Voltage Threshold Optimization: Balancing Leakage and Performance in Integrated Circuits

Filip Raguz filip.raguz@hotmail.com +46 737155521
Joel Qvint qvint.joel@gmail.com +46 763194223

Department of Electrical and Information Technology
Lund University

Supervisor:
Jonas Skeppstedt (LTH)
jonas.skeppstedt@cs.lth.se
Tom Johansson (Xenergic AB)
tom.johansson@xenergic.com

Examiner:
Pietro Andreani
pietro.andreani@eit.lth.se

June 17, 2024

LUNDS UNIVERSITET

xenergic

Printed in Sweden
Tryckeriet i E-huset, Lund

Abstract

This body of work covers the utility of the development of a tool for automatically configuring the transistors in an existing netlist, according to a set of specifications. By swapping out transistors for transistors with other Voltage Thresholds it can achieve target speeds with acceptable accuracy and reduce the static power consumption of a circuit. To make this process feasible, the tool uses approximations to predict what configurations would meet its constraints.

In testing, it was found that the tool could affect the leakage by up to a factor of 246 and frequency by a factor of 2.14. The tool could predict leakage with high accuracy, while the accuracy of frequency estimations was dependent on the architecture in question. As the tool has been iterated on, it has become more reliable in that regard as well.

Using results from the tool it is possible to tell how big an impact Voltage Threshold has on the performance of the circuit varies wildly with size and architecture. Depending on the technology in question, a lot of power can be saved with relatively little effort.

Popular Science Summary

Integrated circuits are paramount in a vast range of everyday devices, in virtually anything that has a power cord or runs on batteries nowadays. They control various components on the devices to enable sophisticated functionality, after specific instructions. For basic functionality, the circuit can be painstakingly designed in a smart way to achieve it.

But if those circuits are going to direct tasks that are more complicated than the very simplest, they need an efficient and modular way of storing the instructions. The industry standard is using an integrated memory.

A customer will want the most efficient memory of sufficient size for their purpose, to save on manufacturing costs, and to make a more energy-efficient product. When using an off-the-shelf memory, this means they have to find a memory that meets their minimum requirements for speed. In all likelihood, the memory will be faster than required, but that is of no benefit to the device. That extra speed will still come at the cost of power consumption.

For this reason, creating designs for integrated memories after precise specifications can be valuable to customers since they can now have access to memories with minimal wasted capacity. However, creating designs is a lengthy process. In this project, an attempt was made to create a means of slightly altering existing designs to fit other constraints, possibly removing the need for new designs if the specs are sufficiently close to existing designs.

To accomplish this, a tool was programmed. The tool automatically adjusts one of the primary components of the memory, the transistors. Swapping transistors in a design is a lesser effort, as different transistor types come in the same size. The challenge is in how many transistors there are, but for an automatic tool that is of little consequence.

Acknowledgments

We would like to say a special thanks to our supervisor Tom Johansson, for making himself available to answer the countless questions we had each day, and always having great solutions for problems encountered throughout the project.

We would also like to thank our academic supervisor Dr. Jonas Skeppstedt for his help in understanding general optimization algorithms and giving ideas on how to approach this difficult problem.

Last but not least, we would like to thank all the fantastic engineers at Xenergic, Reda Boumchedda, Babak Mohammadi, Reza Meraji to name a few, for helping us understand general circuitry and how the circuits developed at Xenergic work.

Table of Contents

List of Figures

List of Tables

List of Acronyms

CMOS Complimentary Metal Oxide Semiconductor
DRC Design Rule Checking
MOSFET Metal-Oxide-Semiconductor Field-Effect Transistor
NMOS N-type Metal-Oxide-Semiconductor
PMOS P-type Metal-Oxide-Semiconductor
$\boldsymbol{V_T}$ Voltage Threshold
ELVT Extreme Low Voltage Threshold
LVT Low Voltage Threshold
SVT Standard Voltage Threshold
HVT High Voltage Threshold
EHVT Extreme High Voltage Threshold
NDA Non-Disclosure Agreement
SoC System-on-Chip
IC Integrated Circuit
Kb Kilobit($2^{10} = 1024$ bits)

Chapter 1
Introduction

Circuits today can contain billions of transistors [1], with several parameters that contribute to power consumption and speed. There are also different types of transistors with varying characteristics that make certain transistors better suited to certain use cases. Given the quantity of components and many different ways of adjusting a circuit, it is unfeasible to manually conduct optimizations. One quality that is of particular interest is static power consumption, also known as leakage.

After a circuit is designed, further optimizations can be achieved by swapping out transistors for other variants. This is possible due to the transistors having the same footprint, rather varying in characteristics such as oxide thickness and amount of doping, among others [2]. By selecting transistor types with different V_T values the circuit can be better tailored to achieve a target frequency. This approach to optimizing circuits is further corroborated as useful by prior work such as that by Daboul et al. [3], and Rahman and Sechen[4], at least in regards to leakage.

Discussed in the aforementioned work are implementations of optimization algorithms that aim to reduce static power leakage while maintaining delay constraints. It was found in these articles that optimizing circuits based on transistor V_T could give a static power reduction as high as 34% [3] and when including changes to transistor sizing as well it could give an average of 37% static power reduction [4].

Reducing the power consumption of an integrated memory has the obvious benefit of directly reducing the overall power consumption of a System-on-chip (SoC), considering that embedded memories account for 50% of the area and 50-70% of the overall power dissipation [5].

1.1 Scope of the thesis

This thesis project aims to design a tool that can automatically assign V_T to the transistors in the circuit in such a manner that:

- It is possible to identify what frequencies a circuit is able to achieve using only changes to V_T.
- Find at least a near-optimal V_T configuration for a given frequency.

The practical reasons for designing such a tool are to save a lot of manual time optimizing these circuits and to easily be able to determine whether a given circuit is able to operate at the desired speed. This allows tailoring of the circuits to their application. It could also be used for characterization of the circuit, to identify weak spots and find which parts of the circuit can be specifically optimized to yield higher performance. Additionally, it can serve as a powerful analytical tool for transistor behaviors in circuits.

Eventually, the tool would be used to further analyze relationships between transistor types and physical properties of circuits such as:

- Areas where V_T has a high impact on performance.
- Total range of transistor impact on current leakage.
- Transistor behaviors between different node sizes and architectures.

This is to hopefully give an overview of the behaviors of circuits in different circumstances, and further characterize them.

Chapter 2
Background

In this chapter, a brief overview of topics that are relevant to the project will be given.

2.1 Basics of Integrated Circuit Components

Integrated circuits (ICs) are small chips that contain electrical components essential for modern electronics. These components enable a variety of functions in devices due to their compact size and high level of customizability. This section highlights some of the fundamental components and concepts critical to IC functionality.

2.1.1 The MOSFET

The MOSFET is the most basic building block in today's digital electronics. It is a switch that can be put in an ON or OFF state, representing the binaries '1' and '0', respectively. There are two types of these MOSFETs, N-type and P-type. These are complimentary to each other, meaning that the N-type MOSFET (NMOS) conducts current when a positive bias is applied to the gate, and the P-type (PMOS) conducts current when a negative bias is applied. These two types allow for the creation of CMOS logic units, the most common technology used in the chip design industry [6].

Looking at Figure 2.1 it is possible to observe the cross-section of an N-type transistor, and the terminals connected to the device. Applying a high enough positive bias to the gate terminal will create a channel beneath the oxide, conducting current between the drain and the source terminals.

2.1.2 The Inverter

By using the MOSFET devices is possible to create the building blocks for digital logic. In Figure 2.2 the most basic of these building blocks, an inverter, can be observed. Giving an input of logic '1' to the inverter will cause the PMOS to be shut off, and the NMOS to be switched on, causing a connection between the output and ground, thus the output will be logic '0'. Similarly, for a logic '0' input

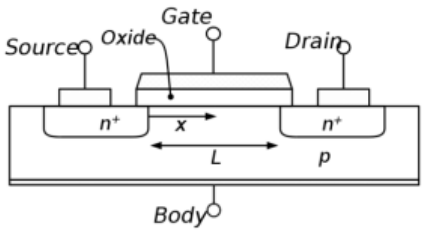

Figure 2.1: Cross-section of an N-type MOSFET. Illustration by [7]

the NMOS will be switched off, and the PMOS switched on, which will pull the output towards V_{dd} and subsequently give an output of logic '1'.

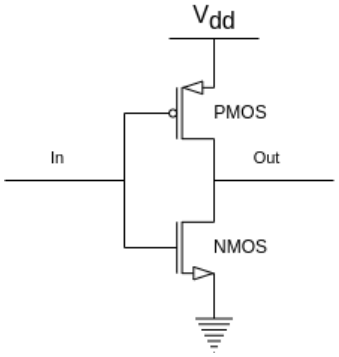

Figure 2.2: Schematic of an inverter

2.1.3 Dynamic Switching

A benefit of CMOS logic over other logic architectures is that it has low power dissipation during activation, as well as low static power consumption. This is due to how the logic units are built. Looking back to Figure 2.2, there cannot be an instance in a steady state where the input to the inverter causes both the NMOS and the PMOS to be switched on, which would cause a short circuit [8].

However, during a switching event, there is a small time frame during which both transistors could be considered half-on, which causes a short-circuit current from supply to ground [9].

Looking at Figure 2.3 the short current spikes as a result of switching can be observed. The contribution of short circuit current to the current spike in the figure is not significant, but rather the discharging and charging of the parasitic capacitance C_L has a large impact on the power consumption during a switching event. The parasitic capacitance C_L arises from the transistors themselves, as well as the wire connected to the output. The capacitance is charged when the output is connected to V_{DD} and subsequently discharged when the output is connected to ground.

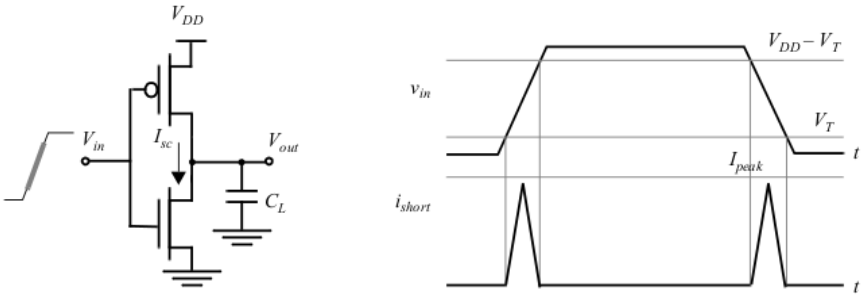

Figure 2.3: Current as a result of input switching. Illustration by [9]

This current spike may not be significant for only one inverter, but for larger circuits where many components are activating simultaneously, this can have a great impact.

2.2 SRAM

The work in this thesis was done on Static Random Memory Access (SRAM) memories and its corresponding control circuitry. An SRAM memory bitcell is a cell consisting of 6 transistors, seen in the schematic in Figure 2.4. It consists of two cross-coupled inverters (transistors Q1, Q3, and Q2, Q4) as well as two transistors (Q5, Q6) which connect the bitcell to the bitlines. Each of these cells can store one (1) bit of data. To increase the number of bits per word, these bitcells can be stacked horizontally, sharing only the wordline (WL) between them, while increasing the number of words possible to be stored in memory, cells can also be stacked vertically, which will cause the cells to share bitlines (BL, BLB). This allows for these memory cells to have a matrix configuration, typical of any RAM cell. Having more cells increases the capacity of the memory but makes it slower since having multiple cells share either WLs or BLs causes these wires to be more capacitive, i.e. slower [9].

The two latched inverters in the middle of the cell are complementary, meaning that one side of the cell will have stored the value '0', and on the other '1'. Which side of the cell that stores the actual memory bit the cell holds is a designer choice, and the control circuitry is designed around it.

2.2.1 Write Operation

During a write operation of the SRAM cell, one of the BLs is set to '1' and the other '0'. Which bitline is '1' and which is '0' is dependent on what bit is to be written to the cell, and is again dependent on design choice and control circuitry. Subsequently, the WL is set to '1' which would open the transistors Q5 and Q6, setting the value at each side corresponding to the bitline at that side [9].

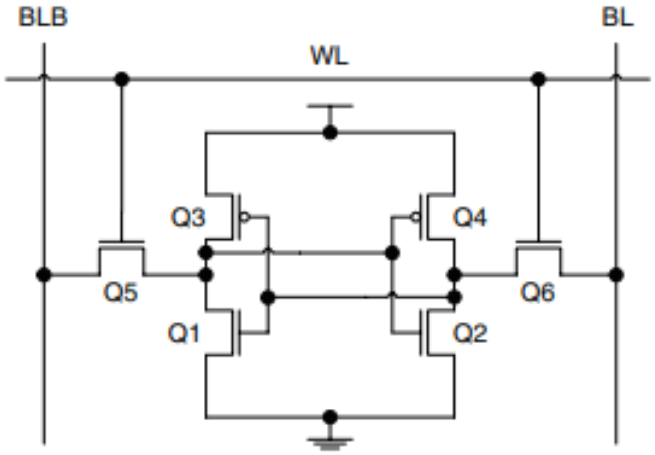

Figure 2.4: Schematic of an SRAM bitcell. Illustration by [10]

2.2.2 Read Operation

For the read operation the bitlines are instead both precharged to '1' and afterwards the WL is set to '1'. This will now evaluate the state inside the cell. Recall that in an SRAM cell, on one side the value '0' is always stored, and on the other '1' is stored. When the WL is set to '1' this will cause one of the bitlines to be discharged through the side of the cell that has '0' stored, causing a difference in voltage level between the bitlines. This difference will be measured by a sense amplifier connected to the bitlines, which will give what value the bitcell is currently storing [9].

2.3 Voltage Threshold

Voltage Threshold represents the minimum voltage required between the gate and source terminals of the transistor to achieve conduction in the channel. V_T is impacted by multiple factors during manufacturing but is ultimately a design parameter. Technology node refers to the semiconductor generation or manufacturing process, they are denoted as 40nm, 22nm, 5nm, etc. The smaller technology node means smaller feature sizes, higher transistor density, and better performance.

At manufacturing level, a higher doping concentration in the transistor channel (highlighted by x in Figure 2.1) will lower the V_T, and decreasing the doping will increase the V_T. Similarly, increasing or decreasing the oxide thickness will increase or decrease the V_T respectively [2].

When designing a circuit, the designer will have a catalog of transistors made available to them by the manufacturer of that particular technology node. This same catalog can be used when choosing transistor types for the tool to use.

2.3.1 Impact of Voltage Threshold on Transistor Behavior

Changing the V_T of a transistor will affect the reactivity of the transistor, power leakage in an idle state, and power consumption during operation.

A lower V_T will make a transistor react to a change in voltage faster, and can significantly reduce the time required for a circuit to produce an output. Conversely, a higher V_T will make the transistor react slower, delaying the output of the transistor.

A lower V_T will also allow more current to pass through in an idle state. In practice, this means there will be a higher leakage from a circuit while it is not operating. Consequently, if a loss of speed is acceptable it would be pertinent to have as high V_T as possible, to reduce this leakage effect.

In Figure 2.5 the effect of lower V_T can be observed, this effect will drastically increase the current in the idle state, while increasing the current for lower input at the gate. This effectively means the transistor reacts to a change in voltage quicker, resulting in a faster transistor.

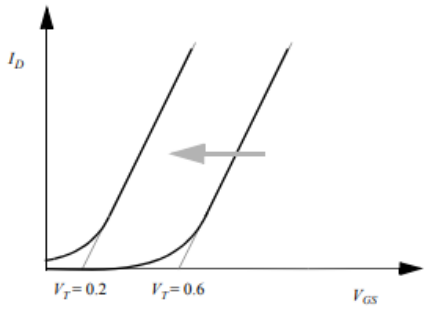

Figure 2.5: Channel current effect of lowering V_T. Illustration by [9]

The relationship between the effects of V_T on speed and leakage is not uniform and varies with circuit parameters. Additionally, depending on the use case, the circuit might spend different amounts of time in its idle state, which may further affect what V_T is ideal.

For this project, the impact of switching transistor types was evaluated in order to understand the effect it has on delay and leakage. This can be seen in Tables 2.1 and 2.2. To measure delay, a series of inverters were designed, as seen in Figure 2.2 with the different transistor types mentioned, where the input was toggled from '1' to '0' and measured the delay of the signal. For leakage measurement the same configuration was used but instead, the input remained idle.

In Tables 2.1 and 2.2 some sample data from these tests can be seen, and it can be observed that in a general scenario having every transistor with as high V_T as possible is the most effective, since changing from Extreme High Voltage Thresh-

Table 2.1: Performance metrics for different transistor types, using SVT as a reference point (40nm technology)

Metrics	ELVT	LVT	SVT	HVT	EHVT
Delay(Factor)	0.50	0.70	1	1.77	2.05
Current leakage(Factor)	612	5.80	1	0.22	0.14

Table 2.2: Performance metrics for different transistor types, using SVT as a reference point (5nm technology)

Metrics	ELVT	LVT	SVT
Delay(Factor)	0.68	0.79	1
Current leakage(Factor)	115.8	11.6	1

old (EHVT) transistors to Extreme Low Voltage Threshold (ELVT) transistors in 40nm technology results in about 4 times higher frequency while increasing leakage by a factor of 4371. Similarly, for 5nm technology, it can be observed that switching from Standard Voltage Threshold (SVT) transistors to ELVT transistors yields a decrease in delay of 0.68 while increasing leakage by a factor of 115.8.

Another effect of switching V_T is the impact on dynamic power consumption during the operation of logic gates as described in section 2.1.3. Since changing the V_T will impact the speed of the circuitry, it will subsequently minimize the time during which both transistors are switched on, as well as the speed with which the parasitic capacitance C_L discharges. This will make the current peak time seen in Figure 2.3 shorter, but steeper.

In Figure 2.6 the effect that different V_T have on dynamic current can be seen. It is known that the switching event is faster for low V_T but has a higher peak current. This means that a circuit with full low V_T transistors can be expected to have a larger power spike when many components are being activated at the same time, as opposed to high V_T.

This data has a very clear trend. The trade-off between speed and leakage is in clear favor of the transistor types with lower leakage, and thus in the optimal scenario that choice would be best. But as will be detailed in this report, this might not always be the case. Depending on how the circuit is built, a circuit might have certain weak spots where lowering the V_T will have a great impact on the overall speed of the circuit, while not contributing much to the overall leakage of the circuit.

2.3.2 Impact of Temperature on Voltage Threshold

Temperature has a range of effects on the behavior of a circuit in general, but for this project, they will be mostly disregarded. If a specific operating temperature is expected, that can be adjusted in the simulations.

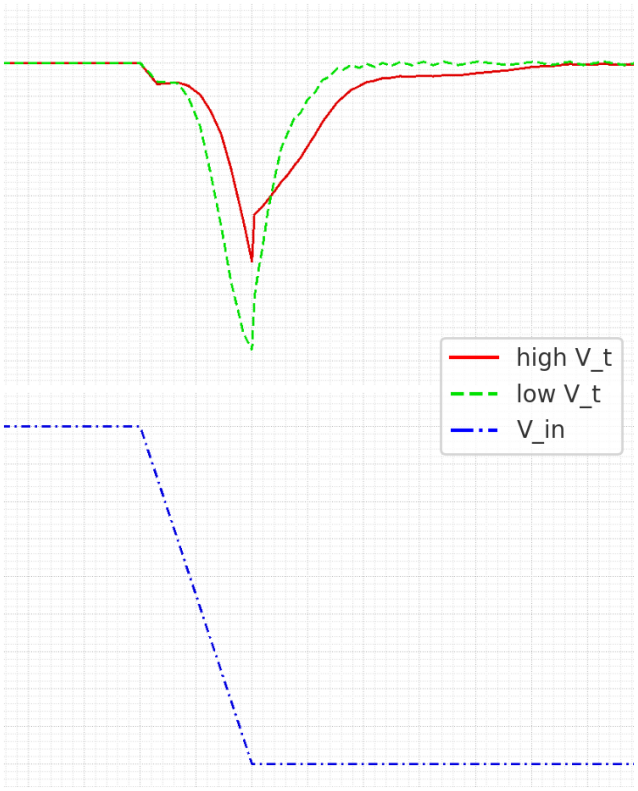

Figure 2.6: Current during a switching event for low V_T and high V_T transistors. Upper: supply currents as a function of time. Lower: Input voltage as a function of time.

One important factor of temperature is its effect on V_T. Higher temperature lowers the effective V_T of transistors, and vice versa. This effect can vary between transistor types.

According to the literature on the subject, increasing the temperature decreases the V_T between 2-4mV/°C [11]. Granted, this literature is from 1993, and since then semiconductors have significantly improved. To better understand this effect in the context of modern technologies, simulations were conducted for different temperatures. It was estimated that the range for the 40nm technology the V_T decrease as a function of increasing temperature was instead 0.24-0.30mV/°C.

To compensate for these effects on temperature, simulations had to account for temperature extremes by being run at the worst case for the desired metrics.

2.4 Interconnect Delay vs Gate Delay

During the initial phase of the project, effort was put into understanding the extent to which transistors influence the delay of the circuit, the two primary of them being the gate delay (transistor delay) and interconnect delay (wire delay). Studies and data have shown that as technologies continue to become ever smaller with each new technology node, the interconnect delay seems to be dominating the total delay of a path [12].

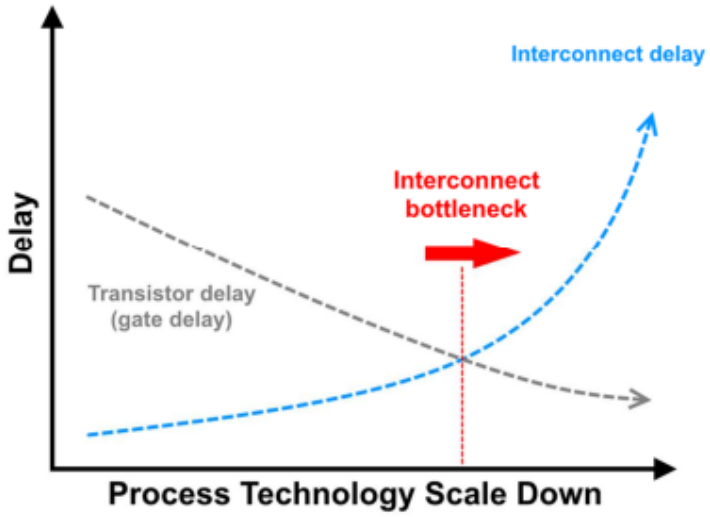

Figure 2.7: Trends in gate delay vs interconnect delay as technology nodes continue to scale down. Illustration by [12]

Studies have shown, as can be seen arbitrarily in Figure 2.7, that the technology nodes scaling down eventually reaches a point where interconnect becomes the dominant contributor to overall delay [12]. In the case of this project, this would imply the power of the tool being developed also goes down as technology nodes continue to scale. This information is further corroborated by [13] where it is discussed how the scaling down of wires does not proportionately increase the speed to the extent that happens when scaling down transistors.

In Figure 2.8 the measured impact that transistors and interconnects have had on overall delay over time is shown, and as technologies continue to scale, interconnects seem to strongly dominate the overall delay. Such information is of interest when developing this tool since it gives an indication of what performance can be expected.

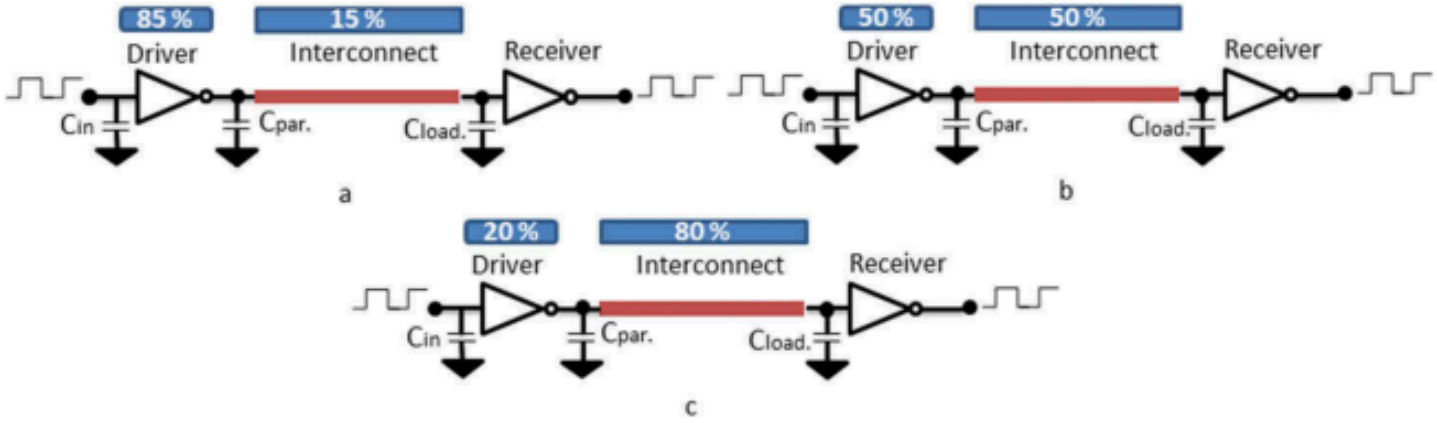

Figure 2.8: Percentage of gate and interconnect delay dominance with technology scaling. (a) Mid-1980's, (b) mid-1990's, (c) 2017. [13]

Chapter 3
Circuit Optimization

When designing a circuit the main consideration is function, but there are other considerations as well. Those might for example be constraints concerning size, shape, inputs, speed, or power usage. Furthermore, if these constraints are fulfilled, it might be possible to optimize for some of these characteristics.

In practice, this can be difficult to do. Modern circuits can have billions of transistors in them [14], which means manual changes are prohibitively time-consuming. Additionally, small incremental changes will often be impractical due to simulation times, which can be prohibitive for larger circuits.

3.1 Optimization Challenges

The purpose of the adjustments being made will generally be to reduce the leakage of a circuit while meeting a target speed constraint. There are several complications to this goal. The largest two are

- Design Rule Checking (DRC) Issues
  Issues with translating from a digital space to a physical one.
- Parameter space
  A too-large number of parameters will make both simulations and estimations challenging.

3.1.1 DRC Issues

DRC is the rulebook that layout designers have to adhere to when designing the layout of the circuit. The layout is what is sent to foundries for manufacturing of the chip and thus has to follow strict rules since not every design is manufacturable. This rulebook is growing ever larger and with each new (smaller) technology node the number of rules increases exponentially, making it increasingly difficult to design circuits that comply with these rules [15].

Figure 3.2 shows the layout of a full adder component with its respective schematic in Figure 3.1, a design that is a lot more complex than the inverter previously discussed. This one instead has 28 transistors, with T different transistor

types each transistor can assume, giving T^{28} different combinations this simple circuit could be arranged into. This would make optimization, even at a scale as small as this, difficult. However, it turns out it is not possible to change the V_T of each transistor in the layout since they are closely packed with each other, and this would violate DRC.

Looking at the layout in Figure 3.2 it is shown just how densely packed these transistors are. The bottom row consists entirely of NMOS transistors and the upper row of PMOS transistors. The lines in between are all interconnects connecting the transistors to the output, input, and between themselves. The straight lines indicate the gate connections of the transistor, and to the sides of these straight lines, the drain and source of the transistor can be seen. At this level, it is simple to change the V_T of the transistor since this would mean simply drawing a rectangle that signifies the desired V_T over the transistor. This would as previously mentioned cause a DRC violation since there needs to be a certain spacing between these V_T rectangles.

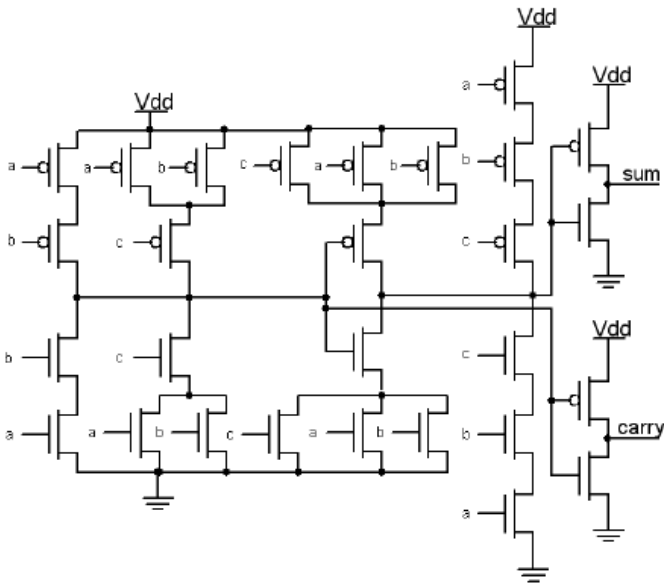

Figure 3.1: Schematic of a full adder. Illustration by [16]

As a silver lining to the issues the DRC presents it is not possible to change the V_T of every individual transistor, which would have most given rise to DRC violations. Instead, the transistors have to be grouped in such a way that does not violate these rules. This significantly reduces the number of changes that are possible, and thus reduces the number of variables on which to optimize the circuit.

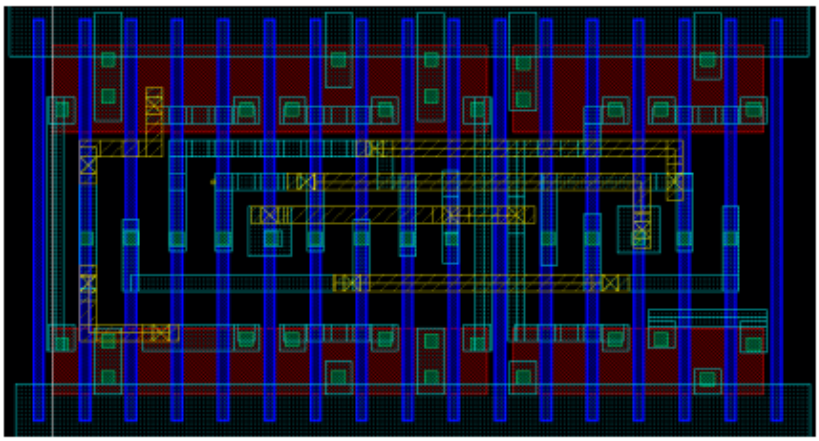

Figure 3.2: Layout of earlier full adder schematic. Illustration by [17]

3.1.2 Parameter Reduction

Even when DRC is accounted for, the number of changes that can be made to transistor V_Ts is too large. As the number of variables scales exponentially, brute force, ie. simulating all possible combinations, is completely obsolete for all but the smallest of circuits. As the granularity of changes increases, so does the need for more effective approaches to optimization.

There are many approaches to reducing the amount of variables that have to be iterated upon. As earlier stated, identifying which areas of the circuit see heavier traffic and which are less critical allows for increasing granularity in the critical areas, while decreasing granularity in less important areas. Another way to reduce variables would be to identify common trends among components and transistors, which could allow for setting some initial components to their most efficient configuration in the initial circuit, removing the need for adding its transistors to the list of configurable transistors.

The number of transistor types during this project often remains at or below 5. The amount of blocks can be slightly larger, but due to the aforementioned DRC issues, they will rarely be more than 10. Only a select few will have a measurable impact on leakage or delay, and thus many blocks without an impact could be discarded given the need for speed. This leaves a relatively small span of variables to work with, in the upper limit reaching single digits of millions. More often it is beneficial to only look at the most important blocks to reduce simulation times, which means barely thousands or sometimes even hundreds of combinations.

3.1.3 Critical Path

Ideally, a circuit would be optimized according to its critical path. A critical path is defined as the series of nodes with the longest combinational path in a circuit, which will be the bottleneck for its frequency (speed). In the case of SRAM memories, which are covered in this thesis, the speed of the circuit is known as the access time, or read time. It is the time from when you ask the memory to receive the stored value at the desired address (Clock is enabled) until you receive the stored value at the output of the memory.

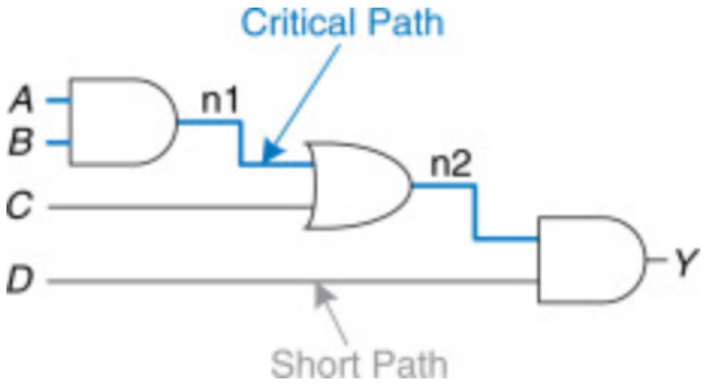

Figure 3.3: Critical Path defined as the longest path the input signals(A,B,C,D) take to reach the output (Y). Paths n1 + n2 is the critical path in this image. Illustration by [18]

Focusing on optimizing the circuit primarily along its critical path would give the best improvement of speed to leakage when shifting voltage thresholds. However, this would be too granular and could cause DRC problems, which is why in this thesis, a more high-level approach is implemented.

3.1.4 Divide and Conquer

Divide and Conquer is a common approach to simplifying a problem by identifying independent areas of the problem and solving those by themselves, reducing the size of the problem temporarily, and finally adding the solutions together to solve the entire problem. The challenge lies in finding which areas can be independently solved, and how they are connected.

This approach was considered, and even trialed with promising results, but demands an excessive amount of manual labor compared to the goal of an automated tool. The reason is that circuits are divided into subcircuits based on functionality, and those subcircuits can still be interconnected. As such, simulating one subcircuit may require the output of other subcircuits, making isolation difficult.

Finally, this manual process would have to be repeated for each circuit design, since the connections between subcircuits can be different, further hindering automation.

A solution could be to implement an algorithm for automatically finding a partition with a minimal amount of connectivity, but in the case of circuits finding a partition with minimal connectivity is an NP-hard problem known as graph partition [19], which is outside the scope of this project.

Chapter 4

Timing Issues

One of the primary components of a memory circuit is the timing block. This component is responsible for making sure the different signals in the circuit are enabled at the right times. This is done by adding capacitors that essentially act as a sequence of timers once triggered. This requires the timers to be tuned to certain conditions, with some pessimism to compensate for variance. Recall in section 2.2.2 that to read an SRAM bitcell, the circuit first needs to precharge the bitlines, followed by switching on the wordline to read the bit stored. Also, recall that bitcells can be stacked in a matrix configuration. This means that there is a variable amount of bitcells on each wordline and on the bitlines that depend on the memory configuration. This will, in turn, affect how capacitive the bitlines and wordlines are, affecting their speed.

As a result of this, bitlines will take a longer time to precharge if the bitline is more capacitive, and similarly it will take a longer time for the bitline to discharge through the bitcell if it has a higher capacitance. In effect, this sequence of timers will determine the speed of the memory circuit, but this raises problems:

- How can the difference in circuit speed be measured as a result of V_T changes when speed is predetermined by the timers?
- Changing V_Ts might impact the conditions that are used to set these timers, further compromising the reliability of the circuit.

The problem is that these timers will have to be attuned to every specific circuit, which means that any adjustment to V_Ts calls for updated capacitances in the timing block.

These timers are not automatically updated to account for changes in V_T. This means measurements of speed will be inaccurate, as the timers will still be set to the time when the original circuit had its conditions fulfilled. This can lead to wasted time if the conditions are now fulfilled faster, or a non-functional circuit if the conditions are fulfilled slower. For this reason, the timers will have to be updated to the new conditions, see Figure 4.1.

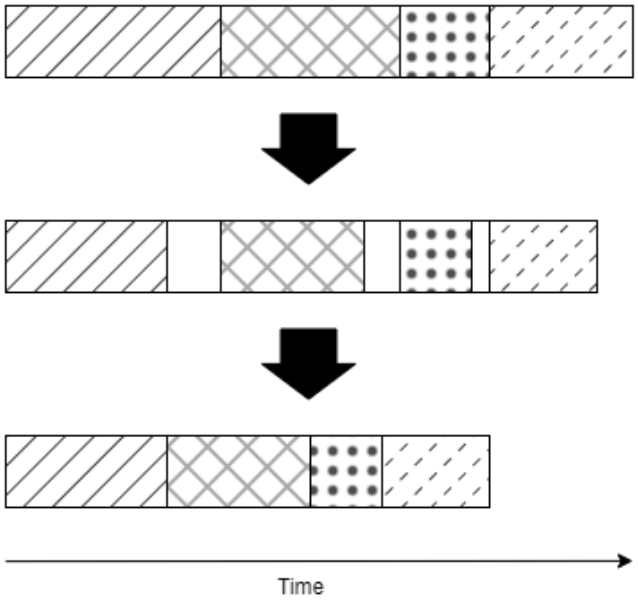

Figure 4.1: Example of how a timer-controller sequence might look before V_T changes, after V_T changes, and after updating timers

4.1 Timers

Sizing the capacitors is a straightforward task, but a time-consuming one. Each change to capacitors needs to be verified with several simulations, and will likely not be correct on the first try. Even when applying smart methods of sampling, it will take multiple simulations to achieve an optimized timing block.

For this project, there needs to be dozens of independent adjustments at a minimum. Since correct measurements of the delay are of the essence, this process of running several simulations per adjustment will become prohibitively large, even for a small circuit. Instead, an approximation method was used to idealize the timing block using logic.

4.2 Overriding the Timing Signals

To circumvent these signals and to save a lot of simulation time, Verilog-A modules were created to control these signals based on conditions rather than predetermined timers. Verilog-A is an analog hardware description language that allows for the creation of any type of analog circuit functionality [20]. With these modules the original timing signals are overridden with the ideal Verilog-A modules, essentially allowing the speed of the circuit to be adjusted on the fly. The conditions used for the timing signals when designing the circuit were determined, which were then subsequently embedded into the Verilog-A modules.

For example: the timer for precharging is set to 1 nanosecond with the default V_T configuration. Lowering the V_T of certain blocks in the circuit now makes the precharge sequence faster, and allows the timer to be shorter, for example, 0.85 nanoseconds. The Verilog-A module will then find that the speed of the signal has changed and will thus fire the response signal at the appropriate time. This means the actual effect of the V_T shift can now be measured, which would be that the effect of lowering these V_Ts gives a speed increase of 0.15 nanoseconds to the circuit.

With this the actual effect of lowering the V_T can be determined and what the timer should be in the new circuit created from the tool. It is possible to do this since the Verilog-A modules evaluate the signals in real-time, as opposed to having to iteratively find the appropriate timer after the creation of the memory. Without these modules, a complete evaluation of V_T shift effects in the memory would take weeks to a month for any mid-size memory and multiple months for large memories. By using these modules, the time consumed is a fraction of the original time.

4.2.1 Example Verilog-A Code

Provided here is an example of the Verilog-A code for the precharge sequence discussed in the previous section. In SRAM memories, to read from the bitcell, it is required that both bitlines connected to the cell are precharged to V_{DD}, and thus the condition the Verilog-A code uses is set such that all bitlines are at V_{DD} or close to V_{DD}.

Algorithm 1 Verilog-A

```
procedure BITLINESPRECHARGED(bitlines)
    for bitline in bitlines do
        if bitline < 0.95*V_DD then
            return False
        end if
    end for
    return True
end procedure
bitlines, enable ← input
if bitlinesPrecharged(bitlines) and enable > 0.5*V_DD then
    precharge_finished ← V_DD
else
    precharge_finished ← gnd
end if
return precharge_finished
```

The pseudocode provided in Algorithm 1 shows the idea behind the precharge module used for the thesis. The module takes all the relevant bitlines, as well as an enable signal as input. When all the bitlines have been precharged to above

95% V_{DD}, it returns that the precharging has finished, and subsequently, the next Verilog-A module uses that to continue the sequence of control signals. It is important to note that since this is a hardware description language, the code does not operate sequentially, as for regular programming languages. Instead it operates concurrently, similarly to the parallel nature of circuits. As a result, the output of all Verilog-A modules will be updated immediately upon any change to the inputs.

Chapter 5
Algorithm

During the project, an algorithm was implemented in the Python language. This chapter outlines how the algorithm processes a netlist to find an optimized version of that netlist that fulfills the constraints. A netlist is a file that describes a circuit and all its components in text format, which allows a simulation tool to simulate it.

The tool developed during the thesis is divided into three subroutines. A full flowchart can be seen in Figure 5.1. The first subroutine uses the original netlist and input from the user to create permutations of the original netlist. The second subroutine simulates the permuted netlists and stores the results in a database. The third subroutine reads the database and outputs estimations based on user-given constraints.

5.1 Preparations

A post-layout schematic of the circuit to be evaluated is required, and it has to have a timing block that has been optimized for the current circuit. A copy of the circuit is created, with Verilog-A code included in it. The Verilog-A code is specific for each architecture and technology, but independent of size, since it probes the circuit at various points and overrides the inherent signals to control them. The names of these nodes and what is to be controlled may change depending on architecture and technology. This means that for the optimization to work appropriately for new architectures or technology, the Verilog-A code needs to be adjusted beforehand.

5.2 Database

Once preparations are completed the tool will read the entire schematic of a circuit with Verilog-A code included, henceforth referred to as the 'default' schematic. Both an unmodified schematic with Verilog-A, as well as the original schematic with optimized memory timings, are simulated, and the results are stored in a database to be used as reference points for the optimization.

When the original circuit was implemented, different sections of it were defined as subcircuits by the designer. These subcircuits may contain other smaller

subcircuits, meaning the entire circuit is essentially built from a hierarchy of subcircuits.

In the configuration file of the tool, the user may specify any subcircuit that is to be changed, though for subcircuits at lower levels the risks of DRC issues (from section 3.1.1) are present.

Memory circuits are standardized enough that the same areas in the same architecture are the usual suspects when it comes to optimizations, giving the option to focus on those areas if the user is familiar with the circuit.

Once the different areas of optimization have been selected, as well as the different transistor types that can be chosen, one schematic is generated from the default for each combination of bounded area and transistor type. The schematics are simulated in a standardized manner, and the results are saved to the database that will be used for optimization. Additionally, the difference between each schematic and the references (default with and without verilog-a) are stored in the database.

Simulations are done in worst-case scenarios for frequency and leakage respectively, accounting for both variations in manufacturing and temperature. While this increases simulation time, it also ensures that the database gains pessimistic data. If simulations were only made with no regard for variations or temperature, the results could be unreliable, as previously discussed in section 2.3.2.

5.3 Optimization

From the database, statistics for all different transistor configurations are read. These statistics are then used to extrapolate all possible combinations of transistor changes, to create a set of theoretical circuits that can be achieved through changes to V_T. By comparing the theoretical to the 'default' schematic, these combinations are each given an estimation of what frequency and leakage they would have.

Any combinations that do not fit the constraint are removed from the set. The set is then converted to a list and sorted by leakage. A sample of the most efficient combinations that still fulfill the constraint will then be provided.

The user may also specify if the tool is to create an actual netlist based on the optimal configuration.

5.4 Algorithm Pseudocode

In this section, the steps of the algorithm are detailed in pseudocode: partitioning the netlists (algorithm 2, **Partitioner** in Figure 5.1), running the simulations (algorithm 3, **Simulation** in Figure 5.1), and finally doing optimizations from the database generated from the simulations (algorithm 4, **Optimizer** in Figure 5.1).

Algorithm 2 Partitioning

```
sub_blocks, transistor_types ← input
InjectVerilogA(default_netlist)
SaveNetlist(default_netlist)
for every sub_block in sub_blocks do
    for every transistor_type in transistor_types do
        new_netlist ← CreateNetlist(sub_block, transistor_type)
        InjectVerilogA(new_netlist)
        SaveNetlist(new_netlist)
    end for
end for
```

Algorithm 3 Simulating

```
sub_blocks, transistor_types ← input
default_netlist ← ReadNetlist(input)
speed_data ← SimulateSpeed(default_netlist)
leakage_data ← SimulateLeakage(default_netlist)
SaveToDatabase(speed_data, leakage_data)
for every sub_block in sub_blocks do
    for every transistor_type in transistor_types do
        edited_netlist ← ReadNetlist(input)
        speed_data ← SimulateSpeed(edited_netlist)
        leakage_data ← SimulateLeakage(edited_netlist)
        SaveToDatabase(speed_data, leakage_data)
    end for
end for
```

Algorithm 4 Optimization

```
database, constraint ← input
combinations ← GenerateCombinations(database)
for combination in combinations do
    if combination < constraint then
        combinations.Remove(combination)
    end if
end for
SortByLeakage(combinations)
```

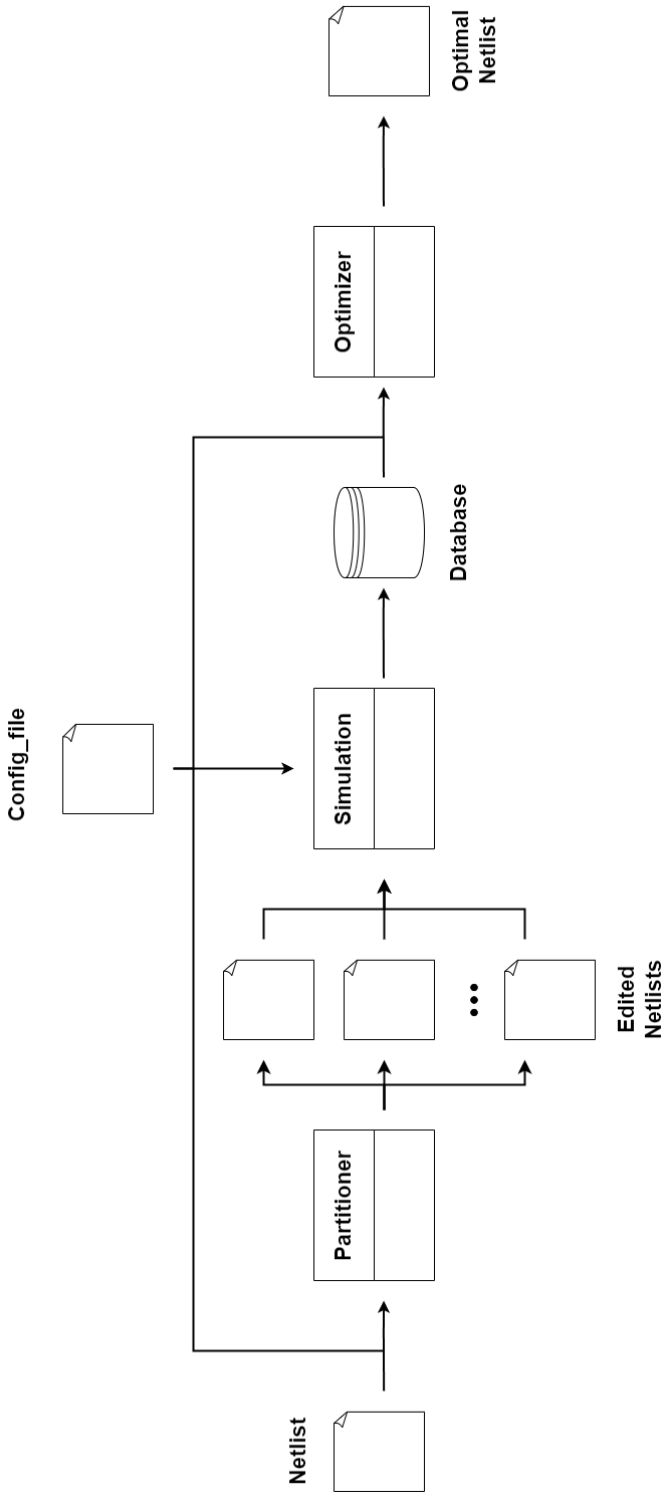

Figure 5.1: Complete flowchart of the tool, the three subroutines can be seen as **Partitioner**, **Simulation** and **Optimizer**

Chapter 6
Results

This chapter gives an overview of the final results of a few different circuits that were adjusted with the tool.

6.1 Circuit spans

A tool was developed that allows optimizing a circuit with little input from the user. A schematic has to be provided and some configurations have to be specified. When optimizing a circuit it is important to keep in mind that the results are approximations and to make sure to have an error margin. Especially in smaller circuits, approximation is more difficult.

The following plots represent all the estimations the tool has made for a specific circuit. Every cross and dot represents one single configuration. Dots represent combinations that may be optimal depending on constraints, and the line indicates the trend of optimal configurations. Frequency is displayed on the horizontal axis, and leakage is displayed on the vertical axis. The vertical axis is logarithmic.

Note that all of the plots in the following result sections have normalized results, to comply with NDAs and instead show factor increase, where the (1.0,1.0) datapoint is the slowest circuit, with the lowest leakage. Circuits will henceforth be referred to by their size in Kilobits(Kb) and their technology node. In this section, four circuits of two different node technologies will be covered, one smaller and one larger each.

6.1.1 4Kb 40nm circuit

Plots of the circuit can be seen in Figures 6.1 and 6.2. The plot in Figure 6.1 has a logarithmic y-axis, to allow for a better view of results in low leakage ranges.

Compared to the permutation with the lowest leakage, the fastest permutation shaved off almost half of the total delay, equivalent to a factor of about 1.88 increase in frequency. However, this comes at the cost of an increase in leakage by a factor of over 230.

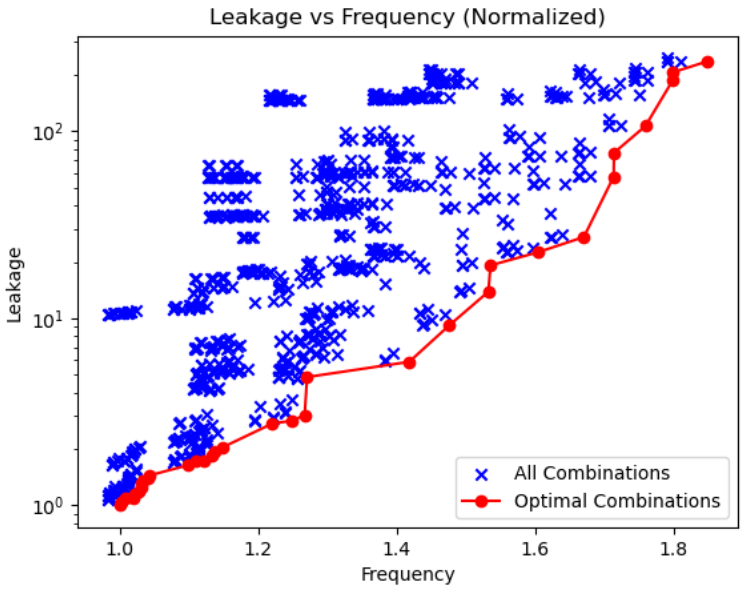

Figure 6.1: Normalized plot of different combinations for a small memory cut in 40nm technology. Circles show the best possible combinations, other combinations shown with crosses

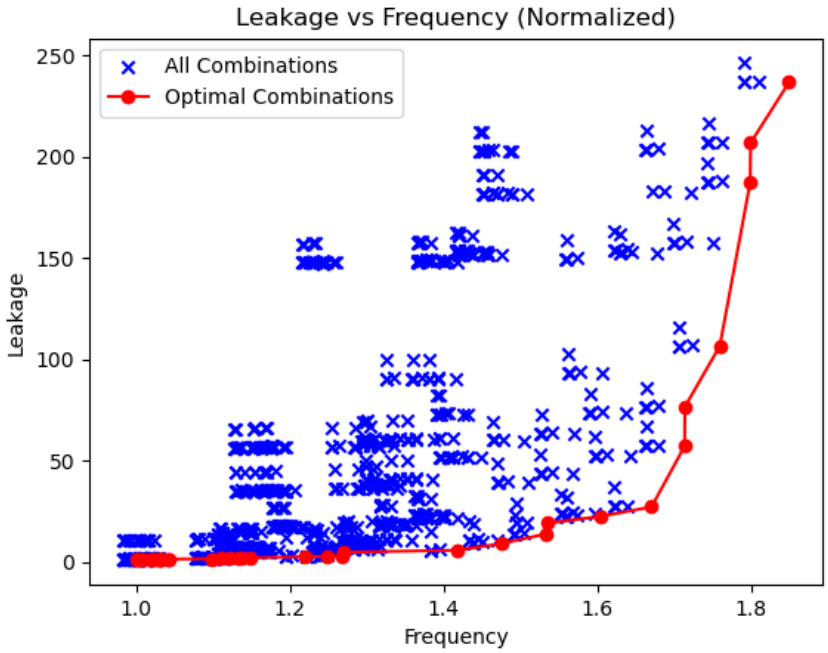

Figure 6.2: Normalized plot of different combinations for a small memory cut in 40nm technology. Circles show the best possible combinations, other combinations shown with crosses

Combination	Frequency	Leakage
Minimum Leakage	1	1.02
Maximum Leakage	1.82	246.1
Minimum Frequency	1.07	1
Maximum Frequency	1.88	236.6

Table 6.1: Relative effects of extremum points in the Small 40nm circuit.

6.1.2 32Kb 40nm circuit

For the large 40nm circuit 5 different V_Ts and 4 different blocks were chosen to optimize, this gave 625 different permutations of the circuit. In Figures 6.3 and 6.4 all of these permutations can be observed as datapoints (denoted by crosses), as well as the line denoting the optimal configurations for the desired frequency. Figure 6.3 has a logarithmic y-axis, which allows results in the lower leakage range to be compared.

For this circuit, the range of frequencies available through V_T changes alone can accomplish a factor increase of 2.14 as opposed to the slowest possible circuit, while in that case also increasing leakage by more than 200. This relationship of a larger leakage increase compared to a smaller speed increase was also observed in 2.5.

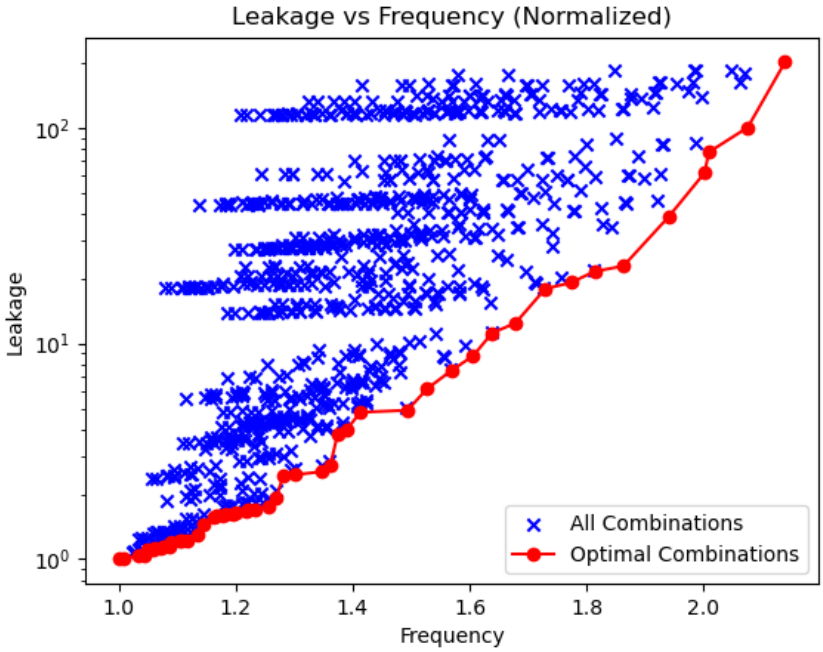

Figure 6.3: Normalized plot of different combinations for a large memory cut in 40nm technology. Circles show the best possible combinations, other combinations shown with crosses. The y-axis is logarithmic.

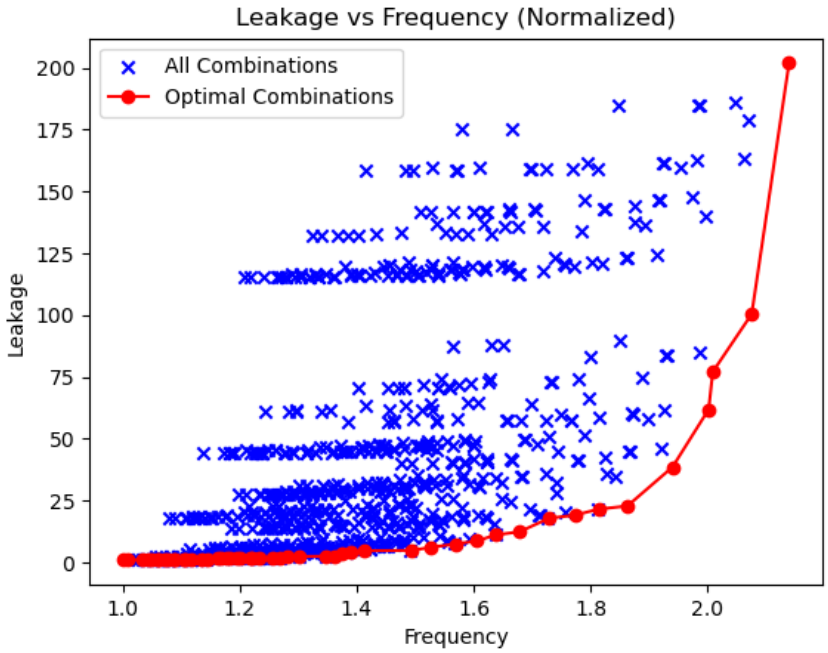

Figure 6.4: Normalized plot of different combinations for a large memory cut in 40nm technology. Circles show the best possible combinations, other combinations shown with crosses. The y-axis is logarithmic.

Combination	Frequency	Leakage
Minimum Leakage	1	1
Maximum Leakage	2.14	202
Minimum Frequency	1	1
Maximum Frequency	2.14	202

Table 6.2: Relative effects of extremum points in the Large 40nm circuit

6.1.3 1Kb 5nm circuit

Figure 6.1 shows the plot extracted from the 1Kb 5nm circuit, this circuit was initially skewed towards lower V_T transistors, which allowed for large leakage reductions. Again, this comes at the cost of reduced frequency. The minimum and maximum frequency differ by a factor of roughly 1.5, while minimum and maximum leakage differ by a factor of almost 2. In the 5nm circuits, fewer variables had a large impact on the speed. This leads to larger gaps between configurations. It is possible that in small circuits like these, it could be useful to increase the granularity of variables to get a better selection of options.

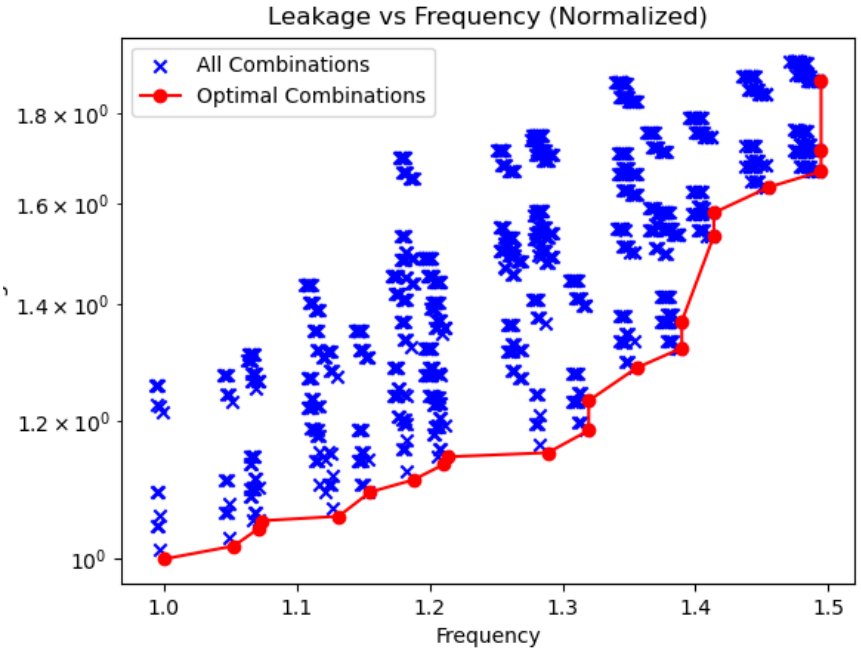

Figure 6.5: Normalized plot of different combinations for a small memory cut in 5nm technology. Circles show the best possible combinations, all combinations shown with crosses

Combination	Frequency	Leakage
Minimum Leakage	1.007	1
Maximum Leakage	1.48	1.93
Minimum Frequency	1	1.26
Maximum Frequency	1.50	1.88

Table 6.3: Relative effects of extremum points in the Small 5nm circuit

6.1.4 40Kb 5nm circuit

In the 5nm circuit, there were not as many variables used as there were for the 40nm circuit. There were only 3 different threshold voltages available as well, and only 3 blocks could be identified that would yield any increase in speed along the critical path. As a result of this, there were only 27 different permutations. It was observed that there is a frequency span of 1.0-1.39 and a leakage span of 1.0-2.3, and again the plot seen in Figure 6.6 displays the optimal configurations for each desired frequency that can be achieved, although the data points are highly discretized as a result of few variables.

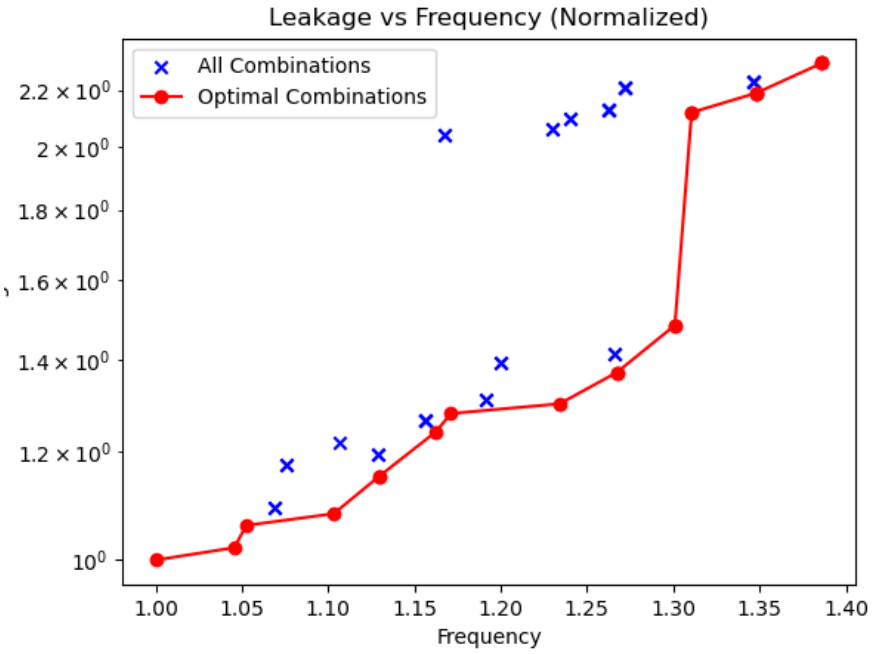

Figure 6.6: Normalized plot of different combinations for a larger memory cut in 5nm technology. Circles show the best possible combinations, all combinations shown with crosses

Combination	Frequency	Leakage
Minimum Leakage	1	1
Maximum Leakage	1.39	2.30
Minimum Frequency	1	1
Maximum Frequency	1.39	2.30

Table 6.4: Relative effects of extremum points in the Large 5nm circuit

6.2 Tool-generated Circuit

Running the tool will output a list of combinations that are estimated to meet the constraints that have been given to the tool. For each combination, the different selections of V_T are displayed along with the estimated frequency and leakage. This list will be sorted by leakage, displaying the lowest leakage combinations first. Table 6.5 provides a few combinations selected from an output using an arbitrary constraint, on an arbitrary circuit. Only a few results were selected to be displayed in this table, to help highlight noteworthy results.

In Table 6.5 the values are normalized. Frequency is normalized to the arbitrary constraint that was set for the tool in this specific case, meaning a frequency above 1 meets the constraint, and a frequency below 1 fails to meet the constraint. Leakage was normalized to the default circuit so that the leakage is represented as a factor of the default circuit leakage.

Circuits	Subcircuit 1	Subcircuit 2	Subcircuit 3	Subcircuit 4	Frequency	Leakage
Default Circuit	Unchanged	Unchanged	Unchanged	Unchanged	0.898	1
Configuration 1	Unchanged	EHVT	SVT	HVT	1.002	1.46
Configuration 2	EHVT	Unchanged	SVT	HVT	1.024	1.555
Configuration 3	EHVT	EHVT	SVT	Unchanged	1.003	1.618
Configuration 4	EHVT	HVT	SVT	HVT	1.034	1.63
Configuration 5	Unchanged	SVT	HVT	HVT	1.003	1.872

Table 6.5: Table showing different circuit configurations to meet frequency constraint. Frequency values are normalized around the constraint, while leakage values are normalized around the leakage from "Default Circuit". "Unchanged" refers to the default configuration in the circuit. V_T types are the types mentioned in table 2.1

6.3 Runtimes

Runtimes from the tool were evaluated for the circuits that the optimization was run on. The results can be seen in table 6.6. For this evaluation, the number of variables ran for 40nm technology was 5 subcircuits and 5 transistor types, while for 5nm technology, it was 3 subcircuits and 3 transistor types. Creating the plots and running the optimizations from the generated databases had a runtime of less than a second in all cases, and were thus omitted from the table.

Run type	4Kb 40nm	32Kb 40nm	1Kb 5nm	40Kb 5nm
Partitioning	00:01:40	00:06:22	00:03:57	00:35:19
Simulation + DB creation	01:05:01	09:22:40	01:36:55	23:08:40

Table 6.6: Runtimes for different chip sizes and technologies, including the different types of runs for the thesis, times shown in HH:MM:SS.

As can be seen in table 6.6, most of the time is spent on the simulations. For this tool, they are the bottleneck when it comes to the runtime of the algorithm. As can be seen in the simulation time of the 40Kb 5nm, not overriding the timing signals would cause runaway simulation times, see section 4.2. There are some other ways of speeding up simulation time, other than reducing the number of simulations, such as increasing the number of cores the simulator is run on or reducing the size of the netlist. Reducing the netlist is a very attractive way of reducing simulation times, however it is at present out of reach. For the presented runtimes, an unreduced netlist with overridden timing signals was simulated using 8 cores.

Comparing the 40nm simulations, an increase in size by a factor of 8 leads to an increase in the total simulation time by about 8.7. For the 5nm, an increase by a factor of 40 in chip size leads to an increase of about 14 in overall simulation time. This data is insufficient to predict the relationship between chip size and simulation time. It is interesting to note the difference in simulation times between the technology nodes, although the 1Kb 5nm chip is a quarter as spacious as the 4Kb 40nm chip, the simulation time, as well as partitioning time were both larger. This can also be observed when comparing the 32Kb 40nm with the 40Kb 5nm chips. The difference in partition and simulation times can be attributed to the increased circuit complexity and larger file sizes for smaller technology nodes.

Chapter 7
Discussion

This chapter will discuss the results and interesting findings during the project. Additionally, potential sources of error that could compromise the validity of the results will also be discussed.

7.1 Results

As can be seen in the earlier results, the effect V_T has on a circuit can vary dramatically. Frequency can be more than doubled in some cases, while in other cases it can not reach more than a 30% speedup. Leakage has an even larger range, sometimes increasing a hundredfold for a modest increase in frequency.

Depending on the use case, even these large leakage increases may be acceptable. An off-the-shelf memory that fits the constraints for frequency and memory size might be too large, or drain an unacceptable amount of power. The limitations of physical space can be as challenging as those of speed.

The plots in section 6.1 give some insight into circuit behavior. Using plots of circuit characteristics when changing V_T, it is possible to compare circuits of the same technology but different sizes. These comparisons help to give an idea of how circuit size affects the flexibility of the tool. At several places in the plots, there are large steps in either the horizontal or vertical axis. This indicates that granularity might be too low for some components, and can be used to identify interesting areas of optimization.

The non-logarithmic plots for the 40nm circuits show that after a certain point, increasing the frequency of these circuits becomes very costly. This can also be observed in the 5nm circuits, though not to the same extent. Knowing at what point this increase in cost occurs can inform if a frequency can be achieved at a reasonable cost or not. It is worth pointing out that due to the low granularity, the real cut-off point might have been offset slightly.

Table 6.5 highlights how the output of this tool can be used to identify trends in circuit behavior. For example, subcircuit 3 seems to consistently be set to SVT in this case, meaning that it is likely an efficient change compared to some of the

transistors of other subcircuits, which are more varied. As earlier mentioned, this may be a result of low granularity, and in this case might hint at good areas for increasing granularity, if possible.

Also of note is that the frequency margin is very small. Table 6.5 shows that many results have less than a percent of margin from the constraint. It is important that the user is aware of this, and adds their pessimism to the initial constraint or makes sure to select a result with sufficient margin.

7.2 Verilog-A

The final method uses the Verilog-A as a key component to reduce the number of simulations and allow the tool to evaluate the effect that V_T shifts have on each subcircuit. This method of using Verilog-A to determine pulse widths just in time can be applicable in many different areas. It could be used for optimization of other types of circuits, not just memories, given that the circuit has control signals with pulse widths that need to be altered as a result of a V_T shift. It could also be used for any process that uses iterations to determine pulse widths, such as the one for sizing the capacitors for the timers discussed in section 4.1. Employing Verilog-A would enable a single-step determination of the timers, with reservation for potential errors that might occur with the Verilog-A method, discussed later in section 7.3.3.

7.3 Potential Sources of Error

In this section, potential sources of errors identified that might cause the approximations from the tool to be incorrect will be discussed. In most cases, the tool was able to correctly estimate speed and leakage within 5%. This 5% error remains unexplained but might come from the reasons listed below.

7.3.1 Linear Approximation

It is important to note here that transistors are nonlinear devices, they can be used to create amplifiers due to their non-linear nature. In the approximations done when optimizing in this thesis, the effect of a V_T shift on a subcircuit is assumed to have a linear impact on the rest of the circuit. The assumption being made is that the measured impact V_T shifts have on each subcircuit could be linearly combined, and in that sense, the total number of circuits that could be created from the tool was substantially increased. As mentioned in 3.1 there can be created T^B combinations of subcircuits and V_T shifts, but to simulate all combinations would be unfeasible. Instead, only the behavior of each subcircuit was simulated for all different available threshold voltages and the estimation was that adding the speed differences together should yield the approximate speed for the specific circuit.

These approximations may cause errors in the calculations since signals are not perfectly sequential and may instead overlap, especially when moving towards lower V_T. Figure 7.1 shows the signals for an inverter chain, where one of the chains contained purely LVT transistors, and the other purely EHVT transistors. The non-linearity can here clearly be seen from the output of the inverters compared to their inputs, this non-linearity effect can also be seen for LVT as an exacerbated case, where the overlap is greater. There is a suspicion that this effect might cause approximation issues when having signals go from a low V_T subcircuit to a higher V_T subcircuit.

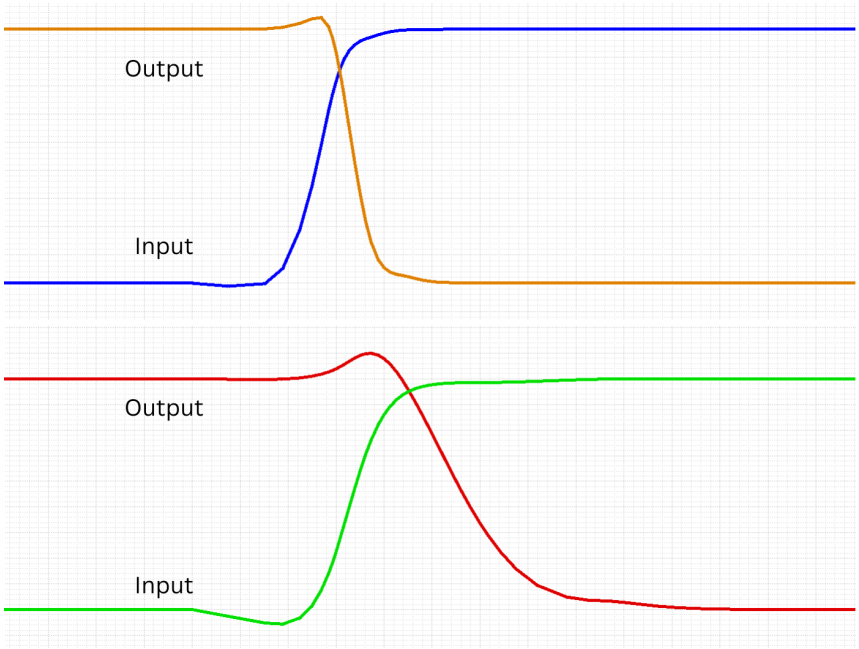

Figure 7.1: Upper: LVT Transistors, Lower: EHVT Transistors.

7.3.2 IR drop

To add to the previous section, IR drop is the effect of supply voltage and ground level converging toward each other during the activation of key elements in the circuit. This happens when many components of the circuit operate at the same time, recall in 2.1.3 that during the switching event of CMOS logic, there is a momentary period in which both transistors of the inverter are on, creating a direct path between supply and ground, as well as parasitic capacitances being charged or discharged. During this time, supply and ground will converge toward each other as seen in Figure 7.2. This affects the whole circuit and makes it slower. To add on to that, lower V_T transistors further exacerbate this effect, by making the drop deeper but faster, as was discussed in section 2.3.1 with direct path currents.

This effect of the change in IR drop when changing the V_T of transistors is difficult to account for when only measuring the effect of switching the V_T of one subcircuit at a time, which was the case for this thesis. As a result of this, the approximations for switching many subcircuits to lower V_T may be compromised.

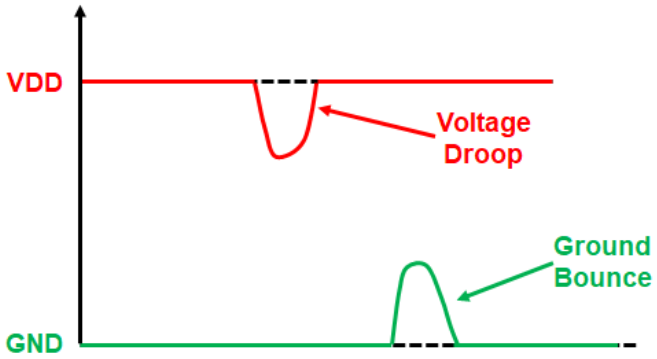

Figure 7.2: Voltage drop and ground bounce as a result of direct path current. Illustration by [21].

7.3.3 Verilog-A Models

Another source of errors in the approximations may be the Verilog-A models themselves. When using these models, the generated signals will be ideal. This means that they are not impacted by parasitic effects and are instead instantaneous, something that does not occur in real circuitry. This can be seen depicted in Figure 7.3.

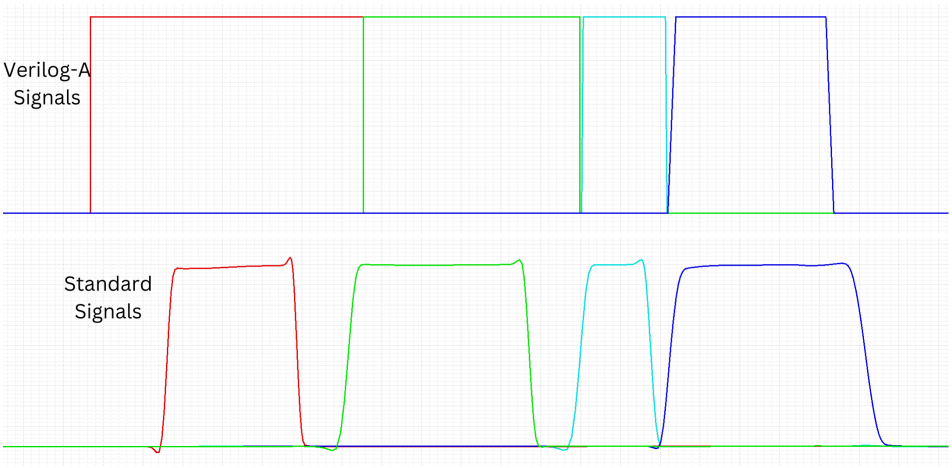

Figure 7.3: Top: Verilog-A generated signals, Bottom: Actual signals

An important thing to note is that the approximation error does not arise from the difference in pulse width between the two, since this can be accounted for, rather the difference in rise and fall time for the signals, as well as the delay between the signals, which the Verilog-A models do not have.

Chapter 8

Conclusion & Future Work

This chapter is a summary of the conclusions that can be drawn by looking at the results and comparing them to initial goals and expectations. Further, it will detail what potential future work can be made with this project as a basis.

8.1 Voltage Threshold

It can be concluded that leakage increases greatly when changing to lower V_T transistors, but not to the extent that was described in section 2.3.1. This is partly due to the memory circuits primarily being comprised of bitcells. For this thesis, the V_T of the bitcells remained unaltered. This choice was made for multiple reasons, but the main one is that changing the type of bitcell can alter the layout of the bitcell, which would then need larger changes in the layout of the circuit.

Furthermore, the observed effect that the tool was able to have greater control over the 40nm circuits was also previously discussed in section 2.4. The frequency span for the 5nm circuits was at a maximum of 1.0-1.49, while the 40nm circuits had a frequency span maximum of 1.0-2.18. The cause of this is believed to be:

- Larger interconnect delay relative to gate delay for smaller technology nodes.
- Fewer available variables with which to optimize the circuit.

The extent to which either of these effects has an impact on the observed frequencies remains uncertain.

The effect of transistors on frequency is also proven to be lower than was observed in isolated transistors, but not as dramatic as the difference in leakage. There is still a meaningful range of changes that can be made to affect the speed of the circuit.

8.2 Tool

In conclusion, the tool achieved the initial goal of providing an optimized version of a circuit given a constraint. The tool allows the user to take a circuit and make

hundreds of variations of it. These all have different properties, and the tool will provide the most optimal combination for the desired frequency.

The tool also allows for identifying parts of the circuit that have a larger effect on the overall speed while not impacting leakage as much. The plots in section 6 only show data points for this thesis, but internally it gives a good understanding of what changes can be made to the chip to speed it up to yield a higher frequency at low leakage cost.

With this tool, instead of manually adjusting an existing circuit, or creating a new one from scratch, a designer can just entice it with one or a few delectable circuits along with some configurations and constraints. The tool will then provide a list of the most optimal permutations of the circuit unless it is not possible to meet the constraint. This can save a lot of time by removing the necessity of designing a new circuit.

8.2.1 Future Relevance

As technology advances, the sizes of transistors decrease. As this thesis has shown, smaller transistors make up less of the contribution to the total speed of a circuit, as discussed in section 2.4. This will mean that the total effect adjustments of transistors can have on a given circuit will likely decrease with time. In all likelihood this will result in the tool having a smaller span of possible circuits, and thus a smaller chance of successfully converting a circuit to one that meets constraints.

However, the tool requires little effort from the user. And if it successfully finds a circuit that meets constraints, it saves a lot of work. Additionally, Moore's Law has been predicted as not holding for much longer [22]. Node size is still decreasing but is unlikely to become much smaller. As such, the tool will retain some relevance as long as transistors do.

8.3 Further Work

While this project has given some interesting results, and can to some extent already be used, there remains work to be done to have a commercially viable V_T optimization tool.

- Increase Accuracy
  While there is no accuracy error unaccounted for to the extent of 5%, subsequent iterations of running this tool gave greater insight into how to improve this accuracy further and the tool should be iterated upon until an acceptable level has been reached.
- Add bitcells as a selection variable
  Since the tool is intended to be used for memory circuits, it would be a great option to allow for different types of bitcells or different bitcell V_T.
- Incorporate dynamic power consumption as a metric
  In this thesis, only speed and leakage were covered as metrics on which to optimize, however, it would be interesting to add dynamic power as a metric.

- Increase granularity and/or design variables
  For this thesis, the choice of design variables was quite shallow to limit DRC issues, it is possible to increase the granularity or add more blocks as part of the optimization, but it requires better insights into the inner workings and layout of the circuits.

Bibliography

[1] G. Moore, "Moore's law," *Electronics Magazine*, vol. 38, no. 8, p. 114, 1965.

[2] D. M. B., "What is the threshold voltage?" https://chipedge.com/what-is-the-threshold-voltage/, 2024, last accessed 12 april 2024. [Online]. Available: https://chipedge.com/what-is-the-threshold-voltage/

[3] S. Daboul, S. Held, J. Vygen, and S. Wittke, "An approximation algorithm for threshold voltage optimization," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 23, no. 6, nov 2018. [Online]. Available: https://doi-org.ludwig.lub.lu.se/10.1145/3232538

[4] M. Rahman and C. Sechen, "Post-synthesis leakage power minimization," in *2012 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2012, pp. 99–104.

[5] "Memory power reduction in soc designs using powerpro mg," https://www.design-reuse.com/articles/21806/memory-power-reduction-soc-design.html. [Online]. Available: https://www.design-reuse.com/articles/21806/memory-power-reduction-soc-design.html

[6] S. Voinigescu, *High-Frequency Integrated Circuits*, ser. The Cambridge RF and Microwave Engineering Series. Cambridge University Press, 2013.

[7] M. Maadi, "Evaluation of spice p-n junction's, bjt's and mosfet's model parameters," p. 5, 2012. [Online]. Available: https://www.researchgate.net/profile/Mohammad-Maadi/publication/263965981_Evaluation_of_SPICE_P-N_Junction's_BJT's_and_MOSFET's_Model_Parameters/links/00b4953c6e49f5993c000000/Evaluation-of-SPICE-P-N-Junctions-BJTs-and-MOSFETs-Model-Parameters.pdf

[8] "Importance of cmos," https://medium.com/@vaibhav.tomar21/importance-of-cmos-1229d79288ac. [Online]. Available: https://medium.com/@vaibhav.tomar21/importance-of-cmos-1229d79288ac

[9] J. M. Rabaey, "Digital integrated circuits. a design perspective: United states edition," p. 219, 1996.

[10] A. Pavlov, *CMOS SRAM Circuit Design and Parametric Test in Nano-Scaled Technologies. Process-Aware SRAM Design and Test.*, ser. Frontiers in Electronic Testing: 40. Springer Netherlands, 2008. [Online]. Available: https://ludwig.lub.lu.se/login?url=https://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,uid&db=cat02271a&AN=atoz.ebs419311e&site=eds-live&scope=site

[11] Weste and Eshraghian, *Principles of CMOS VLSI Design: a systems perspective, Second Edition (1993)*, 1993, pp. 48–49.

[12] T. Grzela, "Comparative stm-based study of thermal evolution of co and ni germanide nanostructures on ge(001)," Ph.D. dissertation, 12 2015.

[13] S. Turkane and A. Kureshi, "Emerging interconnects: A state of the art review and emerging solutions," *International Journal of Electronics*, vol. 104, 02 2017.

[14] K. Rupp, "Microprocessor trend data," https://ourworldindata.org/grapher/transistors-per-microprocessor, 2022, last accessed on 12/06/24. [Online]. Available: https://ourworldindata.org/grapher/transistors-per-microprocessor

[15] E. Sperling, "Design rule complexity rising," https://semiengineering.com/design-rule-complexity-rising/, 2018, last accessed 18 april 2024. [Online]. Available: https://semiengineering.com/design-rule-complexity-rising/

[16] K. Navi, O. Kavehei, M. Rouholamini, A. Sahafi, S. Mehrabi, and N. Dadkhahi, "Low-power and high-performance 1-bit cmos full adder cell," *Journal of Computers - JCP*, vol. 3, pp. 48–54, 02 2008.

[17] T. Johansson, "A technology agnostic approach for standard-cell layout design automation," *LUP Student Papers*, p. 36, 2019. [Online]. Available: https://lup.lub.lu.se/student-papers/search/publication/9001291

[18] "Combinational logic," https://www.sciencedirect.com/topics/computer-science/combinational-logic. [Online]. Available: https://www.sciencedirect.com/topics/computer-science/combinational-logic

[19] A. E. Feldmann and L. Foschini, "Balanced partitions of trees and applications," *Algorithmica*, vol. 71, no. 2, pp. 354–376, 2015.

[20] "Introduction to verilog-a," https://verilogams.com/tutorials/vloga-intro.html. [Online]. Available: https://verilogams.com/tutorials/vloga-intro.html

[21] "Ir drop analysis - ii," http://vlsi-soc.blogspot.com/2019/06/ir-drop-analysis-ii.html. [Online]. Available: http://vlsi-soc.blogspot.com/2019/06/ir-drop-analysis-ii.html

[22] urlhttps://www.nature.com/news/the-chips-are-down-for-moore-s-law-1.19338. [Online]. Available: https://www.nature.com/news/the-chips-are-down-for-moore-s-law-1.19338