Exploring power gating in coarse grained re-configurable architectures

Carboni Munoz, Felipe A.

Award date:
2020

Technische Universiteit
Eindhoven
University of Technology

Department of Mathematics and Computer Science
Architecture of Information Systems Research Group

Exploring power gating in coarse grained re-configurable architectures

ES - Master Thesis document

Felipe Carboni M. (0988340)

Supervisors:
ir. Jos Huisken
Prof. dr. Kees Goossens
Dr. ir. Pieter Harpe
Prof. dr. Henk Corporaal

1.1

Eindhoven, February 2020

Contents

Abstract

Power gating is a widely used technique for low power circuit design, which involves selectively shutting-down regions of an integrated circuit, dropping its power consumption to nearly zero, and all while the rest of the chip remains in operation. It may seem as a win-win strategy, however its implementation comes at a cost in a variety of aspects, some of them being power consumption, area and performance. An important body of research has focused on applying this technique and finding the limits on where this technique can be applied. Let it be in terms of scale and granularity, speed and frequency of switching on and off, etc. Thus, it has been of particular interest to explore this technique in the particular context of re-configurable fabrics.

This research explores power-gating as for reducing energy consumption in coarse-grained re-configurable architectures (CGRA) with the aim of exploring the effects of granularity decisions in this regard. For this end, a method is proposed in order to evaluate power-gating on a cell-to-cell basis, considerably outperforming traditional power-gating strategies. This method substantially extended the reach of power gating in the interconnect network, without compromising the interconnect's functionality. Finally, both area and power trade-offs are analyzed in the back-end stage of a CGRA's development.

Chapter 1

Introduction

In the past years, the use of application-specific embedded systems has reached the point of becoming a basic need in order to satisfy the market demands for cost, performance and power of newer designs. The industry has increasingly adapted the use of (re)configurable architectures in design flows to reduce the time and costs incurred to bring a product to the market, and integrated on high-performance computing devices.

In this section, we will introduce the concept of re-configurable architectures and more importantly, Blocks, the CGRA designed by the TU/e, which will ber where the focus of this research is. We will also discuss general techniques for power optimization, finalizing with the project problem statement ultimately driving this thesis project.

Traditional CPU's have taken us a long way as the workhorse driving most of our devices, ranging from high-end supercomputers, normal computers and smartphones, to small microcontrollers. The one feature that these processors have in common is their capacity for running virtually any type of operation given their rich instruction sets. This completeness comes however at a great cost in terms of power efficiency and speed. This has become a more present challenge as the complexity and computational power that real-time applications require to be effective. Be it visual processing applications, digital signal processing, or multi-variable simulations.

For this reason, the use of application specific integrated circuits started to take over those individual applications, by sacrificing a rich instruction set, and having a streamlined hardware structure, they were able to outperform CPU's by orders of magnitude in both performance and power consumption.

The rise of application specific integrated circuit presented a much more effective way of dealing with different computational challenges as they can perform several orders of magnitude faster, and more efficiently energy-wise than a multi-purpose chip on similar tasks. However, ASIC's strenght in performance would quickly be shadowed by their lack of flexibility. This gap has given way to re-configurable architectures, which would promise near-ASIC performance by mimicking the hardware of an ASIC. However they bring these benefits at the cost of an interconnect area overhead, and lower power efficiency. Different granularities of re-configurable chips have been proposed and have reached various levels of industry use, being the most popular the field-programmable gate arrays (FPGA), and the coarse-grained re-configurable architectures (CGRA). This research is focused on a version of CGRA developed by the TU/e.

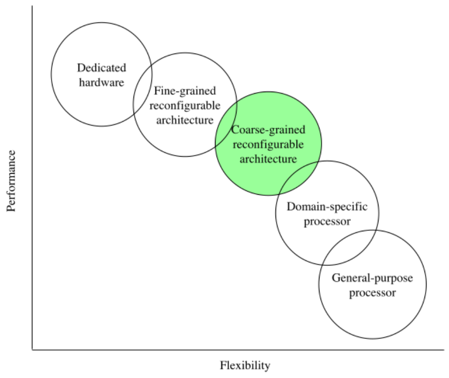

Dedicated hardware
Fine-grained reconfigurable architecture
Coarse-grained reconfigurable architecture
Domain-specific processor
General-purpose processor
Performance
Flexibility

Figure 1.1: Qualitative Flexibility-Performance localization. Source: [7]

1.1 Re-Configurable Architectures

Reconfigurable architectures have increasingly been adopted by industries due to its capacity of adapting an architecture to accelerate defined applications; much like an ASIC, however reversible. This provides a significantly higher performance in both speed and energy utilization [40]. This trend can be supported by looking at the increased use of Xilinx's and now Intel's FPGA's, as well as a broad range of more coarse grained versions of them. Sometimes,

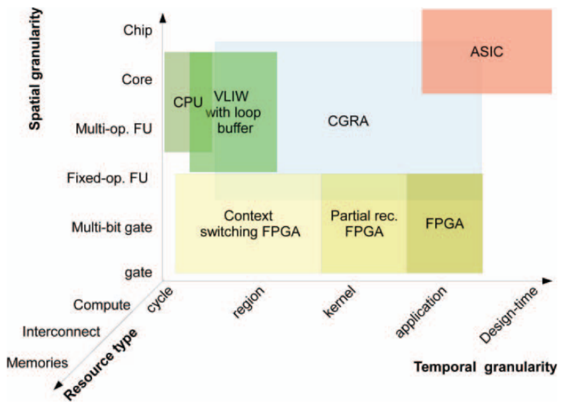


Figure 1.2: Multi-granularity based CGRA definition and comparison. Source: Wijtvliet et al. 2017 [40].

and depending on the type of applications intended, FPGA's can reach better power and performance numbers by locally reducing its per-bit reconfigurability, hence increasing the granularity of its building blocks (see fig. 1.1. for example, the use of multiple DSP's in a standard in FPGA structures.

1.1.1 The CGRA

CGRA's have steadily earning a position between the full reconfigurability of FPGA's and less customizable options, and the major reason for this consists in the recurrent use of pre-determined functions that need acceleration which can be implemented in an ASIC-fashion, but wrapped around a re-configurable layer to support it. The boundaries between ASIC's and programmable architectures in modern processor designs are becoming less and less clear as they implement hybrids and accelerators for specific applications.

CGRA's are more generally defined as a reconfigurable architecture that uses hardware flexibility to adapt the data-path at runtime to the application. Hence it becomes an array of configurable functional units that are also spatially programmable. Some academics have proposed methodologies to classify different types of CGRA's to come up with a more robust definition, therefore bringing to light: a) The wide variety of current CGRA designs, and b) The broad possible range of applications in which CGRA's can shine respect to other architectures for a particular application. Wijtvliet et al. [40] proposed a mix of spatial and temporal granularity metrics to classify a wide range of existent CGRA's, see figure 1.2.

This shift back towards more coarse grained accelerators could mark a trend that will provide hybrid FPGA architectures, or namely completely coarse-grained reconfigurable architectures (CGRA's) a space in the market [40] as a more efficient, cheaper, and potentially easier to configure alternative to current FPGA alternatives. It is hard to directly compare CGRA's with FPGA, given that the first has many varieties proposed where few have made it commercially [7].

.

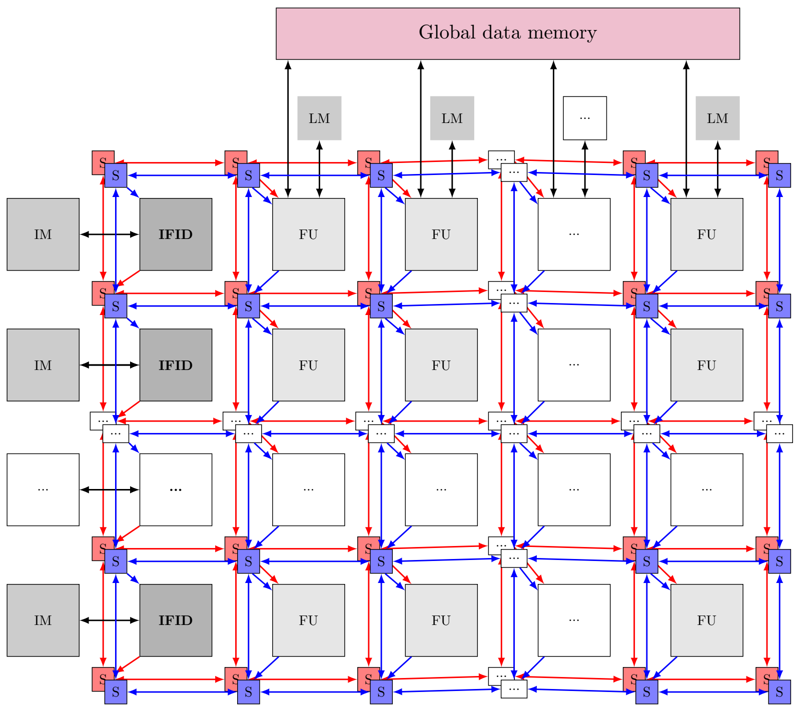


Figure 1.3: Closer look at the structure of Blocks, based on [40]

1.1.2 CGRA-Blocks

The CGRA utilized in this study is called Blocks: a design developed at the Eindhoven University of Technology, which as described by its Wijtvliet, M. as ”is somewhere between reconfigurable processors and coarse grained re-configurable architectures”. This architecture was designed as a fabric of reconfigurable processors (RP’s) that act in a SIMD-VLIW fashion, however with the possibility of extensive explicit bypassing controlled by a specially designed interconnect network that communicates every RP through pre-runtime configured switch-boxes. Figure 1.3 shows the generic structure of Blocks, and one of its main features; the presence of a dual interconnect network. In order to achieve power reduction, the instruction and data paths have separate networks through which they propagate to each of its functional units. This separation allows to reduce the size of the data interconnect, and to exploit instruction re-use.

1.2 Project problem statement

As it has been mentioned in this chapter, the performance that CGRA's can achieve is comparable to that an application specific design. However CGRA's are rather power-hungry mainly due to their massive interconnect network. This issue has been brought up and is one of the main concerns regarding CGRA-based embedded systems [19]. And many strategies have been put in place in order to mitigate the CGRA's power consumption: either by performing the standard power reduction methods discussed in annex .2, or through architecture-specific strategies such as the the interconnect solution in Blocks, or by power-gating sections of the CGRA as it has been applied to FPGA's [5]. For this reason, power gating was chosen as the strategy to study as it targets the most dominant source of power consumption in deep-nanometer designs: leakage. Additionally, power gating is a strategy that involves a careful consideration on the architecture it is applied to, the granularity at which it can be applied, and the ways it can be used effectively. Thus, being an area of research with the potential of bringing novelty while attempting to solve a major challenge in the advance of the CGRA as a standard platform.

Power gating has become almost mandatory in VLSI designs since the leakage is a dominating factor in newer CMOS technologies. It has become specially interesting for re-configurable architectures, where based on the mapping of a function major parts of the architecture remains unused. Power gating comes with logic overhead besides the required power switches themselves, and this overhead logic needs to provide logical isolation and valid logic signals while turned off for each of the switched modules' outgoing wires. This leads us to having to seriously investigate the granularity at which power gating can be applied. This is specially the case with in coarse grain re-configurable hardware, where there are very clear functional/logical dependencies between coarse-grain blocks.

It is for the reasons just argued, that power gating that it has been taken as the most relevant strategy to reduce the power in newer technologies using accelerators and re-configurable fabrics. The case of the CGRA developed at the Technical University of Eindhoven, would make of a suitable test subject to analyze the trade-offs regarding the granularity at which power gating can be applied. This is a topic in which there is not yet a consensus, or a systematic way to quantify the actual impact of power gates.

The problem statement is then an optimization question:
[MQ] At which granularity, in terms of functional units within the context of the coarse grained re-programmable architecture (CGRA), should power gating be applied to be beneficial for power. Analyzing the existing power-switching strategies applied in the industry, and providing results in the context of existing benchmarks and algorithms.

This analysis will weigh present trade-offs based on performance, area, energy savings, and possible flexibility implications of the different granularity settings for these algorithms, and taking into account the overhead of all relevant modifications involved in its implementation.

The relevant sub-questions go then as follows:

[SQ1] Analyze the implementation of power gating from the perspective of functional units, and determine whether is should be added as a default feature of all functional units in the CGRA, investigate the main variables involved and argument a position.

[SQ2] In terms of Floorplanning and the overall physical design, is what is the impact of the different power switching strategies, and how do they compare with the literature in the context of the CGRA?

[SQ3] Quantify the overhead coming from isolation cells, and control logic that may be required.

[SQ4] Can power gating on the CGRA be controlled dynamically e.g. switching functional units on and off during execution? Analyze and quantify. If that were beneficial, how should it be controlled?

1.2.1 Planning

In order to answer these questions, the next chapters are organized as follows:

Chapter 2 introduces the state of the art study on what concerns power gating, analyzing the different aspects that need to be taken into account, as well as drafting an idea of what results were to be expected when applying them to the CGRA.

Chapter 3 introduces the main metrics that will be used to evaluate the power gating strategies adopted in terms of energy and area.

Chapter 4 will review everything related to the workflows and models used to generate the metrics that we need to evaluate. Starting from the specifics of the designs used, tests groups and flows. Finally it introduces the Path traversal algorithm used to optimize power gating in the CGRA.

Chapter 5 showcases the results in terms of the metrics generated by the different strategies applied, the impact that the path traversal algorithm had in the test groups, the overall metrics CGRA-wide both dynamically as well as by switching the CGRA entirely off.

Chapter 6 finally summarizes the results and contrasts it with the initial objectives, highlighting the progress that this research made, as well as the future work that needs to take place.

Chapter 2

State of the art analysis

Thus far, this report lit upon what the current trends in low-power designs are, (excluding fully sub-threshold designs), and motivated a research question based on the granularity of power switches on a re-programmable fabric. This section will now place the focus onto latest research in the field of power gating, trying to map what alternatives are out there that may help answer the problem statement. The section will begin with some identified trends, then pass onto some guidelines on the design of power gates in terms of width and other parameters, to then the start drawing a power model for the different case-base that this research will have, all within the context of 40nm-TSMC technology.

2.1 Current trends in power gating

As introduced in section .2.3.3 and in the problem statement, power gating, or power gating, is one of the most attractive and well adopted techniques for power saving in nanometer technology nodes, we have many of today's microprocessors actually applying block-level power gating when the processors are idling [21].

It is however paramount to being able to apply power gating during the activity of these cores, as generally a small portion of them will be active, accelerators, certain IO's and even parts of the memory could be power switched, hence fine-grained run-time power gating (FRPS or FRPG) has been explored [20] to preserve power in a much smaller temporal and spatial granularity. Fine-grained run-time power gating generally depends on a small bit of control circuitry, and has been applied to memories [35], functional units in microprocessors [16] and re-configurable architectures [26].

An interesting option to fine-grained run-time power gating, or perhaps a complement, consists in the use of state retention within the power gated blocks (see fig. 2.5), however this solution tends to be quite expensive both in terms of power and area. For this reason, smart classification of registers either via netlist analysis or formal methods [13] have allowed to apply retention to only a subset of the registers that would otherwise be introduced in a block. This state retaining method allows for a stop in processors when for example a cache miss occurs and the processor is stalled, then quickly recover from where it left off after the issue has been solved.

In terms of power gating for memories: memories contribute to nearly half of the leakage in deep-nanometer circuits, however they often cannot afford loosing their state and generally a state-retention based power gating scheme would turn to be too expensive area-wise. Thus, a multi-mode power gating strategy was proposed [11]. It consists of a much bigger switch cell, which supports namely 3 modes: On, while active; Sleep while they are off but with memory retention enabled; and Off for completely shut-off. This has also been explored in prior master projects at the TU/e, by Groot [14] who proposed a switch capable of dropping the voltage of a module low enough to reap benefits in leakage, but high enough to allow registers to be able to keep their state.

There is very limited research particularly concerning power gating in CGRA's, however the closest neighbour to this architecture family are FPGA's, which present a substantially more developed literature in terms of granularity and control. Bsoul [4] explored different ways in which the interconnect could be included into the power-gating scheme and investigated dynamic power gating. Additionally, research on power gating in FPGA's generally aims to target the interconnect networks in their designs, on the one side because the blocks within an FPGA are genearlly hard-IP blocks, but also because the interconnect network is one of the main sources of power consumption. Partial and total inclusion of the interconnect structures around particular logic clusters have been proposed [5]. The range of solutions seen in FPGA's do not directly apply to the CGRA, however they face similar issues and therefore their approach in their case seems to lay a solid starting point for tackling the challenges (and pros) that the CGRA presents.

To summarize this quick overview of the main trends with respect to power gating:

1. Leakage keeps growing.
2. Most modern microprocessors apply it at a coarse level, as a standby/ sleep mode, however fine granularity seems to be making big steps due to:
   (a) Vertical integration of power gating in the design flows (eg. Compiler-based power gating + Hardware-based opwer gating).
   (b) Better partitioning algorithms and selective use of state retention.
3. Improvements in power gating for memories has led to various new multi-mode power switches.
4. Power gating research in CGRA' is still very poor, however there is a much more mature knowledge revolving FPGA research, where schemes have been proposed to power-gate at different granularities, as well as including ways of controlling the power gated blocks at run-time.

2.2 Implementing a power gate circuit

The modeling, and latter implementation of power gates require a number of design decisions that need to carefully be revised, This section will present some of the main challenges that this process takes, and what possibilities are in place to take this research into an implementation, as shown in figure 2.1.

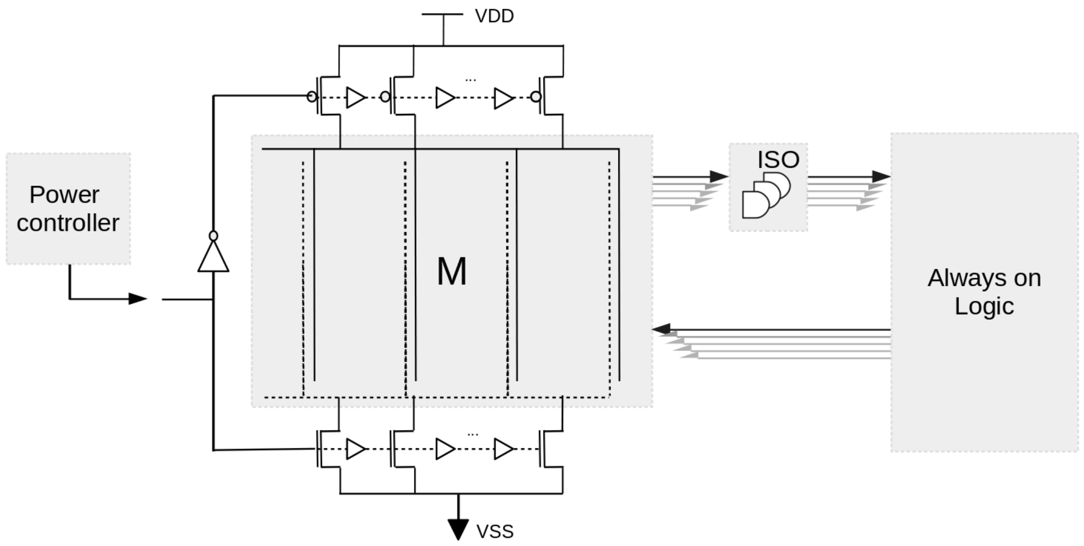


Figure 2.1: Diagram showing the main structures involving a power gating scheme: the controller on the left, the switches themselves (headers and footers on top and bottom of M respectively), the isolation cells on the outputs of the module M, and the always-on block representing the set of non-switched modules.

In the following sections, we will turn into discussing the more practical parts of power gating, treading ever closer to what the final implementation should be.

2.2.1 Sizing of a power gate

The simplest possible way to visualize a power gate is to think about a single single transistor, either PMOS or NMOS (header and footer respectively), then, we can extend this to an array of transistors that act in synchrony as a single switch by inducing a high resistance when the gates are closed. It is used to power gate certain parts of a circuit that are currently not in use. Generally, these sleep transistors are high-V_t.

It is important to remember that the 'optimal' power gating strategy (if any) will depend on specific goals and the actual chosen CMOS technology. Some of these variables are have to do with the use of header and/or footer; if there is any bias; the chosen transistor size and other layout implementation details. We also have to put our search into the context of 40nm bulk CMOS.

Commonly, the minimal sizing of a sleep transistor will depend on the current that the power-gate circuit can draw, and the acceptable IR drop. The following size calculations have been described for both NMOS [1], and for PMOS [17].

We will now perform the estimations for a PMOS. To simplify the analysis, we assume that a single power gate will be used per gated block. We start our analysis by determining the delay of a normal gate delay (eg, in the absence of a PG).

$$\tau_d = \frac{C_L V_{DD}}{(V_{DD} - V_{Tl})^{\alpha}} \tag{2.1}$$

where C_L is the load capacitance, V_{Tl} is the threshold voltage of the (low-Vt) transistor, and α is the velocity saturation index which is technology dependent. Now, in the presence of a sleep transistor, the gate propagation delay inside the power gated block can be calculated as:

$$\tau_d^{PG} = \frac{C_L (V_{DD} - V_{PG})}{(V_{DD} - V_{PG} - V_{Tl})^{\alpha}} \tag{2.2}$$

where V_{PG} is the voltage drop on the power gate. Now, we would like to find the V_{DD}^{PG} such that the delay of the transistor without power gates were equal to the delay of the power gated case:

$$\tau_d = \frac{C_L V_{DD}}{(V_{DD} - V_{Tl})^{\alpha}} = \frac{C_L (V_{DD} - V_{PG})}{(V_{DD}^{PG} - V_{PG} - V_{Tl})^{\alpha}} \tag{2.3}$$

For simplicity, we shall assume $\alpha = 1$, hence the 'supply increase ratio' η is:

$$\eta = \left(\frac{V_{DD}^{PG}}{V_{DD}} - 1 \right) \tag{2.4}$$

Having this relation, we can define the voltage drop in terms of V_{DD}, which will help us in the next steps:

$$V_{PG} = \eta V_{DD} \tag{2.5}$$

Assuming that the PG operates in its linear region, its current can be expressed as:

$$I_{PG} = \mu_p C_{ox} \frac{W}{L} \left[(V_{DD}^{PG} - V_{Th}) V_{PG} - \frac{V_{PG}^2}{2} \right] \tag{2.6}$$

Where if we substitute V_{PG} as in equation 2.5, we will get:

$$\left(\frac{W}{L} \right)_{PG} = \frac{I_{PG}}{\mu_p C_{ox} \eta V_{DD} (V_{DD}^{PG} - V_{Th} - 0.5 \eta V_{DD})} \tag{2.7}$$

This relation puts puts in evidence how the size of the transistor is a function of VDD, and the voltage drop across the power gate (through ηV_{DD}), finally we can calculate the minimal power gate size if we take $I_{MAX} = I_{PG}$, where I_{MAX} is the maximum switching current that the circuit will draw. Repeating the same process, we can get the expression for the size of an NMOS-PG as:

$$\left(\frac{W}{L} \right)_{PG} = \frac{I_{PG}}{\mu_n C_{ox} \eta V_{DD} (V_{DD}^{PG} - V_{Th})} \tag{2.8}$$

often, μC_{ox} is replaced by the value for trans-conductance β, which makes:

$$\left(\frac{W}{L} \right)_{PG} = \frac{I_{PG}}{\beta \eta V_{DD} (V_{DD}^{PG} - V_{Th})} \tag{2.9}$$

where β is the transistor trans-conductance, η is the max relative IR drop, and $V_{DD}^{PG} - V_{Th}$ is the gate-drive voltage. If we now look back at the models presented in section .1, we can start doing the exercise of which power gate would be required for a particular block capacitance.

2.2.2 Trade-offs and break-even point

The implementation of power gates comes at a considerable overhead, as it will require the insertion of wide-enough power switches that will supply of a stable V_{DD} and V_{SS} regardless of: a) the voltage drop induced by the transistor itself, b) the current draw that the circuit will have during its active period. This switches will still have a certain resistance, and hence burn extra active power, as well as introduce extra power consumption when switching on and off a virtual supply (either V_{DD} or V_{SS}). This introduces an overhead that in principle, should be compensated by the gains of reduced leakage power (see fig 2.2).

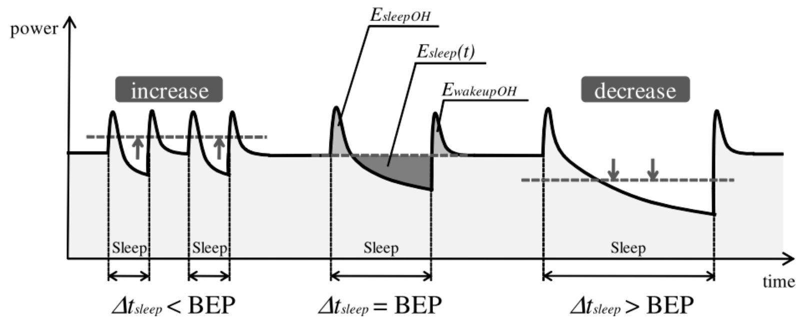


Figure 2.2: The power profile of a curcuit with power gating. Source: Kondo 2014 [20]

Most work available present the idea of a break even point (BEP), in which they look towards compensating the overhead that the PG introduces [20, 32]. In this particular case, the overhead is modeled as $E_{sleepOH}$ and $E_{wakeupOH}$, where the energy gain is a function of the time in which the circuit it in shutoff mode $E_{sleep}(t)$.

$$E_{savings} = E_{sleep}(t) - (E_{sleepOH} + E_{wakeupOH}) \tag{2.10}$$

Niedermeier [28] digs a little deeper into the breakdown of overhead, not only including the extra power used while switching on and off, but rather including explicitly the impact of architectural changes and supporting cells onto the design, expanding on the active power of isolation cells, and other modules. Here, $(E_{sleepOH} + E_{wakeupOH}) = E_{overhead}$ is defined by:

$$\begin{aligned} E_{overhead} = {} & t_{down} * (P_{switch,leak} + P_{iso,leak} + P_{SR,leak}) \\ & + t_{active} * (P_{iso,active} + \Delta P_{SR,active}) \\ & + t_{total} * P_{add.modules} \\ & + N * E_{poweron} \end{aligned} \tag{2.11}$$

Indeed, the power gating is worth it only if $E_{savings} \geq E_{overhead}$.

Since the implementation of the power gates to be done in this project is going to be a mainly hardware-based solution. Other software-based trade-off and analysis schemes are going to be omitted. The logic behind this analysis however, seems to show quite some clarity about how the power gates's performance will be evaluated power-wise.

As was just mentioned, and added into the trade-offs variables; the implementation of power gates often requires a number of other cells, circuit infrastructure, and control mechanisms to become a viable option. The additions that are required are:

1. Decap cells: in order to avoid the power noise caused by the simultaneous switching of IO buffers and logic. The addition of decap cells can considerably reduce the transition noise. The decap cells commonly use on-chip non-switching capacitors C_{ckt} or thin-oxide capacitors C_{ox} [17]. Depending on the noise distribution of the chip or the block in question, decap are distributes around and within the chip to pull noise back to a determined margin. A high capacitance from decap cells is generally used in high performance chips.

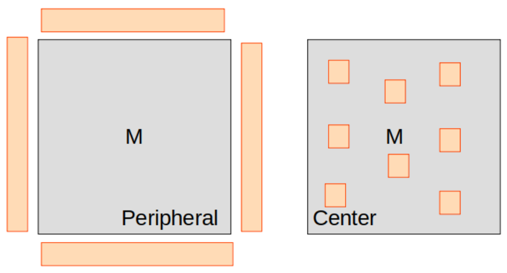


Figure 2.3: Illustration on the insertion of decap cells to denoise a power gated block.

Some of the calculation on how much decoupling, and where it should be located, is presented in the work of [17], where an iteration greedy algorithms and identification of highest noise sensitivity are used to place the necessary decap.

2. Isolation cells: These are simple registers or combinatorial cells that are connected at the outputs of the gated block to prevent the floating outputs of the cut-off circuit to change any states on the parts of the chip that are active. These cells are in the always-on domain and depending on the design and the amount of power islands to be gated, could generate a considerable amount of power overhead [28], there are generally 3 types of standard isolation cells: pull-up, pull-down and with a latch to preserve the last output of the gated block. This last one is generally only used in tandem with state-retention across the power gated block, which can be connected to scan chains or other 'state retention schemes'.

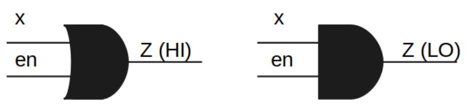


Figure 2.4: schematic of a simple clamp isolation cell

3. Retention registers: one of the challenges of power gates, is that the state of the block that has been shut-off looses its state, since memories and registers are not capable of keeping their information while powered off [33]. Hence, special retention cells have been adopted in most of the commercial standard libraries (eg. TMSC), to support the State retention power gating (SRPG).

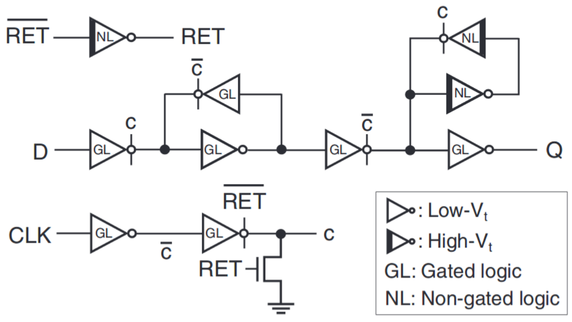


Figure 2.5: Illustration of the schematic of a retention cell. Here, a conventional master-slave D-FF is modified for data retention. Here, the when when the power-gated cells (denoted by GL) are shut down, the information is moved to an adjacent latch that is within the always-on domain. Source: Seomun, 2009 [33].

The use of SRPG requires an extra duplicate of the state latches that need to be retained, thus, the area increases by about an unavoidable 30-50% per retained register [33]. This also poses a great challenge in terms of routing overhead, as some of this registers have to be located sometimes deep within shut-off territory [13], as well as reducing the power-saving effectiveness in contrast to a traditional PG scheme.

A more advanced version of SRPG is selective-SRPG or SSRPG, which is done by only choosing and retaining the registers that are essential for retaining the state of a power-gated block. This assumes that only a small subset of the gated FF's is actually essential for a system-wide state retention, which often has its limitations. Some of the processes in which SSRPG is based consist in the classification of a design's FFs to be power gated [12].

2.3 Fine grained vs. Coarse grained

One of the first decisions that the architect has take when planning power gating, is the issue of granularity. Literature often describes granularity as fine and coarse, depending on whether the power gate is located already as part of each standard cell in the library to be used [18]. However this definition has shifted over time as a) per-cell power gating doesn't seem to justify the overhead that it involves, and b) The literature has shifted the understanding of fine-grained power gating (FGPG) to the order of hundreds of cells, whereas coarse-grained power gating (CGPG) generally ranges in the thousands of cells. Our understanding of granularity could have a distinction on the basis of a functional unit in the CGRA. Meaning that FGPG may involve 1 functional units or less, while CGPG would involve an array of functional units.

2.3.1 Fine-grained power gating

in the case of the smallest possible fine-grained power gating, the power-gate is located inside the standard cell, and since it has to be able to supply the worst case current required by that particular cell, the resulting size of that FGPG ends up being comparable to that of the cell itself; even up to x2-4 of the original cell size [18]. It is important to note that for FGPG footer cells are preferred above headers, for the simple reason that NMOS transistors have roughly twice as higher carrier mobility than PMOS transistors, which will proportionally impact the required size of the power gate in question (see equation 2.9), however, due to their increased mobility, NMOS power gates will present more leakage current than PMOS (see equation 8).

One of the advantages of fine-grained power gating is that the design of each power gate individually has very little problems, as the timing impact of the IR drop can be quite predictable, this means that FGPG could be, if it were included in the standard cell libraries, deployed using a 'normal' design flow. However the sizable amount of overhead could barely justify the use of power gates per cell, also specially because most power-gated circuits use at least 8-bit architectures. It becomes then almost natural to group cells into coarser islands and still call it Fine-grained.

2.3.2 Coarse-grained power gating

In coarse-grained power gating (CGPG), a block of gates is switched by one or a group of power cells. Generally they are placed forming a ring around the gated block or they are distributed within the actual block [18]. This method has been the most used in the past years, as it does not require the extreme area overheads used in fine-grained power gating, however it presents different challenges in regards to the number and size of power gates required for gating a given block. This is due to the difficulties in estimating the worst case current that the gated circuit will draw from the switches.

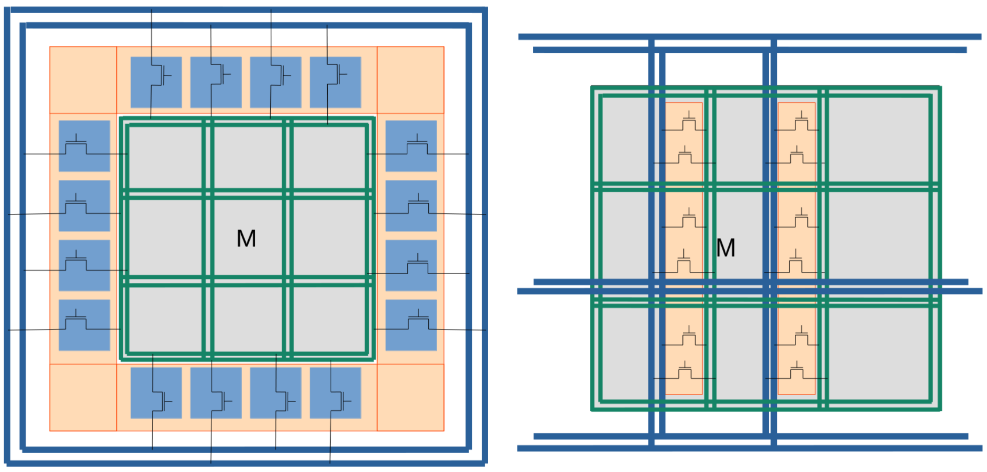


Figure 2.6: structure of a ring and column based power gating techniques

Each of the two methods have their advantages and disadvantages, and it will ultimately be a decision of the designer, on which of the two shall take place. However, this structural CGPG decision will have implications on further design decisions.

2.3.2.1 - Ring-based coarse-grained power gating

It generally is a good option for small logic blocks where the voltage drop across the switch transistors and the VV_{DD} mesh can be easily managed.

+ Simpler power plan due to the separation between the Virtual V_{DD} and the actual V_{DD}. Sleep transistors are not mixed with the other logic cells.
+ Has little negative impact on placement and routing.
– It does not support retention registers (as the whole block is completely cut from V_{DD}).
– Adds a much more significant extra cost compared to a grid approach.

2.3.2.2 - Grid-based coarse-grained power gating

This is a more suitable alternative when large logic blocks are being power gated, as it would supply the VV_{DD} (VV_{SS}) with a better distribution.

+ The switches have to drive smaller portions of the VV_{DD} every time, compared to the ring based power gating.
+ Requires fewer/ smaller sleep transistors for a similar IR drop. Then again, it is because the VV_{DD}'s are are of much smaller depth.
+ Permanent power supply is available across the power-down domain areas.
+ it provides a better trickle charge distribution for management of in-rush current.
+ has less impact on the area of a power gated block.
– It requires changes on the cell routing and physical synthesis.
– Adds much more complexity to the power routing needs of the design.

There are variations of Grid based implementations of power gates, such as column based and row based. These, they are good for reducing the voltage drop across the VV_{DD}s but they impact placement and lower metal layers on the design.

The optimal style will depend on:

1. Design.
2. Library being used and the type of switches available.
3. The technology being targeted and its specific leakage characteristics.
4. The performance and power goals of the design.
5. The use of legacy or highly optimized IP.

Chapter 3

Metrics

The last section in the analysis has to do with the design itself on a back-end perspective, this is a more practical and therefore a slightly less explored area experimented with on the research. Its results would shed some extra light and weights on the trade-off analysis of power switches from both a power and a floorplanning perspective.

3.1 Area overhead (AO)

As presented in section 2.2, we will discuss the two common schemes for used by designers and their trade-offs, namely the ring-based and column-based power switch insertion, additionally, a derivative of the column-based method, generally called the ”checkerboard” method. The first thing we need to find out is how many power switches does the module M require. This information can be obtained by different ways given the tools available and the information that can be gathered from the technology libraries of the TSMC40nm libraries. The most straightforward way of collecting the information about the module M, would be to collect the capacitance, activity factors and energy consumption from Innovus' reports and replicate the 1st order model presented above.

Using the power reports from Innovus, we can collect capacitance, power consumption and the activity factor on which those were calculated. We can thus estimate that the power that the switches need to provide in the worst case as $power(M)/\alpha$, in other words, the module power in case of a 100% chance of activity. With that, the translation from power to required current just requires a division over the supply voltage V_{dd}. Since we are using both headers and footers, we repeat the exercise on both cases:

$$Nr_{PS} = (AP(M)/\alpha)/min(I_{PS}) \tag{3.1}$$

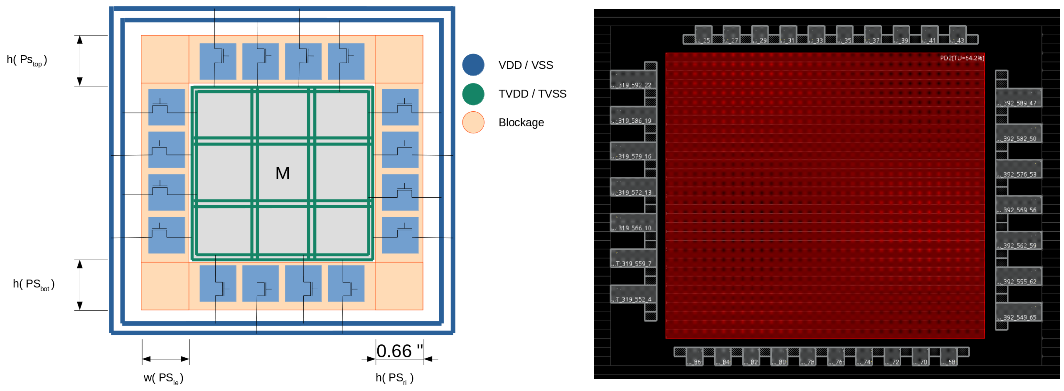
Where the I_{PS} is the current that the power switch can provide given a designer defined IR drop of 5%. Now, knowing how many power switches of every type we would need, we have to distribute it between big and small making sure there are enough small power switches to ensure that the rush-in current stays below a 5x the normal current threshold through a stepped wake-up chain. Having the number of headers and footers of each type to be used for module M, we can analyze the area impact of adding them to the design: The easiest way would be to add the area of the power switches and isolation cells:

$$\begin{aligned} AO_{total} &= AO_{PS} + AO_{ISO} \\ AO_{total} &= \sum area_{PS} + \sum areaISO \end{aligned} \tag{3.2}$$

While this analysis holds for the isolation cells, the area overhead of power switches depend on the type of insertion implemented, we therefore will calculate the AO_{ring} and AO_{cols} for the ring, column and checkerboard types of switch insertion separately, and therefore is it where the analysis will go. Now, the different methods end up collecting basically the same information and can help contrast the results of each other. We will lean towards the first method in this case as we have a way of simulating on the same principles.

3.1.1 Ring-based power switching

Ring-based power switching, as its name indicates, and as discussed in chapter 2, all power switches are inserted around module M, hence creating a ring around it, as shown in figure 3.1.



(a) (b)

Figure 3.1: Area overhead (AO_{ring}) of a ring-based power switch insertion on the area of M. in theory (a), and in practice (b) generated in Cadence Innovus.

Knowing the number of switches needed and assuming a distribution on all 4 sides of M, we can define $h(PS_{side})$ and $w(PS_{side})$ as the maximum height and width of all power switching cells in a particular side plus the separation between the M and the power switches, defined as $halo_M$:

$$h(PS_{side}) = max(h(PS_{side})) + halo_M$$

and

$$w(PS_{side}) = max(w(PS_{side})) + halo_M$$

Now, we can estimate AO_{ring} of as follows:

for the sides:

$w(M) * h(PS_{top})$ +

$w(M) * h(PS_{bot})$ +

$h(M) * w(PS_{le})$ +

$h(M) * w(PS_{ri})$

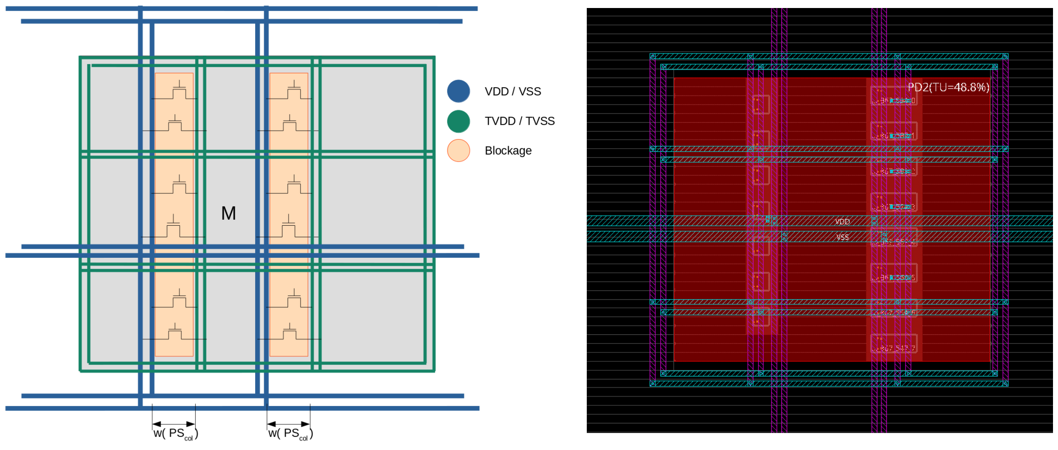
and the corners:

$h(PS_{top}) * w(PS_{le}) + h(PS_{top}) * w(PS_{ri})$ +

$h(PS_{bot}) * w(PS_{le}) + h(PS_{bot}) * w(PS_{ri})$

3.1.2 Column-based power switching

In the case of column based, the switches are placed inside the module M, hence avoiding the requirement for a complete construct around the module as a ring-based scheme would require. This method generally requires less area to be implemented, but increases the routing complexity, as seen in figure 3.3 with respect to 3.1.



(a) (b)

Figure 3.2: Area impact of a column-based power switch insertion on the area of M. in theory (a), and in practice (b) generated by Cadence Innovus.

We can estimate calculate the area impact of the power as, given a COL = nr of PS columns in M, and the width of a column defined as the maximum width of the power switches belonging to that particular column (PS_{col}).

$$w(PS_{col}) = max(w(PS_{col})) + halo_M$$

with makes the area overhead (AO) for columns:

$$AO_{col} = \sum_{i}^{COL} h(M) * w(PS_i) \tag{3.3}$$

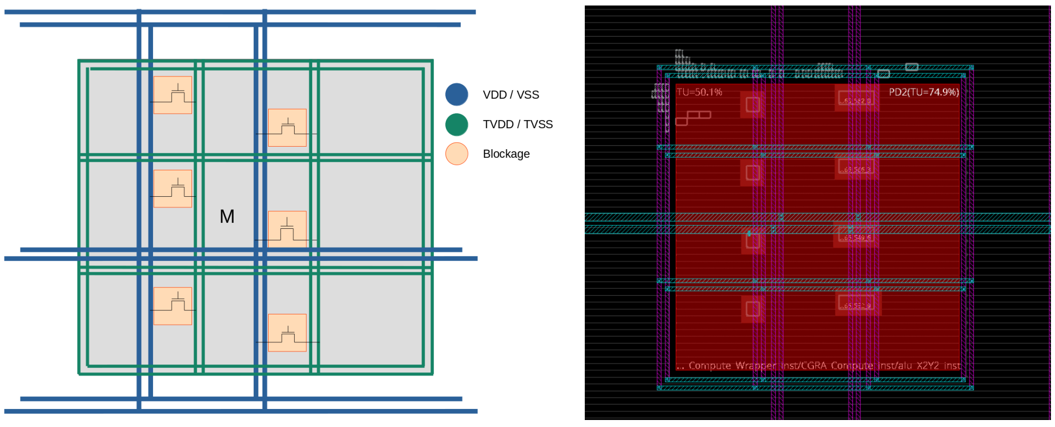
3.1.3 "Checkerboard" power switching

This technically consists in a variation of the column-based power switching, however it places the power switches on every column in a vertical distance to each other, allowing for the blockage that would have been otherwise across the whole column to be interrupted, thus allowing for a minimal area impact on power switching. For this effect, we now hace to calculate the area overhead ao per-switch s, and with $halo = 1.68um$:

$$ao_s = (w_s + 2 * halo) * (h_s + 2 * halo) \tag{3.4}$$

Where the total overhead of the "checkerboard" switching is the sum of all N individual overheads:

$$AO_{check} = \sum_{i}^{N} ao_i \tag{3.5}$$



(a) (b)

Figure 3.3: Area impact of a column-based power switch insertion on the area of M. in theory (a), and in practice (b) generated by Cadence Innovus.

3.2 Energy analysis

Energy is probably the prime reason why power switches are installed, however they do present certain drawbacks in this regard. Citing Niedermejer [28], the complete energy savings of a power switch scheme could be calculated as:

$$E_{savings} = P_{mod,leak} * t_{down} + N * E_{powerdown} \tag{3.6}$$

with N: nr of trantisitons from *on* to *off* and $E_{overhead}$ is defined by:

$$\begin{aligned} E_{overhead} = t_{down} * (&P_{switch,leak} + P_{iso,leak} + P_{SR,leak}) \\ &+ t_{active} * (P_{iso,active} + \Delta P_{SR,active}) \\ &+ t_{total} * P_{add.modules} \\ &+ N * E_{poweron} \end{aligned} \tag{3.7}$$

Sufficient data will be gathered to make this calculation applied to the case of the CGRA, and tested for different modes/granularities of power switching. The results generated will be contrasted to those of a flat design and conclusions can be taken from there.

3.3 Performance

The measured performance of the design given the insertion of power switches is heavily dependent on the impact that the chosen IR drop of the power gated modules have in the critical paths of the design. Therefore topics like timing analysis have receded to a second level of importance to this research. However what the designer would have to weigh during this process is to estimate how much speed is he/she willing to give up given the chosen IR drop for which the power gating structure is put in place.

For the effects of this research, we make the assumption that performance will react solely based on said IR drop, and a simple way to estimate it is by using another first order model for the charging and discharging of an inverter.

Being an inverter's discharge time (t_{pHL}) estimated linearly as the what pull-down network (NMOS) takes to discharge the load capacitance at the gate. Being the same inverter's charge time (t_{pLH}) estimated linearly as what the pull-up network (PMOS)takes to charge the load capacitance at the gate.

$$\begin{aligned} T_{pLH} &= \frac{C_L * V_{dd}}{\frac{W_p}{L_p} * \mu_p (V_{dd} - V_{Tp})^2} \\ T_{pHL} &= \frac{C_L * V_{dd}}{\frac{W_n}{L_n} * \mu_n (V_{dd} - V_{Tn})^2} \end{aligned} \tag{3.8}$$

If in either case, we multiply V_{dd} by a factor $a < 1$, and rearrange the equations, we get:

$$\begin{aligned} T_{pLH} &= \frac{C_L}{\frac{W_p}{L_p} * \mu_p} * \frac{V_{dd} * a}{(V_{dd} * a - V_{Tp})^2} \\ T_{pHL} &= \frac{C_L}{\frac{W_n}{L_n} * \mu_n} * \frac{V_{dd} * a}{(V_{dd} * a - V_{Tn})^2} \end{aligned} \tag{3.9}$$

Note that W_x, L_x, μ_x, C_L are all constants. We can see on the right hand side of the equations that solving for zeros gives us that $V_T/V_{dd} < a < 1$. This give us an asymptotic relationship between $V_d d$ and propagation delay. This is of course not the case in real life, but grossly illustrates the effect of such a change. Considering that we are using LVT cells with $V_T = 0.48V$, $V_{dd} = 1.1V$, and an IR drop of 5%, we can estimate a propagation delay increase of:

$$\begin{aligned} no\_IR\_delay &= 1.1/(1.1 - 0.48) = 1.774 \\ IR\_delay &= 0.95 * 1.1/(0.95 * 1.1 - 0.48) = 1,849 \\ \%\_increase &= 4.249\% \end{aligned} \tag{3.10}$$

In this case, it seems that the difference between V_{dd} and the V_T of the standard cells allowed for a performance cost smaller than the IR drop, however this is not the case when we use HVT cells ($V_T = 0.65V$):

$$\begin{aligned} no\_IR\_delay &= 1.1/(1.1 - 0.65) = 2.444 \\ IR\_delay &= 0.95 * 1.1/(0.95 * 1.1 - 0.65) = 2,646 \\ \%\_increase &= 8.23\% \end{aligned} \tag{3.11}$$

Where the performance cost is now almost doubled for the same IR drop.

The calculations shown in the above example were put in place to illustrate how the designer could estimate the impact of his/her IR drop's decision, however the results of this research have taken that as a granted and fixed value of 5%. And since we can't assert that the changes in performance are going to be what was calculated in the example, we can assume that whatever the real impact is, it will remain constant across our test-groups and benchmarks.

Chapter 4

Methodology

Having outlined the main metrics used in this research for area and power of power gating, the purpose of this chapter is to show the methodology used to gather and mix the information collected for its different parts. Since the data came from varied sources: namely a set of tools as well as documentation. There is a high overlap between them. Information as the power characteristics of the power switches and isolation cells come mainly from TSMC's documentation and liberty (.lib) files, which are also used by the cadence tools used, namely Genus, Innovus, Virtuoso and their respective sub-tools.

4.1 Tested designs

This research was conducted on two benchmark versions of the CGRA blocks, one of relatively small size, and on a bigger scale, these designs went through the design and P+R flows that will be discussed further in this same chapter, for different combinations of power switched modules, and performed one at a time in order to avoid the possible impact that power islands may have on each other, they were all contrasted with a simple flat design of every type.

1. Binarization scalar dynamic: The smallest of the CGRA benchmarks used was the implementation of a binarization algorithm, consisting in a 3x3 grid of functional units and an extra layer of switchboxes (see 4.1).

2. FFT parallel dynamic: A more upscale version of the CGRA containing 5x more FU's than the initial benchmark, this design will prove useful to test the granularity of power switching on the CGRA, as well including the MUL functional unit, which was not available in the binarization benchmark (see 4.2).

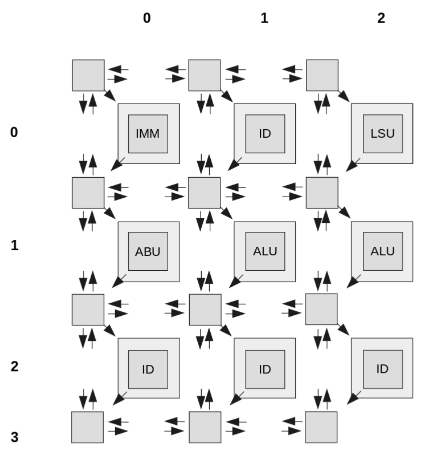


Figure 4.1: Binarization scalar dynamic CGRA architecture.

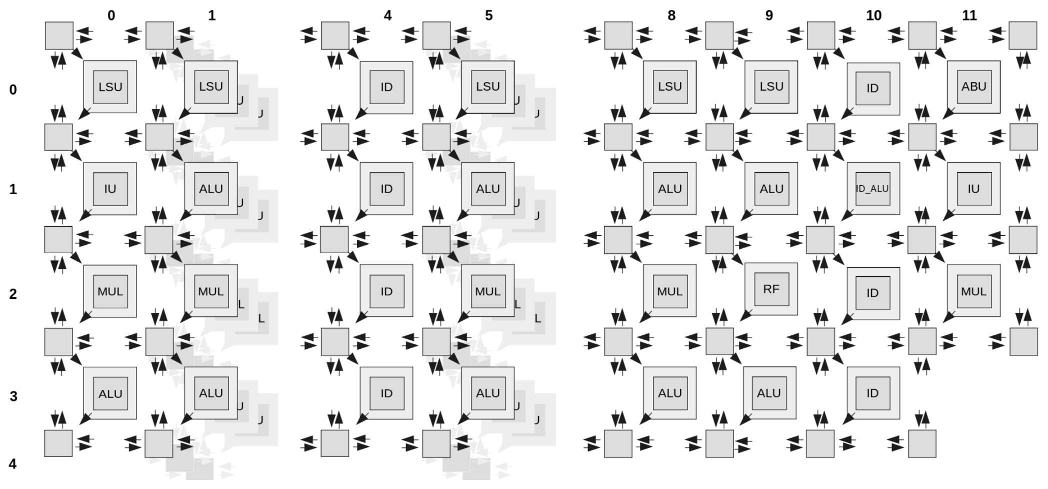


Figure 4.2: FFT parallel dynamic CGRA architecture.

4.2 Test groups

As mentioned, in the beginning of the section, there are 3 test groups consist in a) only switchboxes, b) only functional units, and c) extended functional units. In here we will briefly go through what we will find in each one of them, and give the cumulative measures on them based on the two benchmarks used; Binarization and FFT.

4.2.1 Switchboxes

They are the most numerous modules, accounting for [63-65]% of the CGRA's logic on both benchmarks used (excluding memory). They also account for with the most variability in their possible configurations, and subsequently in the number of cells, and outputs they present.

1. Control switchboxes: are the smallest, and their size in the benchmarks used ranges between [40 - 572] cells. Their number of outputs range between [33-321].

Control SWB	min	max	average	median
cells	40	572	296.67	255
area	69.38	932.33	542.61	539.78
oports	33	321	100.64	97

2. Data switchboxes: the bulk of the CGRA interconnect, their size ranges between [110-4834] cells. Their number of outputs range between [49-449].

data SWB	min	max	average	median
cells	110	4834	2255.65	1451
area	196.62	7563.09	3778.17	2868.97
oports	49	449	309.12	353

The immense variability that they present depends uniquely on how many paths does the switchbox need to support. In the case of the smaller ones it would be just a redirection, where on the biggest ones it would support redirection from/to either cardinal point, as well as connections from and to 2 different FU's. For this same reason, the results of switchboxes were separated in 5 intervals depending on their number of instances:

Module	instances	count
*swb*_500	$i <= 500$	7
*swb*_1000	$500 < i <= 1000$	5
*swb*_1500	$1000 < i <= 1500$	12
*swb*_2000	$1500 < i <= 2000$	15
*swb*_3000	$2000 < i <= 3000$	7
*swb*_4000	$3000 < i <= 4000$	5
*swb*_4000+	$4000 < i$	24

4.2.2 Functional units

They are computing part of the CGRA, accounting for the resting [35-37]% of the CGRA's logic. They present just a handful of different functional units, namely: IMM, ID, ALU, MUL, RF and LSU, which account for every module needed to make a processor. Limited in number as they are, there is much less variability involved: where IMM and ID's are less than 200 cells, most other FU's are in the 2000 cell size.

FU	min	max	average	median
cells	74	2290	1161.23	812
area	334.22	6924.99	3210.86	1851.49
ports	46	238	95.87	66

In terms of the results that will be presented in the following chapters, the functional units were grouped throughout both benchmarks based on their type:

Module	count
mul	9
lsu	10
abu	2
rf	1
id	16
imm	4

4.2.3 Extended functional units

They are the same functional units that were counted before, but extended using the path traversal algorithm, to select the biggest possible number of cells from their respective switchboxes, without altering the functionality of the interconnect. The results were rather positive, And the measured data on them can be found in the following table.

FU+	min	max	average	median
cells	196	3379	1977.92	1893
area	477.22	8418.98	4821.31	3842.93
ports	46	238	95.87	66

In terms of the results presented in the next chapter, the extended functional units were grouped in the same way as the functional units as shown in the prior section.

4.3 First order power switch

Based on the work of Groot, 2007 [14] a first order model of the power gated module M was generated. Where the core will represent the a module named "M". We will use a first order approximation of the core's behavior as composed by a resistor, a capacitor, and a current source. Where: V_{DDV} represents the voltage of the core during active period, it equals V_{DD} minus the voltage drop across the power switch, and V_{SSV} represents the virtual ground of the core (see fig.4.3).

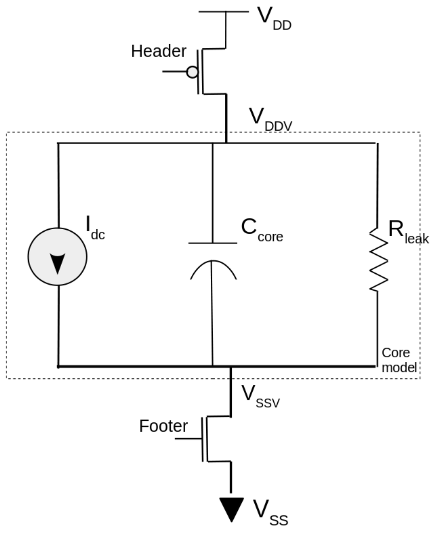


Figure 4.3: First-order model for the core.

Where the current source I_{DC} represents the dynamic power consumption of the core. During simulations, the current source is either turned into an open switch to simulate inactivity, or into a current source that will compete with the supply provided by the header (and the ground of the footer), here α represents the switching activity and f the circuit frequency:

$$I_{dc} = \alpha C_{core} * V_{DDV}^2 * f \tag{4.1}$$

the capacitor C_{core} represents the total capacitance of the core:

$$C_{core} = C_{decap} + \sum_{gate \in core} C_{gate} \tag{4.2}$$

and the resistor R_{leak} is the calculated resistance of the core, from which we can model the leakage current I_{leak}:

$$R_{leak} = \frac{V_{DDV}}{I_{leak}} \tag{4.3}$$

In the model, the total energy used during wake-up can be approximated as equation 4.4, being t_{wakeup} the total time it takes C_{core} to charge up to V_{DDV}:

$$E_{corecharge} = \int_0^{t_{wakeup}} V_{DDV} * i(t)dt = V_{DDV}^2 * C_{core} \tag{4.4}$$

With this model, it is possible to generate a baseline estimation of the charging and discharging behavior in different operating modes. The models described in this section were implemented using cadence RC (Virtuoso) and tested for the various ranges that the analyses may require. Some of the handles moved to simulate the possible requirements were:

1. Number/type of power switches: The switches available vary in their size and their capacity to supply power to the V_{DDV} and V_{SSV}, the decision on type and numbers were based on the current required to cover for a determined voltage drop.

2. Delay: Although the schematic shown in figure 4.4 contains buffers between the gates, it was simpler to model the delay directly by assigning a delay on the enable of each power switch. Additionally, the delay was useful to balance the behavior of headers and footers, as well as to keep the rush-in current in check.

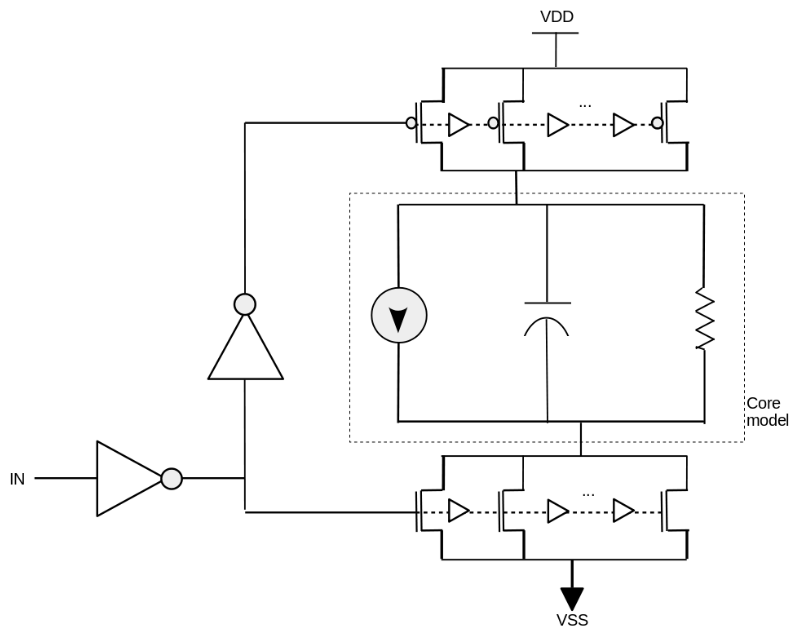


Figure 4.4: Illustration of the test model using a variable number of switches which for a determined W_{eff}, they may have different rush-in current characteristics due to the size, and the signal delay of the switches used. This model, adjusted for switch number; size; and delay, will be used to calculate the transient behavior of the switches used later on in this research.

After all parameters were set, we could sweep through variables. Since this method would not be able to use the power switches, but only simulate HVT -CMOS of similar characteristics, it presented a good way of double checking that the design decisions were made correctly. For example: those done on a number and size of power switches for module M was taken in order to ensure a maximum IR drop of 5%, since all characterizations were used in worst case, pretty much all the tests shown a compliant IR drop in the simulations when running the first order model.

4.4 Synthesis workflow

The Synthesis of the different CGRA versions was scripted in TCL and executed by *Cadence Genus v19.11.000.* The flow uses some of the features from the stylus version of the tool (newest versions available), that allow for a multi-mode, multi-corner synthesis which can be kept consistent throughout both Front-end and Back-end development of the CGRA design. The design flow consists in a series of steps, summarized in figure 4.5 and detailed below. Additionally, the scripts used in this case can be found in the annexes to this report.

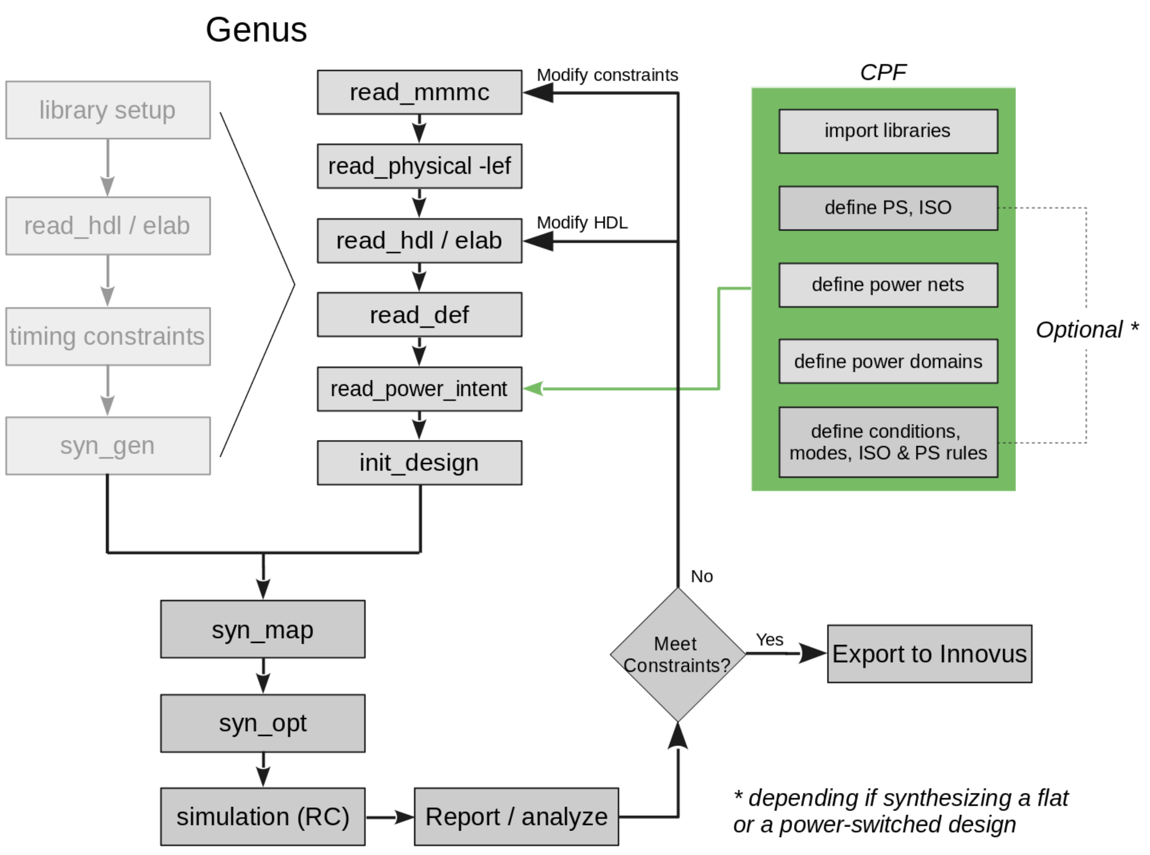


Figure 4.5: Genus synthesis flow used in the report. The traditional flow (light grey on the left) was replaced by the more complete MMMC flow, allowing for a more consistent power-aware development throughout both front- and back-end

- *read_mmmc*: corresponds to reading the multi-mode, multi-corner file, libraries for best, worst and typical conditions are defined for all cells in the design, defining a set of operating views consisting of combinations of temperature, voltage, libraries and rc corners.

- *read_lef*: reads the different physical characteristics of each layer in the libraries, as well as their dimensions and combinations corresponding for each library cell.

- *read_hdl* and*elaboration*: imports the design written in HDL language, generally VHDL or/and verilog, then elaborates the design, i.e. checks the consistency of the design for errors, unconnected IO's and other unresolved declarations.

- *read_def*: further checks for consistency, this time between the elaborated design and the DEF files, which contains physical information.

- *read_power_intent*: imports the CPF file, which contains all the command necessary for a low power design, this file contains the definition of the separate power domains and their correspondence to which instances in the design. It also isolation, switches and level shifter cells and rules. likewise, if the plan is to make a flat design, certain steps here are omitted.

- *init_design*: The tool here steps through the defined MMMC objects, the design, and the power requirements, building the full design and leaving it ready for manipulation, e.g synthesis.

- *syn_map*: Can also be preceded by *syn_generic*, which synthesizes the design and generates a full netlist in HDL code. The *syn_map* step maps the generic gate-level synthesis to the technology libraries provided in the beginning of the synthesis.

- *syn_opt*: optimizes de design to try match the timing and power constraints.

- *simulation*: runs the testbench of the CGRA and generates a toggle count file (TCF), which can be fed back to genus in order to make a more precise power and timing analysis. If The constraints are still within bounds, the design can be exported for the back-end development.

- *reports/analysis*: generate preliminary results using the TCF from the simulations. Items like cell area of a module are now well known, as well as some ideas on power and timing.

- *export to innovus*: write the design in a ready-to-use format for Innovus stylus (v19.11.000), which will pass on the MMMC data, as well as the CPF power configurations.

4.5 Back-end and testing workflow

Starting with a design generated on genus, we use a 2-tool approach to simulate and generate the required results with regards to our power switches. First, a flat floorplan is generated and tested, which will provide the most precise information on items like total capacitance of a module including nets, accurate path delays and power consumption values.

Note that the Back-end flow was not completed entirely as for performing post-route simulation and signoff due to issues with a malfunctioning memory macro that the design uses as a global memory (*TSDN40LPA8192X32M8M*). Hence a toggle feed to Innovus could not be generated to calculate the most accurate power and timing results. The most information attainable consisted in static analyses post routing-optimization which use a constant activity factor across the whole logic. These results provide however sufficient data in terms of power consumption, leakage, and certain paths that can be tested individually. Additionally, the scripts used in this case can be found in the annexes to this report.

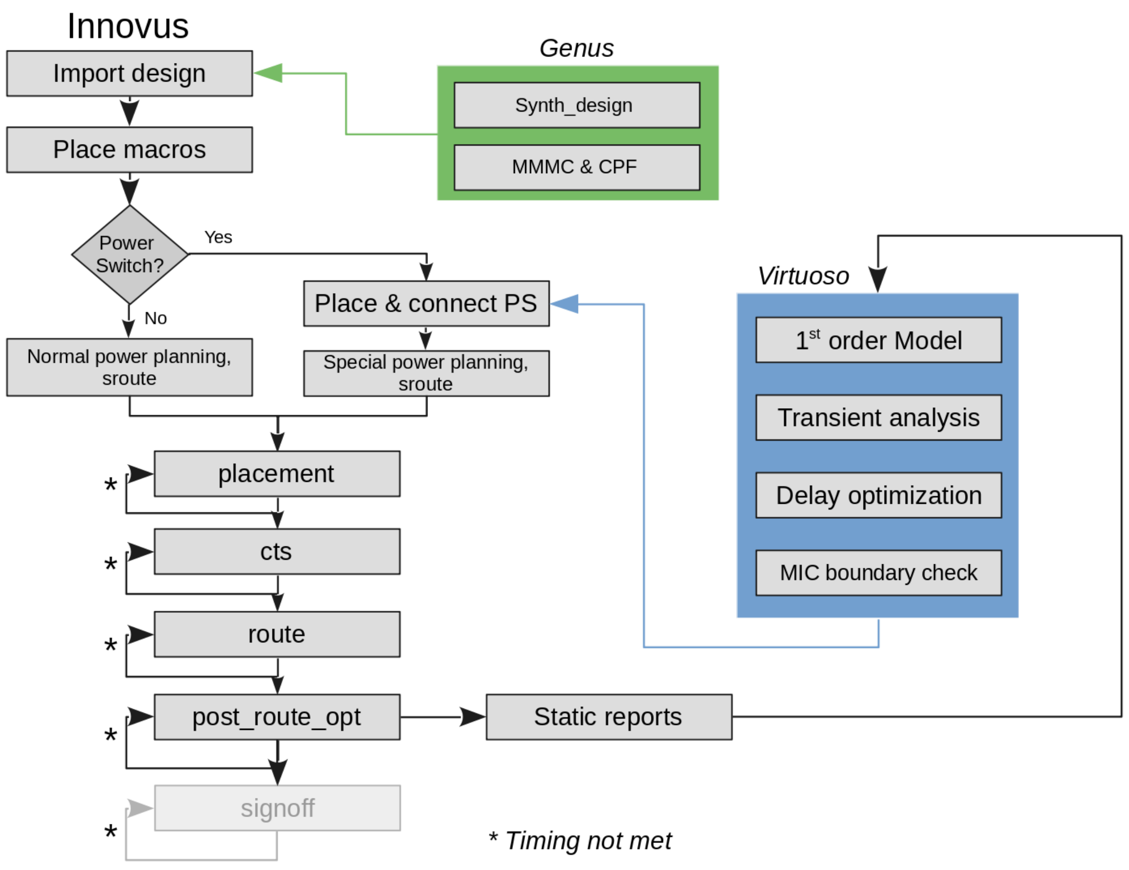


Figure 4.6: Innovus back-end flow

The used flow goes as follows:

- *import_design*: the output from Genus is read, importing the design and all its defined configurations, settings and constraints, this means that of a low-power design flow is not needed in Innovus anymore, other than in terms of floorplan and the rest of back-end. It is worth noting that the CPF configuration can be also edited and re-applied in case of modification on power domains, or re-assignation of instances to/from a power-switched domain.

- *place_macros*: as its name indicates, corresponds to the physical allocation of memory modules and other fixed modules. The process is very similar in the case of a flat or a power switched designs, depending of course on which module(s) are to be gated. If the latter is the case, then additional power domains have to be allocated as blocks, setting up their target density from a start.
- *Place and connect PS*: This part has to be done before *route_sppecial* is performed, here the switches are placed in either a ring or column configuration and have to be connected with their respective VDD, VSS, TVDD and TVSS. Once that step is performed along with the power rings and lines of the design, we can proceed to run special route, which will introduce the VDD and VSS of every row and every power domain with their respective macro power nets. If the design is flat, the procedure is relatively standard: insert rings, rows and special route.
- *placement*: Corresponds to the placement of all the standard cells in the design, along with preliminary routing connections, however none of those are fixed.
- *cts*: inserts the clock tree and tests the design for timing violations, this prioritizes the clock hence the routed put during placement will be removed if needed.
- *route*: only after the clock tree has been successfully inserted, the routing of the rest of the logic takes place, this is also a rather automated step.
- *post_route_opt*: this step cycles through the nets of the routed design checking for timing violations (for both hold and setup), and performs the required changes in order to fix them. This process can take several iterations to get a clean output and will have more difficulty in designs that are more dense.
- *static reports*: after the optimizations have taken place and the design has no violations, parasitic capacitances are extracted and reports are generated. They include breakdowns in power consumption, timing, area, capacitance and so on. These reports are then used to simulate the transient behavior of the switched module.
- *Virtuoso*: Having the data about the module, we can calculate the number and type of power switches used and run a simulation on a first order model when charging and discharging the module. Here, items like the Max. immediate current (MIC) can be generated and changed by adding proper delay between the switches and building a "switch propagation tree". This simulation would give us the detail of the switch power consumption, as well as the nr of cycles that it would take to wake up.

The results of this final analysis can be used to feed the back-end development on a following run, providing the proper information on buffers; signal propagation; nr of cells; IR drop, etc. which in turn can lead to modifications in the floorplan, power-switch methodology used and maybe even timing constraints.

4.6 Power switching granularity: the path traversal method

The decision around granularity of power switching is highly architecture dependent for pretty much every time that it is attempted. The case case of the CGRA is no exception. And as with other configurable architectures, generally the isolated case of power gating one single functional unit (or LUT / DSP slice in case of an FPGA) generally will have little impact on the network and its traffic. However as one adds more functional units to the power-gated pool, parts of the interconnect start becoming unused, leaving the opportunity of saving their idle leakage if they were power gated. Since the decision on where and what to allocate to the power islands becomes static: How do we conciliate extending the reach of the pool starting on an idle functional unit to other modules; either fully or partially; without sacrificing the possibility of re-configuration? In other words, is it possible to power switch the interconnect aware of the functional units' state at a given time? We explore here 2 possible answers:

1. *Yes: put a switch on every switchbox independently and leave it to the power controller.*
2. *Maybe: if we manage to partially extend the switches' reach by checking every cell individually.*

In the case of the first answer, it will simply become the justification for a test group where the switchboxes are tested individually, in what we could call a naive approach towards the interconnect.

To answer the second question: the path traversal method aims to check on a cell-by-cell basis, and determine whether they qualify for being power switched or not. To do this a simple algorithm has been put in place to establish the maximum granularity possible given a minimally sized "seed". It works as follows:

- We define a cell c, as a construct composed of two lists, one for input cells (c_{in}) and outputs cells (c_{out}). Being c_{in} and c_{out} represented as a list of other cells that are connected to the the input and output of c respectively. Likewise, we can take the inputs and outputs of a set of cells, as the union of all inputs and outputs of the cells in the set. - We define a module M, containing any number of cells. - We define a seed S, as a set of outputs in a module M, this is a list of special cells that only contain a list of inputs (S_{in}).
- Given M and S, we now accumulate all the cells backwards in the path towards S until there are no more cells to add, defining the set K of candidate cells:

$$\begin{aligned} let\ (C0 \subseteq M);\ \forall\, c \in M; (c \in S_{in} &\implies c \in C0) \\ let\ (C1 \subseteq M);\ \forall\, c \in M; (c \subseteq C0_{in} &\implies c \in C1) \\ let\ (C2 \subseteq M);\ \forall\, c \in M; (c \subseteq C1_{in} &\implies c \in C2) \\ &\quad ... \\ let\ (Ci \subseteq M);\ \forall\, c \in M; (c \subseteq C(i-1)_{in} &\implies c \in Ci) \\ &\quad i \in \mathbb{N} \end{aligned} \tag{4.5}$$

$$K \equiv C_0 \cup C_1 \cup ... \cup C_N$$

Having accumulated all the cells that are part of the paths leading to S, We now check for every cell in K that its outputs connections are self contained in K, and if so, they belong to the power switch set PS. In other words, we want to select the cells that belong exclusively to paths leading to S:

$$\forall\, c \in K;\ \forall\, cell_{out} \in c;\ c_{out} \in K \longrightarrow c \in PS \tag{4.6}$$

Implementing this algorithm took a functional-before-efficient approach, as it requires a massive amount of iterations and re-iterations lists of cells in M. In fact, this naive implementation has a worst case scenario of *O(n!)*. For a set K containing n cells, every cell's next cell should be present in K in order for it to stay; however if at any point in time a cell needs to be removed from the set, then all formerly marked cells need to be checked again. So for n cells with n output cells, it could take up to n! checks if always the last output of the last cell were not contained in K.

Fortunately, the modules in the CGRA are of a manageable size (up to 5000 cells), allowing for the checks to complete in a reasonably low time. Additionally, given the inherently modular structure of any circuit design, it can be run throughout big designs by turning one module's inputs as the seeds for the next one. figure 4.7 illustrates an example of applying the method on a circuit using Out_1 as the seed. The whole process begins with the Cx subgroups that were added while propagating all the cells whose path led to the seed, then adding it all on a single pool K. Then we check for every cell in K, that its output cells are also contained in K, else we have an escaping path. If that is the case, then remove said cell from K and try again until all cells are checked. What is left on K will correspond to the largest path(s) exclusively towards the seed, and our qualifying cells to be power switched.

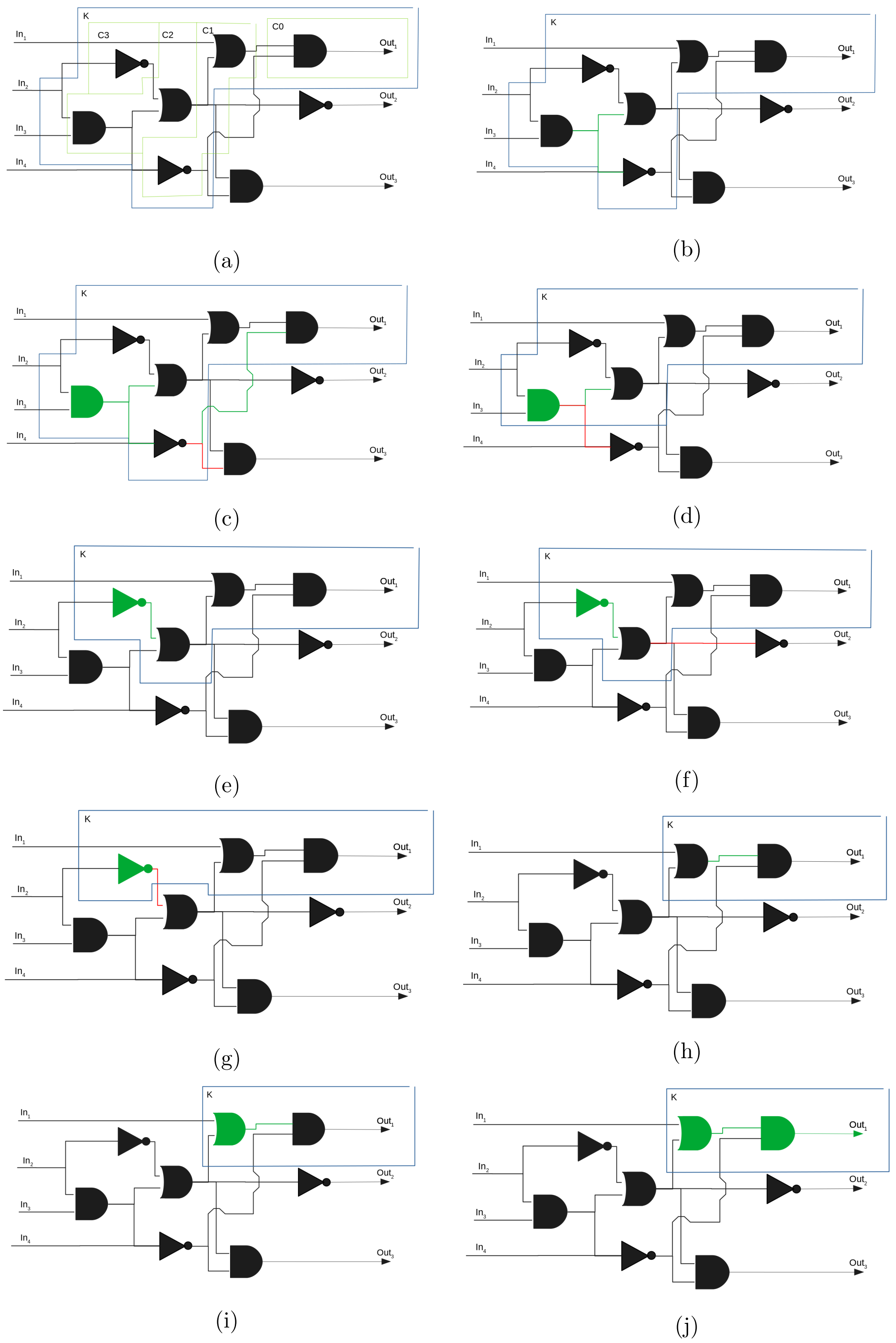


Figure 4.7: Example of the path-traversal algorithm proposed and used for power switch assignation.

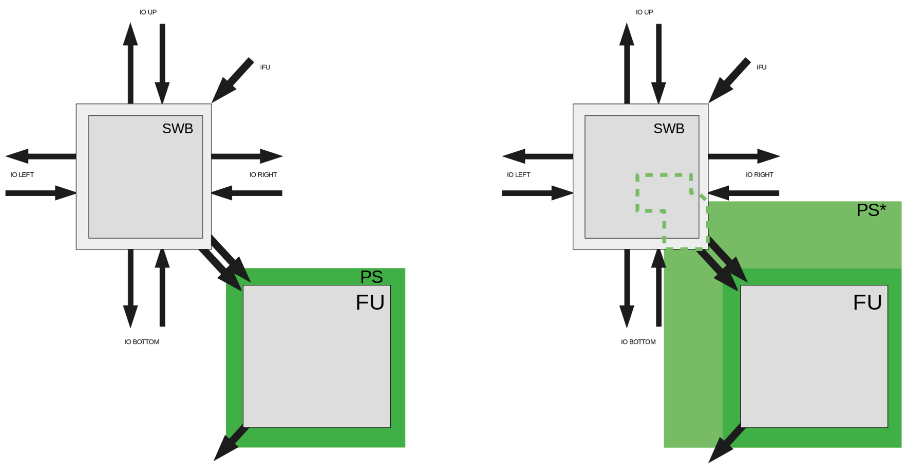


Figure 4.8: The objective of the path traversal method is to expand the power-gated area farther from the functional unit into the interconnect, without altering the functionality of the interconnect.

Naturally, if all outputs of a selected set are defined as seeds, the whole of the set will be switched which in this case often justifies switching a whole functional unit. This method is proposed to extend that to part of the switchboxes connected to it. This would allow us for a bigger switch-off area without changing functionality when cutting the power to any particular functional unit (see figure 4.8). Since the size of the switchboxes varies from a 100 and 5000 cells depending on how it was generated and what type of connections it supports, the results of this addition could end up having an important impact on power consumption.

4.7 Control

One of the biggest advantages of power switching on the CGRA, is that its behavior is deterministic, meaning that at any particular time, we should be able to know the state of each of its functional units. This means that the control logic needed to run the power switches timely is minimal. In fact, the activation of power switches could be triggered the same way the algorithms are implemented in the PASM code developed for the CGRA. This allows for implementing dynamic power-switch control rather easily. while constraining the shut-off and waking-up times to a limited number of clock cycles, we could send power-switch instructions directly into the PASM code and exploit any long idle periods.

Additionally, in case that the CGRA is not needed at all as an accelerator, we could implement a CGRA-wide power switch controlled by the main processor, normally an ARM or RISC5 processor as it has been proposed in some official designs including a CGRA. This is indeed a very attractive possibility as accelerators are generally idling for long periods before jumping into activity.

Chapter 5

Experimental results

Having defined the research approach and all the gears that this report used to gather data and reach conclusions, we will proceed to the tests themselves. Every dot in the graphs represent different power-switching settings, applied on both benchmarks, the settings correspond to the following:

1. Functional units only (FU): picking every single functional unit independently based on its outputs

2. Switchbox only (SWB): This case grouped both data and control switchboxes for the analysis, namely because control switchboxes are generally extremely small.

3. Extended functional units (FU+): using the path-traversal algorithm proposed in chapter (4), every FU switching was extended to its respective pair of data/control switchboxes.

This chapter will begin in section 5.1, which presents the results of the path-traversal algorithm proposed in the methodology, applied to the functional units to form the extended functional unit test-group. Section 5.2 briefly summarizes the impact that isolation cells had in each of the test-groups, as it will be a crucial factor in determining whether power-gating would be effective in each case. Section 5.3 shows the area impact of power-gating throughout all groups. Section 5.4 constructs the complete energy analysis and shows the break-even points in the tests made. Section 5.5 applies power gating dynamically in both benchmarks. And finally, section 5.6 summarizes the area and power analyses when applied to the complete CGRA.

5.1 Path traversal method

One of the test groups proposed in the methodology section, consisted in an algorithmic extension of the power island around a single functional unit. This extension is designed to maximize the number of cells switched off when a functional unit is, without altering the functionality of the interconnect network. Hence it would allow the designer to maintain the maximum flexibility in terms of running different applications and configurations. The results of the extensions on the functional units are shown in figures 5.1 with absolute results, and figure 5.2 for relative results.

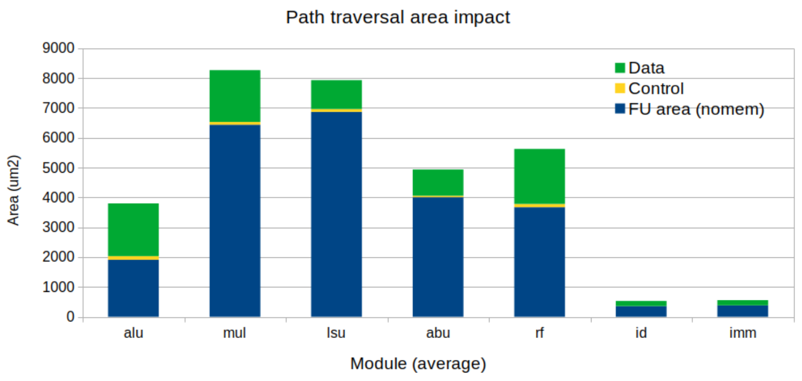


Figure 5.1: Impact of the path traversal method to extend the reach of a FU's switch decision in terms of instance area.

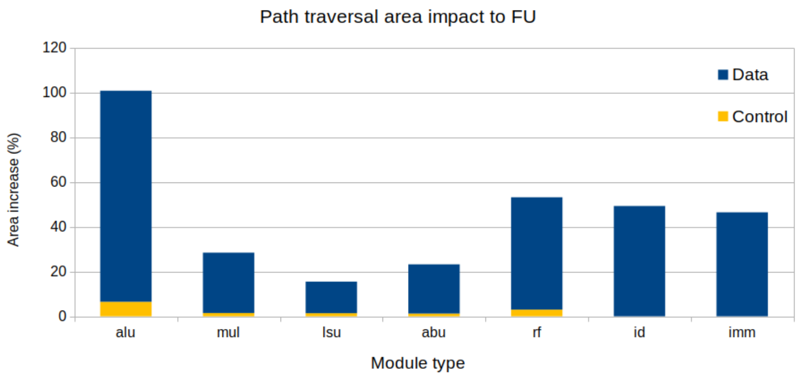
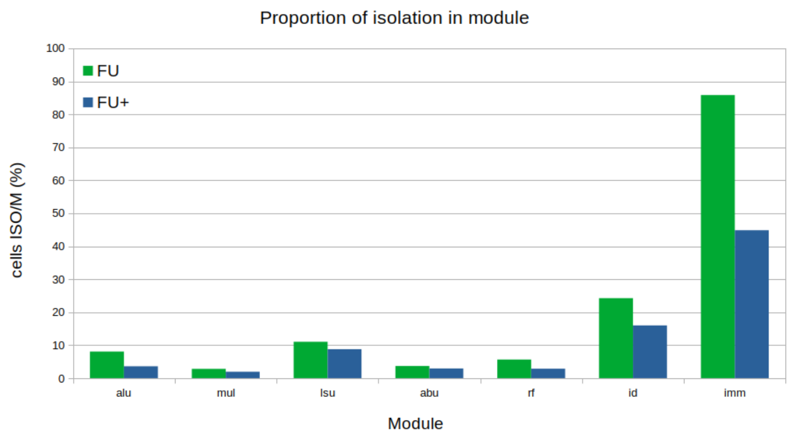


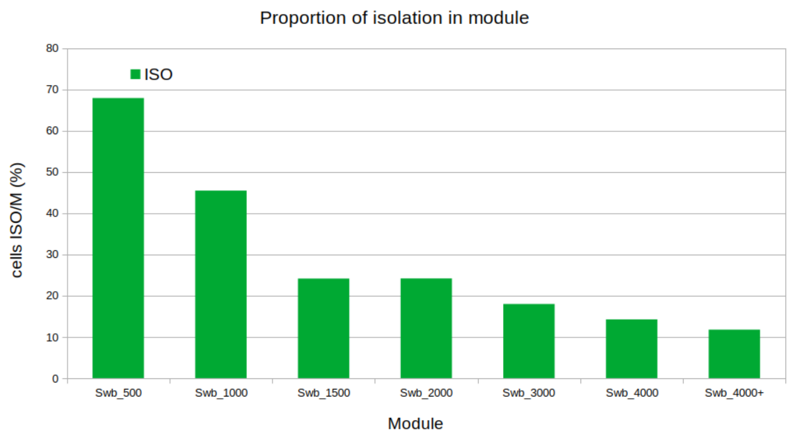
Figure 5.2: Same path traversal method impact measured as a fraction of the functional unit. As can be seen, relatively small functional units doubled in size while keeping the same amount of isolation.

5.2 Isolation

As discussed in chapter 2.3, isolation cells can have a significant impact on the power consumption of the modules that have been power gated. As shown in figures 5.3 functional units require a rather low number of isolation cells, due to having a single array of outputs to their respective switchbox, this is not the case of instruction decoders and immediate units because they are functional units too small to disseminate the impact of isolation. Finally, the switchboxes are the complete opposite of what happened to the functional units, where the arrays of outputs go from 2 to 6, and the only logic involved corresponds to the configuration and redirection of traffic needed.



(a)



(b)

Figure 5.3: Proportion of isolation cells required for power-gating every particular module.

5.3 Area Overhead (AO)

Testing all different PS settings the combinations on the benchmarks, one can note that there is a clear benefit to scaling the power switching to bigger island, where the asymptotic minimum overhead corresponds to the set distance between the adjacent power domains. For the tests, the distance between power domains was set to: $halo = 1.68um$.

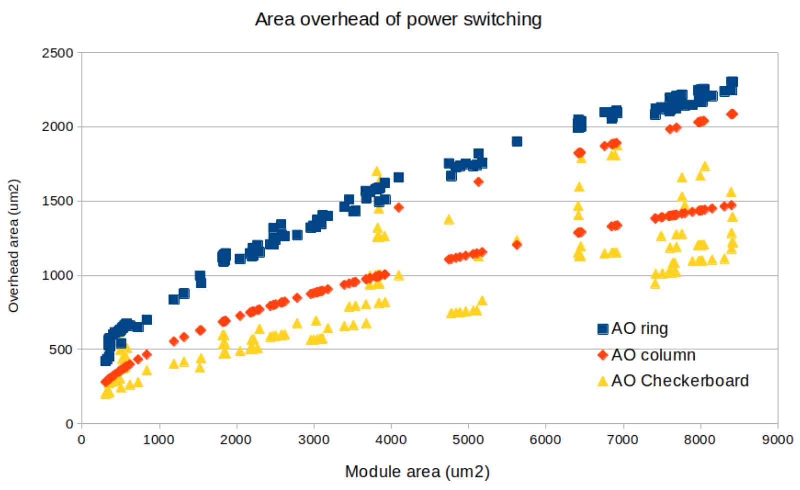


Figure 5.4: Area impact of power switching in floorplanning.

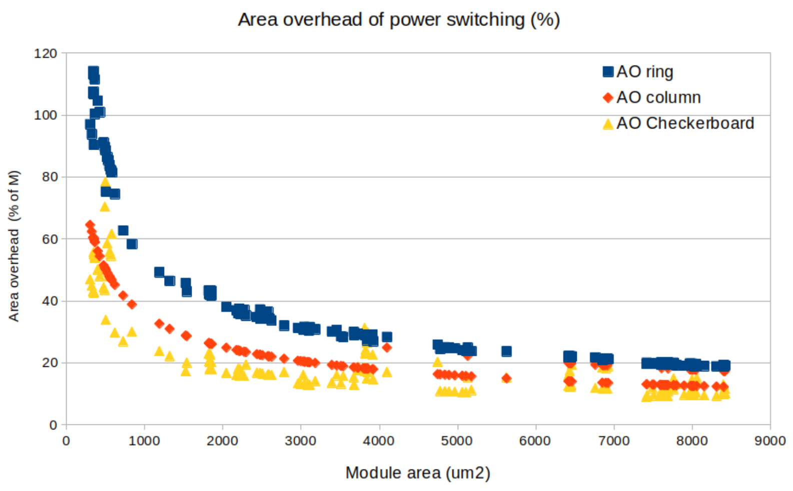


Figure 5.5: Area impact of power switching relative to the size determined for the module M.

As it can be seen in both figure 5.4, and 5.5, there are areas with more and less data density: on the X axis has to do with the type of functional units tested, where certain sizes had much more recurrence than others. and the jumps in the Y axis corresponds to the assignation of power switches to the modules in every case. This assignation is based on the power characteristics of the modules and the power switches available for it, where roughly, one big power switch will replace 15-20 smaller switches, provided that this change would leave enough small switches available for waking up slowly (hence limiting the voltage drop on the main power domain).

5.4 Energy impact

In this section, we broke the analysis into all the different states in which the switched module M will go through, there are two constant states and two transient states, and these will be first addressed separately, and later on combined to make an energy analysis.

1. Steady states: represent sustained activity of module M (Active mode), and when the module M is switched off (Sleep mode) . The data for these states was generated by Innovus after place and route.

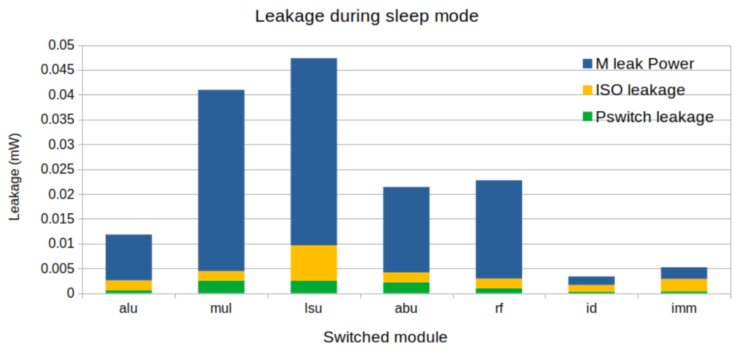
2. Transient states: represent the switching from Active mode to Sleep mode (Sleep), and the switching from Sleep mode to Active mode (Wakeup). The transient analysis data for every point is generated by Virtuoso / Spectre using the first order model.

5.4.1 Steady state: Sleep mode

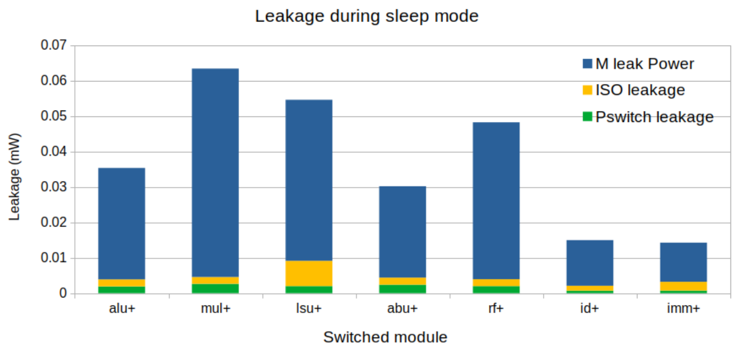
During inactivity we assume that the module has gone into an off state, then, the only relevant power consumption corresponds to the leakage of the switches and isolation cells. Figure 5.6 shows the total impact of the power switches and isolation cells on the modules tested, and figure 5.7 shows the relative reduction of power gating onto the leakage of the modules tested.

In the case of the power switches themselves, their leakage across all three groups averaged a 6.88% of the benchmark and ranged between [3.18% – 19.13%], leaving the field open for a substantial reduction potential. However, the gains in power from the switches are quickly eroded by the inclusion of isolation cell; and this is where the test groups differ the most, as the isolation cells averaged 10 times that of the switches, 68.71%.

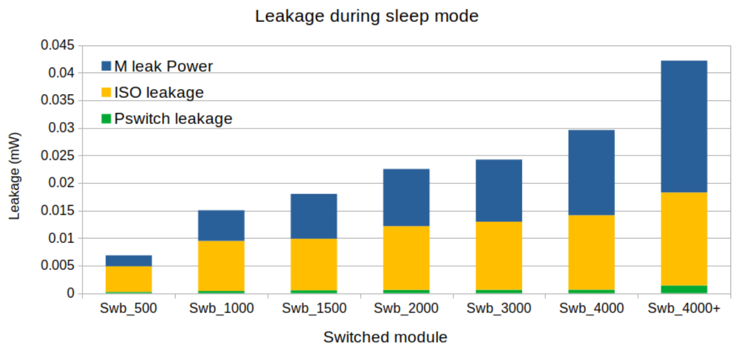
It seems that the switchboxes (fig. 5.6 and 5.7) themselves are not great candidates for power gating, given the fact that too many outputs require too many isolation cells that end up canceling the initial gains, or plainly making it even worse. This is also the case of some small functional units that have a relatively high number of outputs, these being instruction decoders (ID) and immediate units (IM). In the case of LSU's, even though the number of isolation cells is by far the highest among all FU's, their size allow for them to compensate for that extra overhead by providing relatively more gains when switched, the proportions of isolation cells to each functional unit was discussed in section 5.2.



(a)

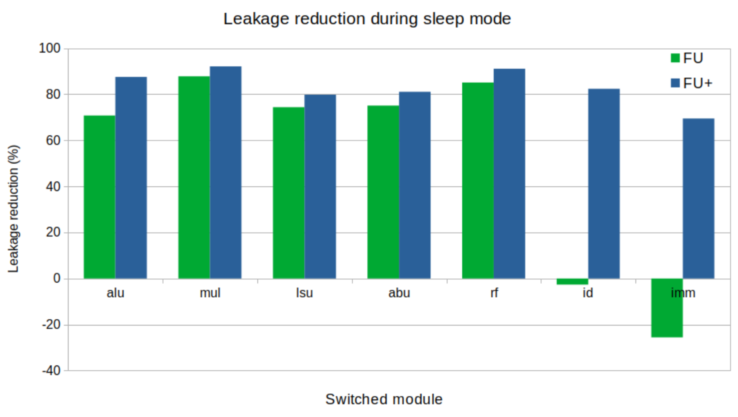


(b)

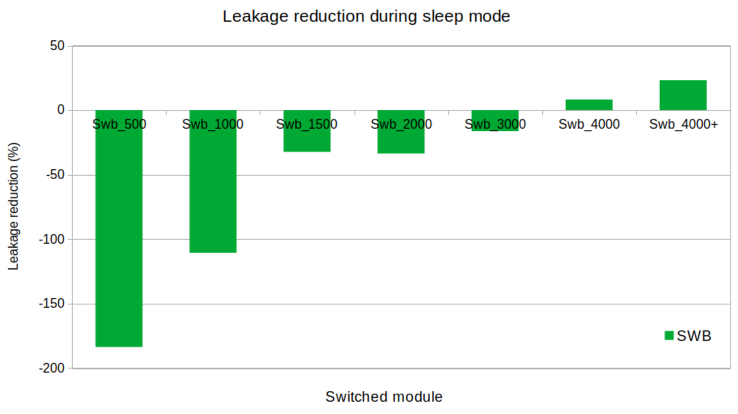


(c)

Figure 5.6: Absolute leakage on sleep for a) functional units, b) extended functional units, and c) switchboxes.



(a)



(b)

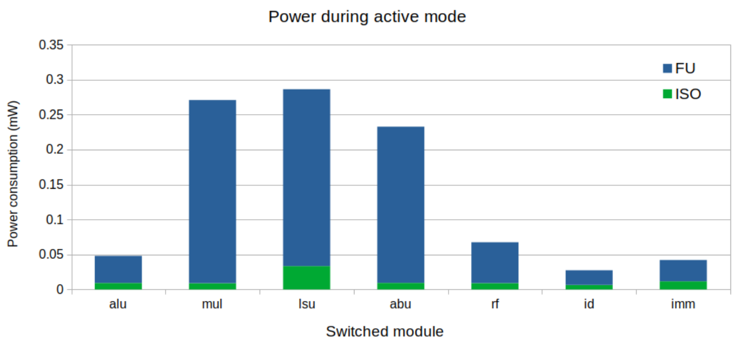
Figure 5.7: Relative reduction of leakage leakage on sleep for a) functional units, and b) switchboxes. Calculated as 100% minus the proportion of leakage of the switches and isolation, with respect to the leakage of the module switched.

In the case of the functional units, smaller immediate units and instruction decoders neither seemed to be able to compensate their isolation overhead. Interestingly, the cases of FU's extended using the path traversal method (seen with a "+") had, without exception, better results than the non extended ones, and the reason for this improvement is simple: the number of isolation cells remained constant, therefore the incremental cost of every added cell is only the leakage from the extra power switches, which have proven to be extremely efficient. This can be more clearly seen in figure 5.7 where the relative leakage reductions are shown. If we separate the results of each group, we get that the weighed leakage reductions are:

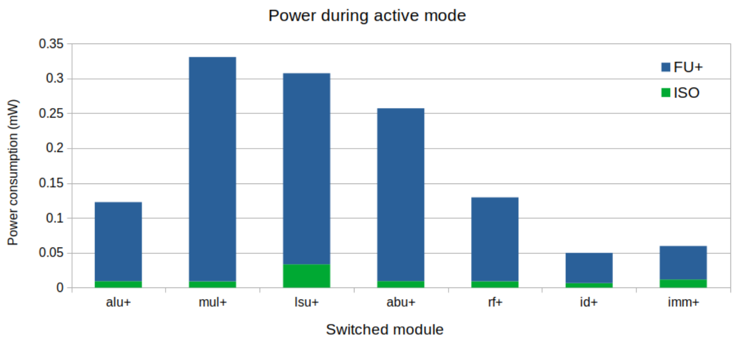
- Switchboxes: 3.12%
- Functional units: 75.9%
- Extended functional units: 85.64%

5.4.2 Steady state: Active mode

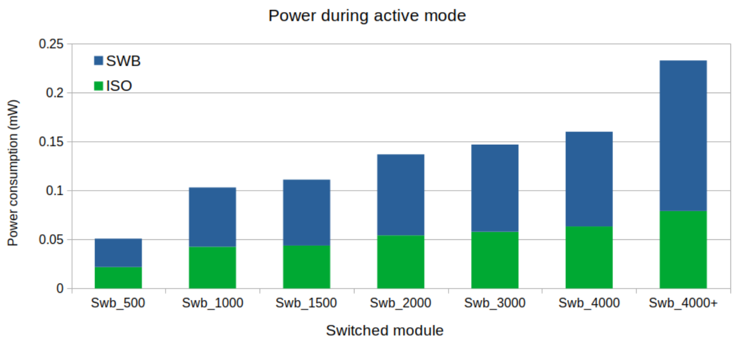
For analyzing the power consumption in active mode, both leakage and dynamic power of the module M were including the isolation overhead were collected. Similarly to the leakage, the impact is heavily dependent on the number of isolation cells present in M. From figure 5.8 one can see results consistent to those of the leakage analysis, leaving the switchboxes at seemingly greater loss than gain. And in terms of the other functional units; instruction decoders (ID) and LSU's seem to loose more due to their higher number of isolation cells relative to their original number of cells (as shown in section **??**).



(a)



(b)



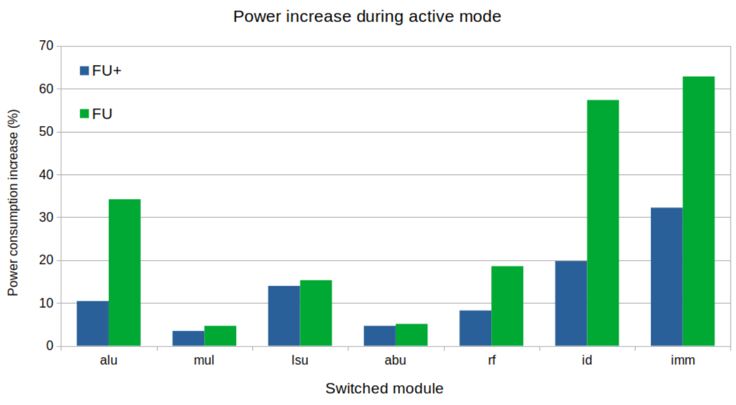
(c)

Figure 5.8: Absolute values on active mode power for a) functional units, b) extended functional units, and c) switchboxes.

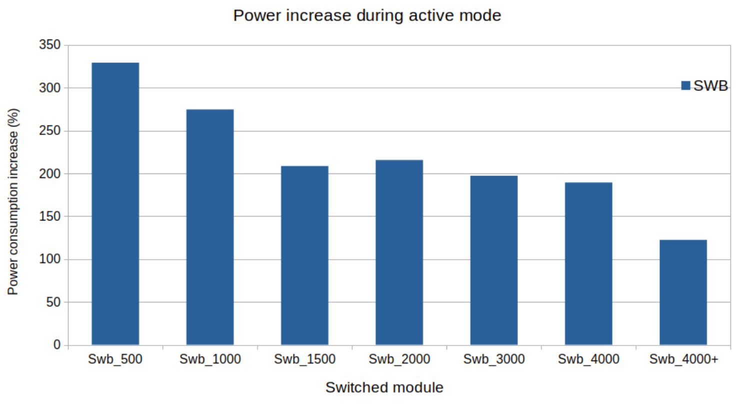
The weighed average power increase for each group was:

- Switchboxes: 143.22%
- Functional units: 13.54%
- Extended functional units: 9.59%

Even though the extended functional units have a higher power consumption in activity, the introduction of the isolation cells generated a substantially lower increase in their power. This is better shown in figure 5.9, where in the smallest functional units, the relative increase was cut by half, and remained consistently lower throughout the FU's. Conversely, sheer size could not save the switchboxes, where the power numbers were prohibitively high for most of them.



(a)



(b)

Figure 5.9: Relative values for power increase during active mode for a) functional units and extended functional units, and b) switchboxes. Calculated as 100% minus the proportion of leakage of the switches and isolation, with respect to the leakage of the module switched.

5.4.3 Transient states: Wake-up and Shut-off

Having calculated the static states using the data generated by Innovus, we further generated the information to feed the first order model used to estimate the transient behavior of our switched module. The results were plotted, and as expected from the simplicity of our estimation model, the relation between capacitance and the wake-up energy was rather apparent (see figure 5.10). The results were consistent to the well known model for capacitors charge: $Q = C * V$.

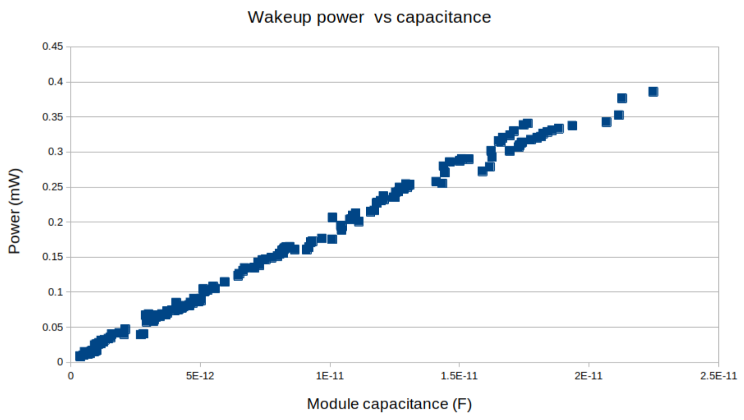


Figure 5.10: The wake-up energy results were similar to charging a single capacitor, the slight variations on the results come from the modelled resistance, thus leakage, in the core. This figure shows wake-up power instead of energy to facilitate its comparison with the other states, it was done so by taking the wake-up period as three clock cyces ($30ns$)

Having the data from the transient analysis we can now estimate the total power consumption throughout an entire cycle of sleep and activity. Hence adding up the energy savings and the overhead as described in the metrics 3 we calculate that, on a single on-off cycle:

$$E_{savings} = P_{mod,leak} * t_{down} + E_{powerdown} * N \tag{5.1}$$

being $N = 1$, and $E_{overhead}$ is:

$$\begin{aligned} E_{overhead} = t_{down} * (P_{switch,leak} + P_{iso,leak}) \\ + t_{active} * P_{iso,active} + E_{wakeup} * N \end{aligned} \tag{5.2}$$

We need to know the relation between t_{active} and t_{sleep}, so that $E_{overhead} < E_{savings}$, and when replacing the different values on each side, we obtain:

$$\begin{aligned} t_{active} * P_{iso,active} < t_{sleep} * (P_{mod,leak} - P_{switch,leak} - P_{iso,leak}) + (E_{powerdown} - E_{wakeup}) * N \\ P_{iso,active} < \frac{t_{sleep}}{t_{active}} * (E_{net\_savings}) + \frac{c}{t_{active}} \end{aligned} \tag{5.3}$$

If we separate $E_{powerdown} - E_{wakeup}$ and group then in a constant c, as well as calling $E_{net\_savings} = P_{mod,leak} - P_{switch,leak} - P_{iso,leak}$, we can note that as time passes, the constant c becomes less relevant to the analysis (in the tests, it had a net vale equivalent of a couple clock cycles worth of energy), This leaves us with the relation:

$$\frac{t_{active}}{t_{sleep}} > \frac{E_{net\_savings}}{P_{iso,active}} \tag{5.4}$$

This is in principle the same relation proposed in the research prior to the study, accommodated to the particular case of the CGRA, where no state retention, nor a control infrastructure were used.

5.4.4 Break - even point

Having applied the calculations, the break-even points per each tested module are presented in 5.12 for the functional units and their extended versions. It is clear to see the great impact that the path traversal method had on the CGRA, where now without exception every functional unit can be optimized given the constraints of the algorithms used and the activity that they present.

Since the behaviors in the CGRA are of deterministic nature, the designer can now precisely know whether each functional unit would benefit from power switching by comparing the expected activity against the allowable ratios of their BEP.

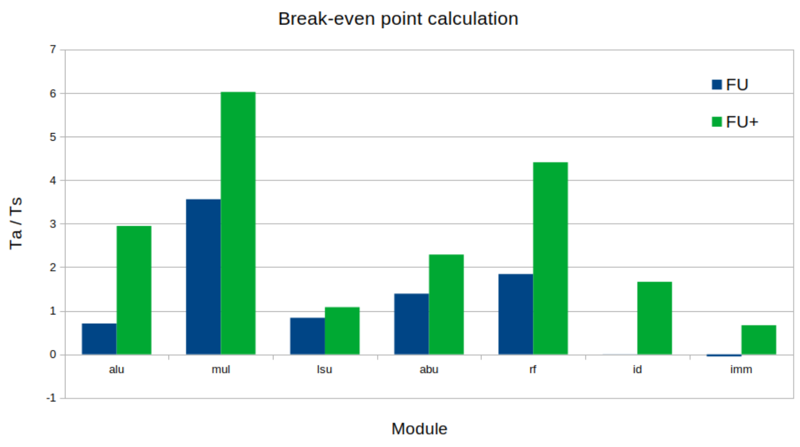


Figure 5.11: The break-even point calculated on the functional units with respect to their extended versions, now shows the real performance of the path traversal method to increase the gain from power switching in the CGRA.

Conversely, the results for the switchboxes showed negative values for the great majority, meaning that there is no solution to the relation proposed earlier, as both leakage and power during activity have increased. This closes of the question of viability of power switching in what concerns switchboxes.

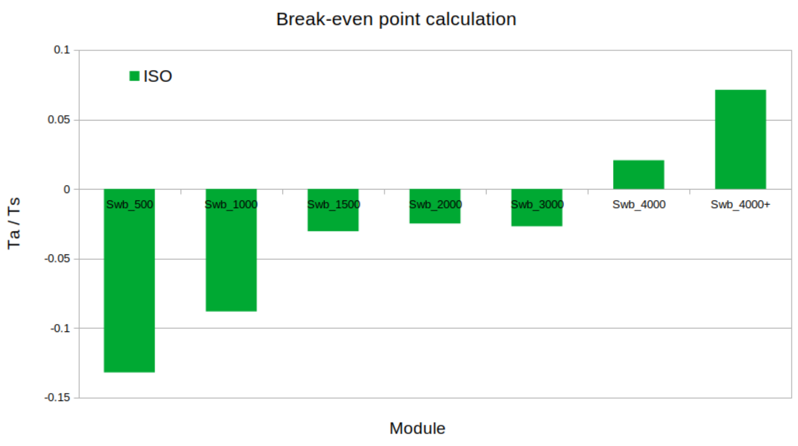


Figure 5.12: the break-even point for most switchboxes was negative, meaning that there is no possibility of producing any gains because both the leakage and the power during active mode have increased with respect to the non-switched version.

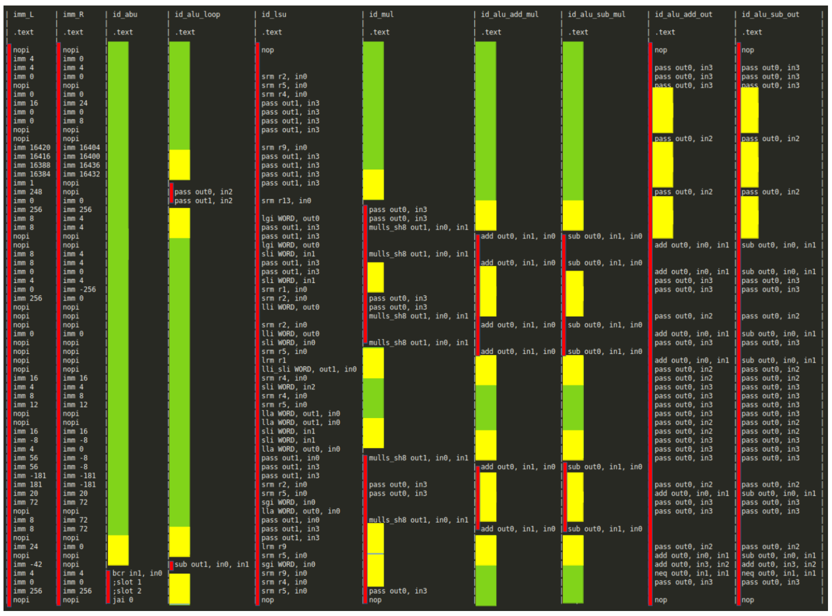
5.5 Dynamic power-gating

Having calculated the break-even points in all cases, we can conclude that the extended functional units (FU+) are, without exception, a better implementation of power gating when compared to the other two test groups. It would be therefore interesting to see how it could be applied on the actual benchmarks we have begun with. On each benchmark, the idle times of every functional unit can be identified by looking at the PASM code controlling the CGRA.

One of the biggest advantages of power switching on the CGRA, is that its behavior is deterministic, meaning that at any particular time, we should be able to know the state of each of its functional units. This means that the control logic needed to run the power switches timely is minimal, therefore the behavior of dynamic power gating can be easily estimated in each of the benchmarks.

5.5.1 FFT

Having analyzed the instructions on which the FFT runs, the gaps in activity that are long enough to merit power savings were selected and tested using the models developed in this research. Figure 5.14 shows the loop in which the parallel FFT occurs, highlighting the possibilities for power gating, where green corresponds to energy saving, and yellow represents transients states (where energy is consumed), as well as the red areas, which are times of activity, or gaps between activity too small to fit a power-gating cycle.



```
imm_L

.text
nopi
imm 4
imm 4
imm 0
nopi
imm 0
imm 16
imm 0
imm 0
nopi
nopi
imm 16420
imm 16416
imm 16388
imm 16384
imm 1
imm 248
imm 0
imm 256
imm 8
imm 8
nopi
nopi
imm 8
imm 8
imm 0
imm 4
imm 0
imm 256
nopi
nopi
nopi
imm 0
nopi
nopi
nopi
nopi
imm 16
imm 4
imm 8
imm 12
nopi
nopi
imm 16
imm -8
imm 4
imm 56
imm 56
imm -181
imm 181
imm 20
imm 72
nopi
imm 8
imm 8
nopi
imm 24
nopi
imm -42
imm 4
imm 0
imm 256
nopi
```

```
imm_R

.text
nopi
imm 0
imm 4
imm 0
nopi
imm 0
imm 24
imm 0
imm 8
nopi
nopi
imm 16404
imm 16400
imm 16436
imm 16432
nopi
nopi
imm 0
imm 256
imm 4
imm 4
nopi
nopi
imm 4
imm 4
imm 0
imm 4
imm -256
imm 0
nopi
nopi
nopi
imm 0
nopi
nopi
nopi
nopi
imm 16
imm 4
imm 8
imm 12
nopi
nopi
imm 16
imm -8
imm 0
imm -8
imm -8
imm -181
imm -181
imm 20
imm 72
nopi
imm 72
imm 72
nopi
imm 0
nopi
nopi
imm 4
imm 0
imm 256
nopi
```

```
id_abu

.text
bcr in1, in0
;slot 1
;slot 2
jai 0
```

```
id_alu_loop

.text
pass out0, in2
pass out1, in2

sub out1, in0, in1
```

```
id_lsu

.text
nop


srm r2, in0
srm r5, in0
srm r4, in0
pass out1, in3
pass out1, in3
pass out1, in3
pass out1, in3

srm r9, in0
pass out1, in3
pass out1, in3
pass out1, in3
pass out1, in3

srm r13, in0

lgi WORD, out0
pass out1, in3
pass out1, in3
lgi WORD, out0
sli WORD, in1
pass out1, in3
pass out1, in3
sli WORD, in1
srm r1, in0
srm r2, in0
lli WORD, out0

srm r2, in0
lli WORD, out0
sli WORD, in0
srm r5, in0
lrm r1
lli_sli WORD, out1, in0
srm r4, in0
sli WORD, in2
srm r4, in0
srm r5, in0
lla WORD, out1, in0
lla WORD, out1, in0
sli WORD, in1
sli WORD, in1
lla WORD, out0, in0
pass out1, in0
pass out1, in3
pass out1, in3
srm r2, in0
srm r5, in0
sgi WORD, in0
lla WORD, out0, in0
pass out1, in0
pass out1, in3
pass out1, in3
lrm r9
srm r5, in0
sgi WORD, in0
srm r9, in0
srm r4, in0
srm r5, in0
nop
```

```
id_mul

.text
pass out0, in3
pass out0, in3
mulls_sh8 out1, in0, in1

mulls_sh8 out1, in0, in1

pass out0, in3
pass out0, in3
mulls_sh8 out1, in0, in1

mulls_sh8 out1, in0, in1

mulls_sh8 out1, in0, in1

pass out0, in3
pass out0, in3

mulls_sh8 out1, in0, in1

pass out0, in3
nop
```

```
id_alu_add_mul

.text
add out0, in1, in0

add out0, in1, in0

add out0, in1, in0

add out0, in1, in0

add out0, in1, in0

add out0, in1, in0
```

```
id_alu_sub_mul

.text
sub out0, in1, in0

sub out0, in1, in0

sub out0, in1, in0

sub out0, in1, in0

sub out0, in1, in0

sub out0, in1, in0
```

```
id_alu_add_out

.text
nop

pass out0, in3
pass out0, in3
pass out0, in3

pass out0, in2

pass out0, in2

add out0, in0, in1

add out0, in0, in1
pass out0, in3
pass out0, in3

pass out0, in2

add out0, in0, in1
pass out0, in3

add out0, in0, in1
pass out0, in2
pass out0, in2
pass out0, in3
pass out0, in3
pass out0, in3
pass out0, in3
pass out0, in2
pass out0, in2
pass out0, in3
pass out0, in3
pass out0, in3

pass out0, in2
add out0, in0, in1
pass out0, in3
pass out0, in3

pass out0, in2
add out0, in0, in1
add out0, in3, in2
neq out0, in1, in1
pass out0, in3

nop
```

```
id_alu_sub_out

.text
nop

pass out0, in3
pass out0, in3
pass out0, in3

pass out0, in2

pass out0, in2

sub out0, in0, in1

sub out0, in0, in1
pass out0, in3
pass out0, in3

pass out0, in2

sub out0, in0, in1
pass out0, in3

sub out0, in0, in1
pass out0, in2
pass out0, in2
pass out0, in3
pass out0, in3
pass out0, in3
pass out0, in3
pass out0, in2
pass out0, in2
pass out0, in3
pass out0, in3
pass out0, in3

pass out0, in2
sub out0, in0, in1
pass out0, in3
pass out0, in3

pass out0, in2
sub out0, in0, in1
add out0, in3, in2
neq out0, in1, in1
pass out0, in3

nop
```

Figure 5.13: PASM configuration of the parallel FFT algorithm, highlighting opportunities for switching off dynamically.

As it can be seen from the figure, the accumulate -branch unit (abu) and a single alu_loop are idle on all 63 cycles that the algorithm takes, except for a couple of instructions. Less optimally but still promising are the 8 multiplier units (mul) and 8 more ALU's. The colored columns indicate in green: where power gating provides savings in leakage; yellow: that the functional unit is in a transient state either turning on or off; and red: that the functional unit is busy or does not have enough time to switch off before having to switch on again. This can be seen in the 2 last columns, where even though there are windows of 3 clock cycles, that is exactly the time on which the FU's are configured to switch on/off, thus not being able save energy, in fact, they would consume more energy that way.

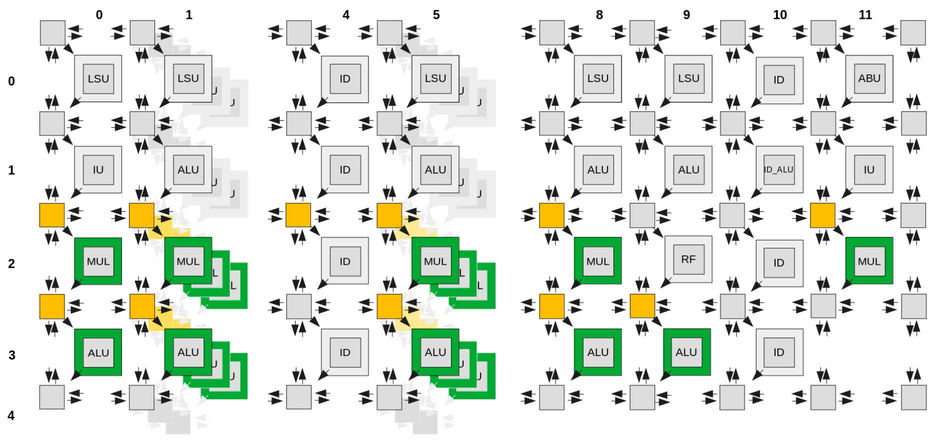
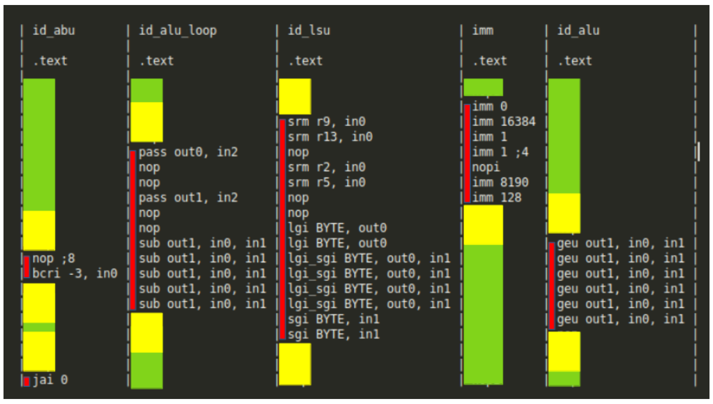


Figure 5.14: green: functional units that were dynamically switched. yellow: switchboxes that were partially switched, along with their corresponding functional unit.

Even having selected a reduced set of functional units, the energy savings on the CGRA accounted for 14%. was calculated using the same methodology used to calculate the break-even points, though this time, since the power is calculated in a window of 63 clock cycles, the energy consumed during all transitions becomes very relevant. Another point to highlight is the fact that 4 instruction decoders, namely the ones controlling the power switched functional units could potentially also be power switched along with their slave functional units. Should this be done, the power savings would increase by an extra 2%, however this would end up falling into the designer's choice on implementing the power switch control mechanisms: should it be integrated into every FU's as an instruction, then we would need to keep the decoders on to control the switches; should it be controlled by an additional functional unit, would allow for switching both instruction decoders and functional units alike.

5.5.2 Binarization

In the case of our smaller benchmark, the binarization algorithm, we find ourselves in a much smaller instruction loop, meaning that the CGRA is much more active if compared to the FFT, as shown in figure 5.15. There are however still opportunities to power off the relevant functional units on certain intervals. Similarly to the case of the the FFT, the results presented omitted switching the immediate units and the instruction decoders, under the assumption that the power switches would be controlled by instructions on the functional units.



```
| id_abu        | id_alu_loop           | id_lsu                        | imm          | id_alu                  |
| .text         | .text                 | .text                         | .text        | .text                   |

                                                                          imm 0
                                          srm r9, in0                     imm 16384
                                          srm r13, in0                    imm 1
                  pass out0, in2          nop                             imm 1 ;4
                  nop                     srm r2, in0                     nopi
                  nop                     srm r5, in0                     imm 8190
                  pass out1, in2          nop                             imm 128
                  nop                     nop
                  nop                     lgi BYTE, out0
                  sub out1, in0, in1      lgi BYTE, out0                                   geu out1, in0, in1
  nop ;8          sub out1, in0, in1      lgi_sgi BYTE, out0, in1                          geu out1, in0, in1
  bcri -3, in0    sub out1, in0, in1      lgi_sgi BYTE, out0, in1                          geu out1, in0, in1
                  sub out1, in0, in1      lgi_sgi BYTE, out0, in1                          geu out1, in0, in1
                  sub out1, in0, in1      lgi_sgi BYTE, out0, in1                          geu out1, in0, in1
                                          sgi BYTE, in1                                    geu out1, in0, in1
                                          sgi BYTE, in1

  jai 0
```

Figure 5.15: PASM configuration of the binarization algorithm, highlighting opportunities for switching off dynamically.

And having applied power gating in the functional units identified in the pasm code, and their respective modules in the architecture are shown in figure 5.16. This strategy led to a 7.6% energy reduction on the CGRA, value calculated using the same methods on which the rest of this report is based.

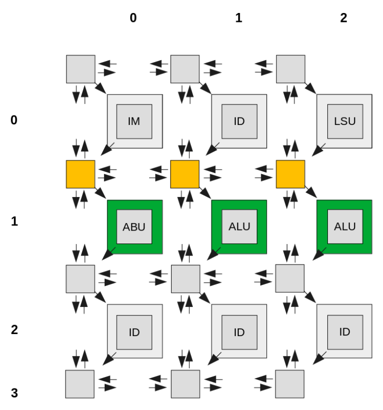


Figure 5.16: The units in green have been power-gated accordingly in the times their PASM code allows, and the switchboxes highlighted in yellow have been partially switched together with their respective functional unit.

5.6 Static power gating

The last strategy tested corresponds to the simple fact of switching the entire CGRA, this is a very relevant solution that could be implemented together with the dynamic power-switching strategies. Additionally, this is a chance of minimizing the power consumption of a chip where the CGRA is utilized as a hardware accelerator. Assuming that local memories are not required to maintain their state, the CGRA is power switched excluding only the global memory block. In terms of area, we calculate the Overhead of each method reviewed and present it as a percentage of overhead relative to the CGRA's area. As seen in the table, the bigger the module, the smaller the overhead.

In terms of power, using the same methods discussed above, it was possible to calculate the Break-even point between active time and sleep time. It important to note that since the CGRA as a whole contains a relatively small number of outputs, the number of isolation cells in both benchmarks was only 240, which has an almost negligible impact on the active power and leakage on both designs. In case of binarization, the ratio was similar to that of an extended multiply functional unit. However the massive size of the FFT allows to reducing the impact of isolation to a barely noticeable level, allowing for power consumption 30x smaller when the CGRA is switched off.

Benchmark	Cell count.	Cell area (mu^2)	AO ring	AO col	AO C.board
FFT	236,215	459,677.94	2.49%	2.35%	2.09%
Binarization	20,341	44,392.12	8.05%	7.56%	4.77%

Table 5.1: Area overheads of the power switching methods researched: ring, column and checkerboard floorplans were applied on the entire CGRA fabric placed with the model at 70% cell density.

Benchmark	Leak p. reduction	Active p. increase	BEP (T_a/T_s)
FFT	96.83%	0.44%	50.19
Binarization	93.98%	3.19%	6.58

It is important to note that even though the power-switched FFT seems to win in every front, it still depends on having long sleep times, as the power savings increase over time as the module slowly discharges through the power switches. Similarly, it would require a much longer wake-up period in order to avoid instability in the supply when switching it on. This issue would depend heavily on the use-case of the CGRA alongside a general processing unit, therefore it was not further tested.

Chapter 6

Conclusion

As integrated circuits become more specialized in the form of accelerators, more power-hungry and shifting towards leakage, it has become paramount to find and implement efficient and scalable ways to save energy through architectural decisions. This has become increasingly challenging as the use of re-programmable hardware has been stepping steadily into the spotlight on modern processors. This lead to the following research questions:

[MQ] At which granularity, in terms of functional units within the context of the coarse grained re-programmable architecture (CGRA), should power gating be applied to be beneficial for power.

[SQ1] Analyze the implementation of power gating from the perspective of functional units, and determine whether is should be added as a default feature of all functional units in the CGRA, investigate the main variables involved and argument a position.

[SQ2] In terms of Floorplanning and the overall physical design, is what is the impact of the different power switching strategies, and how do they compare with the literature in the context of the CGRA?

[SQ3] Quantify the overhead coming from isolation cells, and control logic that may be required.

[SQ4] Can power gating on the CGRA be controlled dynamically e.g. switching functional units on and off during execution? Analyze and quantify. If that were beneficial, how should it be controlled?

In order to answer this questions, we went through a process summarized in the next section.

6.1 Summary

In chapter 1 the context of this research was presented, we introduced the concept of reconfigurable hardware and the CGRA. Highlighting Blocks, the CGRA developed by the TU/e. Similarly, the issue of power was raised and the use of power-gating was presented as one possible solution, as well as a possible contribution in what research concerns. This lead to the research questions mentioned above.

Chapter 2 dives deeper into the challenges and complexities that power gating has to offer, touching on topics like the sizing of the power switch, and the ways of implementing them. Having established the general trade-offs that granularity presents in terms of area and power, and the concept of a break-even point that weighs the energy savings and overhead, we can set constraints in terms of a circuits' activity on which power gating would be an admissible strategy at all.

Chapter 3 establishes the main metrics on which the research results would be evaluated given the technology used. The area overhead consists on the impact that the different techniques of power-switch insertion have on a power-gated block, as well as their ways of calculating them. The energy metrics establish the measuring of power for the energy savings as well as for the overhead of power gates, leading to the calculations of the break-even point for this particular context, this section also explains why performance was not one of the main topics of this research as its results are a derivative of a design decision that for effects of this research, remained constant: The voltage drop across the switches. A simple first order estimation on an inverter does however illustrate what the designer should expect to have in terms of performance, given a decision in terms of voltage drop. .

Chapter 4 walks through the testing environment starting by the designs used: namely CGRA designs as accelerators for 2 algorithms: namely a simple binarization function, using a grid of 3x3 functional units; and a parallel FFT implementation that uses a grid of 12x4 functional units. Then the test groups separated all the functional units and switchboxes of the benchmarks into three: switchboxes independently; Functional units independently, and the newly added extended functional units group which is the result of an optimization proposed later in the same chapter. As the benchmarks and test groups have been established, now the first order model is defined, tested using Cadence's Virtuoso tool, and used as the source of data for the transient behavior of the power gated blocks, namely the waking-up and the shutting-off transitions. The next steps consisted in establishing a workflow in Cadence's Genus tool for synthesis in which the power gates, their supporting power domains and isolation rules are established. Since synthesis does not take the power switches themselves into consideration, back-end design took place using Cadence's Innovus tool in order to do floor-planning, placing, routing and optimizing the design, to generate the relevant information to feed back into the first order model and establish a method to systematically determine the number and type of power switches used, as well as the transient behavior of the power-gated circuit for evaluation. Finally, our methodology introduces an path traversal algorithm based on a binary search, that aims to answer the main research question about finding the best possible granularity of power gating in a CGRA, doing so by selecting exclusive paths within a module, that lead exclusively to a set of outputs that we defined as a seed.

Chapter 5 begins by showing the impact that the path traversal algorithm had in extending the reach of a power switched functional unit into the interconnect network, showing that we could shut-off up to more than double the number of cells without extra overhead, should we decide to switch off a particular functional unit, later to highlight the use of isolation cells in each of the test groups, as isolation is most definitely the defining factor for determining the viability of power-gating. The general results in terms of area and power are then presented, aggregating the functional units of both into test groups mentioned above. We managed to show the area costs that power switching have at different granularities by establishing a relation between area overhead and circuit size. And we went through every step mentioned in the methodology to construct the break-even points for every functional unit and achieving several important points:

1. Presented the trade-offs in area an power that power gating bring on different granularities within the boundaries of a functional unit, meaning sizes ranging from less than 100 cells in case of the smallest switchboxes, up to more than 5000 in the case of the biggest switchboxes, passing through the complete set of functional units available to the CGRA, excluding memories.
2. A method was presented to optimize the insertion of power switches, bringing considerable power reductions increasing the viability of power switching dynamically.
3. The issue of control is discussed and made assumptions on, however no actual implementation was made.
4. A usable workflow was developed along this research, allowing the implementation of power switches in future projects.

The results chapter concludes with coming back to seeing the big-picture of this analysis, returning to the energy savings using fine-grained power gating dynamically, putting the results generated on a functional-unit basis to the test. Interestingly, even extended functional units do not qualify for power gating if their utilization is high, or if its idle periods are too fragmented. It was shown that even when a fraction of the functional units in the CGRA are power-switched, there are significant power savings even during activity, reducing the energy consumption of the FFT by 14%, and that of the binarization by 7.6%. Finally, the question of statically switching off the entire CGRA was brought to discussion, highlighting how well does power-gating behaves when the overhead is low and the size of the power island grows.

6.2 Closing remarks

In this thesis, we have presented and evaluated some of the mainstream methods of applying power gating under specific technology constraints, as well as for a specific type of architecture. And the reason of this research consisted in taking the first steps onto applying and consistently including power switching onto the coarse-grained re-configurable architecture developed by the Eindhoven's University of Technology. A method for evaluating the energy impact of power gating was re-applied to this particular case and put some light on the potential benefits of power gating in the coarsest granularity that allows for no trade-offs in terms of the resulting flexibility of the CGRA fabric and its interconnect.

In what concerns the research questions, we managed to establish a way to calculate tradeoff's and earnings of power gating at different granularities, doing so by implementing a physical design, and including all overhead coming from the switches and isolation cells. This answers the main question as well as three of the sub-questions presented.

The final sub-question was answered and tested upon, showing that there are significant power savings if power-gating were controlled dynamically while an algorithm is computing, 2 possibilities were discussed however not implemented, in terms of how would power-gating be controlled: either as an instruction within the CGRA, or via an additional functional unit that would orchestrate it. Finally we tested switching off a complete CGRA, showing the potential power savings it would bring and motivating its use-case. It was however not further tested.

6.3 Future work

Since this project's ambitions went through covering an extensive number of issues worth researching, there are many issues that for the sake of simplicity were assumed as either a design constraint or a requirements, however some of these topics could be targets of future project and/or research, some of the most important are:

1. Design in smaller technologies: This particular project was planned to be done in 40nm, however there are several projects that are being taken on using smaller and also different technologies (FSOI, finFET) in which leakage keeps dominating, therefore making power-gating a very attractive option. Thus, as the CGRA gets replicated into other technology nodes, the challenges seen in this particular research may come afloat just as much as new challenges may come to light.

2. The control problem: As it was noted along this report, the issue of control on the CGRA was touched and speculated upon, with the exception of simple externally controlled power switches. Regrettably, no dynamic control strategy was implemented, and as it was discussed, there could be several ways to make it take place, and all the options would have their trade-offs. For example: if the control of power gates were added as functional units' instruction, then the instruction decoder should remain on. Conversely if an external power controller is put in place, the instruction decoders could be switched off together with their respective functional units, but other challenges would take place

instead as for making sure that the power-switch signal propagates on time, that the controller remains in low utilization, etc.

3. The variables on transient behavior of the design: There is a consistent amount of research touching on the transient behavior of power switches and their switched blocks, and the discussion generally circles around stability, power switch signal distribution and minimization of rush-in current. These topics were established as reasonable constraints, however no optimization, nor great deal of research went into this field.

4. Dropping first-order models: probably one of the reasons why the transient behavior was taken more superficially comes from the fact that the transient behavior was modelled using very simple models. Next steps would certainly begin from reliable and simple models, but a great deal of complexity can be added to its analysis for further improvement.

5. implementing power-gating on a SoC: Even though this project was initially meant to take place in a SoC implementation of the CGRA, it was later rolled back onto simpler benchmarks. Combined versions of the CGRA could be beneficial for testing combinations of power switches acting dynamically (and internally to the CGRA), and more statically e.g. controlled by an external chip. This would allow us to explore the limits on how much power can be saved using this strategy.

Bibliography

[1] Mohab Anis, Shawki Areibi, Mohamed Mahmoud, and Mohamed Elmasry. Dynamic and Leakage Power Reduction in MTCMOS Circuits Using an Automated Efficient Gate Clustering. pages 480–485, 2002. 11

[2] Yannick Bonhomme, Patrick Girard, Loïs Guiller, Christian Landrault, Serge Pravossoudovitch, and Arnaud Virazel. A Gated Clock Scheme for Low Power Testing of Logci ICs or Logic Cores. *Journal of Electronic Testing*, 22(1):89–99, 2001. 72

[3] Stephen P. Boyd, Seung-Jean Kim, Dinesh D. Patil, and Mark A. Horowitz. Digital Circuit Optimization via Geometric Programming. *Operations Research*, 53(6):899–932, 2005. 68

[4] Assem A M Bsoul and Steven J E Wilton. An FPGA Architecture Supporting Dynamically Controlled Power Gating Relevant Work : Power Gating for FPGAs. *Computer Engineering*, 24(1):1–11, 2016. 9

[5] Assem A.M. Bsoul and Steven J.E. Wilton. An FPGA with power-gated switch blocks. *FPT 2012 - 2012 International Conference on Field-Programmable Technology*, (May):87–94, 2012. 6, 9

[6] Jun Cheng Chi, Hung Hsie Lee, Sung Han Tsai, and Mely Chen Chi. Algorithm for Power Optimization Under Timing Constraint. 15(6):637–648, 2007. 67

[7] Kiyoung Choi. Coarse-Grained Reconfigurable Array: Architecture and Application Mapping. *IPSJ Transactions on System LSI Design Methodology*, 4:31–46, 2011. 3, 4

[8] Aniryudh Reddy Durgam and Ken Choi. Optimized clock gating cell for low power design in nanoscale CMOS technology. *Proceedings of the 5th Asia Symposium on Quality Electronic Design, ASQED 2013*, pages 85–88, 2013. 72

[9] Horst Fiedler, Ralf Brederlow, Roland Thewes, Jorg Berthold, and Christian Pacha. Efficiency of Body Biasing in 90 nm CMOS for Low Power Digital Circuits. pages 175–178. 71

[10] T. Ghani, K. Mistry, P. Packan, S. Thompson, M. Stettler, S. Tyagi, and M. Bohr. Scaling challenges and device design requirements for high performance sub-50 nm gate length planar CMOS transistors. pages 174–175, 2002. 69

[11] Ankur Goel, R.K. Sharma, and AnilKumar Gupta. Area efficient diode and on transistor inter-changeable power gating scheme with trim options for SRAM design in nano-complementary metal oxide semiconductor technology. *IET Circuits, Devices & Systems*, 8(2):100–106, 2014. 9

[12] Shlomo Greenberg, Joseph Rabinowicz, and Erez Manor. Selective state retention power gating based on formal verification. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 62(3):807–815, 2015. 15

[13] Shlomo Greenberg, Joseph Rabinowicz, Ron Tsechanski, and Eugene Paperno. Selective state retention power gating based on gate-level analysis. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 61(4):1095–1104, 2014. 9, 15

[14] Cas Groot. A Dynamic Power Gating Method to Reduce Standby Energy Consumption. (October):1–11, 2007. 9, 28

[15] Scott Hanson, Mingoo Seok, Dennis Sylvester, and David Blaauw. Nanometer device scaling in subthreshold logic and SRAM. *IEEE Transactions on Electron Devices*, 55(1):175–185, 2008. 70

[16] Zhigang Hu, Alper Buyuktosunoglu, Viji Srinivasan, Victor Zyuban, Hans Jacobson, and Pradip Bose. Microarchitectural Techniques for Power Gating of Execution Units. pages 32–37. 9, 69

[17] Hailin Jiang, Malgorzata Marek-sadowska, and Sani R. Nassif. Benefits and Costs of Power-Gating Technique. 2005. 11, 13, 14, 65, 73

[18] M. Keating. *Low Power Methodology Manual for System-on-chip Design*. Springer, 2007. 15, 16

[19] Yoonjin Kim and Rabi N Mahapatra. *Design of Low-Power Coarse-Grained Reconfigurable Architectures*. 2011. 6

[20] Masaaki Kondo, Hiroaki Kobyashi, Ryuichi Sakamoto, Motoki Wada, Jun Tsukamoto, Mitaro Namiki, Weihan Wang, Hideharu Amano, Kensaku Matsunaga, Masaru Kudo, Kimiyoshi Usami, Toshiya Komoda, and Hiroshi Nakamura. Design and evaluation of fine-grained power-gating for embedded microprocessors. *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2014*, (25220002):1–6, 2014. 9, 13

[21] Jinson Koppanalil, Gus Yeung, Dermot O'Driscoll, Sean Householder, and Chris Hawkins. A 1.6 GHz dual-core ARM Cortex A9 implementation on a low power high-K metal gate 32nm process. *Proceedings of 2011 International Symposium on VLSI Design, Automation and Test, VLSI-DAT 2011*, pages 239–242, 2011. 9

[22] Zhiyu Liu and Volkan Kursun. Leakage Power Characteristics of Dynamic Circuits in Nanometer CMOS Technologies. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 53(8):692–696, 2006. 69

[23] Sven Lutkemeier and Ulrich Ruckert. A subthreshold to above-threshold level shifter comprising a Wilson current mirror. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 57(9):721–724, 2010. 67

[24] Maurice Meijer, Bo Liu, Rutger Van Veen, and Jose Pineda De Gyvez. Post-silicon tuning capabilities of 45nm low-power CMOS digital circuits. *2009 Symposium on VLSI Circuits*, (June):110–111, 2009. 71, 72

[25] Maurice Meijer and José Pineda De Gyvez. Body-bias-driven design strategy for area- and performance-efficient cmos circuits. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 20(1):42–51, 2012. 71

[26] Pradeep S. Nair, Santosh Koppa, and Eugene B. John. A comparative analysis of coarse-grain and fine-grain power gating for FPGA lookup tables. *Midwest Symposium on Circuits and Systems*, pages 507–510, 2009. 9

[27] Ship Navigation and Northern Sea Route. *Low Power Design Essentials.* 2009. 65, 66, 67, 68, 69, 70, 71

[28] Anja Niedermeier, Kjetil Svarstad, Frank Bouwens, Jos Hulzink, and Jos Huisken. The challenges of implementing fine-grained power gating. *Proceedings of the 20th symposium on Great lakes symposium on VLSI - GLSVLSI '10*, page 361, 2010. 13, 14, 22

[29] S. Pant, L. Nazhandali, S. Hanson, J. Olson, a. Reeves, M. Minuth, R. Helfand, T. Austin, D. Sylvester, and D. Blaauw. Energy-Efficient Subthreshold Processor Design. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 17(8):1127–1137, 2009. 67, 69

[30] Gracieli Posser, Guilherme Flach, Gustavo Wilke, and Ricardo Reis. Transistor sizing and gate sizing using geometric programming considering delay minimization. *2012 IEEE 10th International New Circuits and Systems Conference, NEWCAS 2012*, pages 85–88, 2012. 68

[31] Kaushik Roy, Saibal Mukhopadhyay, and Hamid Mahmoodi-Meimand. Leakage current mechanisms and leakage reduction techniques in deep-submicrometer CMOS circuits. *Proceedings of the IEEE*, 91(2):305–327, 2003. 65, 70

[32] Soumyaroop Roy, Nagarajan Ranganathan, and Srinivas Katkoori. A Framework for Power-Gating Functional Units in Embedded Microprocessors. 17(11):1640–1649, 2009. 13

[33] Jun Seomun and Youngsoo Shin. Self-retention of data in power-gated circuits. *2009 International SoC Design Conference, ISOCC 2009*, pages 212–215, 2009. 14, 15

[34] Ambika Prasad Shah, Nandakishor Yadav, Ankur Beohar, and Santosh Kumar Vishvakarma. On-chip adaptive body bias for reducing the impact of nbti on 6t SRAM cells. *IEEE Transactions on Semiconductor Manufacturing*, 31(2):242–249, 2018. 71

[35] Xing Su and Shinji Kimura. Optimization of area and power in multi-mode power gating scheme for static memory elements. *2016 IEEE Asia Pacific Conference on Circuits and Systems, APCCAS 2016*, 0:214–217, 2017. 9

[36] Siong Kiong Teng and Norhayati Soin. Low power clock gates optimization for clock tree distribution. *Proceedings of the 11th International Symposium on Quality Electronic Design, ISQED 2010*, pages 488–492, 2010. 72

[37] Thompson, Young, Greason, and Bohr. Dual Threshold Voltages And Substrate Bias: Keys To High Performance, Low Power, 0.1 /spl mu/m Logic Designs. pages 69–70, 1997. 71

[38] Liqiong Wei, Student Member, Zhanping Chen, Student Member, Kaushik Roy, and Senior Member. Design and Optimization of Dual-Threshold Circuits for low-voltage Low-power Applications. 7(1):16–24, 1999. 69, 71

[39] Neil H E Weste and DAvid Money Harris. *CMOS VLSI Design : A Circuit and Systems Perspective*, volume 53. 2011. 66

[40] Mark Wijtvliet, Luc Waeijen, and Henk Corporaal. Coarse grained reconfigurable architectures in the past 25 years: Overview and classification. *Proceedings - 2016 16th International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation, SAMOS 2016*, pages 235–244, 2017. 3, 4, 5

[41] Martin Wirnshofer. *Variation-Aware Adaptive Voltage Scaling for Digital CMOS Circuits (Springer Series in Advanced Microelectronics)*, volume 49. Springer, 2013. 71

[42] Bo Zhai, David Blaauw, Dennis Sylvester, and Krisztian Flautner. Theoretical and practical limits of dynamic voltage scaling. *Proceedings - Design Automation Conference*, pages 868–873, 2004. 69

Appendices

.1 Some poower-analysis elements

Energy consumption, in its principle consists in the amount of charge that passes through the circuit in a given period of time, or after a certain activity (or inactivity) period.

$$E = \int_0^\infty CV(\frac{dV_c}{dt})dt = \frac{1}{2}CV^2 \tag{1}$$

This simple model is true as long as we apply a step voltage with no swing. Now, if we put ourselves in the simples of CMOS circuits, the inverter, we can determine the energy dissipation in terms of transitions, where the power $P = E_{transition} * N_{transitions}$, where a transition can be a $T_{0->1}$ or $T_{1->0}$, just like a circuit clock would do. This means that per clock cycle we would always have a positive and a negative edge, meaning that power dissipation in a circuit would be:

$$P = CV^2 f \tag{2}$$

Where f represents the clock frequency applied to the circuit. Equation 2 can be generalized to 3 by adding an activity factor $0 \leq \alpha \leq 1$ that will determine the amount of switching that a circuit will be estimated to have. The accuracy of this addition largely depends on how precisely the activity estimations are.

$$P = \alpha CV^2 f \tag{3}$$

Since in reality we are dealing with clock transitions that are not ideal, the positive and negative edges on the transition generally cause a small SC current as for a small window of time, both the pull-up and the pull-down networks (PMOS and NMOS respectively) are conducting. The short circuit power can be modeled as:

$$C_{SC} = k(a\frac{\tau_{in}}{\tau_{out}} + b) \tag{4}$$

where a and b are technology related parameters, and k is a function of supply [27], threshold voltages and transistor sizes. Then using the same capacitor charge model from equation 3, we can express the short circuit capacitance as:

$$P_{SC} = C_{SC}V_{DD}^2 f \tag{5}$$

For an idling circuit, there is still going to be current leaking through, some of the effects modulating this phenomenon are the diffusion currents, the drain-induced barrier lowering (DIBL), the gate-induced drain leakage (GIDL), tunneling through the gate oxide and other static sources of leakage (bias, drain-substrates). The impact of leakage is actually one of the greater concerns in sub-micron CMOS technologies as it is becoming a constantly larger portion of the total power consumption of an integrated circuit [31, 17]. The static power can be modeled as:

$$P_{static} = (I_{DC} + I_{leak})V_{DD} \tag{6}$$

where I_{DC} is static current, I_{leak} is the leakage current, and V_{DD} is the supply voltage. It is quite important to remember what factors dominate leakage specifically, as will be a recurring topic in further sections:

$$I_{leak} = I_{ds0} e^{\frac{(V_{GS} - V_T + \delta_d V_{DS})}{n v_T}} (1 - e^{\frac{-V_{DS}}{v_T}}) \tag{7}$$

where I_{ds0} is source current at threshold voltage, V_{GS} is the gate voltage, V_T is the threshold voltage, $\delta_d V_{DS}$ is an approximation of the effect of drain-induced barrier lowering on the threshold voltage, v_T is the thermal voltage constant, and n is a process dependent term affected by the depletion region characteristics (normally within $1.3-1,7$ for CMOS processes)[39]. where I_{ds0} can be estimated in function 8, where β is transconductance, and $e^{1.8}$ is an empirical constant [39]:

$$I_{ds0} = \beta v_T^2 e^{1.8} \tag{8}$$

Other types of transistor leakage include gate leakage (GIDL), and other sources of I_{DC} such as tunneling through the depletion region, bias-induced leakage, junction leakage, band-to-band tunneling, etc [39]. Finally, our circuit power consumption can be calculated as function 9.

$$P = \alpha(C_L + C_{SC}) V_{DD}^2 f + (I_{DC} + I_{leak}) V_{DD} \tag{9}$$

where again α is the switching activity, C_L is the load capacitance, C_{SC} is the short circuit capacitance, f is the frequency, I_{DC} is the static current, and I_{leak} is the leakage current. This, in other words, boils down to:

$$P = \frac{energy}{operation} * rate + static\_power \tag{10}$$

.2 Related work in power optimization techniques

There is a myriad of different techniques that can be applied to at its different design stages, and the purpose of this section is introduce the most relevant ones for this particular research, this list is based on the classifications presented in [27], where optimizations are divided between active and static power optimizations. These optimizations however almost never only affect one or the other, but rather both.

.2.1 Active power

As its name indicates, this type of optimizations attempt to improve the active- or dynamic-power consumption of a particular circuit, these decisions will however still have an important impact on static power consumption. To put in perspective, the only difference between the power and the energy, is that the latter is the aggregate of the active power, throughout the duration of a determined routine.

$$P_{active} = \alpha C_L V_{swing} V_{DD} f \tag{11}$$

For effects of our analyses, we will assume that the voltage swing V_{swing} is equal to the supply voltage V_{DD}, which takes us straight back to equation 3.

.2.1.1 Multi-supply Voltage Domains

Multi-supply voltage domain is a quite effective technique used to reduce both dynamic and leakage power in nowadays CMOS chips [6]. This approach leverages the quadratic effect of supply voltage in the power consumption of a circuit (see eq. 3). This approach consists of partitioning the design into separate voltage domains, each operating at its own voltage level depending on its timing requirements. Here, islands where critical paths are located are assigned a high supply voltage to maximize its performance (VDDH), where non-critical domains are assigned a lower voltage, to exploit their slack as power savings (VDDL), this approach allows for saving power without compromising on system performance.

In the case that the circuit is an ultra low-power design, the possibility of having sections of it operating in sub-threshold and near sub-threshold regimes can become an important and valuable addition to the scheme [29]. The implementation of MSVD often requires the insertion of level shifter cells (LS's) on the boundaries between the logic at different supplies, the use and design of LS's will heavily depend on what supplies the circuit will have, specially if there is sub-threshold involved [23].

To illustrate the impact of MSVD, designs with 2 up to 3 voltage domains are compared to the single domain case, where adding an extra power domain drastically reduces power in a circuit, but the effect does quickly saturate due to the extra overhead and infrastructure that every extra power domain requires [27].

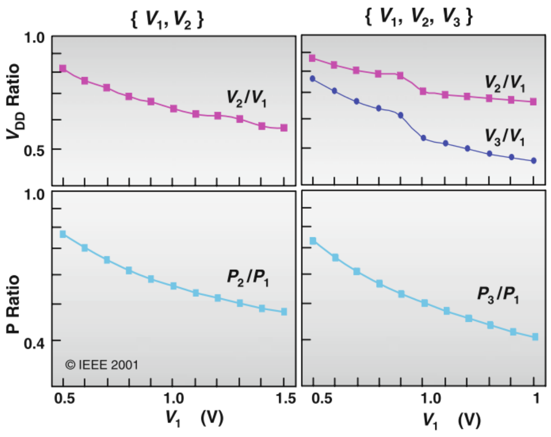


Figure 1: The addition voltage domains reduce the circuit's power, but the effect quickly saturates, yet an additional power domain just gives a fraction (5-10%) of what the savings the first one provided. Source: Rabaey, 2009, [27].

.2.1.2 Transistor sizing

If we look at our simple power model in eq. 3, the next group of optimizations has to do with the choice of load capacitances C_L along the different paths of a circuit, the principle is rather simple: more drive strength gives a speed-up of the circuit but also increases its switching power. Likewise, smaller $C'_L s$ reduces the power consumption but also degrades the circuit speed. This methodology can be used to both speed up critical paths and to collect energy savings without loosing performance.

A great deal of these optimizations are solved via iterative optimization runs by common IC design tools or constrained optimization problems [3, 30], making use of rich technology libraries, who have several gate design options for every type of combinatorial operation, for example larger and more complex gates would reduce overall capacitance at the price of speed, whereas other combinations may prioritize speed over power. The main takeaway of this type of optimization is that the designer counts with a set of options to map a particular function into logical gates, and is able to generate different ”profiles” depending on what the main design objectives are; whether it is focused on low energy, high performance, or a point in between.

.2.1.3 Activity and structural modifications

The reduction of the activity factor α comes along with a series of different optimizations and transformations regarding circuit topology. Some of them are factoring and restructuring:

1. Restructuring: is the optimization in which converging logical paths are given similar delays, allowing to tackle dynamic hazards. There are two main ways of restructuring a circuit: the first one corresponds to permuting cells from one of the paths to the other, whenever the function could remain unaffected, and when this is not possible, the next step is to insert delay buffers in paths that are hazardous due to path imbalance. This process is mostly automated and are considered a standard step in the back-end part of a design.
2. Factoring: corresponds to the transformation of certain logical expressions used in a gate or a group of gates, to an equivalent combination of gates that may be simpler, reduce capacitance or simply save energy by balancing the circuit. This optimizations are commonly visualized directly onto a logical expression, where purely logical transformations (de Morgan, factorization, distributivity, etc.) help in determining a new topology.

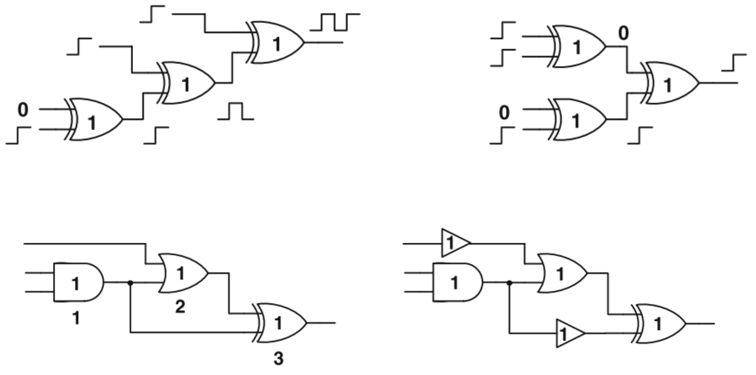


Figure 2: Illustration of circuit balancing through restructuring (up) and buffer insertion (down). Source: Low power design essentials, Rabaey, 2009 [27].

.2.2 Static-power optimizations

Some of the most energy efficient designs have a considerable portion of leakage energy, in some the that portion reaches up to 40% of the total power consumption of a design [16], this is mainly driven by the consistent lowering of the threshold voltages and gate insulator thickness in smaller technologies, impacting exponentially in the sub-threshold leakage current [38, 22].

Leakage has passed to be one of the main topics in circuit design, where it has become an increasingly difficult challenge to reduce it [27], it has also become a source of opportunities to designs that are pushing the limits of ultra-low V_{DD} [29, 42]. Some of the solutions proposed to maintain the efficiency of transistors in deep nanometer scales are changes in the design, such as different channel lengths; in the manufacturing process, such as reduced amount of doping or variations on the gate insulation [10]. In more extreme cases, a different flavor in the technology itself; such as FDSOI, finFET, etc. Since our focus will remain within the boundaries of 40nm-bulk technology, we will not discuss about other technologies in this section.

As was just mentioned, there are series of leakage-targeting optimizations, and although most of them have a direct impact on threshold voltage, which in turn has an exponential impact on leakage (see eqn. 13). They have a wide range of results in both active behavior (performance, power).

.2.2.1 Increasing channel length

The principle behind the changing of the channel length is rather simple; the longer L becomes, the higher the V_{TH} becomes, hence the leakage also drops (see eq. 13). This measure has been proposed ultra low-power deep nanometer technologies, where in order to accommodate the slower down-scaling of gate oxide thickness the gate length should scale down in the same fashion. A higher channel length does come at a cost: as the gate capacitance increases, so does the active power.

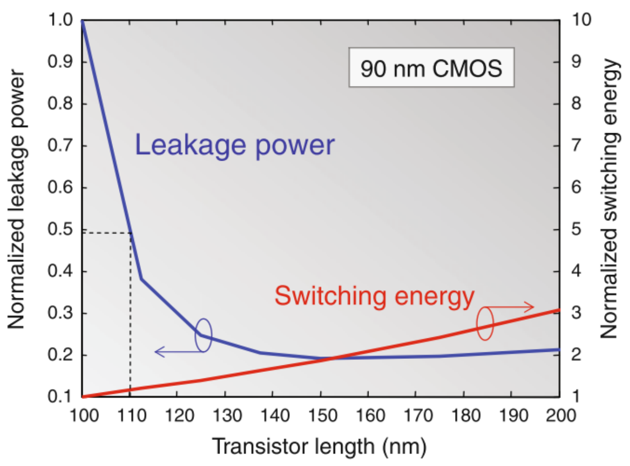


Figure 3: Illustration of the effect of channel length on threshold voltage and active energy, in the context of 90nm technology. Source: Rabaey, 2009 [27].

Additional measures to raise the threshold voltage are reducing the amount of doping applied to the substrate [31, 15], since substrate and halo doping affect almost linearly the threshold voltage of a transistor, hence having an important impact in sub-threshold leakage power.

.2.2.2 Circuit stacking

Circuit stacking is a very effective strategy against leakage, as it reduces also exponentially reduces the leakage of a circuit as the leakage of one transistor becomes the supply of the next one and so on, however it cannot always be applied, as it is limited to topology changes that maintain functional equivalenc.

.2.2.3 Multi-threshold libraries

Use of multi-threshold cells: modern libraries have nowadays 3 versions of their normal cells meant for different objectives on a design, they naturally are libraries are named depending on their threshold voltage. Thus, HVT, SVT, and LVT for high-, standard- and low-threshold cells. These libraries are also generally made using some of the technology related techniques here mentioned, for example: HVT cells may have thicker gate oxides, longer channels and reduced doping. In the next section, figure 4 shows the differences that multi-threshold cells present in terms of leakage and speed, while also including the dynamic optimization of body-biasing.

.2.3 Dynamic optimizations

Out of the optimizations that have briefly been discussed, this section will aim to present those strategies that combine some of these and apply them dynamically, or at runtime. These optimizations tend to be, more efficient, more resilient however they often add a considerable overhead in terms of control, distribution, retention of data, etc. The solutions discussed in this section are a) dynamic body biasing, b) clock gating and c) power gating. These three approaches are generally used together with the voltage islands generated by the MSVD approach. Different permutations of these methodologies are used to tackle the power challenge, as to selectively increase the threshold of a group of logic cells to reduce its leakage; limiting the clock activity to spread onto unused parts of a design; or completely cutting off an idling region of a chip.

.2.3.1 Body biasing

Namely reverse body biasing (RBB) has been a common measure to reduce sub-threshold current by means of raising the threshold voltage of the biased transistors [24, 9]. The effect of body-biasing on the threshold voltage is presented in [9], but for simplicity, we will stick to its linear approximation [27]:

$$V_{TH} = V_{TH0} - \gamma V_{BS} \tag{12}$$

Where γ is a fixed parameter. Thus, the updated calculation of sub-threshold leakage can be obtained by slightly modifying our expression in 13:

$$I_{leak} = I_{ds0} e^{\frac{(V_{GS} - V_T + \delta_d V_{DS} + \gamma_d V_{BS})}{n v_T}} (1 - e^{\frac{-V_{DS}}{v_T}}) \tag{13}$$

The use of FBB ad RBB is still a powerful tool to improve circuit performance by speeding a circuit in active period (FBB) and reducing its leakage during inactivity [9]. It has also been used to narrow down best- and worst-case delays in the synthesis process, yielding total area reductions [25].

The effect of body-biasing however is negatively affected by thinner gate-oxides, shorter gate lengths and smaller V_T's [9]. This means that the efficiency of BB would drop with the deep nanometer technologies. This has been reflected in a reduction of nearly half on the gains in active and leakage power for 40nm with respect to 90nm [24]. In some cases, RBB even presented increased leakage for reverse-biased HVT cells. see figure 4.

The use of body biasing is a standard in most technology libraries, and regardless of its efficiency reduction in deep nanometer technologies, it still can improve the underlying challenges of downscaling [37]. Thus, most commercial libraries have body contacts for both PMOS and NMOS have a well tap which is either left connected to VDD and VSS (PMOS and NMOS) for the non-biased case, and to a particular voltage domain in the case of MSVD [38], in some cases, even variable VBB schemes have been presented [34] estimating optimal thresholds to match a performance in, for example, variable temperature conditions [41].

.2.3.2 Clock gating

The principle of clock gating came up as the preferred way to keep the clock distribution away from parts of the chip that were idling for long periods of time. Having an important impact on active power of a chip. For example, (Bonhomme, 2006) demonstrated how by

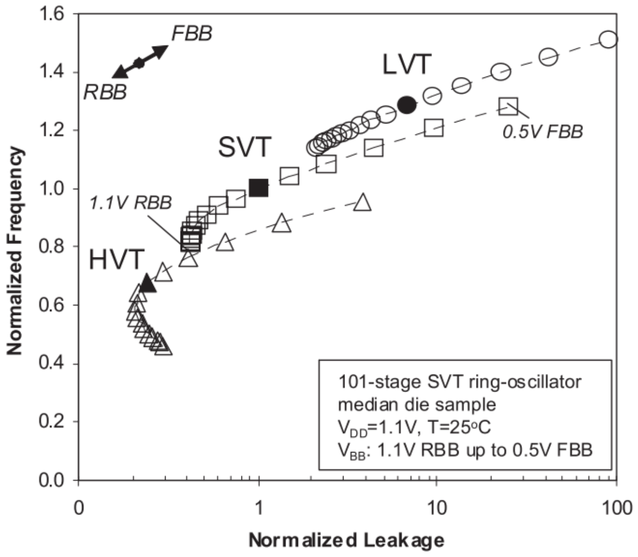


Figure 4: Frequency vs leakage chart for cells with BB=[0.5-1.1]V. The graph shows that for RBB specially in SVT and HVT cells, the attainable leakage reductions are becoming quite limited, where the most room for improvement in terms of leakage goes to LVT cells, whereas leakage reductions quickly saturate in SVT, and even increases in the case of HVT cells. Source: Meijer 2009 [24].

clock gating the design-for-testability" (DFT) circuitry [2], the power of a circuit could be slashed by up to 64%. Figure 5 shows some of the basic clock-gated circuit designs.

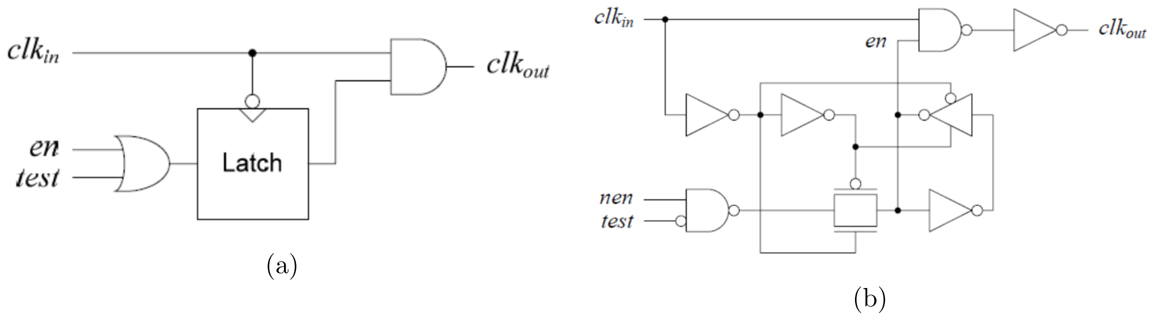


Figure 5: Designs of CGC vary by using variations with set-Reset latch, where a the single bit memory will indicate whether the clock clk_{out} will spread. a) basic structure of a CGC. b) structure of a conventional CGC. Source: Durgam 2013, [8]

The principle and the control used for CGC's is rather simple, but its complexity occurs when placing it throughout the clock-tree synthesis. This is a process that has been embedded in many design flows, and its implementation heavily relies on algorithmic optimizations to merge - permute and relocate clock gate cells throughout the clock three during synthesis [36].

In the CGRA-Blocks, there are CGC's in place to shut-off whole regions and functional units from the clock distribution. This has helps a great deal to reduce average active power, however it does not tackle the leakage issue.

.2.3.3 Power gating

The principle of power-gating (see fig 6) consists in placing a large transistor, or a series of smaller ones, between VDD and/or VSS of a logic block, providing an individual power domain that can be isolated from the always-on supply (and ground). This creates an intermediate power distribution network, namely VVDD for virtual-VDD, and VVSS for a virtual-VSS.

Nowadays, with the increasing use of on-chip accelerators, the activity in these areas becomes more predictable, and generally these big accelerators will idle for long periods. In the case of CGRA-Blocks, different sets of FU's will act as standalone accelerators making this method a very attractive one to reduce leakage power. For example it has been shown that the leakage of a circuit could be cut by 47% while incurring in an only-header scheme, on a 4 and 5% of total area and active power respectively [17].

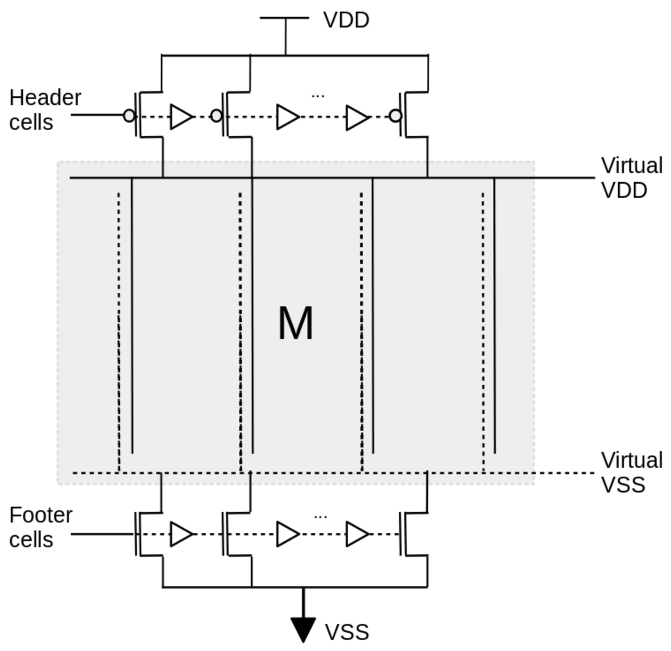


Figure 6: Simple schematic of a header and footer cells around a module M.

.3 Power Switch characterization

As opposed to the case with isolation cells, the power switches require a bit more explanation in order to be characterized and that is the reason for the existance of a section dedicated to

Cell	Inputs	DC Current (mA)		
		Min	Max	Avg
Header DI	Input 1	1.06	3.48	2.27
	Input 2	0.02	0.08	0.05
Footer DI	Input 1	-1.51	-2.46	-1.99
	Input 2	-0.09	-0.14	-0.12

Table 2: Summary table of best- and worst- case current supply capacity for Headers and footers, assuming an IR drop of 5% across the power switch in question.

them (again!) but now in the context of the technology utilized. Thus this section will briefly define the different aspects of the power switches that are relevant for further analyses and decisions as well being used as the base of models presented further. The data here gathered was taken from the available power-switch liberty libraries and documentation. This particular case uses the TSMC 40nm libraries on the worst-case condition, which is characterized as:

Corner	Slow-slow
Temperature	125^oC
Voltage	$0.99V$

The power switches used and characterized for this research corresponded to those described in table 6, and whose names were simplified for clarity.

Simplified name	Cell type Characteristics	Cell name
HDRSID1	Header, single-input, $P_{drive} = 1$	HDRSID1BWP12TM1PHVT
FTRSID1	Footer, single-input, $P_{drive} = 1$	FTRSID1BWP12TM1PHVT
HDRDID1	Header, double-input, $P_{drive} = 1$	HDRDID1BWP12TM1PHVT
FTRDID1	Footer, double-input, $P_{drive} = 1$	FTRDID1BWP12TM1PHVT

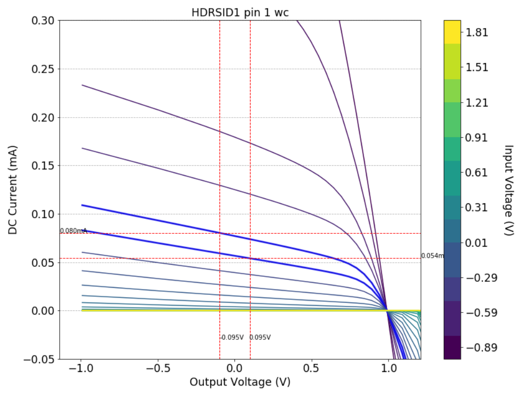
Table 1: Power switch cells tested, and their nomenclature used in the research

.3.1 Current capacity

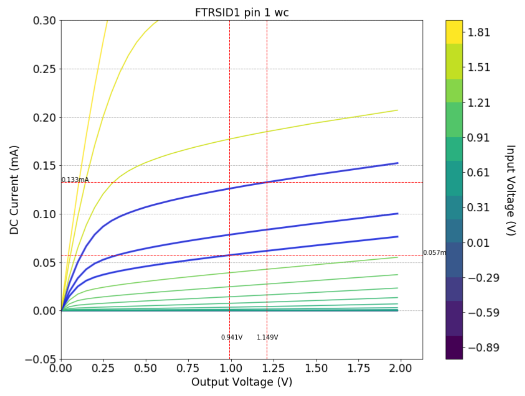
The current DC current characteristics of every power switch has been extracted from the TSMC liberty files (.lib), providing directly a relationship between the input voltage, output voltage and current.

Looking at the plots in figure 7, Every line here represents the voltage at the input, which we will assume to be between *0.99* and *1.21V* as the voltages considered acceptable between the best-, and worst-case scenarios. Thus, the lines that fall in the range were highlighted with a thicker blue line.

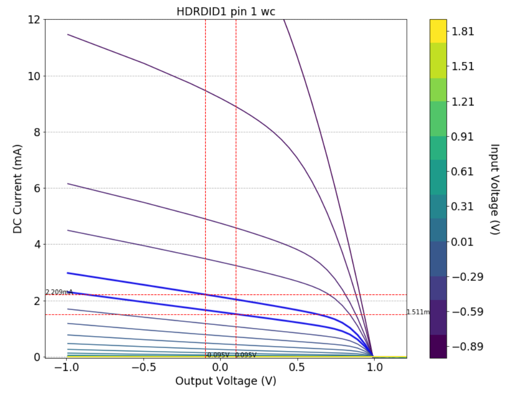
To bound the possible currents that the switches can provide, a 5% IR drop was determined, hence the boundaries on the X-axis are now located between the same BB-WC scenarios, with a 5% variation on them. The intersections of these boundaries make an area of operation where the min and max values were taken.



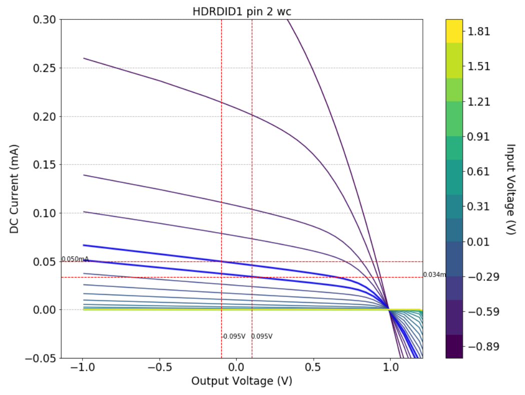
(a)



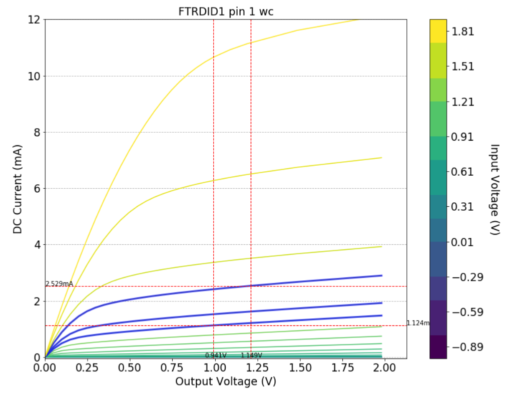
(b)



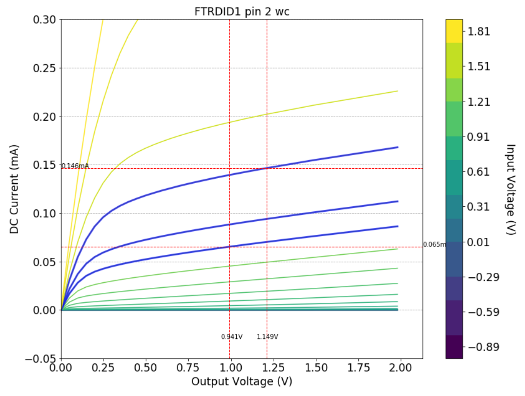
(c)



(d)



(e)



(f)

Figure 7: Current capacity plots for footer cells and header cells based on input voltage (color bar) and output voltage (x-axis). First row: single-input headers and footers. Second-third row: 2-input switch cells. The V_{in} in the range $[0.99, 1.21]$ were highlighted as blue lines, likewise the V_{out} in said range were highlighted vertically on the x-axis including a 5% voltage drop across the switch. The red horizontal lines show the DC values on those intersections, marking the min and max DC values found for that interval.

.3.2 Leakage

Leakage of the power switches as represented in the TSMC documentation, only takes into account the leakage of the paths between the input(s) and output(s) of the power switches, However they will be used to complete the leakage measurements from the models used in the methodology (4). The selected cells' leakage can be seen in table 3.

Cell name (simplified)	Leakage		
	min (nW)	average (nW)	max (nW)
HDRSID1	0.130	0.135	0.140
FTRSID1	0.137	0.140	0.143
FTRDID1	0.319	0.392	0.465
HDRDID1	0.302	0.369	0.436

Table 3: Leakage of the selected PG cells. Cell names were simplified with respect to the original: Where *HDR/FTR* indicates the type of cell, *SI/DI* indicates whether it is a single- or double-input cell, and *DX* indicates the drive power. Source: TSMC40nm Documentation

.3.3 Dimensions

The area impact on a design when inserting power switches is one of the most important costs that reflect monetarily in a design, therefore it is important to be able to estimate what that impact is going to be given different implementations of power gates. In this section, we will only address the dimensions of the different power switches used, as the area impact will be discussed later.

The area of each of the cells can be found in the cell libraries provided by TSMC, whereas the dimensions were obtained through Cadence Innovus, the dimensions of the selected power switches are presented in table 4.

Cell name (simplified)	area (um2)	height (um)	width (um)
FTRDID1	10.35	2*1.68	3.08
HDRDID1	29.17	2*1.68	8.68
HDRSID1	12.23	2*1.68	3.64
FTRSID1	2.82	1*1.68	1.68 height

Table 4: Table with power switch dimensions made with data from TSMC's cell libraries. It is worth noting that the height of the cells is measured in nr of rows it uses, which for 12-track corresponds to 1.68um

.3.4 Gate delay

The gate delay corresponds to the time that it takes for an edge to propagate from the input pin until the output pin of a PG cell, namely from the point the input edge raises/falls to 50%, until the respective output edge raise/fall to 50% . This delay will be used to later on evaluate the switch-on and switch-off delays on a power switched block. The propagation delay is modeled in the TSMC documentation as equation 14:

$$T_{worst} = T_{intrinsic} + F * C_{load} \tag{14}$$

where

T_{worst} = propagation delay at Worst case (125^oC) (ns)

$T_{intrinsic}$ = the intrinsic delay of each cell/path (ns)

F = load delay factor (ns/pF)

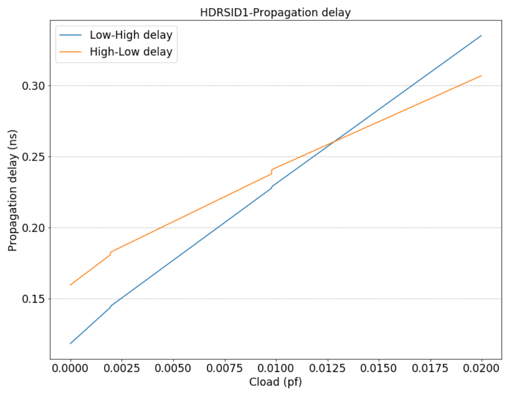
C_{load} = total output load capacitance (pF)

The delay models in the TSMC documentation present 3 groups of equations, depending C_{load}/C_{igate}, which is the ratio between load capacitance at the output (C_{load}), and the input gate capacitance (C_{igate}). The three groups combined make a single delay curve composed of 3 intervals which are detailed in table 5. The resulting plots for a 2-input header and footer cells are presented in figure 8.

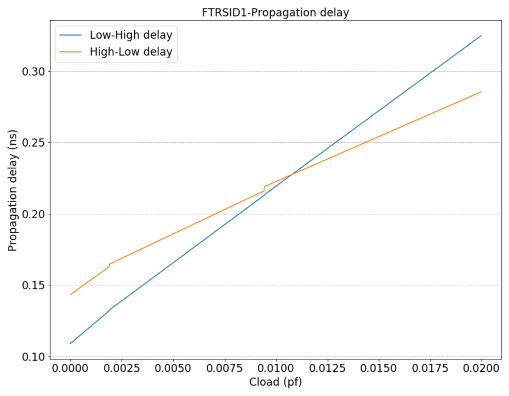
	C_{load}/C_{ipin}
Group 1	<= 2
Group 2	2 <x <= 10
Group 3	10 <x

Table 5: Equation intervals for modeling propagation delay

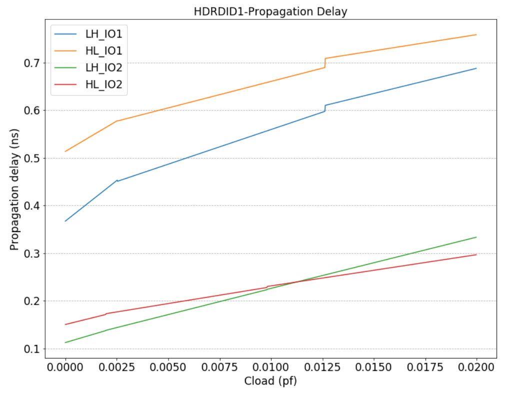
in order to present the analysis framework, we will arbitrarily decide for $2C_{input} < C_{load} <= 10C_{input}$ which would place us in the middle section of the propagation delay estimation models, or in "Group 2". Even though the latter implementations of the power gates use a simple daisy chain connecting all the power switches in increasing size order, this decision will provide us with a more pessimistic view which in general design terms is more desirable than a too optimistic one.



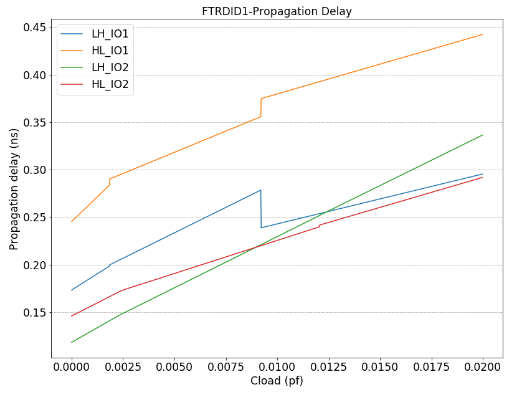
(a)



(b)



(c)

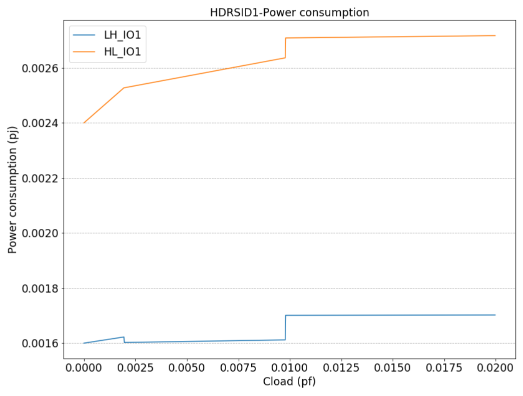


(d)

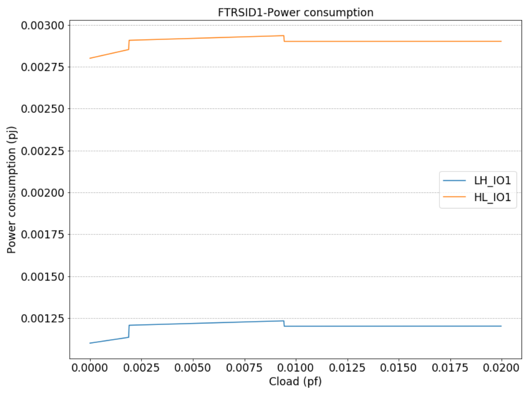
Figure 8: Propagation delay using the 3-stage models described in the TSMC documentation, presented for single- and double-input header (left) and footer cells (right) for paths IO1 and IO2. for both Low-High (LH) and High-Low (HL) transitions

.3.5 Switching power

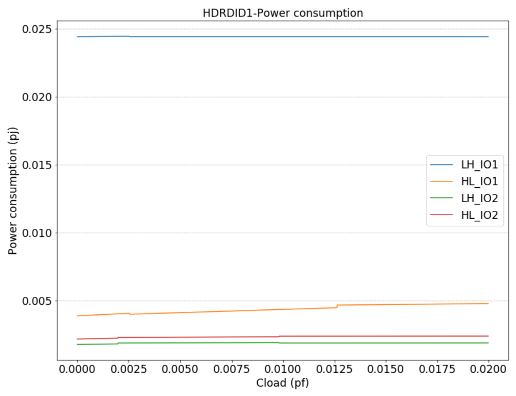
The power given in the TSMC documentation corresponds to the output pin power consumption on the cell when the respective pin changes state, the active power estimations are given on the same 3-group basis as the propagation delay was presented. Using a similar 3-group model from the TSMC documentation, the power consumption of a single- and double-input switches are represented in figure 9.



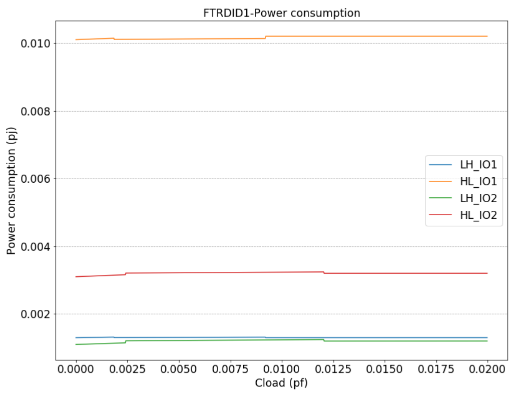
(a)



(b)



(c)



(d)

Figure 9: Active power of header (top) and footer (bottom) cells, where the power on the path IO1 is depicted on the left (a,d) and the power on the path IO2 is depicted on the center (b,e). Curiously the power consumption of both pins are almost identical, their models match in great measure but their difference is depicted on the left.

As seen in figure 9 the active power consumption of each cell is rather stable among its pins, this information will help us calculate the trade-offs on the insertion of power gates, but still does not answer the questions regarding the design of a gated module, these will come the further the framework is developed.

.4 Isolation cell characterization

The use of isolation cells was described in chapter 2, they tie all output paths of the switched module to either 0 or 1 in order to prevent unexpected behavior down those paths caused by having an otherwise dangling set of outputs. The isolation cells used from the TSMC libraries have 2 inputs: a data input *I*, and an enable *"ISO"*, and one output *"Z"*. A subset of the available isolation cells was selectes, each type vary in their drive power, as the outputs of a power switched block may have a relatively high fanout, and as with the case of the power switches, they have been characterized on their worst case corner.

Similar to the power switches, the names of the cells used in this report are simplified versions of the ones presented in the TSMC libraries, just for simplicity and readability. the nomenclature used is presented in table.

Simplified name	Cell type Characteristics	Cell name
ISOHID2	Iso to 1,$P_{drive} = 2$	ISOHID2BWP12T40M1PLVT
ISOHID4	Iso to 1,$P_{drive} = 4$	ISOHID4BWP12T40M1PLVT
ISOLOD2	Iso to 0, $P_{drive} = 1$	ISOLOD2BWP12T40M1PLVT
ISOLOD4	Iso to 0, $P_{drive} = 1$	ISOLOD4BWP12T40M1PLVT

Table 6: Isolation cells used, and their nomenclature used in the research

.4.1 Leakage

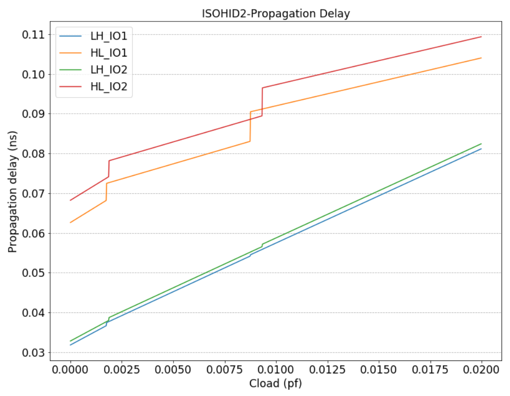
Leakage for isolation cells is also obtained from the TSMC documentation, and is described in table 7.

Cell name (simplified)	Leakage (nW)		
	Min	Avg	Max
ISOHID2	15.96771	21.06084	28.24807
ISOHID4	30.71693	34.47088	37.99739
ISOLOD2	14.61419	21.03096	28.75779
ISOLOD4	25.21779	32.88731	44.0453

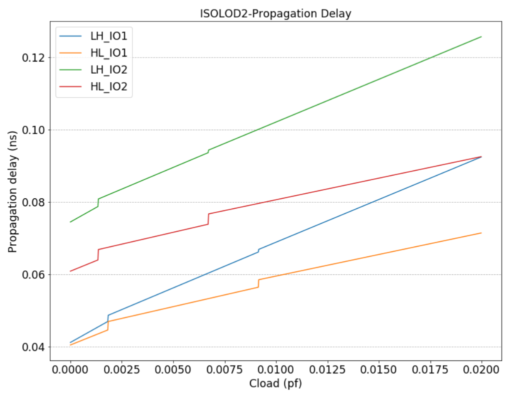
Table 7: cell leakage for isolation cells with Driving power 2 and 4 on their different paths *I-Z* (IO1) and *ISO-Z* (IO2)

.4.2 Propagation delay

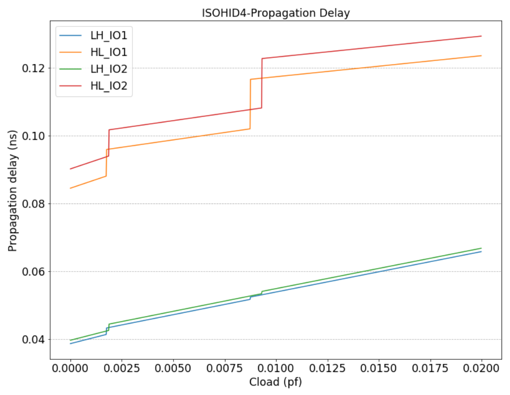
The propagation delay of the isolation cells, as well as with the power switches, is described in a set of 3 equations depending on the ratio between input and load capacitance as described in table 5, the behaviors of the selected cells are shown in figure 10.



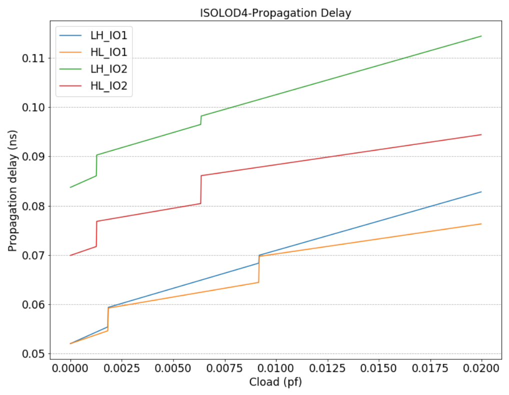
(a)



(b)



(c)

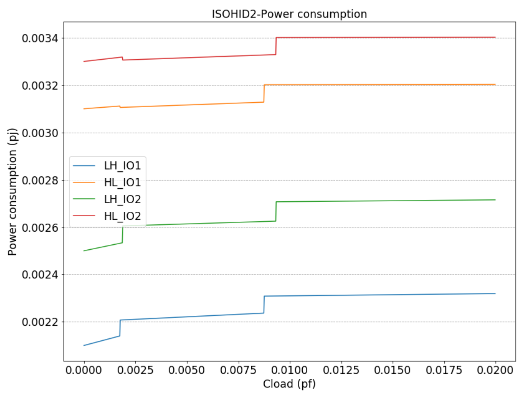


(d)

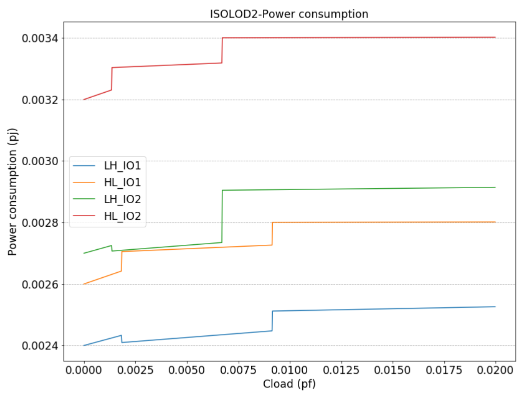
Figure 10: Propagation delay models for cells of driving power 2 (top) and 4(bottom)

.4.3 Active power

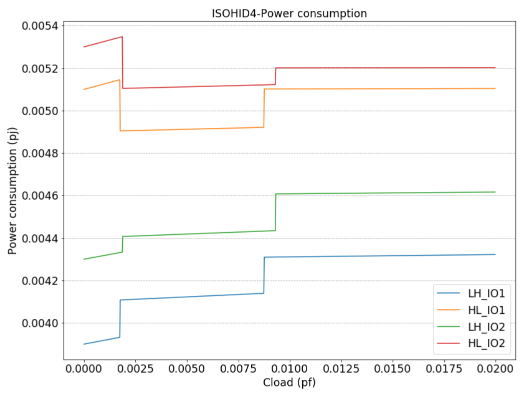
The estimation of active power is determined using the 3-equation models presented on the TSMC, and also described in table 5, the plots of the respective models on the selected cells are presented in figure 11



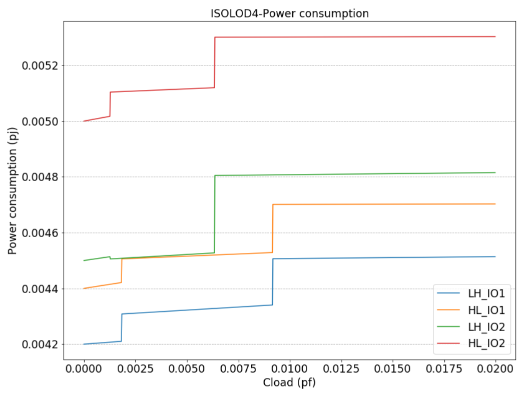
(a)



(b)



(c)



(d)

Figure 11: Models for power consumption on cells with drive power 2 (top) and 4 (bottom), for their paths *I-Z* (IO1) and *ISO-Z* (IO2)

.4.4 Dimensions

The dimensions of the selected isolation cells are shown in table 8 and was collected from TSMC's liberty files (.lib). All isolation cells have the same height of one row (1.68um) hence the area depends directly on the width of the each of the cells.

Cell name (simplified)	area (um2)	height (um)	width (um)
ISOHID2	1.6464	1.68	0.98
ISOHID4	2.352	1.68	1.4
ISOLOD2	2.352	1.68	1.4
ISOLOD4	3.0576	1.68	1.82

Table 8: Isolation cell dimensions for driving power 2 and 4

.5 The switched module characterization

Having defined the characteristics of the switches and isolation cells, as well as a model for their transient behavior, we have to characterize the module/set of cells that we want of switch off. It is in this phase where we can select the granularity of the scheme, and therefore attempt to answer our main research question.

.5.1 Area / density

The area of the module M is defined by a simple relation, namely:

$$area_M = \frac{\sum_i^{C(M)} area_i}{D_{target}} \qquad (\mu m^2) \tag{15}$$

where,

$C(M)$ is the set of cells in module M,

$area_i$ is the area of cell i,

D_{target} is the target cell density of module M.

We can assign any aspect ratio *a:b* given a particular target density $d\%$, by solving for x, and later assigning *a*x* and *b*x* to width and height.

$$x = \frac{area_M}{(a+b)} \qquad (\mu m) \tag{16}$$

with

$height_M = roundup_{1.68}(\sqrt{area_M})$

This roundup on the height dimension is done automatically by the design tools, adding a small variation to the area in the range $[0 - 1.68] * width_M$. This small variation can be mitigated in the design tools by specifying the dimensions of M in terms of width and height rather than in terms of density and aspect ratio.

.5.2 Capacitance

This information is necessary to estimate the the power-on and power-off times and consumption, as well as to calculate the number of power switches needed to achieve a particular IR drop. We define the capacitance of module M as the sum of the capacitances of all cells and nets inside M.

$$Cap_M = \sum_{i}^{C(M)} cap_i + \sum_{j}^{N(M)} capn_j \tag{17}$$

where,

$C(M)$ is the set of cells in module M,

cap_i is the capacitance of cell i,

$N(M)$ is the set of nets in module M,

$capn_j$ is the capacitance of net j.

```

set_cpf_version 2.0

##################################
# Define Library settings
#################################

define_library_set -name libs_wc -libraries $opcon_wc
define_library_set -name libs_tc -libraries $opcon_tc
define_library_set -name libs_bc -libraries $opcon_bc

##############################
##      PG and isolation cells
#################################
define_isolation_cell \
  -cells {ISOHI*} \
  -valid_location to \
  -enable ISO
define_isolation_cell \
  -cells {ISOLO*} \
  -valid_location to \
  -enable ISO

###################
## Headers & Footers
###################

define_power_switch_cell \
  -cells {HDRSI*} \
  -power_switchable TVDD \
  -power VDD \
  -stage_1_enable !NSLEEPIN \
  -stage_1_output !NSLEEPOUT \
  -type header

define_power_switch_cell \
  -cells {HDRDI*} \
  -power_switchable TVDD \
  -power VDD \
  -stage_1_enable !NSLEEPIN2 \
  -stage_1_output !NSLEEPOUT1 \
  -type header

define_power_switch_cell \
  -cells {FTRSI*} \
  -ground_switchable TVSS \
  -ground VSS \
  -stage_1_enable SLEEPIN \
  -stage_1_output SLEEPOUT \
  -type footer

define_power_switch_cell \
  -cells {FTRDI*} \
  -ground_switchable TVSS \
  -ground VSS \
  -stage_1_enable SLEEPIN2 \
  -stage_1_output SLEEPOUT1 \
  -type footer


######################
# Design part of the cpf
######################
set_design CGRA_Top

# Create global nets and pins
create_power_nets   -nets VDD   -voltage $tc_voltage
create_power_nets   -nets TVDD  -voltage $tc_voltage -
    external_shutoff_condition {iPG_signal}
create_ground_nets  -nets VSS
create_ground_nets  -nets TVSS  -external_shutoff_condition {iPG_signal}

# Create power domains
create_power_domain \
  -name {PD1} \
  -default
create_power_domain \
  -name {PD2} \
  -instances {POWER_SWITCHED_INSTANCE_LIST} \
  -base_domains {PD1} \
  -shutoff_condition {iPG_signal}

create_global_connection -domain {PD1} -net {VDD} -pins VDD
create_global_connection -domain {PD1} -net {VSS} -pins VSS
create_global_connection -domain {PD2} -net {TVDD} -pins TVDD
create_global_connection -domain {PD2} -net {TVSS} -pins TVSS

update_power_domain -name {PD1} -primary_power_net VDD -primary_ground_net VSS
update_power_domain -name {PD2} -primary_power_net TVDD -primary_ground_net
    TVSS

##################################
# Define Nominal Conditions & modes
##################################

create_nominal_condition -name ON_STATE -voltage $tc_voltage
create_nominal_condition -name OFF_STATE -voltage 0.0 -state off

update_nominal_condition -name ON_STATE -library_set {libs_tc}
update_nominal_condition -name OFF_STATE -library_set {libs_tc}

```

```
create_power_mode -name PM1 \
  -domain_conditions "PD1@ON_STATE PD2@ON_STATE" -default
create_power_mode -name PM2
  -domain_conditions "PD1@ON_STATE PD2@OFF_STATE"

#################################
### Isolation and PS rules
#################################

create_isolation_rule -name ir1 \
  -isolation_condition "iPG_signal" \
  -from PD2 -to PD1 \
  -isolation_output high \
  -isolation_target to
update_isolation_rules \
  -names ir1 \
  -location to \
  -prefix isorule1

create_power_switch_rule \
  -name psr1 \
  -domain PD2 \
  -external_power_net VDD
update_power_switch_rule \
  -name psr1 \
  -cells {HDRSI*} \
  -prefix PSHDR_

create_power_switch_rule \
  -name psr2 \
  -domain PD2 \
  -external_ground_net VSS
update_power_switch_rule \
  -name psr2 \
  -cells {FTRSI*} \
  -prefix PSFTR_

########################
# Define operation corners
########################

create_operating_corner -name wc_rcworst \
  -library_set libs_wc \
  -process 1 \
  -voltage $wc_voltage \
  -temperature 0

create_operating_corner -name bc_rcbest \
  -library_set libs_bc \
  -process 1 \
  -voltage $bc_voltage \
  -temperature 125

##########################
# Design Analysis view
##########################

```

```
create_analysis_view -name wc_AV_rcmax_hold_PM1 \
  -mode PM1 \
  -domain_corners "PD1@wc_rcworst PD2@wc_rcworst"
create_analysis_view -name AV_bc_setup_PM1 \
  -mode PM1 \
  -domain_corners "PD1@bc_rcbest PD2@bc_rcbest"

end_design
```

.6 Genus Synthesis flow - synthesis.tcl

Snap of the Genus synthesis flow used in the report, variables and other settings have been removed from the original code in order to improve readability, making this code non functional as-is.

```

####################################################
## Library setup
####################################################

set_db / .init_lib_search_path {. ./$LIB_DIR}
set_db / .script_search_path {. }
set_db / .init_hdl_search_path {. ../src sources}
set_design_mode -process 40

::legacy::set_attribute init_blackbox_for_undefined true /
::legacy::set_attribute write_vlog_empty_module_for_logic_abstract false /

source "./script/tech_settings_tsmc40.tcl"

set_db / .library "$opcon_wc $opcon_tc $opcon_bc"
set_db / .lef_library $tech_lef
set_db / .cap_table_file $rcw_captables

####################################################
## Load Design
####################################################
source script/read_hdl.tcl

read_mmmc ./script/mmmc.tcl
elaborate $DESIGN

check_design -unresolved ${DESIGN}

init_design
read_power_intent -module $DESIGN -cpf cpf/design.cpf

####################################################
## Constraints Setup
####################################################
define_clock -period $CLOCK_PERIOD -name CLK { iClk } -mode *

write_hdl -generic
report timing -lint -verbose

```

```
#######################################################
## Synthesizing to generic
#######################################################

commit_power_intent

syn_generic
report_summary
write_hdl -generic

########################################################
## Synthesizing to gates
########################################################

syn_map

report_summary
report_dp

#########################################################
## Optimize Netlist
#########################################################

syn_opt

time_info OPT
report_summary

#################################
### write the mapped design and sdc file
#################################

puts "Write Design and Netlist"

write_design -base_name
write_design -innovus
write_sdf -edges check_edge -setuphold split

puts "Write reports"
report area
report timing
report gates
report design_rules

report power -power_mode PM1
report power -power_mode PM2

report summary

puts "Final Runtime & Memory."

write_sdc -view wc_AV_rcmax_hold

puts "=============================="
puts "Synthesis Finished ........."
puts "=============================="
```

.7 Innovus - floorplan.tcl

snap of the Innovus floorplan flow used in the design, variables and other settings may have been removed from the original code in order to improve readability. note that the variable "MODE" was used as a dummy variable to determine whether the intended flow would use a ring or a column configuration.

```

################################
## Plan block placement      ##
################################

source script/floorplan_macros.tcl

######################
## Plower Planning ##
######################
if {$MODE =="ring"} {
  source ./script/floorplan_power_ringPG.tcl
} else {
  source ./script/floorplan_power_colPG.tcl
}

###################################
### insert power Switches
####################################

if {$MODE =="ring"} {
  source ./script/floorplan_add_ringPG.tcl
} else {
  source ./script/floorplan_add_colPG.tcl
}

###################################
### Save database
####################################

write_db DB/$_OUTPUTS_PATH/floorplan_PG.enc
```

.8 Innovus placement.tcl

```
##########################
## Setup Timing options ##
##########################

set_analysis_view -setup {wc_AV_rcmax_setup tc_AV_rcnom} -hold {
    bc_AV_rcmin_hold wc_AV_rcmax_hold}
set_interactive_constraint_modes [ all_constraint_modes -active ]

#####################
## Timing Derating ##
#####################

source ./script/timing_derate.tcl

######################
## Place the Design ##
######################

set_db plan_design_boundary_place true
set_db plan_design_effort high
set_db plan_design_fix_placed_macros false
plan_design

####################
## Pin Assignment ##
####################

assign_io_pins -move_fixed_pin -pins *
set_db plan_design_incremental true
set_db plan_design_effort high
plan_design

set_db finish_floorplan_active_objs {core macro}
set_db finish_floorplan_drc_region_objs {macro macro_halo hard_blockage min_gap
     core_spacing}
set_db finish_floorplan_add_blockage_direction xy
set_db finish_floorplan_override false

place_design

#########################
## PreCTS Optimization ##
#########################

opt_design -pre_cts -incremental

###################
## Report Timing ##
###################

time_design -pre_cts
write_db placement.enc
```

.9 Innovus cts.tcl

```
##########################
## Setup Timing options ##
##########################

set_analysis_view -setup {wc_AV_rcmax_setup} -hold {bc_AV_rcmin_hold}
set_db timing_analysis_type OCV
set_db timing_analysis_cppr both
set_db timing_analysis_check_type setup

#####################
## Timing Derating ##
#####################

source ./script/timing_derate.tcl

#########################
## ClockTree Synthesis ##
#########################

ccopt_design -check_cts_config
ccopt_design -report_dir ./$_REPORTS_PATH/cts_reports/

set_interactive_constraint_modes [ all_constraint_modes -active ]
set_propagated_clock [ all_clocks ]

##########################
# PostCTS optimization ##
##########################

set_analysis_view -setup {wc_AV_rcmax_setup} -hold {bc_AV_rcmin_hold}
opt_design -post_cts -report_dir $_REPORTS_PATH/timing_reports/cts -report_prefix ctsSetup

###################
## Report Timing ##
###################

time_design -post_cts -num_paths 10 -report_dir $_REPORTS_PATH/timing_reports/cts
time_design -post_cts -hold -num_paths 100 -report_dir $_REPORTS_PATH/timing_reports/cts

write_db DB/$_OUTPUTS_PATH/cts.enc
```

.10 Innovus route.tcl

```
##########################
## Setup Timing Options ##
##########################

set_analysis_view -setup {wc_AV_rcmax_setup tc_AV_rcnom} -hold {
    bc_AV_rcmin_hold wc_AV_rcmax_hold}
set_interactive_constraint_modes [ all_constraint_modes -active ]
set_propagated_clock [ all_clocks ]
set_db timing_analysis_type OCV
set_db timing_analysis_cppr both
set_db timing_analysis_check_type setup

#####################
## Timing Derating ##
#####################

source ./script/timing_derate.tcl

############################
## Route Clock Nets First ##
############################
set_route_attributes -nets ${CLK_PORT_NAME} -bottom_preferred_routing_layer 3 -
    top_preferred_routing_layer 4 -preferred_extra_space_tracks 1

set_db route_design_concurrent_minimize_via_count_effort "high"
set_db route_design_antenna_diode_insertion false
set_db route_design_reserve_space_for_multi_cut true
set_db route_design_selected_net_only true
set_db route_design_strict_honor_route_rule "false"
set_db route_design_with_si_driven true
set_db route_design_with_timing_driven true

route_global_detail

set_db route_design_selected_net_only false
route_global_detail

#######################
## Route Signal Nets ##
#######################

set_db route_design_detail_post_route_swap_via multi_cut
set_db route_design_detail_use_multi_cut_via_effort  high

fix_via -min_cut
route_design -via_opt

###################
## Report timing ##
###################
time_design -post_route
time_design -post_route -hold

write_db route.enc
```

.11 Innovus post_route_opt.tcl

```
##########################
## Setup Timing Options ##
##########################

set_analysis_view -setup {wc_AV_rcmax_setup tc_AV_rcnom} -hold {
    bc_AV_rcmin_hold wc_AV_rcmax_hold}
set_interactive_constraint_modes [ all_constraint_modes -active ]
set_propagated_clock [ all_clocks ]
set_db timing_analysis_type OCV
set_db timing_analysis_cppr both
set_db timing_analysis_check_type setup

create_basic_path_groups
get_path_groups *

set_db delaycal_equivalent_waveform_model propagation
set_db delaycal_combine_mmmc none

set_db opt_post_route_fix_glitch true
set_db opt_post_route_fix_clock_drv true

####################
## Timing Derating ##
####################

source ./script/timing_derate.tcl

#############################
## Remove Std Filler Cells ##
#############################

delete_filler -prefix FILL

#############################
## Post Route Optimization ##
#############################

opt_design -post_route
opt_design -post_route -setup -incr

set_dont_use   [get_lib_cells *DEL*] false
set_dont_touch [get_lib_cells *DEL*] false
opt_design -post_route -hold
opt_design -post_route -hold -incr

#############################
## Insert Std Filler Cells ##
#############################

add_fillers -base_cells $filler_cells -prefix FILL -check_drc true -
    check_via_enclosure true -check_min_hole true -power_domain {PD1}
add_fillers -base_cells $filler_cells -prefix FILL -check_drc true -
    check_via_enclosure true -check_min_hole true -power_domain {PD2}

opt_design -post_route
opt_design -post_route -hold

############
# Report
############

if {$MODE == "flat" } {
source ./script/report_flat.tcl
} else {
source ./script/report_pg.tcl
}

#####################
# Write outputs
#####################

write_netlist -exclude_leaf_cells $_OUTPUTS_PATH/optRoute.v
write_netlist -exclude_leaf_cells $_OUTPUTS_PATH/optRoute.phys.v -phys

write_sdf -view tc_AV_rcnom     -no_escape -edges check_edge -delimiter . $
    _OUTPUTS_PATH/optRoute.sdf_typ
write_sdf -view wc_AV_rcmax_setup   -no_escape -edges check_edge -delimiter . $
    _OUTPUTS_PATH/optRoute.sdf_rcw
write_sdf -view bc_AV_rcmin_hold    -no_escape -edges check_edge -delimiter . $
    _OUTPUTS_PATH/optRoute.sdf_rcb

extract_rc
write_parasitics -spef_file $_OUTPUTS_PATH/${TOP_DES_NAME}.spef_typ
write_parasitics -spef_file $_OUTPUTS_PATH/${TOP_DES_NAME}.spef_rcw
write_parasitics -spef_file $_OUTPUTS_PATH/${TOP_DES_NAME}.spef_rcb

write_db ./DB/$_OUTPUTS_PATH/post_route_opt.enc
```

.12 Innovus report.tcl

```

set_analysis_view -setup {wc_AV_rcmax_setup tc_AV_rcnom} -hold {bc_AV_rcmin_hold wc_AV_rcmax_hold}
set_interactive_constraint_modes [ all_constraint_modes -active ]
set_propagated_clock [ all_clocks ]
set_db timing_analysis_type OCV
set_db timing_analysis_cppr both
set_db timing_analysis_check_type setup

create_basic_path_groups
get_path_groups *

set_db delaycal_equivalent_waveform_model propagation
set_db delaycal_combine_mmmc none
set_db opt_post_route_fix_glitch true
set_db opt_post_route_fix_clock_drv true

####################
## Timing Derating ##
####################

source ./script/timing_derate.tcl
time_design -post_route -num_paths 10 -report_dir $REPORTS_PATH/timing_reports/post_route -report_prefix op_route_setup
time_design -post_route -hold -num_paths 10 -report_dir $REPORTS_PATH/timing_reports/post_route -report_prefix op_route_hold

########################
# Report power
########################
reset_power_activity
report_power -view tc_AV_rcnom        -out_file $REPORTS_PATH/power_reports/power_rpt_typ.txt
report_power -view wc_AV_rcmax_setup -out_file $REPORTS_PATH/power_reports/power_rpt_wc.txt
report_power -view bc_AV_rcmin_hold  -out_file $REPORTS_PATH/power_reports/power_rpt_bc.txt


#report the capacitance of all nets, (python to parse on module M)
report_power -view tc_AV_rcnom -cap  -out_file $REPORTS_PATH/power_reports/cap_rpt.txt

#power of isolation cells
report_power -insts  *isorule* -view wc_AV_rcmax_setup -out_file $REPORTS_PATH/power_reports/iso_power_rpt.txt

#power of the module M
report_power -insts  *alu_inst* -view wc_AV_rcmax_setup -out_file $REPORTS_PATH/power_reports/alu_inst_power.txt

# power of al switches (however it is already included in the module M)
report_power -insts  *PSFTR* -view wc_AV_rcmax_setup -out_file $REPORTS_PATH/power_reports/pg_power_rpt.txt

```

```
#cell area of the module M
report_area -hinst CGRA_Core_inst/CGRA_Compute_Wrapper_inst/CGRA_Compute_inst/
    alu_inst -out_file $REPORTS_PATH/pg_area.txt

#floorplan area of PD2 = module M
get_db [get_db groups *PD2] .area

#get the total area of all isolation cells
set tot 0.0
set isonr 0
foreach {cells} [get_db [get_db insts *isorule*] .area ] {
set tot [expr $tot + $cells]
set isonr [expr $isonr +1]
}
puts "total area of $isonr ISO cells is $tot \n"

#detailed energy per switch
report_inst_power [get_db insts .name *PSFTR*] -out_file $REPORTS_PATH/
    pg_inst_power.txt
report_inst_power *isorule* -out_file $REPORTS_PATH/iso_inst_power.txt
```

.13 Some design querying functions used (tcl)

```

## declare functions
####################
proc count_set { set } {
set a 0
foreach output $set {
incr a
}
return $a
}
proc get_outcells { cell hinst_boundary_name} {
#global hinst_boundary_name
return [get_db [get_db [get_db $cell .pins -if {.direction == out}] .net.loads.
    inst -if {.parent.name == $hinst_boundary_name } ] -if  {.name != *
    rConfig_reg*}  ]
}

proc get_incells { cell hinst_boundary_name} {
#global hinst_boundary_name
set aux  [get_db [get_db $cell .pins -if {.direction == in} ] .net.drivers -if
    {.name != *iClk*} ]
set aux2 [get_db $aux .inst -if {.parent.name == $hinst_boundary_name }]
set aux3 [get_db $aux2 -if {.name != *rConfig_reg*} ]
return $aux3
}

proc back_propagate { list hinst_boundary_name} {
foreach item $list {
set nr_outcells [count_set [get_outcells $item $hinst_boundary_name]]
#puts "$item has --> $nr_outcells"
set matches 0
foreach ocell [get_outcells $item $hinst_boundary_name] {
foreach othercell $list {
if {$ocell == $othercell} {
#puts "match!!"
incr matches
}
}
}
set broke 0
if {$matches >= $nr_outcells} {
#puts "matches $matches vs $nr_outcells --> $item +++++++"
} else {
puts "matches $matches vs $nr_outcells --> $item --------"
set rem [lsearch $list $item]
set list [lreplace $list $rem $rem]
set broke 1
break
}
}
if { $broke == 0 } {
return $list
} else {
back_propagate $list $hinst_boundary_name
}
```

```
}

proc append_cells {list input} {
foreach cell $input {
lappend list $cell
}
return $list
}

proc print {list } {
foreach item $list {
puts $item
}
}


proc getports {target_hinst} {
set targets [get_db hinsts $target_hinst]
set portlist {}
foreach target $targets {
set ports [get_db $target .hports -if {.direction == out}]

foreach port $ports {
#puts $port
lappend portlist $port
}
}
return $portlist
}

proc get_area {list} {
set full_list {}
foreach t $list {
set totarea [get_db $t .area]
lappend full_list "$t area: $totarea "
return $full_list
}
}

proc get_ps_cells { hinst_boundary_name seed_name } {

set nrcells [count_set [get_db [get_db hinsts $hinst_boundary_name] .insts ]]
if {$seed_name == "all"} {
set seeds [get_db [getports $hinst_boundary_name] .hnet]
} else {
set seeds [get_db [get_db hinsts $hinst_boundary_name] .hnets $seed_name ]
}

if {[count_set $seeds] == 0} {
puts "could not find seeds by name $seed_name"
return 0}
set 1_layer_cells [get_db $seeds .net.drivers.inst -if {.parent.name == $
    hinst_boundary_name }]

set last_list $1_layer_cells
set done 0
set counter 0
set superset {}
set superset [append_cells $superset $1_layer_cells]

for {set i 0} {$i < 50} {incr i} {
set current_list [get_incells $last_list $hinst_boundary_name]
set current_list [lsort -unique $current_list]
set superset [append_cells $superset $current_list]
puts [count_set $current_list]
set last_list $current_list
}

set superset [lsort -unique $superset]
set count2 [count_set $superset]
puts "The count: $nrcells -> $count2 "

set clean_list {}
set clean_list [back_propagate $superset $hinst_boundary_name]
set count3 [count_set $clean_list]
puts "Final count: $nrcells -> $count2 -> $count3 after back propagation "

return $clean_list
}


#for set the search parameters
set seed_name "oRIGHT oLEFT"
set hinst_boundary_name *CGRA_Compute_inst/SWB*
set targets [get_db [get_db hinsts $hinst_boundary_name -if {.name != *buffer_
    *}] -if {.name != *PREFIX*}]
print $targets

#do the search

set full_list {}
foreach seed $seed_name {
foreach t $targets {
set totarea [get_db $t .area]
set nrcells [count_set [get_db $t .insts ]]
# puts $nrcells
set pow_dyn [get_db $t .power_dynamic]
set pow_leak  [get_db $t .power_leakage]
set pow_total [get_db $t .power_total]

#run the path traversal
set test [get_ps_cells [get_db $t .name] $seed]
set nrpscells 0
set nrpscells [count_set $test]

set psarea 0
set pspow_dyn 0
set pspow_leak 0
set pspow_total 0

foreach cell $test {
if {$cell != 0} {
set psarea [expr $psarea + [get_db $cell .area]]
set pspow_dyn [get_db $t .power_dynamic]
set pspow_leak  [get_db $t .power_leakage]
set pspow_total [get_db $t .power_total]
}
}
lappend full_list "$t: $nrcells -> $nrpscells // area: $totarea -> $psarea // pow_total: $pow_total -> $pspow_total // pow_dyn: $pow_dyn -> $pspow_dyn // pow_leak: $pow_leak -> $pspow_leak"
#lappend full_list "$t: $nrcells -> $nrcells // area: $totarea -> $totarea // pow_total: $pow_total -> $pow_total // pow_dyn: $pow_dyn -> $pow_dyn // pow_leak: $pow_leak -> $pow_leak"
}
}

print $full_list

foreach t $targets {
set ports [get_db $t .hports -if {.direction == out}]
puts $ports

puts "$t ports -> [count_set $ports]"
}
```