switches are used, the voltage that at the source of the NMOS is known as the virtual ground (VGND). When the footer is active, VGND $\approx$ GND. When off, VGND $\approx V_{dd}$. A full realisation of a power gated circuit is shown in figure 3.

2) Synthesis Techniques: Power gating can be split into two techniques, fine and coarse grain. Fine grain gating is where individual cells (e.g. in ASIC) have their own gating transistor. Coarse grain techniques are where gating transistors are applied on a larger module scale. [3] compares the fine grain and coarse grain methods on a 65nm FPGA technology. Here, a SRAM block is used as the test circuit. Fine grain, in this case, is where the memory of each cell is individually gated, and coarse is where a 16×1 SRAM module is gated. It

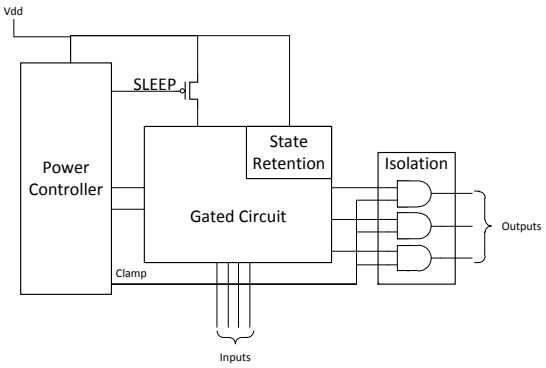

Fig. 3. The realisation of a power gated module. State retention, isolation and a power manager are all needed

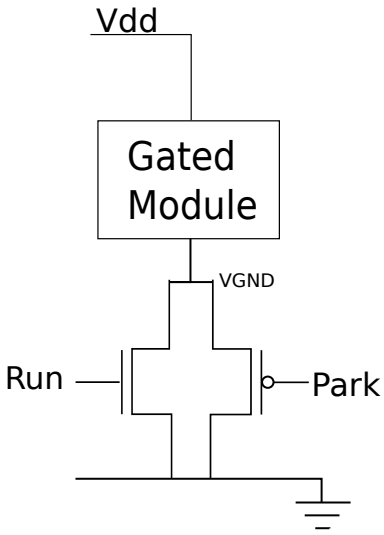

Fig. 4. Middle power state implementation used in [11]

was conclusively shown that coarse grain gating reduced power consumption more than fine grain due to being able to turn off the read/write circuitry.

When deciding to sleep a module, there are a number of overheads. Due to the charging / discharging of the gated circuit (depending on the use of header or footer switches), when the circuit is re-enabled, there is an amount of time needed to allow the capacitors to return to normal [8]. This is referred to as the wake-up time and can also include the time to restore state if applicable. The (dis)charging can also poses ground bounce issues to the device due to the sudden draw of current, [9], [10].

This wake up time can then impact the performance of the device - if more energy is consumed waiting for the disabled module than it would have consumed being awake, the overall power is not reduced. [11] discusses a method where an intermediate power saving mode is introduced. This is done by using a standard footer power gating NMOS. An extra PMOS is then added in parallel with the power gating switch. Table I summarises the states of the transistors in the circuit in figure 4

The RUN and COLD states operate as previously discussed. The PARK state, however, is where the virtual ground voltage does not rise to V_{dd} due to the PMOS conducting. The virtual ground is at the threshold voltage of the PMOS, so the leakage current is less than in the RUN state, but not so high that state retention is needed. This drastically reduces the wake up overhead of the circuit.

TABLE I
POWER SAVING MODES OF [11]

State Name	NMOS State	PMOS State	Virtual Ground Voltage (V)
RUN	ON	OFF	GND
COLD	OFF	OFF	V_{DD}
PARK	OFF	ON	V_{th} PMOS

A further improvement to this single extra state is proposed in [12]. Here, multiple intermediate power states are implemented by using different sub-threshold gate voltages on the NMOS footer switch. The compromise of wake up time against leakage reduction can then be more finely controlled to best suit the circuit. In general, the closer the virtual ground is to ground, the more leakage current occurs, but a faster wake up is available.

B. Power Scaling

I want to try and find use of two modules, one with a Low V_t for high performance, high power, and a high V_t for low performance, low power, which can be switched between. If not, just look at multi V_t technologies

Or I could look into adaptive body bias?

1) Theory:

2) Synthesis Techniques:

C. Power Domains

1) Theory:

2) Synthesis:

III. Clock Techniques

A. Frequency Domains

Lorem Ipsum...

B. Frequency Scaling

Lorem Ipsum...

C. Clock Gating

1) Theory: The clock in a sequential circuit can consume 15-45% of the power [13]. Therefore it is a large area of potential power saving. Clock gating is an approach of controlling the clock to individual modules of a design by either stopping or slowing down the clock with respect to a master clock [14]. An approach, seen in [15], involves stopping the clock to unused modules.

This method can be realised using two simple circuits seen in figures 5 and 6. Although figure 5 is functionally complete, in reality, a latch is needed to remove any glitches in the circuit. These are fundamentally different to load-enable registers, where the input is multiplexed between the current value or the input. The load-enable registers are still clocked at the master clock frequency.

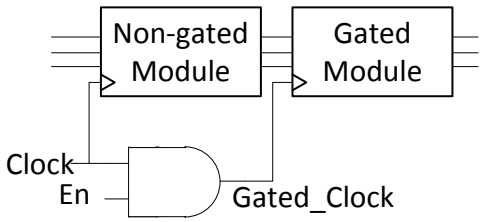

Fig. 5. Clock gating circuit using an AND gate

Gated Module or Gated Circuit? Consistency with the other diagrams

Is this even needed?

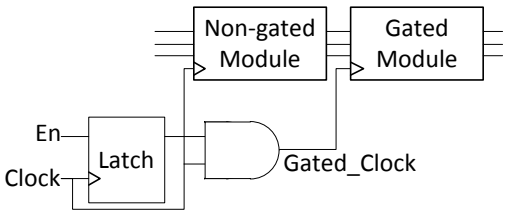

Fig. 6. Clock gating circuit using an AND gate and a latch

Gated Module or Gated Circuit? Consistency with the other diagrams

2) Synthesis Techniques: A gating function is typically defined by the designer within the RTL design stage. However, a more common approach is to allow the synthesis tool to obtain the gating functions from a gate-level netlist [16], [17].

The general outline for the synthesis is to find the clock gating function for each flip-flop. The flip-flops are then grouped so that they are driven by the same function. The problem of simplifying the gating function is looked at in [18]. Here, an algorithm is suggested where the gating function is shared by existing combinational logic. This was shown to reduce the logic added

by introducing clock gating.

However, sometimes the addition of clock gating is not advantageous. The gating function can be large and therefore can cause timing violations, resulting in an unsuitable synthesis. If the gating function is large enough, it can also consume more power than it saves. Both of these issues are addressed in [19]. Here, the author proposed solutions to large gating functions by reducing the depth of logic by an approximation. The approximation is made such that the resulting logic does not gate the clock more than the original function. It results in the flip-flop being clocked more often, but reduces the logic so that it can be utilised, thereby saving some power. [19] also proposes the use of a clustering algorithm that looks at grouping similar gating functions. This can then reduce the overall logic needed to implement the clock gating and maximise the energy saved.

3) Conclusion: Clock gating is a simple principle to implement on a small scale. The underlying theory is to disable a module when it is not in use. This is done by identifying a gating function which disallows clock propagation if the function is asserted.

However, clock gating can produce large gating functions which violate the timing constraints of the circuit, or even consume more power than they save. This results in two problems that need solving - group formation and simplification [18], [20].

IV. Asymmetric Multi-Cores

A. Theory

get buzz word 'hetrogeneous' in here a few times

Discuss symmetric, performance asymmetric and asymmetric multi-cores Multi-core architectures are not a new concept. A multi-core processor is a single die with two or more independent identical CPUs. They can have a shared cache and be connected on a bus. An asymmetric multi-core processor is a chip which has two or more independent CPUs (usually two). The difference to a symmetric multi-core processor, is that the cores are not the same. The cores can differ in cache, clock frequency, area and power, but maintain the same Instruction Set Architecture [21]. The cores therefore can have different optimisations.

Asymmetric multi-core architectures can be split into two categories - function or performance asymmetry [22]. Performance asymmetry is where the cores differ only by their performance, and subsequently power. This is achieved by having a multi-core processor with cores that can be individually gated, or scaled with DVFS. The Intel Single-chip Cloud Computer is such a device that supports this [23].

Functional asymmetry is where the processor has heterogeneous with memory, architecture (and usually also performance). One large upcoming functional asymmetric multi-core architecture is the ARM big.LITTLE. The architecture of the device is seen in figure 7. There are two cores in the big.LITTLE - a Cortex A15 and Cortex A7. The A15 is a high performance, high energy consumption, out of order execution processor.The A7 is the opposite of the A15 as it is an in order, energy efficient processor, and a compromise of performance. This asymmetric configuration allows performance critical tasks to be performed on the larger core, which can be powered down during periods of inactivity while the smaller, energy efficient core conducts other, non performance reliant tasks. big.LITTLE implements clock and power gating, and Dynamic Voltage and Frequency Scaling on the cores to aid the energy savings.

discuss cache differences

Discuss pros/cons to AMC

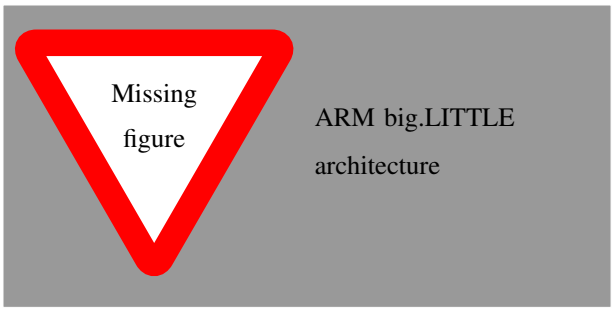

Fig. 7. ARM big.LITTLE Architecture diagram

B. Scheduling Techniques

The main issue to asymmetric multi-core architectures is around the scheduling of tasks to the correct core. Parallel programming also is not directly applicable to a asymmetric architecture. This is due to the assumption of symmetry in parallel programming - that each task would take the same time on any core. This is not so due to the performance difference between the cores.

A primitive, but reasonable, scheduler is outlined in [24] and [25]. Here, a hard fast allocation is made. Memory intensive workloads are given to the small core and computationally heavy tasks are allocated to the larger core. This is beneficial as memory intensive tasks are generally slow due to the access delays to the memory. However, with varying sizes of caches between the architectures, this method is suboptimal. A more in-depth method is necessary, looking at more than compile time directives.

Run time can provide useful information of the tasks. [26] proposes a scheduler for an asymmetric multi-core processor which uses the history of the tasks to allocate which core to schedule to. A task stealing policy is also implemented. This allows the bigger core to take over a task from the smaller core if it takes a long time. Task stealing is also scalable. [26] showed that their algorithm gave an 82.7% increase in the performance of the processor by using history based scheduling.

Scheduling can also be done at a higher level. [27] looks a popular mobile task - web browsing. Parsing web pages can take varying amounts of time, but are generally very complex tasks. An asymmetric multi-core set up is used using an ARM Cortex A9 for the ‘big’ core and an A8 for the ’little’ core. It shows that even at a high level of task, a big/little configuration can yield energy savings without a drastic performance decrease compared to performance focussed systems.

Virtual machines (VMs) are commonly used on servers. Some VMs can perform intensive tasks, where others do not need large performance capabilities. However, it can be difficult to predict when and if a VM needs performance, as the applications run may not be known to the host. [22] proposes a task scheduler for use on an asymmetric multi-core. Current schedulers are not aware of the heterogeneities of the processor cores. The scheduling algorithm here uses only run time characteristics of the virtual machine, but achieves between 13.5% and 55% increase in performance per watt.

A combination of compile time and run time can aid optimisations further. [21] gives a brief overview of using these measurements to the tread scheduling as well as lower level energy saving aspects such as clock gating and DVFS.

In the ARM big.LITTLE architecture, the two cores differ greatly. Not only the frequency of the core is different, but there are differing numbers of pipeline stages, cache sizes and branch predictions. The A15 also has cache miss and branch misprediction counters, where as the A8 does not. The main thing the two cores have in common is that they share the same instruction set. Due to the differences of the cores, it is difficult to predict

the performance of a task on a core. One technique used is looking at the Cycles per instruction of the core [28]. This gives an insight into the differences of the cores. This measure can then be used to predict the power consumption of the task when run on the other core. However, this is unable to identify data dependencies of the task. Compile time directives are therefore used to help predict stalls in the pipeline.

Performance and energy consumption are not the only things to be considered on a low energy core. The thermal characteristics can also be important, especially in mobile devices where cooling is not generally available. [29] suggests a scheduler where the large core is used sparingly, and all cores are run at minimum frequency when possible. A power budget is implemented here also, which is to reduce the thermal expenditure of the device. If the power budget is spent, then the performance of the device is lowered until it is possible to recover. This concept is a different view on the energy consumption, but involves solving the same problem.

C. Conclusion

Asymmetric multi-core architectures are very promising to achieving a good compromise between the performance and energy consumption of a processor. The main issue that this architecture faces is correctly scheduling the tasks to the correct cores. If the allocation is sub-optimal, then the asymmetry will be a hindrance to the processor. It has been shown, however, that correct task allocation results in energy savings, compared to non-symmetric architectures. The scheduling can be done purely on run time statistics to a satisfactory level, but by utilising hardware counters available and compile time directives, the energy savings can be much larger.

V. Other Techniques

Brief description of some other techniques. Found a bit on GA's and the such. Worth talking about if I have space?

1) Genetic Algorithms used in synthesis

VI. Conclusion

Lorem Ipsum...

References

[1] A. A. Bsoul and S. J. Wilton, "An fpga architecture supporting dynamically controlled power gating," in *Field-Programmable Technology (FPT), 2010 International Conference on*. IEEE, 2010, pp. 1–8.

[2] Altera, "Statix ii vs. virtex-4 power comparison & estimation accuracy," 2005, online at http://www.altera.co.uk/literature/wp/wp_s2v4_pwr_acc.pdf.

[3] P. S. Nair, S. Koppa, and E. B. John, "A comparative analysis of coarse-grain and fine-grain power gating for fpga lookup tables," in *Circuits and Systems, 2009. MWSCAS'09. 52nd IEEE International Midwest Symposium on*. IEEE, 2009, pp. 507–510.

[4] K. Bernstein, C.-T. Chuang, R. Joshi, and R. Puri, "Design and cad challenges in sub-90nm cmos technologies," in *Computer Aided Design, 2003. ICCAD-2003. International Conference on*. IEEE, 2003, pp. 129–136.

[5] S. Borkar, "Design challenges of technology scaling," *Micro, IEEE*, vol. 19, no. 4, pp. 23–29, 1999.

[6] W. M Arden, "The international technology roadmap for semiconductorsperspectives and challenges for the next 15 years," *Current Opinion in Solid State and Materials Science*, vol. 6, no. 5, pp. 371–377, 2002.

[7] C. Apte, "Power gating implementation in SoCs," University of California, Los Angeles, Tech. Rep., 2014, online at http://nanocad.ee.ucla.edu/pub/Main/SnippetTutorial/PG.pdf.

[8] A. Abdollahi, F. Fallah, and M. Pedram, "An effective power mode transition technique in mtcmos circuits," in *Proceedings of the 42nd annual Design Automation Conference*. ACM, 2005, pp. 37–42.

[9] S. Kim, S. V. Kosonocky, and D. R. Knebel, "Understanding and minimizing ground bounce during mode transition of power gating structures," in *Proceedings of the 2003 international symposium on Low power electronics and design*. ACM, 2003, pp. 22–25.

[10] Y.-S. Chang, S. K. Gupta, and M. A. Breuer, "Analysis of ground bounce in deep sub-micron circuits," in *VLSI Test Symposium, 1997., 15th IEEE*. IEEE, 1997, pp. 110–116.

[11] S. Kim, S. V. Kosonocky, D. R. Knebel, and K. Stawiasz, "Experimental measurement of a novel power gating structure with intermediate power saving mode," in *Low Power Electronics and Design, 2004. ISLPED'04. Proceedings of the 2004 International Symposium on*. IEEE, 2004, pp. 20–25.

[12] H. Singh, K. Agarwal, D. Sylvester, and K. J. Nowka, "Enhanced leakage reduction techniques using intermediate strength power gating," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 15, no. 11, pp. 1215–1224, 2007.

[13] M. Pedram, "Power minimization in ic design: principles and applications," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 1, no. 1, pp. 3–56, 1996.

[14] Q. Wu, M. Pedram, and X. Wu, "Clock-gating and its application to low power design of sequential circuits," *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, vol. 47, no. 3, pp. 415–420, Mar 2000.

[15] G. E. Téllez, A. Farrahi, and M. Sarrafzadeh, "Activity-driven clock design for low power circuits," in *Proceedings of the 1995 IEEE/ACM international conference on Computer-aided design*. IEEE Computer Society, 1995, pp. 62–65.

[16] L. Benini, G. De Micheli, E. Macii, M. Poncino, and R. Scarsi, "Symbolic synthesis of clock-gating logic for power optimization of synchronous controllers," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 4, no. 4, pp. 351–375, 1999.

[17] A. P. Hurst, "Automatic synthesis of clock gating logic with controlled netlist perturbation," in *Proceedings of the 45th annual Design Automation Conference*. ACM, 2008, pp. 654–657.

[18] I. Han and Y. Shin, "Synthesis of clock gating logic through factored form matching," in *IC Design Technology (ICICDT), 2012 IEEE International Conference on*, May 2012, pp. 1–4.

[19] E. Arbel, C. Eisner, and O. Rokhlenko, "Resurrecting infeasible clock-gating functions," in *Design Automation Conference, 2009. DAC '09. 46th ACM/IEEE*, July 2009, pp. 160–165.

[20] S. Paik, I. Han, S. Kim, and Y. Shin, "Clock gating synthesis of pulsed-latch circuits," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 31, no. 7, pp. 1019–1030, July 2012.

[21] B. de Abreu Silva and V. Bonato, "Power/performance optimization in FPGA-based asymmetric multi-core systems," in *Field Programmable Logic and Applications (FPL), 2012 22nd International Conference on*. IEEE, 2012, pp. 473–474.

[22] Y. Wang, X. Wang, and Y. Chen, "Energy-efficient virtual machine scheduling in performance-asymmetric multi-core architectures," in *Proceedings of the 8th International Conference on Network and Service Management*. International Federation for Information Processing, 2012, pp. 288–294.

[23] Intel Labs, "The SCC platform overview revision 0.7," 2010.

[24] H. Esmaeilzadeh, E. Blem, R. St Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of multicore scaling," in *Computer Architecture (ISCA), 2011 38th Annual International Symposium on*. IEEE, 2011, pp. 365–376.

[25] V. J. Jiménez, L. Vilanova, I. Gelado, M. Gil, G. Fursin, and N. Navarro, "Predictive runtime code scheduling for heterogeneous architectures," in *High Performance Embedded Architectures and Compilers*. Springer, 2009, pp. 19–33.

[26] Q. Chen, Y. Chen, Z. Huang, and M. Guo, "WATS: workload-aware task scheduling in asymmetric multi-core architectures," in *Parallel & Distributed Processing Symposium (IPDPS), 2012 IEEE 26th International*. IEEE, 2012, pp. 249–260.

[27] Y. Zhu and V. J. Reddi, "High-performance and energy-efficient mobile web browsing on big/little systems," in *High Performance Computer Architecture (HPCA2013), 2013 IEEE 19th International Symposium on*. IEEE, 2013, pp. 13–24.

[28] M. Pricopi, T. S. Muthukaruppan, V. Venkataramani, T. Mitra, and S. Vishin, "Power-performance modeling on asymmetric multi-cores," in *Compilers, Architecture and Synthesis for Embedded Systems (CASES), 2013 International Conference on*. IEEE, 2013, pp. 1–10.

[29] T. S. Muthukaruppan, M. Pricopi, V. Venkataramani, T. Mitra, and S. Vishin, "Hierarchical power management for asymmetric multi-core in dark silicon era," in *Proceedings of the 50th Annual Design Automation Conference*. ACM, 2013, p. 174.

Henry Lovett Henry is a fourth year MEng Student at the University of Southampton.