NTNU – Trondheim
Norwegian University of
Science and Technology

Low Energy Implementation of Robust Digital Arithmetic in Sub/Near-Threshold Nanoscale CMOS

For Ultrasound Beamforming

Lars-Frode Schjolden

Master of Science in Electronics
Submission date: June 2013
Supervisor: Snorre Aunet, IET
Co-supervisor: Trond Ytterdal, IET

Norwegian University of Science and Technology
Department of Electronics and Telecommunications

Abstract

This thesis will show combinatorial digital design using the 65nm transistor technology operating in near/sub-threshold region. Designing a 16By9Bit adder optimized with regard to power consumption with a speed requirement of 50MHz per operation for micro-beamforming. To optimize the addition of the 16, 9 bit numbers, studies of different building block are performed to find the best building blocks optimized for low power consumption, robustness and regular layout design without breaking the speed requirement. A new digital building block for standard digital building blocks optimized for subthreshold performance are proposed. In addition there will be shown a way to make regular layout designs.

As a final result there will be shown a 16by9bit adder layout design with a delay equal to 17.7nS = 56.5MHz with a power consumption of 25μW at 20°C and delay equal to 10nS = 100MHz with a power consumption of 36.2μW at 80°C. The design are build up from 6736 transistor and uses a area of 240μm * 84μm = 20.1mm^2.

Preface

This thesis is the final part of the master of science degree in electronics within the field of circuit and system design at NTNU(Norwegian University of Science and Technology) at the Department of Electronics and Telecommunication.

Working with this project has been very interesting and given me a grate insight in IC design, designing circuits for sub/near-threshold operations and the challenges by doing circuit layouts.

I want to thank my supervisor Snorre Aunet, and co-supervisor Trond Ytterdal for valuable technical support and guidance trough the master thesis. As well i want to thank the other master students at the study room Joacim, Magne and Jonathan. At last i want to thank my family that has supported me trough my 5 years as a student.

Contents

List of Figures

List of Tables

1 Introduction

Arithmetic operations play an important part in most VLSI applications, and one of the commonly used arithmetic operations is the adder. Upgrading the adder performance will have a great impact on the circuit performance. And with a increased demand for battery operated applications the need for power efficient design has grown significantly[1]. Scaling the threshold voltage downto the sub-threshold and nearthreshold region is a method to achieve low power solutions, but it comes at the cost of slower operating speed and increased sensitivity due to process variation[2].

1.1 Motivation

The motivation for this project is to make a 16 times 9 Bits adder used for ultra-sound beamforming used in a probe fore image views. This is further described in section 2. The additions have a time requirement of 50Mhz in room temperature and above. When it comes to power consumption the goal is to uses as little power as possible where the final goal of power usage is set to be bellow 50μW. The adder are named 16by9bit adder in the thesis.

1.2 Previous Work

There is not found any previous result for a 16 times 9 bits addition in the sub/near-threshold region. It is presented a adder that sums 128 numbers for micro-beamforming, made in VHDL and synthesized by the synopsis tool for 0.18μm CMOS logic, then to show a result for a 10bit adder to have a circuit delay equal to 500MHz [3]. But the paper does not present anything about supply voltage or power consumption. It is shown some previous work implementing a micro-beamformer in probes[4] where there are presented some numbers for power consumption, this is further described in section 2.

1.3 Overview of the Thesis

- Section 1 gives a short introduction of the thesis, the motivation and previous work.
- Section 2 present a more complex description of the problem description and present how this has been done earlier.
- Section 3 gives a description of the theory behind sub/near-threshold operations and some basic design challenges.

- Section 4 present the basic building blocks design for sub/near threshold operations.
- Section 5 is a short study of a new 8T implementation, where comparison of 2-input NAND implementations is performed.
- Section 6 shows testes of threshold voltages, comparison of transistor types and comparison of adder topologies to find the best implementation for the 16by9bit adder. The method for layout will as well be shown in this section.
- Section 7 present the simulation and results from the 16by9bit adder.
- Section 8 is a discussion of the results in section 7.
- Section 9 state the conclusion of the thesis.

  There are as well three appendix chapters:

- Appendix A.1 present the verilogA code used for the circuit verification.
- Appendix A.2 presents the circuit layout designs.
- Appendix A.3 will present some schematic circuits.

2 Ultrasound Beamforming

Beamforming is a process that are used in combination with an array of sensors to give a flexible form of spatial filtering. The beamform sends out a echo signal and the purpose is to form a beam of the returning echo signal summed together to form a strong echo signal of the point of interest [5].

The beamforming will be used in a probe that should obtain image from inside the human body, and one the mayor issues is the large numbers of signals that had to be transmitted to the external imaging system. Connecting each signal element with separate cables will not be possible, and applying micro-beamforming inside the probe will reduce the channel count while maintaining valuable information[6].

One way to handle this receive signal is to to use the sub-array beamforming architecture[4]. The signal processing chain is divided into a front-end realized in the probe tip and a back-end that is implemented in an external image system as seen in figure 1.

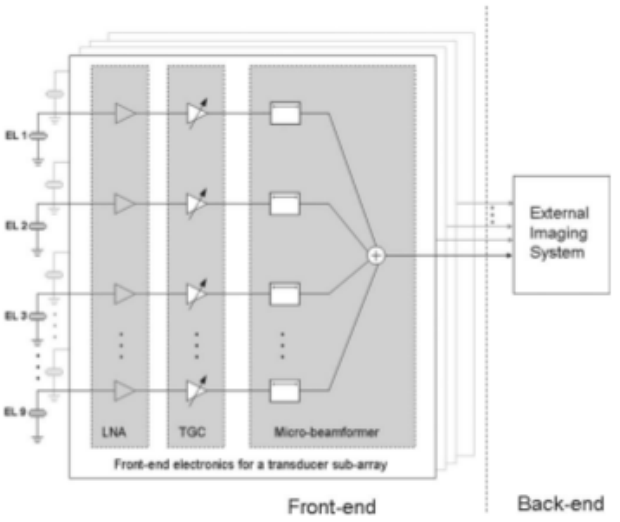


Figure 1: Signal Process Architecture
[4]

As seen in figure 1 the electronic inside the probe consist of three function blocks, low-noise amplifier(LNA), TGC amplifiers and a micro-beamformer circuit. The micro-beamformer is applied to align the signals for the elements and

then add them. This beamform addition is done using a analog design using voltage to current converters then to sum the signals in the current domain[4]. In this project the purpose is to do this summation with a combinatorial digital adder operating in sub/near-threshold to achieve savings in the power consumption with a speed equal to and higher than 50MHz. The power usage for a anlaog implementation of the micro-beamforming is previously shown to use 480μW with a step size of 40nS = $\frac{1}{40nS}$ = 25MHz[4].

3 Sub/Near-threshold Operations and Basic Design Challenges

In this section some basic theory about near/sub-threshold operations and some theory for the understanding of these operations behaviour will be described. Tools that are used for simulation and verification of the building blocks will also be described in this section. All the schematic designs, layout designs and simulations are performed in the Cadence Virtuoso platform.

3.1 Subthreshold Operation

Subthreshold design has emerged as a method to achieve low power consumption for digital circuits where speed is of second concern. Operating circuits in weak inversion at supply voltages bellow the transistor's threshold voltage V_T provides considerable energy savings at the cost of slower operating speed and increased sensitivity toward process variation. [2].

3.2 Nearthreshold Operation

Nearthreshold also called moderate inversion is the point between the weak-inversion (subthreshold) and strong-inversion (super-threshold). The behaviour of the transistor does not jump directly from the exponential behaviour of subthreshold to the quadratic behaviour of super threshold. There is a smooth transition between the two, where neither effect is dominant called the nearthreshold region. The behaviour in this area can be understood as a cross between weak- and strong-inversion[7].

3.3 The 65nm Technology Library

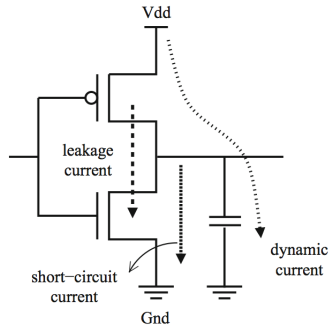
This project uses two transistors from a 65nm library. The lvtgp and the svtgp transistors both are general purpose transistors but they behaves different because they have different threshold voltages (V_t). The svtgp is the transistor that uses standard V_t voltage, while the lvtgp have a lower V_t voltage. Lower threshold voltage will increase the speed but it comes at the cost of additional power consumption. The transistor is further tested and bench-marked in section 6.1 and section 6.3.

3.4 CMOS Power Consumption

The power consumed by a CMOS transistor based circuits can be divided into three different sources: dynamic, static and short circuit power consumption. Static power consumption is also described as leakage power consumption. The total power consumption in a CMOS circuit can be calculated from equation 1 [8].

$$P_{Total} = P_{Dynamic} + P_{leakage} + P_{shortcircuit} \tag{1}$$

In figure 2 the three components of power consumption are shown.



[8]

Figure 2: Dynamic, leakage and short circuit power cunsumption in a basic inverter

Further down in this section these three types of power consumption will be described.

3.4.1 Dynamic Power Consumption

The Dynamic power consumption occur when the logic state is changing (switching). The energy is drawn from the power supply to load the output capacitance. This power consumption due to switching activity is given in equation 2.

$$P = \alpha * f_{clk} * V_{dd}^2 * C[1] \tag{2}$$

Where f_{clk} is the clock frequency, V_{dd} is the supply voltage, α is the switching activity and C is the capacitive load. Reducing any of this factors leads to reduction in the dynamic power consumption. This equation is given for static

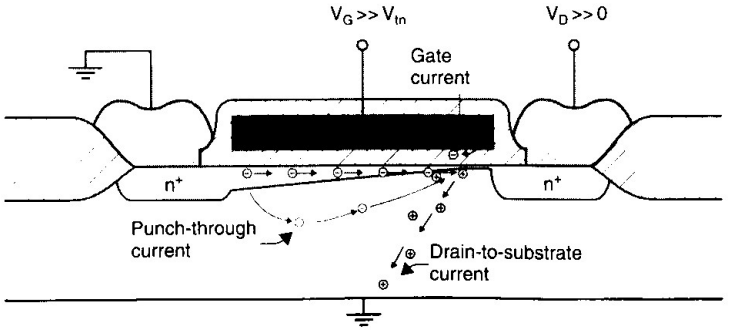
circuits with a system clock, but in this project all circuits are combinational circuits with no clock. Hence equation 2 can be adjust simply by using the gate input frequency instead of the clock frequency.

3.4.2 Subthreshold Static Power Consumption

The main contribution to leakage is the subthreshold current between the drain and source [9]. The power consumption due to static power consumption are shown from equation 3.

$$P = V_{dd} * I_{lsub}[1] \tag{3}$$

The source for current leakage in the subthreshold region mainly comes from three sources. The weak inversion effect: when the gate voltage is below V_T, carriers move by diffusion along the surface. This effect becomes significant when the supply voltage is smaller then, and close to the threshold voltage [8]. The Drain-Included Barrier Lowering(DIBL): The reduction of threshold voltage of the transistor at higher drain voltages. The DIBL effect is enhanced at shorter effective channel length and higher drain voltage [8]. The direct punch-through current: This is the punch-trough of the electrons between drain and source. When the drain and source depletion regions approach each other and electrically touch deep in the channel as seen in figure 3. This effect can occur as a result of the DIBL as well[8].



[10]

Figure 3: Punch trough illustration

Equation 4 shows the basic equation for modelling subthreshold current.

$$I_{D:sub-threshold} = I_o * exp(\frac{V_{GS} - V_T}{n * U_T}) \tag{4}$$

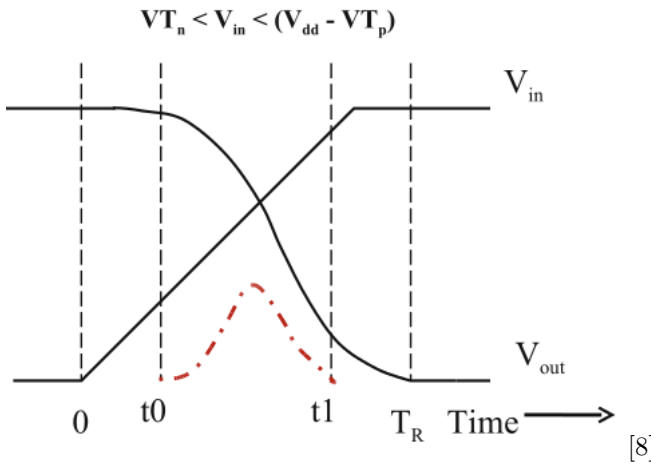
Where I_o er equal to:

$$I_o = \mu_o * C_{ox} * \frac{W}{L} * (n-1) * V_{th}^2 \tag{5}$$

where U_T is the thermal voltage $\frac{kT}{q}$, n is the subthreshold slope factor($n = 1+\frac{C_d}{C_{ox}}$), V_T is the transistor threshold voltage, μ is the carrier mobility, C_{ox} is the oxide capacitance and W and L are the effective transistor width and lengths[11].

3.4.3 Short Circuit Power

In digital circuits there is always a short time where pull-up and pull-down paths of a CMOS gate are one simultaneously, thus creating a parasitic current that is wasted as illustrated in figure 4. And this additional power consumption is called short-circuit power. This effect can account for 10 percent of the dynamic power consumption[12] depending on the technology that are used.



[8]

Figure 4: Short circuit power illustration

3.5 Delay

Propagation delay is is often described by equation 6 in subthreshold:

$$t_d = \frac{C_L * V_{DD} * K}{I_o * exp(\frac{V_{DD}-V_T}{n*U_T})} \tag{6}$$

Where k is a constant, V_{DD} is the supply voltage, V_T is the transistor threshold voltage and U_T is the thermal voltage [11]. From equation 6 it is shown that the delay is strongly affected by the supply voltage, when scaling down the supply voltage it is seen from equation 2 and equation 3 that the power consumption are decreased but this leads to a increase in the propagation delay. Moreover the delay is effected by the slew rate, the equation for the slew rate are give in equation 7 [10].

$$SR = \frac{I_D}{C_L} \tag{7}$$

Where I_d is given by equation 8 in weak inversion.

$$I_D = \mu * C_{ox} * \frac{W}{L} * e^{-\frac{V_{T0}}{nU_T}} \tag{8}$$

Where μ is the mobility of electrons in the channel, C_{ox} is the gate capacitance per unit area, W is the gate width, L is the gate length, B_{T0} is the gate threshold voltage for the channel at equilibrium and U_T is the thermal voltage [13].

3.6 Power Delay Product and Energy Delay Product

Power delay product(PDP) is a measurement that describe the relation between power and delay and are estimated from equation 9 [14].

$$PDP = Power * DELAY = Energy \tag{9}$$

Energy delay product(EDP) is a measurement that then describe the relationship between the energy and the delay and are estimated from equation 10 [14].

$$EDP = PDP * Delay \tag{10}$$

As seen from equation 9 and equation 10 both EDP and PDP can be calculated from the measured delay and power consumption.

3.7 PMOS / NMOS Imbalance and Transistor Sizing

Imbalance between NMOS and PMOS often occur due to different threshold voltage between the P- and N- MOS transistors which can give large strength difference. Imbalance between the NMOS and PMOS can then lead to Noise Margin and $V_{dd,min}$ degradation and a increase in leakage energy [15]. The

transistor sizing will as well have a great impact on the circuit performance. Gate widths and lengths will affect both the subthreshold current and the threshold voltage, meaning that finding good transistor lengths and widths can optimise the circuit performance for the circuit specifications.

3.8 Process Variation and Robustness

Process variation is a production error, as CMOS technology scale down there is more exposed to process variation. The major factors leading to process variation are wafer misalignment, random doping fluctuations, and imperfections in planarizarion steps [16]. Different transistors at a chip can get a variation in critical process parameters such as threshold voltage or effective channel length and can result in fluctuations in the switching speed and leakage power consumption [16]. The MOS transistor's in the subthreshold region is extra sensitive to both temperature and process variation. That because the variation in the threshold voltage will affect the subthreshold current exponentially[17].

3.9 Temperature Variations

For higher temperatures the effective threshold voltage and the mobility factor μ_o both decreases as seen from equation 11.

$$\begin{aligned} \mu(T) &= \mu(T_0) * (\frac{T}{T_0})^{-M} \\ V_T(T_0) &= K * T \end{aligned} \tag{11}$$

In strong inversion the lower mobility will dominate and lead to slower circuits at high temperatures. In the subthreshold region the lower V_T dominates, and hot carriers grows faster exponentially. This lead to raised leakage currents and to decreased circuits delay at high temperatures [11].

3.10 Monte Carlo Simulation

To check the circuits robustness to process variation (described in section 3.8) and mismatch the Monte Carlo simulations are used. The Monte Carlo simulations perform risk analyses by building models of possible results by randomly changing the transistor parameters. The cadence Monte Carlo simulators that are used in this project allow random variation of process parameters, mismatch parameters and process & mismatch parameters. This is done because wafer production always gives some variations of the technology parameters and the Monte Carlo simulations is used as a tool to estimate the effect of these variations. The

Monte Carlo simulations will as a result present the mean value of the simulations together with a sigma(σ) value, where this sigma value is the standard deviation. Typical 30 to 50 Monte Carlo runs are enough to get meaningful statistics [18].

3.11 Parasitic Effects

When doing layout simulations the circuit performance changes because of the parasitic extraction. This parasitic extraction is necessary because not all of the electrical parameters can be considered during schematic analyses. The parasitic extraction includes capacitance, resistance and inductance from wiring. The parasitic parameters can account for 70% or more of the circuit delays depending on the technology used. For near/sub-threshold circuits the delay caused by wiring is of great importance, and today four main parasitic extractions of layout design are used: one, two and three-dimensional and extraction of dynamic capacitance effects. In this project the three dimensional extraction is used because it consider the capacitance between neighbouring wires, as well as capacitance by different levels of metallization. This level of accuracy comes of the cost of CPU time and main memory requirement but it is as well the most exact parasitic extraction[19].

3.12 Verilog-A

Verilog-A is a hardware description Language (HDL) and is derived from the IEEE 1364 Verilog HDL specification. The intention of Verilog-A is to let designers make high performance modules, that describe the module behaviour mathematically in terms of the external parameters applied to the module[20].

4 Basic Building Block Implemantation for Sub/Near-Threshold Operations

In this chapter all the building blocks used in the project will be presented. All basic building blocks uses stacked transistors because this is shown to increase the robustness for minority-3 gates [21]. For the other standard building blocks a new stacked 8 transistor implementation is introduced in this section and further tested in section 5. The stacked transistors are as well shown to reduce the subthreshold leakage current [22]. Bringing the standard of stacked transistors into all the building blocks will as well contribute to a more regular layout designs. The layouts are presented in the appendix A.2 and will be referred to in the different sections. The way of doing the layouts are further explained in section 6.4.

4.1 Basic Building Block

In this section the basic building blocks used for larger designs will be presented.

4.1.1 Minority-3 Gate

The minority-3 gate is a basic digital building block that has 3 binary inputs and one binary output. As a result of two or more logic 1 at the input the output will be zero. And if there are one or less digital 1 at the input the result will be a logical 1 at the output, as seen in table 1.

X	Y	Z	OUT
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Table 1: Truth Table Minority-3

The minority-3 gate can be a base for many different building blocks and can depending on their input wiring, implement either NAND, NOR, Inverter or Inverter Carry[21]. If an extra inverter is available and set on the output the AND, OR and CARRY gates can be implemented as well. There are various

alternatives of how the minority-3 gates are implemented and each of them have their advantages. In this project a 10T implementation of the minority-3 gate will be used as seen in figure 5.

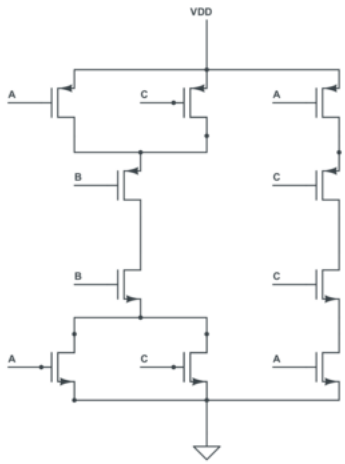


Figure 5: 10T Minority-3

Previous studies have looked at different minority-3 topologies[21]. Their results show that a 10T implementation used the smallest amount of power consumption and that a 22T implementation has the best robustness potential. In addition the 10T implementation is shown to, in most cases dominate the 6T implementation on robustness and the 12T implementation on circuit delay[21]. The minority 3 layout is shown in figure 43.

4.1.2 4T Inverter

The inverter is a very simple logic gate with one logic input and one logic output where the logical input are inverted to the output. The inverter normally consist of one NMOS and one PMOS gate, but in this project the robustness can be a issue, therefore the inverter implementation used in this project is based on a 4T inverter as seen in figure 6. The 4T implementation will as well contribute to the regular layout with stacked transistors.

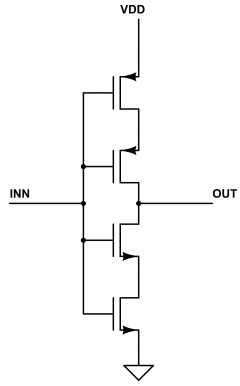


Figure 6: 4T inverter

The layout of the 4T inverter are shown in figure 42.

4.1.3 8T Implementation

There are several ways to implement basic building blocks for digital design, in subthreshold circuits the minority-3 gate are in some cases chosen to get more robust circuits. This because it has been reported to be a more reliable design compared to implementation based on boolean logic [23]. As an alternative to these two implementations a new standard implementation has been made. This new implementation is called an 8T (8 Transistor) implementation, after an idea by Snorre Aunet and Jonathan Bjerkedok.

By making the 8T design the idea is to keep the basic building block for all part of the design. In the same way as minority-3 it are made to implement blocks like NAND, NOR, INVERTER, XOR, AND and OR only by changing the wiring and using inverted signals. The 8T implementation take the design from the basic boolean NAND and NOR and add transistors either by stacking or setting two extra in parallel to make a basic building block as seen in figure 7. This is done to make a robust design of the basic boolean building blocks that shod outperform the minority-3 gate on delay times and the boolean gates on robustness. The implementation of a 8T NAND is further tested compared to boolean NAND and a minority-3 NAND in section 5.

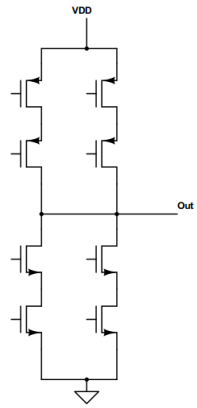


Figure 7: 8T implementation

4.1.4 XOR Gate

In figure 8 a implementation of a Xor gate are shown using a standard Xor implementation because this implementation already consist of stacked transistors, and the 4T inverter described in section 4.1.2. This way of design is chosen to keep the regular design with stacked transistor. The Xor gate layout are shown in figure 44.

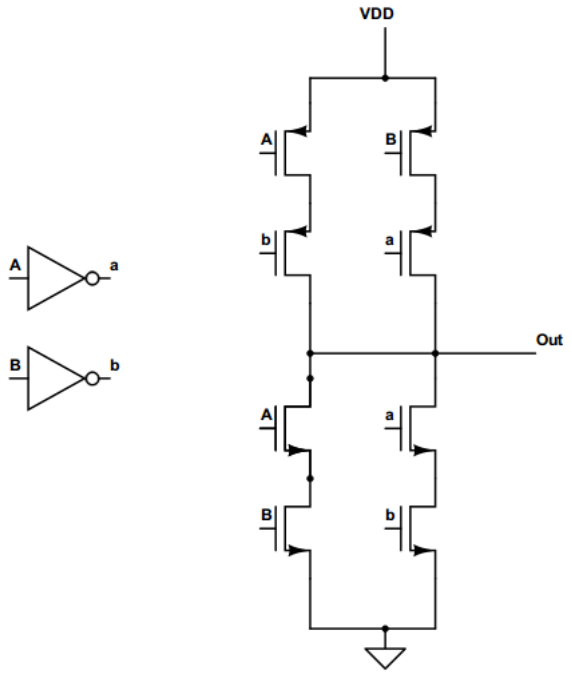


Figure 8: XOR

4.2 Adder Implementations

In this section the different adder are shown, this section start by showing the implementation of a half- and full Adder. Then the Ripple-Carry adder and Kogge-Stone adder are presented follow-up by a 16by9bit adder implementation.

4.2.1 Half Adder

The half adder is a combinational circuit that add two inputs, that provide two outputs sum and carry[24]. The half adder is build from the Xor and 8T And gates seen in section 4.1.4 and section 4.1.3 as shown in figure 9. The 8T And uses the 8T implementation of a NAND gate as seen in figure 17b and a 4T inverter to create the 8T AND gate.

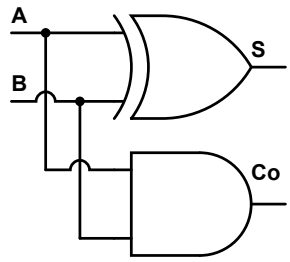


Figure 9: Half Adder

The Half Adder truth table is shown in table 2.

Table 2: Half Adder Truth Table

A	B	Sum	C_o
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

The half adder delays and power consumption can roughly be calculated as seen in equation 12.

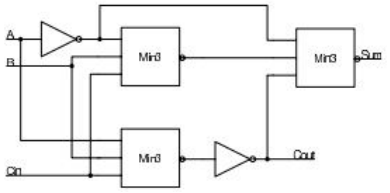
$$\begin{aligned} Sum &= XOR_{Delay} \\ C_o &= AND_{Delay} \\ Power &= XOR_{Power} + AND_{Power} \end{aligned} \tag{12}$$

This type of adder is used for adder stages that not receive any carry input. Using the 8T implementation technique this implementation will consist of 28 transistors. The layout for the half adder are shown in figure 45.

4.2.2 Full Adder

A full-adder is one of the basic blocks in digital design. The Full-Adder is a combinational circuit with three inputs A, B and a carry (C_{in}) and two outputs

Sum and carry out (C_{out}) [25]. This combinational circuit is based on minority-3 gates, consisting of three minority-3 gates and two inverters as seen in figure 10[1].



[1]

Figure 10: Minoruty-3 based Full Adder

The Full Adder truth table is shown in table 3.

Table 3: Full Adder Truth Table

A	B	C_i	Sum	C_o
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

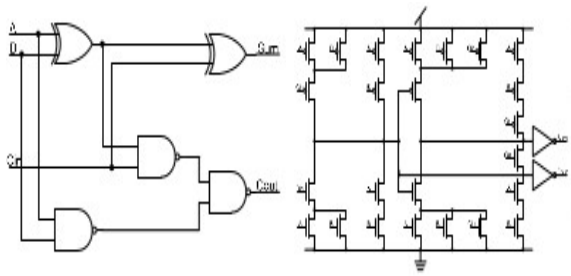
The full adder delays and power consumption can roughly be calculated as seen in equation 13.

$$\begin{aligned} Sum &= 2 * miority - 3_{Delay} + inverter_{Delay} \\ C_o &= minority - 3_{Delat} + inverter_{Delay} \\ Power &= 3 * minority - 3_{Power} + 2 * inverter_{Power} \end{aligned} \tag{13}$$

This type of adder is different from the half adder because it handles a carry input. But this adder is bigger and consists of 38 transistors in schematic design,

in addition 6 dummy transistors are used to create a regular layout and the transistor count raise to 44 transistors in layout. The full adder layout are shown in figure 46.

This design is chosen for the full adder because this implementation is proved to be as fast and faster than the standard CMOS implementation and direct synthesis implementation[26], the standard CMOS implementation and the Direct synthesis implementation are shown in figure 11 .In addition the minority-3 implementation provide stacked transistors that lead to more robust and regular design than the standard CMOS. One possibility could have been to implement a direct synthesis implementation using the 8T implementation but this would give a transistor count of 64 that will require more area and higher power consumption.



(a) Direct synthesis **(b)** Standard CMOS

Figure 11: FullAdder implementations
[26]

4.2.3 N-Bit Ripple-Carry Adder

Binary addition has previously been studied starting with the ripple-carry adder going toward parallel implementations, such as the Kogge-Stone adder. It is commonly accepted that the Ripple-Carry Adder is the slowest, while the Kogge-Stone Adder is the theoretically fastest [27]. It is shown that the 32-bit Kogge-Stone Adder is 4.5 times faster than the 32-Bit Ripple-Carry Adder not considering the wire delays [28]. While this factor is reduced to 2.2-2.4 taking wire delays into account. The number of layer the signals need to go trough can be calculated to be [27]:

$$Layers_{RCA} = n \tag{14}$$

$$Layers_{KS} = 2 + log_2(n) \tag{15}$$

Where n is the number of bit, RCA is short for the ripple carry adder and KS is short for the kogge-stone adder.

The ripple-carry adder are build from the 10T minority-3 gates 4.1.1 in combination with the 4T inverters4.1.2. Making FullAdders 4.2.2 put in series depending in the number of bits for the adder. The delay can then be calculated from the following equation.

$$\begin{aligned} &Delay = (N-1) * Cout + SUM => \\ &(N-1)(min3_{Delay} * inverter_{Delay}) + ((2 * min3_{Delay}) * inverter_{Delay}) \end{aligned} \tag{16}$$

The schematic of a 8 Bit ripple-carry adder is shown in appendix A.3.

4.2.4 N-Bit Kogge-Stone Adder

The Kogge-Stone adder consist of three basic stages [29]. Bitwise PG Logic, Group PG Logic and Sum logic as seen in figure 12.

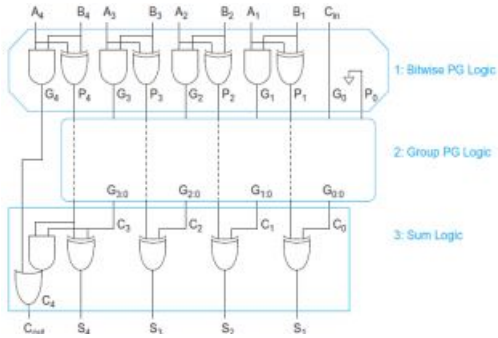


Figure 12: Adder schematic
[29]

The group PG stage is build on black and gray building blocks as seen in figure 13.

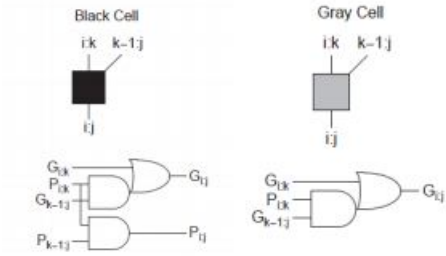


Figure 13: Black and Gray Building Blocks
[29]

A schematic of the Kogge-Stone adder are shown in figure14.

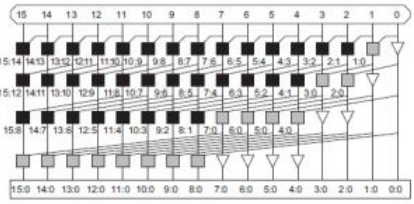


Figure 14: Black and Gray Building Blocks
[29]

From equation 14 a 8 Bit Kogge-stone adder will have five layers of blocks to propagate trough. These five layers will consist of one start block from the Bitwise PG Logic, three black cells and one cell from the sum Logic. Giving the following calculation for the 8-Bit Kogge-Stone delay seen in equation 17.

$$Delay = StartBlock_{Delay} + 3 * blackCell_{Delay} + SumBlock_{Delay} =>$$
$$Delay = XOR_{Delay} + 3(AND_{Delay} + OR_{Delay}) + (AND_{Delay} + OR_{Delay}) \tag{17}$$

The kogge-stone adder logic gates are not made from the minority-3 gate although that is possible, but from some the new 8T implementation of NAND and NOR gates because they are proven to work better than the minority 3 basic gates as seen in section 5. The schematic of a 8 Bit kogge-stone adder is shown in appendix A.3.

4.2.5 16By9Bit Adder Design

The 16by9Bit adders purpose is to add 16 9bit numbers and are shown in figure 15. This adder consist of eight 9 Bit Adders (stage 1) followed up by four 10 Bit Adders (stage 2), two 11 Bit Adders (stage 3) and one 12Bit Adder (stage 4) working in parallel as seen in figure 15. Each adder stage propagate the result to the next adder with the carry as the highest bit, meaning that the adder increasing with one bit for each stage. This is done to avoid overflow and the result from 16, 9 Bit numbers added together will give a 13 Bit answer. By doing the addition this way there will not flow any carry into the adder stages and the 9,10,11 and 12 bit adders can be made without a carry input, how to do this is described in section 6.6.1.

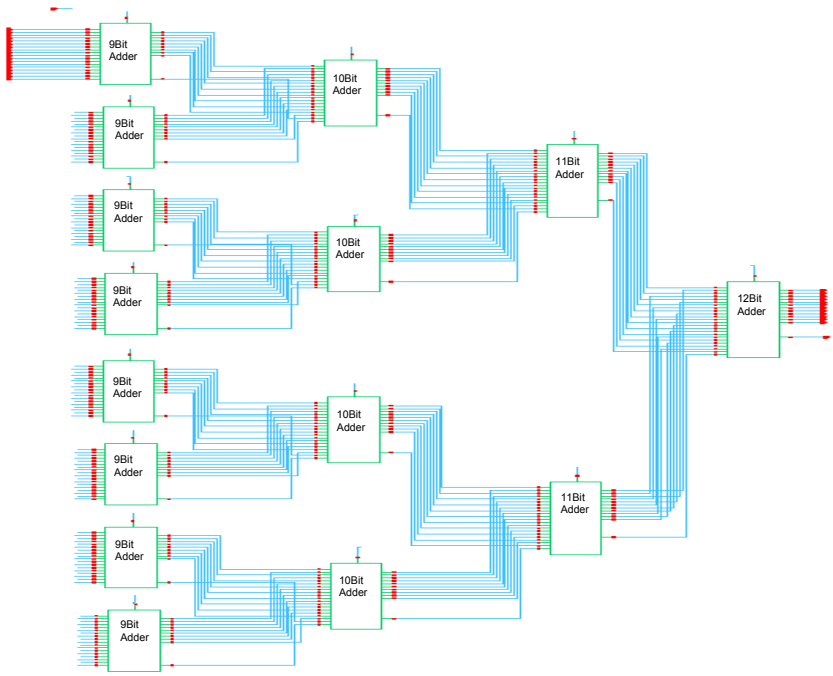


Figure 15: 16 By 9 Bit Adder

The layout for the 16by9bit adder are shown in figure 48.

The propagation out from each stage in the 16by9bit adder are shown as an example in figure 16, all 16 9 Bit inputs are in this example are set to toggle between zero and one, and shows the output of each adder stage. The delay is the working time before the signal settle and the delay time increase with the numbers of stages the signal has to propagate trough. Stage one are represented by S_1, stage two by S_2, stage three by S_3 and stage four by S_4.

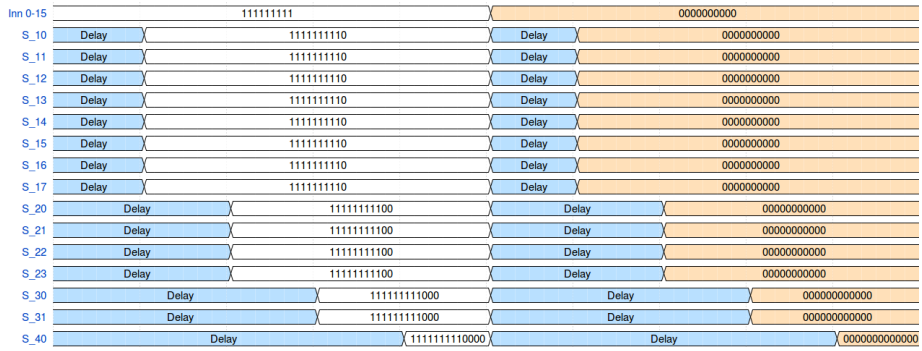
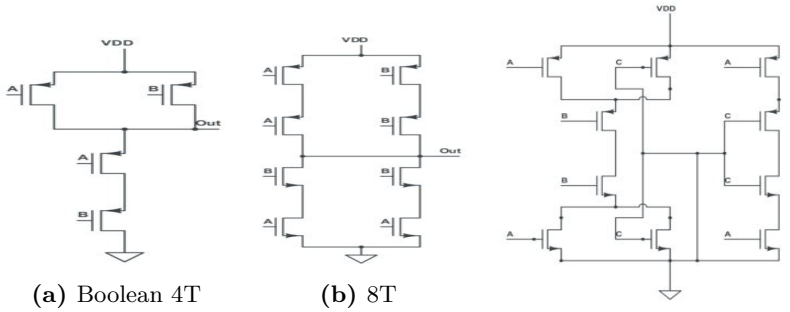


Figure 16: Propagation of the signal in the 16 By 9 Bit Adder

5 Comparison of 2-input NAND for Subthreshold Operations

In this section the study of three different ways making basic building blocks for subthreshold operations are shown. A 2-input NAND implementation using Minority-3, standard boolean implementation and the 8T implementation described in section 4.1.3 are tested looking at delay, power consumption, PDP, leakage and robustness. The implementations of the NAND gates are shown in figure 17.



(a) Boolean 4T **(b)** 8T **(c)** Minority 3

Figure 17: NAND Implementations

5.1 Balancing the Gates

For testing of the 2-input NAND gates the svtgp transistors from the 65nm library is chosen using a standard gate length equal to 90nm, because it is shown to give the best compromise between good circuit performance and low sensitivity toward process variation[2]. The threshold voltages for the p- and nMos transistors are then measured to be 318mV and 361mV respectively. The circuits are then balanced for 250mV supply voltage(V_{DD}) by setting $\frac{V_{DD}}{2}$ at the inputs then to tune the transistor widths until the output is $\frac{V_{DD}}{2}$.

For a NAND gate there will be two ways of sizing the transistors, either by putting both inputs at $\frac{V_{DD}}{2}$ or by setting one input to V_{DD} and the other to $\frac{V_{DD}}{2}$ then to tune the transistor widths. All the implementations are balanced with both methods then the gates are run through the test-bench described in section 5.2 to check which way of balancing that gives the best power delay product(PDP).

5.2 Test-Bench

The test-bench takes the 2-input NAND implementations and put three of them in series, and connect them as a ring oscillator. Then by getting the output to toggle the delays, max power consumption and power delay product can be measured. This because the oscillator will work at maximum speed with each NAND-gate having two NAND-gates as input/output load. The output of the NAND gate will toggle either when one input toggle and the other input is set to V_{DD} or when both inputs are set to toggle. The result shows the cases where the worst PDP is found. The static leakage is tested by setting static inputs to the oscillator, then there will not be any switching by the transistors and the static leakage can be measured.

To check the robustness due to process variation and mismatch 200 Monte-Carlo runs are used for each simulation, this because 50 Monte Carlo runs are needed to get meaningful statistics as described in section 3.10 and 200 runs was used to be shore to get enough data. The simulation times is still short when running 200 Monte Carlo runs for this circuits. Then to check the robustness towards temperature variations, the gates are simulated at -40 degrees, 20 degrees and 80 degrees. The simulations are only done in schematic not taking parasitics from wiring into account.

5.3 Results

Here the results for the comparison of the 2-input NAND implementations will be shown.

5.3.1 Dimension Results

In table 4 the gate length and widths that are found after balancing the circuits are shown.

Table 4: Gate widths and lengths after balancing

Gate implementation	P_{length}	N_{length}	P_{width}	N_{width}
Boolean(4T)	90nm	90nm	290nm	300nm
8t	90nm	90nm	339nm	300nm
Minority-3	90nm	90nm	372nm	300nm

5.3.2 Delay Results

In figure 18 the mean delay after 200 Monte-Carlo runs are shown together with the % deviation measured by one sigma(σ).

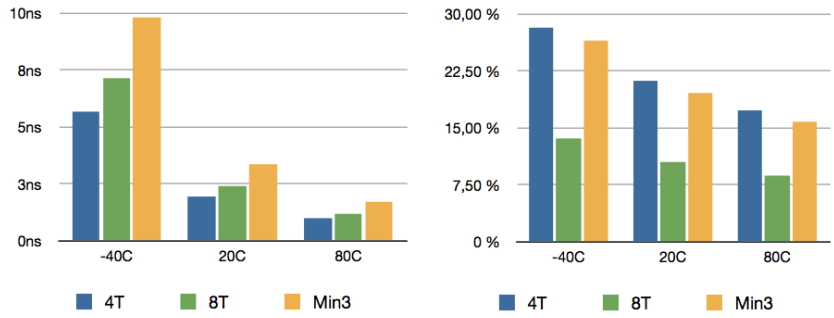


Figure 18: Delay at different temperatures.

In table 5 the % deviation due to temperature variation is shown.

Table 5: Delay reduction going from -40C to 80C.

Topology	Reduction
4T	82.54%
8T	83.47%
Min3	82.57%

5.3.3 Power Consumption Result

In figure 19 the mean power consumption after 200 Monte-Carlo runs are shown together with the % deviation measured by one sigma(σ).

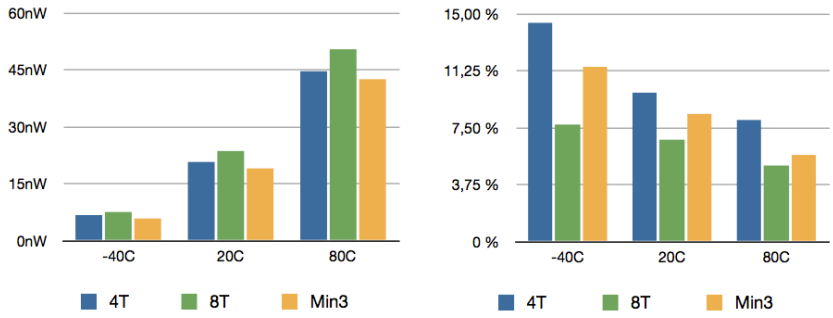


Figure 19: Power consumption at different temperatures.

In table 6 the % deviation due to temperature variation is shown.

Table 6: Power increasing when going from -40 to 80C.

Topology	Increasing
4T	665.67%
8T	671.11%
Min3	732.76%

5.3.4 Power Delay Product Results

In figure 20 the mean power delay product(PDP) after 200 Monte-Carlo runs are shown together with the % deviation measured by one sigma(σ).

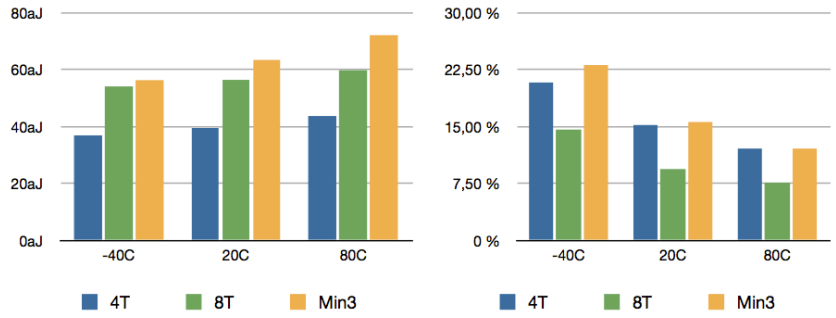


Figure 20: PDP at different temperatures.

In table 7 the % deviation due to temperature variation is shown.

Table 7: Power delay product increasing when going from -40 to 80C.

Topology	Increasing
4T	118.43%
8T	110.54%
Min3	128.06%

5.3.5 Static Leakage Results

In figure 21 the mean static leakage after 200 Monte-Carlo runs are shown together with the % deviation measured by one sigma(σ).

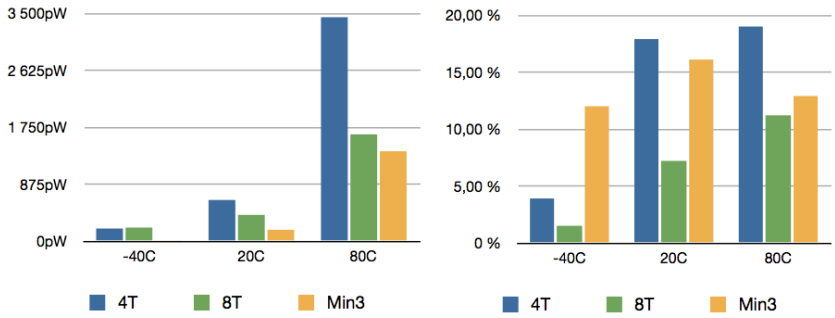


Figure 21: Leakage at different temperatures.

In table 8 the % deviation due to temperature variation is shown.

Table 8: Leakage power increasing when going from -40 to 80C.

Topology	Increasing
4T	1791%
8T	792%
Min3	11111%

5.4 Quantification

In table 9 the different NAND implementations are quantified by rating each implementation with regard to delay, power, pdp and leakage. The rating are given from one to three stars where three stars are is the best and one star is the worst.

Table 9: NAND implementation quantification

Measurements	Boolean NAND	8T NAND	minority-3 NAND
$Delay$	★★★	★★	★
$Delay_{Robustness}$	★	★★★	★★
$Power$	★★	★	★★★
$Power_{Robustness}$	★	★★★	★★
PDP	★★★	★★	★
$PDP_{Robustness}$	★★	★★★	★
$Leakage$	★	★★	★★★
$Leakage_{Robustness}$	★	★★★	★★

5.4.1 Delay

It is shown in figure 18 that the boolean implementation is the fastest while the minority-3 is the slowest while the 8T implementation delay is between the two other. When looking at robustness it is shown that the 8T implementation is far better than both the minority-3 and the boolean implementation. Looking at the robustness to temperature variation there is no big difference between the topologies as seen in table 5.

5.4.2 Power Consumption

From figure 19 It is shown that the 8T implementation have the highest power consumption while the minority-3 have the lowest power consumption. But when looking at power there is just minor differences, it is only 4nW between the 8T and the minority-3 power consumption at 20 degrees. But it is yet again shown that the 8T implementation is more robust than the two other implementations. And as with the delay the deviation due to temperature variation seems to be close to equal as seen from table 6.

5.4.3 Power Delay Product

Figure 20 shows that the boolean 4T implementation have the best power delay product while the minority-3 gate have the worst power delay product. It is again shown that the 8t implementation is the most robust looking at the PDP. And as seen in table 7 the deviation due to temperature is miner between the different implementation.

5.4.4 Static Leakage

Looking at the static leakage in figure 21 it is shown that the minority-3 implementation consumes least power while the boolean implementation is shown to be the worst. Again the 8T implementation is the most robust. But when looking at static leakage the different temperatures have different effects on the implementations as seen from table 8.

5.5 Discussion

In subthreshold design robustness is a issue and by this mini study it is shown that the 8T implementation contributes to a much more robust design than the standard boolean and the minority-3 implementation of standard digital building blocks like the NAND gate. The minority-3 gate is proven to be a bit more robust than a standard implementation of a NAND gate but it is much slower. Therefore a it is not recommend to use the minority-3 gates for basic building block designs like NAND, NOR ,AND etc. Still the minority 3 gate will be useful in other design like the Full-Adder as seen from section 4.2.2. In super-threshold the boolean implementation of the NAND gate should still be the best alternative because in super-threshold the robustness is not that big of a issue as in subthreshold. In subthreshold the 8T implementation of basic digital building blocks seems to be the better alternative because of the improved robustness without increasing the delay as much as the minority-3 gate, compared to the standard boolean implementation. The 8T implementation will as well contribute to much more regular layout design than the boolean implementation.

6 Method

In this section testes of threshold voltages, comparison of transistor types and comparison of adder topologies are shown to find the best implementation for the 16by9Bit adder. A regular method for layout design are shown and the test-benches for the 16By9Bit adder design are described.

6.1 Threshold Voltages

It is important to know the transistor used for the design, therefore some tests are done to check the behaviour of the lvtgp and svtgp transistors threshold voltages, the threshold voltage is important for the transistor choice as well. The change in threshold voltages is tested as a factor of transistor lengths, and as a factor of nWell doping in the layout design.

6.1.1 Threshold Voltage Test-Bench

The test-bench used to check the gate length effect on the threshold voltage uses lvtgp and svtgp transistors implemented as 4T inverters balanced for 200mV. The supply voltage V_{DD} is set to 200mV, there are no additional load on the test-bench and the input pulse is generated from a ideal voltage source, then to measure the DC operation points. The sizes are shown in table 10

Table 10: Widths and lengths for a 4T inverter balanced for 200mV gate lenght = 90nm

Transistor	P_{length}	N_{length}	P_{width}	N_{width}
lvtgp	90nm	90nm	590nm	300nm
svtgp	90nm	90nm	330nm	300nm

To check the nWell effect the same test-bench are used but this time the 4T inverter is drawn in layout and extracted with three dimensional parasitic extraction. Then the nWell is put at different distances from the p- nMos gates, then to measure the DC operation points of the transistors to check the threshold voltages.

6.1.2 Transistor Sizing Effect on the Threshold Voltage

In figure 22 the threshold voltage is shown as a factor of transistor gate lengths.

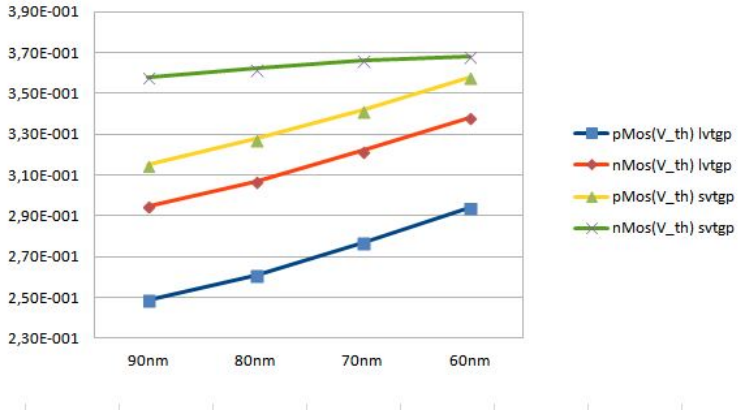


Figure 22: Gate length effect on the threshold voltage

6.1.3 nWell Sizing Effect on the Threshold Voltage

In figure 23 the threshold voltage is shown as a factor of distance from nWell edge to the transistor gate.

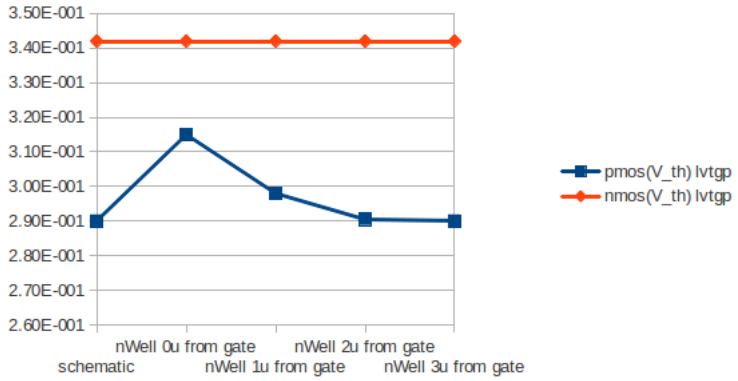


Figure 23: nWell effect on the threshold voltage

6.1.4 Threshold Voltage Summation

As shown in figure 22 the transistor gate have a great impact on the threshold voltage, smaller gate lengths gives higher threshold voltages. In figure 23 there is shown that the distance from pMos gate to nWell edge will affect the threshold voltage for a pMos transistor. But it is also shown that when the distance is bigger than 2μm it will no longer have an impact on the threshold voltage.

6.2 Transistor sizing

It is previous shown that gate length $L_p = L_n = 1.5 * min$ will give a good comparison between good circuit performance and low sensitivity toward process variation in the 65nm technology[2]. With this gate length as a base the transistor widths are adjust. To adjust the gate widths for a block $\frac{V_{dd}}{2}$ are set to the input gate and the gate widths are adjust to get $\frac{V_{dd}}{2}$ at the output. This way of sizing is used as a base for block comparison. When the blocks are chosen the sizing is change to optimize the circuit performance for the 16By9Bit Adder. In this project all blocks uses stacked transistors, leading to a proportional design where all nMos transistors are equal and all pMos transistors are equal sized.

6.3 Transistor Choice

The choice of transistor type is based on the transistor threshold voltages, power delay product, power and minimum delays. To check this effects a 10T implementation of the minority-3 gate are used and tested with both the svtgp and the lvtgp transistors for comparison.

6.3.1 Transistor Comparison Test-Bench

To test the minority-3 performances the gates are put in a ring-oscillator as seen in figure 24 to measure max speed, power usage at max speed and to calculate the power delay product (PDP).

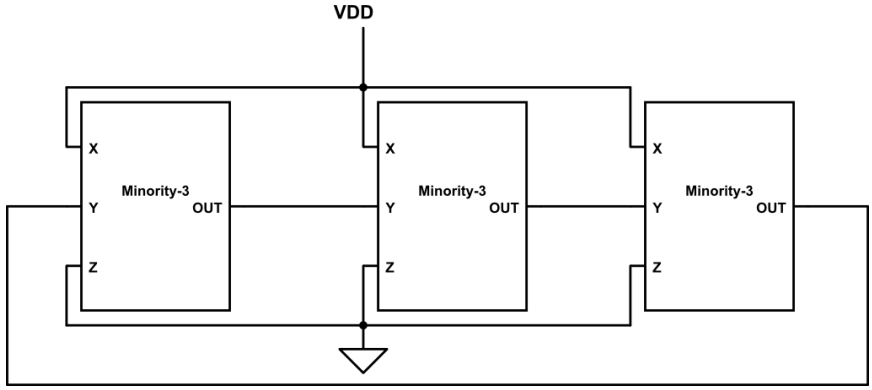


Figure 24: Minority-3 gates oscillation

The gate are balanced for 200mV resulting in the sizes seen in table 11.

Table 11: Widths and lengths balanced for 200mV gate lenght = 90nm

Transistor	P_{length}	N_{length}	P_{width}	N_{width}	pMos V_{th}	nMos V_{th}
lvtgp	90nm	90nm	590nm	300nm	244mV	294mV
svtgp	90nm	90nm	330nm	300nm	316mV	358mV

6.3.2 Transistor Comparison Result

In figure 25, figure 26 and figure 27 the delay, power and PDP is shown as a factor of the supply voltage for both the svtgp and the lvtgp implementation.

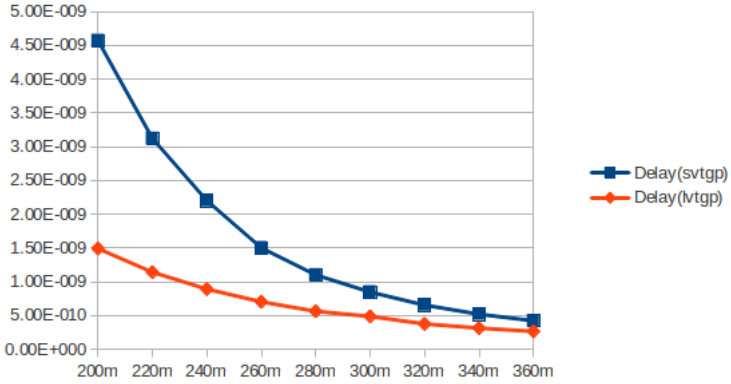


Figure 25: Minority-3 Delays

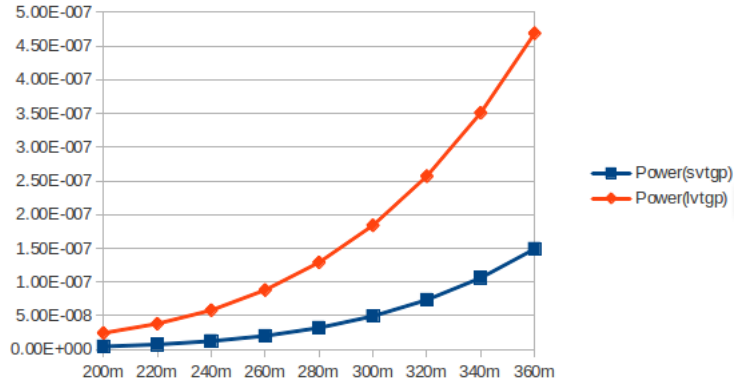


Figure 26: Minority-3 Power

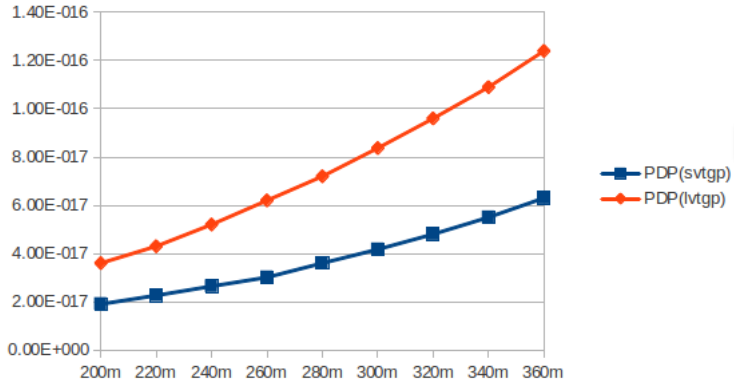


Figure 27: Minority-3 PDP

6.3.3 Summation

As seen in figure 25 the lvtgp implementation is faster than the svtgp implementation at the same supply voltage. And from figure 26 and figure 27 it is shown that the svtgp are more energy friendly at the same supply voltages. The svtgp is as well shown to give the best PDP product at set speeds. This results show that the svtgp transistors have better potential for energy saving for circuits with slower working speed, but in this project it is foreseen that time requirement can be hard to accomplish in sub/near-threshold therefore the lvtgp transistor is chosen although it comes at the cost at some more energy usage per operation.

6.4 Layout Design

In this section the a standard method for all layout designs in this thesis are shown and described.

6.4.1 65nm Design Rules

In table 12 the major 65nm design rules are shown [30].

Table 12: 65nm Design Rules

Design Rules	Minimum pitch	Line/Space	
OD(nm)	190	90/100	
PO(nm)	180	70/110	
CO(nm)	200	90/110	
M1(nm)	180	90/90	[30]
Via-x(nm)	210	100/110	
M-x(nm)	210	100/110	
PO-CO distance(nm)	210	100/110	
n+ /p+ distance(nm)	190	-	

The figure describes minimum pitch, line and spacing for different layers and vias.

6.4.2 Well Proximity Effect

The WPE (Well proximity effect) is the effect caused by substrate implant ions being reflected off the well edge leading to increased threshold voltage [31]. This effect is tested in section 6.1.3 and its shown that the WPE does not affect the threshold voltage when using nWells equal to 2μm or wider from the pMos gates.

6.4.3 Guard Rings

The pMos transistors uses nWell vias connections to V_{DD} as guard rings at the left and right hand side while the nMos use pTap vias connected to GND as guard rings at the right and left. This is to reduce small carrier disturbance, and to minimize stray electrons and stray holes from affecting the transistors[32].

6.4.4 Dummy Transistors

Dummy transistors are used to sure that each element sees the same surroundings. The pMos dummy wire all connections to V_{DD} while the nMos wire all connections to GND. The dummies are used to fill empty spaces to achieve regular design as seen by the top middle pMos and the bottom middle nMos in the minority 3 layout seen in figure 43. Dummies are as well used at the end of building blocks to give the working transistors inside the blocks equal working

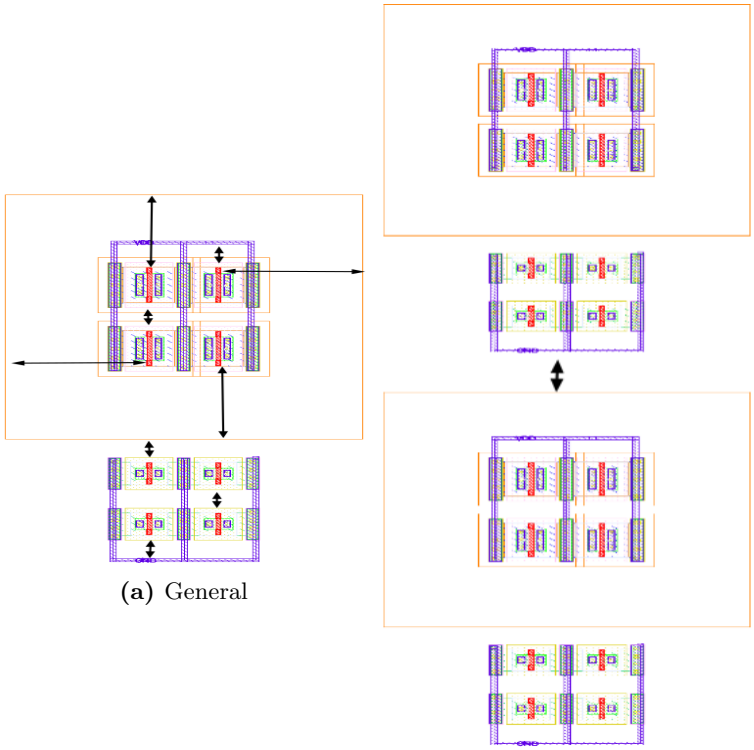
environment and additional space for wiring between the building blocks.

6.4.5 Set Poly Pitch

In this project set poly pitch are used in a single direction. Poly pinching ore rounding can contribute to mismatch errors and to increased gate leakage, especially at the gate edge [31]. Regular poly pattern with a set width require less optical proximity correction in production, the high poly density will as well reduce the poly Reactive-ion etching loading [31].

6.4.6 The Layout Outline

In figure 28a the outline used for layout design are shown. The long arrows indicated the distance from the pMos gated to the nWell edge and are equal to 2μm. The short arrows indicate the distance between the transistors and the distance from the nMos to the nWell gate. This distance are set to 0.5μm to make space for wiring. As seen in the figure the transistors are not wired, the wiring will determine what type of building blocks this should be. By adding or removing lines of stacked transistors the different blocks used in this project are created from this outline. Blocks as seen in figure 28a can as well be set on top of each other giving layout as seen in figure 28b where the small arrow still indicates a distance equal to 0.5μm. It is then important to keep the poly pitch directly under each other to keep the regular outline. This method is used when putting together the 16by9bit adder seen in figure 48 to get a more quadratic layout and shorter wiring distances.



(a) General

(b) Stacked

Figure 28: Layout Outline

6.5 Kogge-Stone vs Ripple-Carry

It is commonly known that the kogge-stone adder is the fastest adder while the ripple-carry adder is considered to be the slowest one. For a 32 Bit Adder the Kogge-Stone adder are shown to be 4.5 times faster than the Ripple-Carry Adder, while with wire delays it is shown to be only 2.2-2.4 times faster for a 32Bit implementation[28]. In this project the each adder will only reach a maximum number of additions equal to 12Bit. From the calculation in equation 14 and 15 the Ripple-Carry adder will consist of 8 block while the Kogge-Stone adder will consist of 5 blocks for the 8Bit adders. But for a 32 bit adder the Ripple-Carry adder will have 32 delay blocks and the Kogge-Stone adder will only have 7 delay blocks. Based on this it is possible that it can lead to major energy saving using Ripple-Carry Adders for smaller additions jet not have a major delay disadvan-

tage. Therefore a comparison between a 8Bit Ripple-Carry and 8Bit Kogge-Stone adder are executed.

6.5.1 8Bit Adder Test-Bench

For adder comparison the lvtgp transistors operating at 200mV are chosen. The Ripple-Carry adder are build from the Full Adder blocks shown in section 4.2.2. While the Kogge-Stone Adder are build from the new 8T basic building blocks shown in section 4.1.3. The Adder test bench set all 8 bits for one signal to logic one(V_{DD}) while setting all 8 Bits signals for the other signal equal to logic zero(GND). Then the carry input is tuned to toggle between logic one and zero at max frequency, to measure the maximum delay and power usage. The toggling of the carry input will in this situation lead to a propagating of the carry through the whole adder, leading to the longest timing delay at the highest summation output. The test-bench is illustrated in figure 29.

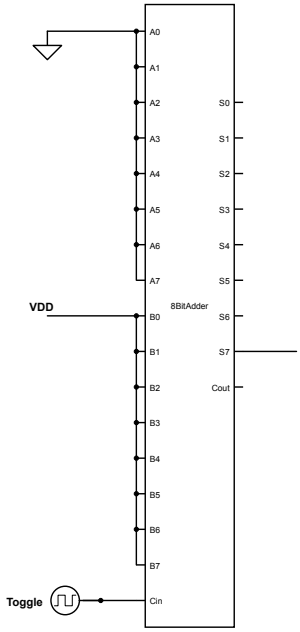


Figure 29: 8Bit Adder test-bench

To check the robustness due to process variation and mismatch 100 Monte-Carlo runs are used for each simulation. This because 50 Monte Carlo runs are needed to get meaningful statistics as described in section 3.10 and 100 Monte Carlo runes are used to be sure that enough data is obtained still not having very long simulation time. Then to check the robustness towards mismatch and process variations, the gates are simulated at 27 degrees. The simulations are only done in schematic not taking parasitics from wiring into account.

6.5.2 Transistor Size

After sizing the transistor widths with gate lengths equal to 90nm and supply voltage equal to 200mV the transistor sizes used in this two 8Bit adders are shown in table 13.

Table 13: Widths and lengths balanced for 200mV gate lenght = 90nm

P_{length}	N_{length}	P_{width}	N_{width}
90nm	90nm	590nm	300nm

6.5.3 Kogge-Stone and Ripple-Carry Simulations

In figure 30 the result of delay and power simulations for the Kogge-Stone adder and the Ripple-Carry adder are shown.

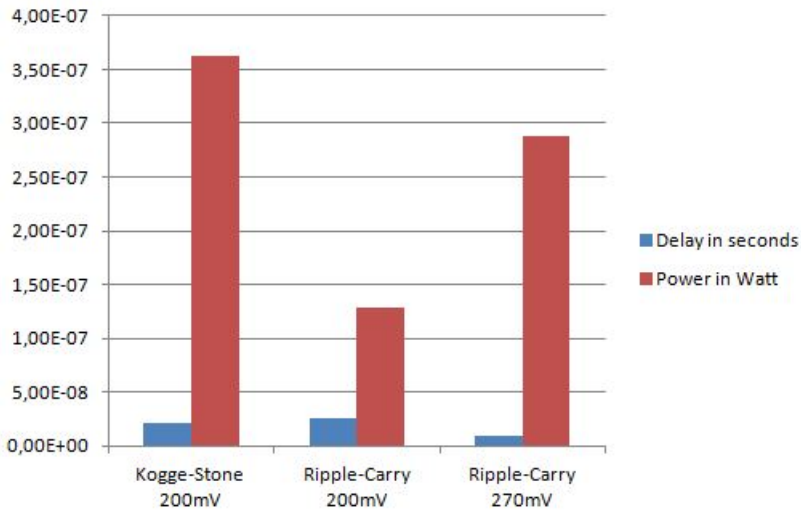


Figure 30: Kogge-Stone and Ripple carry Power and Delay

The results are as well shown in table 14.

Table 14: Kogge-Stone and Ripple carry Power and Delay

Implementation	V_{DD}	Delay in seconds	Power in Watt
Kogge-Stone Adder	200mV	20.9nS	0.36μW
Ripple-Carry Adder	200mV	26.3nS	0.13μW
Ripple-Carry Adder	207mV	9.2nS	0.29μW

In figure 31 the deviation due to process variation and mismatch are shown as percent deviation from the delay and power consumption shown in figure 30. The deviation are set by sigma(σ) that represent the standard deviation from the Monte Carlo simulations.

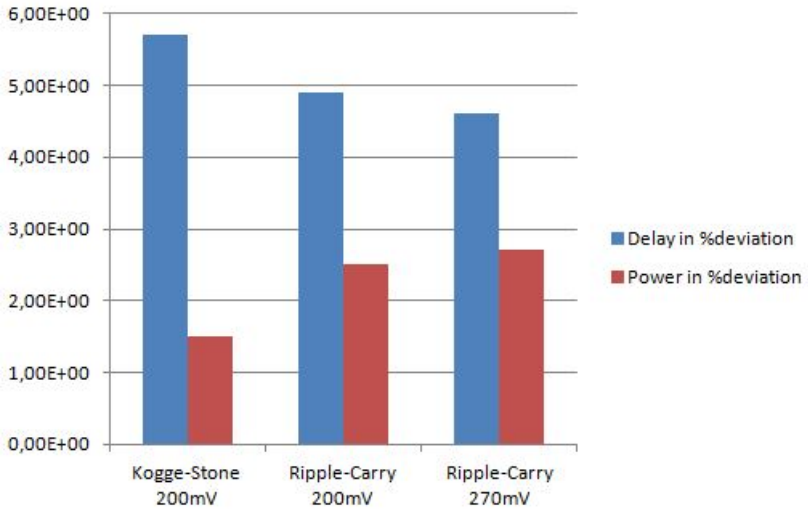


Figure 31: Kogge-Stone and Ripple carry Deviation

6.5.4 Adder Conclusion

As seen from figure 30 it is shown that the Kogge-Stone adder is 20% faster than the Ripple-Carry adder at equal supply voltage, but it also uses 180% more power at a supply voltage equal to 200mV. In figure 30 it is also shown that when the Ripple-Carry supply voltage is turned up to 270mV the Ripple-Carry

adder is 55% faster yet uses 35% less power than the Kogge-Stone adder working at 200mV. In figure 31 the deviation due to process variation and mismatch are shown, it is shown that the Ripple-Carry adder are more robust looking at delay variation while the Kogge-Stone adder is shown to be more robust looking at power in % deviation.

The simulations are schematic simulations that not consider wirering that are foreseen to degrade the speed advantage for the Kogge-Stone adder[28]. This results shows that the Ripple-Carry adder is the better alternative for smaller adder implementations like the adders that will be used for the 16By9Bit adder.

6.6 16By9Bit Adder Design

The 16 time 9 Bit adder will add 144 inputs giving $2^{144} = 2.23 * 10^{43}$ different input stages. The testing of all this input stages will take a lot of time therefore some restrictions are set. The circuit are made for beamforming for ultrasound, and all the highest most important bits will be equal while there will be minor differences in the smallest bits that not is so important for the image. Taking this into consideration all high and low bits can be set equal for simulation and testing of the circuit. Meaning that all the 9 bits from the 16 input signals are set equal giving only $2^9 = 512$ different input stages.

6.6.1 Adders

The adders used in the 16by9bit adder have no carry inputs meaning that there is no need for a Full adder to calculate the first Bit. The fist bit are calculated from a Half adder 4.2.1 while the rest of the adders uses Full adders 4.2.2. The N-Bit adders delay and power consumption can then be calculated from equation 18

$$T_{Delay,Power} = H_{Adder} + (N-1)F_{Adder} \tag{18}$$

Where $T_{Delay,Power}$ is the total delay or power consumption, H_{Adder} can be used as the Half adder delay or power consumption and F_{Adder} can be used as the Full adder delay or power consumption.

6.6.2 Transistor Sizing

Timing will become a issue therefore the lvtgp transistor are used. But when sizing the lvtgp transistor it has been chosen to use a gate length equal to 60nm. As seen in figure 22 this will increase the threshold voltage leading to a bit slower

circuit delay, but it will as well lead to savings in power consumption. The transistor widths are found after balancing the minority-3 circuit for operation voltage V_{DD} equal to 270mV and gate lengths equal to 60nm, the transistor sizes are shown in table 15.

Table 15: Widths and lengths balanced for 270mV gate lenght = 60nm

Transistor	P_{length}	N_{length}	P_{width}	N_{width}	pMos V_{th}	nMos V_{th}
lvtgp	60nm	60nm	650nm	300nm	290mV	338mV

6.6.3 16By9Bit Adder Verification

A verification test bench is made to check if the 16by9bit adder works for all 512 different input combinations. The test bench are shown in figure 32 and consist of a counter that count trough all 512 stages, the counter is a 9 bit adder that always add one bit to the previous result giving a 9 Bit counter. To verify the adder the 16by9bit adder are as well written in verilogA code for comparison with the 16by9bit adder CMOS adder design. And as seen in figure 32 this two adder results are run into a Sample and hold circuit that hold the signals at a set time and run the signal into a comparator block that compare the output from the two 16by9bit adders and gives a '1' on the output if a difference occurs. The sample and hold, comparator and FullAdders used for the verilogA 16by9bit adder are written in verilogA code and are shown in appendix A.1.1, A.1.2 and A.1.3.

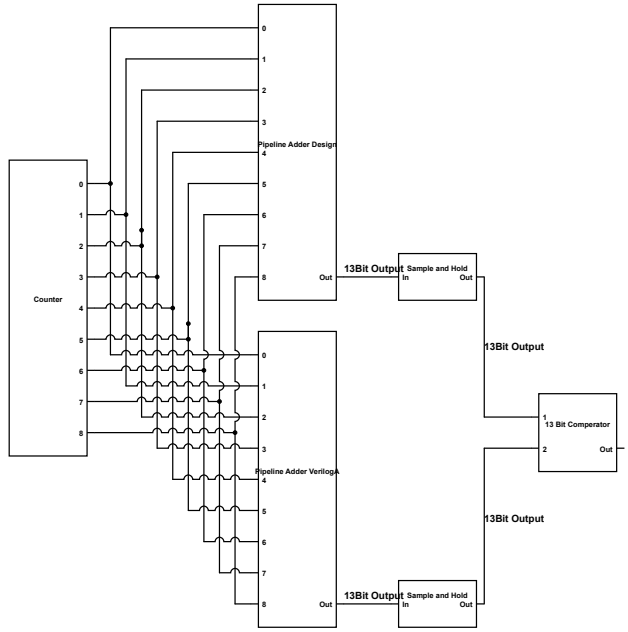


Figure 32: 16 times 9 Bit Adder verification test-bench

6.6.4 16By9Bit Adder Test-Bench

The test bench in figure 33 are made to check the 16by9bit adder speed and power consumption at a input signal of 50MHz. The inverters are set as load to give the circuit a load and more real input signals. The worst delay time is calculated to be when all inputs toggle between zero and one. At the time when all inputs go from zero to one, the carry will propagate trough all the Full Adders meaning that this cause the longest delay time for the 512 different input stages. But this is only correct for this restrictions, meaning that longer delay times can be found by not setting all the 9bit from the 16 input signals equal to each-other. But this is not considered in this project.

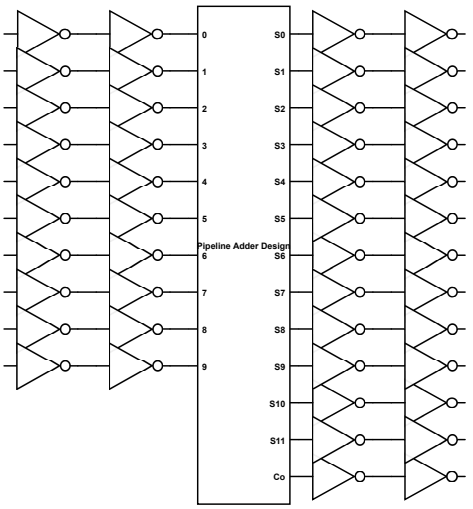


Figure 33: 16 times 9 Bit Adder test-bench

6.6.5 Process Variation, Mismatch and Temperature Simulations

To check the temperature variations single runs at different temperatures are run in schematic and of the three dimensional parasitic layout extraction. Checking the robustness toward process variation and mismatch was done with Monte Carlo simulations in schematic. But when running Monte-Carlo simulations of the parasitic extracted layout for this design a problem occurs, a 12 core Intel(R) Xeon(R) CPU X5680 @ 3.33GHz server with a main memory of 48G and swapping memory of 50G can't handle this data without running out of memory. A downgrade of the parasitic extraction from three dimensional to two dimensional that not considers wiring parasitics between the layers where done, yet the server runs out of memory. Therefore there has been chosen to split up the 16By9Bit adder to a simple 9Bit Ripple-Carry adder whit no carry input that is a part of the 16by9bit adder design. Then to test the parasitic extracted layout design of this adder to give some indication of the circuit's robustness towards process variation and mismatch. The test-bench for this adder uses the worst case condition shown in section 6.6.4 where all inputs switches from zero to one at 50Mhz to check the variation in delay and power for analyses of the 16by9bit adder.

The Ripple Carry adder simulations will give a opportunity to estimate the effect the process variation and mismatch have on the 16by9bit adder in layout. The layout of the 9Bit adder are shown in figure 47. The test-bench runs 100 Monte Carlo runs, the deviation are measured at one sigma and the temperature is set to 27°C.

The process variation in a 9Bit adder can be used to estimate the process in the 16By9Bit adder because it represent one of the four 16 times 9 adder stages described in section 4.2.5. And because all the stages consist of adder this simulation will give a understanding of the robustness towards mismatch and process variation at each 16By9Bit adder stage.

7 Simulations and Results

After the 16by9bit adder has passed the verification test-bench described in section 6.6.3 the adder are run trough the test-benches and the results are presented in this section.

7.1 Transistor Count and Area

With Dummy transistors the 16by9bit adder consist of 6736 transistors where 172 of them are dummy transistors. The layout is 240μm long and 84μm high giving a area of 20.1mm^2 and are shown in figure 48.

7.2 Delay and Power

In figure 34 the delay as a function of temperature simulated in schematic are shown. The figure shows rising and falling delay for the 16by9bit adder at different supply voltages.

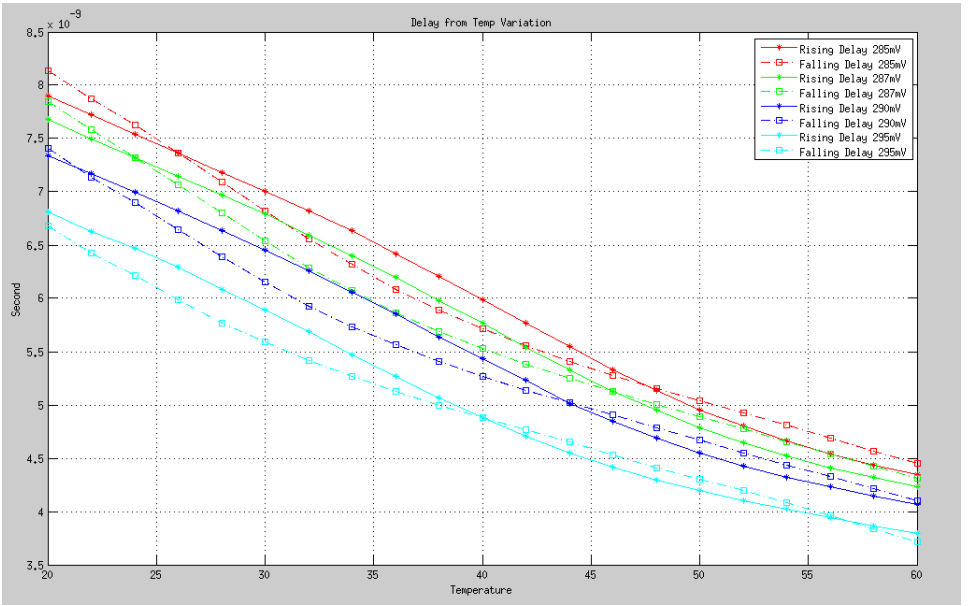


Figure 34: Delay at different temperatures after schematic simulations

In figure 35 the power as a function of temperature simulated in schematic

are shown. The figure shows the power consumption for the 16by9bit adder at different supply voltages.

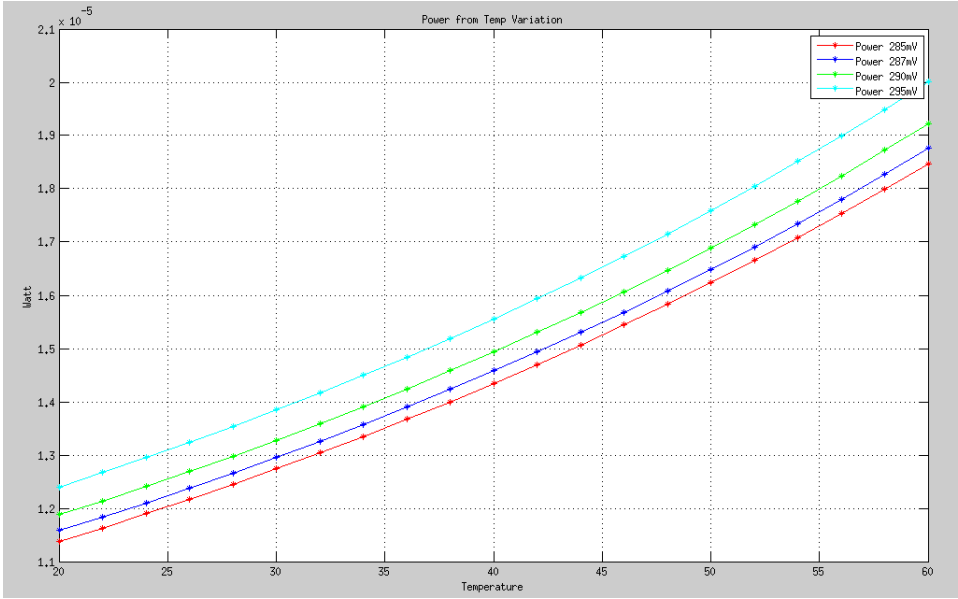


Figure 35: Power at different temperatures after schematic simulations

In figure 36 the delay as a function of temperature simulated from layout are shown. The figure shows rising and falling delay for the 16by9bit adder at different supply voltages.

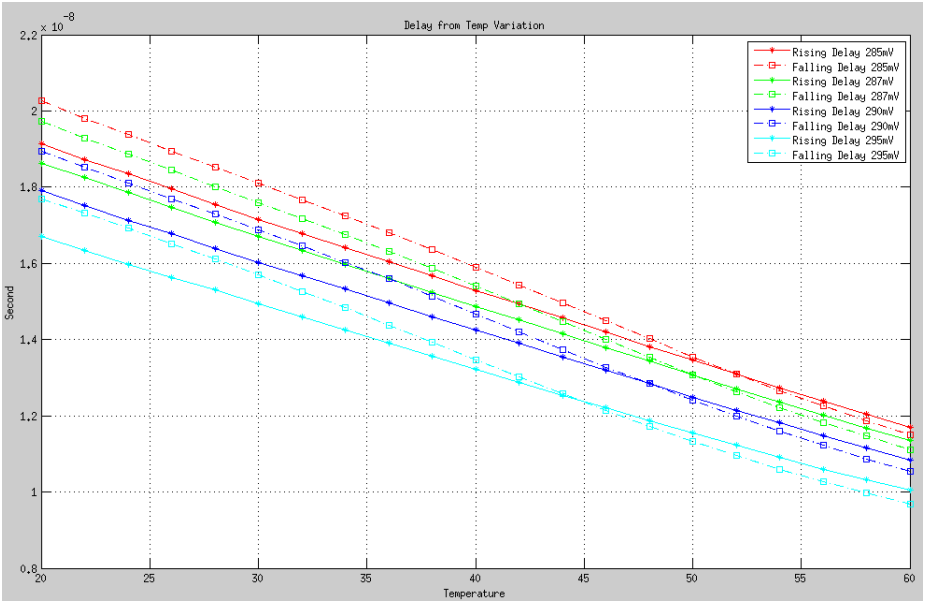


Figure 36: Delay at different temperatures after layout simulations

In figure 37 the delay as a function of temperature simulated from layout are shown. The figure shows the power consumption for the 16by9bit adder at different supply voltages.

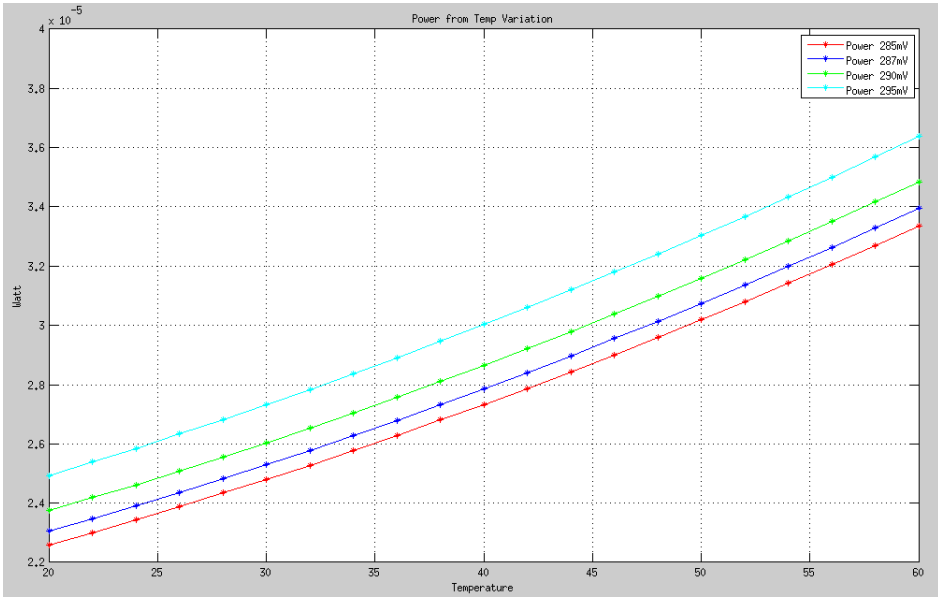


Figure 37: Power at different temperatures after layout simulations

7.3 Process Variation and Mismatch

In this section the results from 100 Monte Carlo runs at a temperature equal to 27°C are shown.

In figure 38 the results after 100 Monte Carlo simulations are shown in delay together with σ deviation for the 16by9bit adder at 285mV and 295mV after schematic simulations.

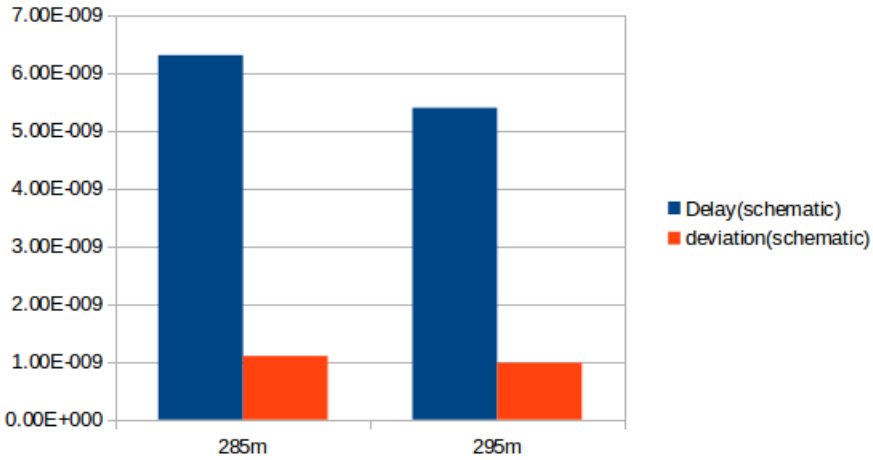


Figure 38: Delay simulations in schematic for the 16by9bit adder

The results are as well shown in table 16 that also shows the % deviation.

Table 16: Delay simulations in schematic for the 16by9bit adder

V_{DD}	*Delay*	*Deviation*	*%Deviation*
285m	6.3nS	1.1nS	17.5%
295m	5.39nS	0.98nS	18.2%

In figure 39 the results after 100 Monte Carlo simulations are shown in power consumption together with σ deviation for the 16by9bit adder at 285mV and 295mV after schematic simulations.

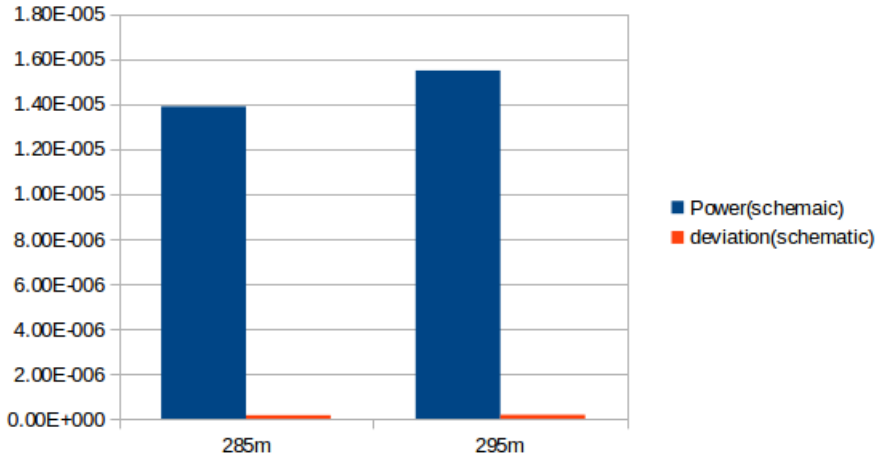


Figure 39: Power simulations in schematic for the 16by9bit adder

The results are as well shown in table 17 that also shows the % deviation.

Table 17: Power simulations in schematic for the 16by9bit adder

V_{DD}	*Power*	*Deviation*	*%Deviation*
285m	13.9μW	0.16μW	1%
295m	15.5μW	0.18μW	1%

In figure 40 the results after 100 Monte Carlo simulations are shown in delay together with σ deviation for the 9Bit adder at 285mV and 295mV after both schematic and layout simulations.

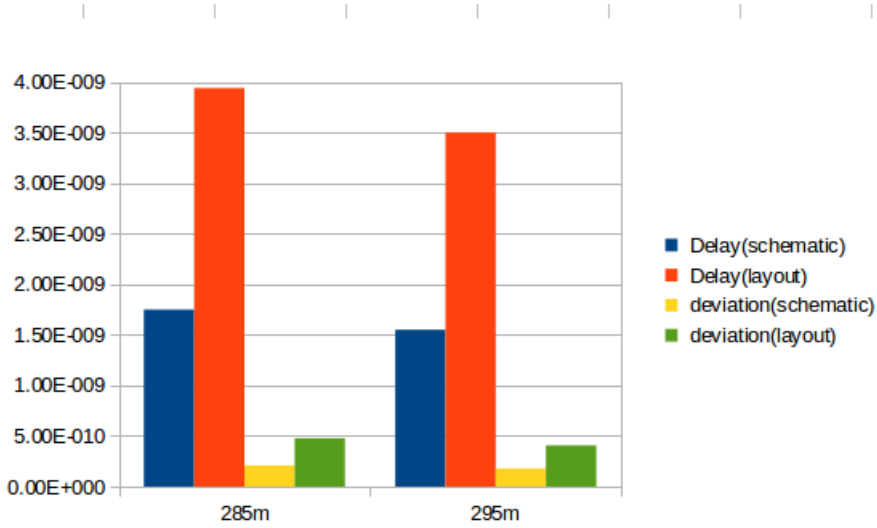


Figure 40: Delay simulations in layout and schematic for the 9Bit adder

The results are as well shown in table 18 that also shows the % deviation. Where Dev is short for deviation schem is short for schematic and lay is short for layout.

Table 18: Delay simulations in layout and schematic for the 9Bit adder

V_{DD}	$Delay_{lay}$	$Delay_{schem}$	Dev_{lay}	Dev_{schem}	$\%Dev_{lay}$	$\%Dev_{schem}$
285m	3.9nS	1.75nS	0.48nS	0.21nS	12.1%	11.9%
295m	3.5nS	1.55nS	0.41nS	0.18nS	11.6%	11.5%

In figure 41 the results after 100 Monte Carlo simulations are shown in power consumption together with σ deviationfor the 9Bit adder at 285mV and 295mV after both schematic and layout simulations.

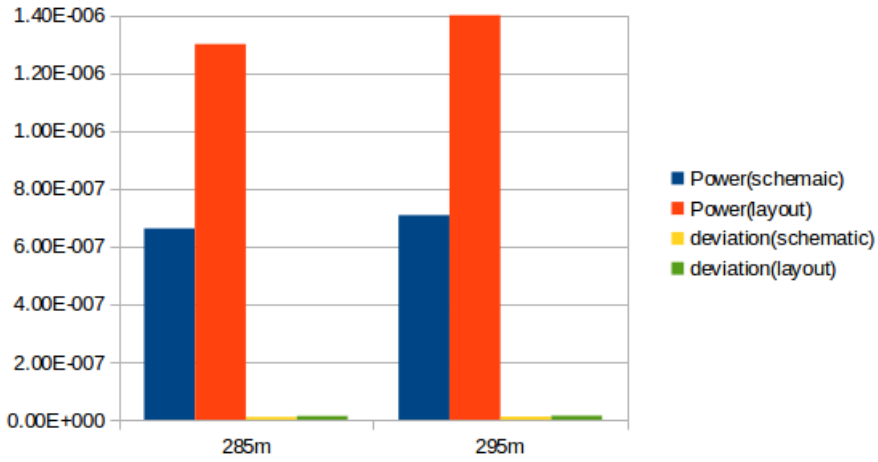


Figure 41: Power simulations in layout and schematic for the 9Bit adder

The results are as well shown in table 19 that also shows the % deviation. Where Dev is short for deviation schem is short for schematic and lay is short for layout.

Table 19: Power simulations in layout and schematic for the 9Bit adder

V_{DD}	$Power_{lay}$	$Power_{schem}$	Dev_{lay}	Dev_{schem}	$\%Dev_{lay}$	$\%Dev_{schem}$
285m	1.3μW	0.66μW	13nW	10nW	1%	1.5%
295m	1.4μW	0.71μW	14nW	10.1nW	1%	1.5%

8 Discussion

In this section the results from the 16by9bit adder seen in section 7 will be discussed.

8.1 Delay

As seen from figure 34 and figure 36 the delays increase with a factor of approximately 2.5 when going from schematic to layout implementation. This is due to the parasitics created from the three dimensional parasitic extraction. The circuit was benchmarked to work from 20 degrees up to 80 degrees with a working speed faster than 50MHz that means that the delays has to be smaller than $\frac{1}{50Mhz} = 20ns$. As seen from the figure 36 the 16 times 9 bit adder will at 20 degrees not meet the delay requirement when the power supply is equal to 285mV. But when tuning up the power supply voltage to 295mV the worst case time delay is shown to be 17.7nS = $\frac{1}{17.7ns} = 56.5$MHz at 20 degrees and will then still have some margin towards process variation and mismatch. It is as well shown that when the temperatures increase the delay time decreases as foreseen from the temperature theory shown in section 3.9 meaning that the circuit have the worst delay time at 20 degrees. The factor for decreased delay times due to power supply scaling can be calculated from figure 36, and are calculated to be 6.5% faster per 5mV increasing of the supply voltage at 20°C.

8.2 Power Consumption

As seen from the power simulation results the power consumption increase with a factor of two approximately, from schematic to layout simulations. And in figure 37 it is shown that the power consumption reach 36.2μW at max when the temperature is 80 degrees and the supply voltage is 295mV. This is well below the goal of less than 50μW meaning that the supply voltage can be increased even further to achieve even faster circuit delays without braking the goal for power consumption. It is as well shown that the power consumption degrees with the temperature, this because the leakage current are raised by the temperature as described in section 3.9. The factor due to supply voltage scaling can be found from figure 37 by looking at the increased power consumption when increasing the supply voltage, this factor is calculated to be 4.3% per 5mV increase in supply voltage at 80°C.

8.3 Process Variation and Mismatch

In figure 40 the delay Monte Carlo simulation results are shown from both layout and schematic for a 9 Bit Adder. Calculating the relative deviation in percent as seen in table 18 it is shown that the deviation % is almost equal looking at delay deviation. This can indicate that the process and mismatch variation are dominated by the changes in the transistors parameters, meaning that the wiring parasitic variations has very little effect on the relative deviation when going from schematic to layout Monte Carlo simulations. Meaning that a good estimate can be done for the 16By9Bit adder when it comes to process variation and mismatch in layout. In table 16 it shown that the % deviation is equal to 17.5% and 18.2% in delay after Monte Carlo simulations in the schematic at power supply equal to 285mV and 295mV. Then the layout delay times from the 16 time 9 bit adder are read out from figure 36 at 27°C at 285mV and 295mV to be 18.2nS and 16nS then to estimate the deviation due to process variation and mismatch to be $18.2nS * 0.175 = 3.2nS$ and $16nS * 0.182 = 2.9nS$. Meaning that the implementation using 295mV power supply still will have a maximum delay under 20nS at 27°C and higher temperatures. Using the same factor for calculations of the impact caused by process variation and mismatch at 20°C the deviation will be equal to $17.7nS * 0.182 = 3.2nS$ giving a max delay at 20.9nS, that breaks the limit of 20nS but this can be fixed by tuning up the supply voltage even further. The goal of power consumption was set to 50μW and the max power consumption was measured to be 36.2μW at 295mV power supply, meaning that the supply voltage can be set higher whit-out braking the goal of 50μW.

Looking at the power deviation after 100 Mote Carlo runs it is shown in figure 39 and figure 41 that the deviation due to mismatch and power consumption are small, close to 1%. This power deviation is probably this low because the input has a frequency of 50MHz and the circuit is much faster than this in the Monte Carlos simulations meaning that the circuit is in a steady state for longer periods. When the temperature is 80 degrees the power consumption is highest and the circuit have a low delay time meaning that a deviation of 1% is likely because of longer steady state times. It is as well reasonable to assume that the % deviation will increase when the circuits work at 20 degrees because of a slower circuit and shorter steady time, but at 20 degrees the power consumption is lower and the deviation most likely not cause the power consumption to be higher than the power consumption at 80 degrees.

8.4 Improvements

After parasitic extraction of the 16 times 9 bit adder the delay increase with more than a factor 2.5 compared with schematic simulation, this was more than

accounted for and thereby the supply voltage has to been tuned above the 270mV that the circuit was balanced for. For a improvement of the design it will may have been better to stick to the 90nm gate length that would give a lower threshold voltage and thereby a faster working speed, but this would had come at the cost of higher power consumption. Another approach that may would have improved the circuit performance could be to balance the circuit for a supply voltage equal to 295mV still keeping gate length at 60nm. Instead of balancing for a power supply at 270mV that was proven to not give a circuit that was fast enough. To get a better estimate of the 16by9bit adders robustness towards mismatch and process variation in layout. A idea could be to recreate the longest delay path in the adder circuit, this can be done by setting one 9, 10, 11 and 12 bit adder in series and measure the delay propagating trough this adders. A parasitic extraction of this layout design is far smaller than the parasitic extraction of the 16by9bit adder and the computer would more likely be able to run a Monte Carlo simulation on this circuit without running out of memory.

9 Conclusion

The thesis shows a 16by9bit adder implementation with a working speed higher than 50MHz for temperature between 20°C and 80°C after parasitic extraction of the layout design. It is shown that designing the circuit for the near/sub-threshold operating point it is achievable to get a working layout design with a power consumption lower than 50μW, still not braking the 50MHz requirement. With a operating point equal to 295mV the adder are shown to have a delay equal to 17.7nS = 56.5MHz with a power consumption of 25μW at 20°C and delay equal to 10nS = 100MHz with a power consumption of 36.2μW at 80°C. There are as well calculated some robustness values towards process variation and mismatch after Monte Carlo simulations. Taking this variation into account the time limit would be hard to hold at 20°C. This value was calculated to be 20.9μS at 20°C. But it is shown that turning up the supply voltage will solve this problem and still have a power consumption less than the goal of 50μW.

This result shows that a digital implementation of the micro-beamformer shown in section 2 operating in the sub/near-threshold region leads to major savings in the total power consumption compared to an analog implementation.

9.1 Future Work

- The next step will be to do a tape out on a circuit board for future testing of circuit delay, power consumption and robustness.
- A paper regarding the 8T implementation are under working progress together with Snorre Aunet and Jonathan Bjerkedok.

References

[1] K. Granhaug and S. Aunet. Six subthreshold full adder cells characterized in 90 nm cmos technology. In *Design and Diagnostics of Electronic Circuits and systems, 2006 IEEE*, pages 25 –30, 0-0 2006.

[2] M. Blesken, S. Lu andtkemeier, and U. Ru andckert. Multiobjective optimization for transistor sizing sub-threshold cmos logic standard cells. In *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, pages 1480 –1483, 30 2010-june 2 2010.

[3] Liji Chen, Ruoyu Xu, and Jie Yuan. An efficient bscan-sample-based #x03a3; #x0394; beamformer for medical ultrasound imaging. In *Biomedical Circuits and Systems Conference, 2009. BioCAS 2009. IEEE*, pages 285–288, 2009.

[4] Zili Yu, S. Blaak, Zu yao Chang, Jiajian Yao, J.G. Bosch, C. Prins, C.T. Lancee, N. de Jong, M. A P Pertijs, and G. C M Meijer. Front-end receiver electronics for a matrix transducer for 3-d transesophageal echocardiography. *Ultrasonics, Ferroelectrics and Frequency Control, IEEE Transactions on*, 59(7):1500–1512, 2012.

[5] B.D. Van Veen and K.M. Buckley. Beamforming: a versatile approach to spatial filtering. *ASSP Magazine, IEEE*, 5(2):4–24, 1988.

[6] Zili Yu, S. Blaak, Zu yao Chang, Jiajian Yao, J.G. Bosch, C. Prins, C.T. Lancee, N. de Jong, M. A P Pertijs, and G. C M Meijer. Front-end receiver electronics for a matrix transducer for 3-d transesophageal echocardiography. *Ultrasonics, Ferroelectrics and Frequency Control, IEEE Transactions on*, 59(7):1500–1512, 2012.

[7] R.R.Harrison. Mosfet operation in weak and moderate inversion.

[8] P.R.Panda. *Power-efficient System design.* Springer Science+Business Media, 2010.

[9] P. Nilsson. Arithmetic reduction of the static power consumption in nanoscale cmos. In *Electronics, Circuits and Systems, 2006. ICECS '06. 13th IEEE International Conference on*, pages 656 –659, dec. 2006.

[10] Ken Martin David A. Johns. *Analog Integrated Circuit Design.* John Wiley & Sons Inc, 2009.

[11] Alice Wang, Benton H Calhoun, and Anantha P Chandrakasan. Subthreshold design for ultra low-power systems. 2006.

[12] Philippe Royannez Amara Amara. Vhdl for low power. *Taylor and Francis Group, LLC*, 2006.

[13] Eric A Vittoz. Micropower techniques. 1994.

[14] Jabulani Nyathi Robert R. Rydberg Walid Ibrahim Valeriu Beiu, Snorre Aunet. Serial addition: Locally connected architectures. 2007.

[15] M. Alioto. Impact of nmos/pmos imbalance in ultra-low voltage cmos standard cells. In *Circuit Theory and Design (ECCTD), 2011 20th European Conference on*, pages 536 –539, aug. 2011.

[16] K. Raghavendra and M. Mutyam. Process variation aware issue queue design. In *Design, Automation and Test in Europe, 2008. DATE '08*, pages 1438 –1443, march 2008.

[17] H. Soeleman, K. Roy, and B.C. Paul. Robust subthreshold logic for ultra-low power operation. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 9(1):90–99, 2001.

[18] A.V.Kordesch P.B.Y.Tan and O.Sidek. How to design for analog yield using monte carlo mismatch spice models, 2005.

[19] Michael Reinhardt. *Automatic Layout Modification*. Kluwer Academic Publishers, 2002.

[20] Verilog-a language reference manuall, 1996.

[21] H.K.O. Berge, A. Hasanbegovic, and S. Aunet. Muller c-elements based on minority-3 functions for ultra low voltage supplies. In *Design and Diagnostics of Electronic Circuits Systems (DDECS), 2011 IEEE 14th International Symposium on*, pages 195 –200, april 2011.

[22] H. Al-Hertani, D. Al-Khalili, and C. Rozon. A new subthreshold leakage model for nmos transistor stacks. In *Circuits and Systems, 2007. NEWCAS 2007. IEEE Northeast Workshop on*, pages 972–975, 2007.

[23] H. Kristian, O. Berge, and S. Aunet. Multi-objective optimization of minority-3 functions for ultra-low voltage supplies. In *Circuits and Systems (ISCAS), 2011 IEEE International Symposium on*, pages 2313–2316, 2011.

[24] N.K. Tiwari, S. Akashe, J. Shrivas, and R. Sharma. Impact of technology scaling and supply voltage variation on half adder design in nanometer era. In *Information and Communication Technologies (WICT), 2012 World Congress on*, pages 33–38, 2012.

[25] A. Ghosh and D. Ghosh. Optimization of static power, leakage power and delay of full adder circuit using dual threshold mosfet based design and t-spice simulation. In *Advances in Recent Technologies in Communication and Computing, 2009. ARTCom '09. International Conference on*, pages 903 –905, oct. 2009.

[26] K. Granhaug and S. Aunet. Six subthreshold full adder cells characterized in 90 nm cmos technology. In *Design and Diagnostics of Electronic Circuits and systems, 2006 IEEE*, pages 25–30, 2006.

[27] V. Beiu, S. Aunet, J. Nyathi, III Rydberg, R.R., and A. Djupdal. On the advantages of serial architectures for low-power reliable computations. In *Application-Specific Systems, Architecture Processors, 2005. ASAP 2005. 16th IEEE International Conference on*, pages 276 – 281, july 2005.

[28] Snorre Aunet Valeriu Beiu, Asbjørn Djupdal. Ultra low-power neural inspired addition: When serial might outperform parallel architectures. 2005.

[29] Z. Moudallal, I. Issa, M. Mansour, A. Chehab, and A. Kayssi. A low-power methodology for configurable wide kogge-stone adders. In *Energy Aware Computing (ICEAC), 2011 International Conference on*, pages 1 –5, 30 2011-dec. 2 2011.

[30] F. Arnaud, F. Boeuf, F. Salvetti, D. Lenoble, F. Wacquant, C. Regnier, P. Morin, N. Emonet, E. Denis, J.-C. Oberlin, D. Ceccarelli, P. Vannier, G. Imbert, A. Sicard, C. Perrot, O. Belmont, I. Guilmeau, P. O Sassoulas, S. Delmedico, R. Palla, F. Leverd, A. Beverina, V. DeJonghe, M. Broekaart, L. Pain, J. Todeschini, M. Charpin, Y. Laplanche, D. Neira, V. Vachellerie, B. Borot, T. Devoivre, N. Bicais, B. Hinschberger, R. Pantel, N. Revil, C. Parthasarathy, N. Planes, H. Brut, J. Farkas, J. Uginet, P. Stolk, and M. Woo. A functional 0.69 mu;m2 embedded 6t-sram bit cell for 65 nm cmos platform. In *VLSI Technology, 2003. Digest of Technical Papers. 2003 Symposium on*, pages 65–66, 2003.

[31] L.L. Lewyn, T. Ytterdal, C. Wulff, and K. Martin. Analog circuit design in nanoscale cmos technologies. *Proceedings of the IEEE*, 97(10):1687–1714, 2009.

[32] Mark Lambert Cayanes Lee Eng Han, Valerio B.Perez and Mary Grace Salaber. Cmos transistor layout kungfu, 2005.

A appendix

A.1 verilogA

A.1.1 Sample And Hold

```
1  //SH block
2
3  `include "constants.vams"
4  `include "disciplines.vams"
5  `define   RISING   +1
6  `define   FALLING  -1
7
8  module SH_work(vin,clk,vout);
9  input vin, clk;
10  output vout;
11  electrical  vin, vout, clk;
12    parameter real   vth  = 0.187;
13    parameter real slack  = 100.0p from (0:inf);
14
15       real samp;
16
17
18     analog begin
19
20     // on Rise  edges of clk, sample vin
21       @( cross(V(clk)-vth, `RISING, slack, clk.potential.abstol )
             ) begin
22
23              samp=V(vin);
24
25     end
26
27     // assign output
28     V(vout) <+ samp;
29
30     end
31
32 endmodule
33
34 `undef   RISING
35 `undef   FALLING
```

Listing 1: SA

A.1.2 Comparator

```
1  //comporator made to check if two 13 bit signals matches.
2
3  `include "constants.vams"
4  `include "disciplines.vams"
5  `define RISING +1
6
7  module comp_working(A, B, OUT);
8  input [0:12] A;
9  input [0:12] B;
10
11 output OUT;
12 electrical [0:12] A;
13 electrical [0:12] B;
14 electrical OUT;
15
16 real Ai [0:12];
17 real Bi [0:12];
18 real high, low, out;
19 analog begin
20   @(initial_step("dc","ac","tran")) begin
21     high = 0.250;    // define 250mV as high output
22     low = 0;     // define 0 as low input
23   end
24
25   generate i (12,0) begin
26     Ai[i]=abs(V(A[i]));    // set the A inputs to the Ai variable
27     Bi[i]=abs(V(B[i]));    // set the B inputs to the Bi variable
28   end
29   //check if one of the bits are unequal
30   if((Ai[0] != Bi[0]) || (Ai[1] != Bi[1]) || (Ai[2] != Bi[2])
31              || (Ai[3] != Bi[3]) || (Ai[4] != Bi[4]) || (Ai[5] !=
                   Bi[5]) || (Ai
32              [6] != Bi[6]) || (Ai[7] != Bi[7]) || (Ai[8] != Bi[8])
                    || (Ai[9] != Bi
33              [9]) || (Ai[10] != Bi[10]) || (Ai[11] != Bi[11]) || (
                   Ai[12] != Bi[12]))
34     out = high;    //set the out variable to high if one of the
           bits are unequal
35   else
36     out = low;     //set the output if all bits matches
37
38   V(OUT) <+ out;     //set the out variable to the comparator
         output
39 end
40 endmodule
41
42 `undef RISING
```

Listing 2: Comparator

A.1.3 FullAdder

```
1  // simple fullAdder
2
3  `include "constants.vams"
4  `include "disciplines.vams"
5
6  module FullAdder(A, B, Ci, Co, Sum);
7    input A, B, Ci;
8    output Co, Sum;
9    electrical A, B, Ci, Co, Sum;
10
11   //variables
12   real tempA, tempB, tempC;
13   integer x, y ,z, a,b,ci,sum,co;
14
15   analog begin
16
17     // set the inputs to the variables
18     tempA = V(A);
19     tempB = V(B);
20     tempC = V(Ci);
21
22     //define if the input shoud be a set 1 or a set 0
23     //and set the integer variables
24     if(tempA < 0.050)
25       a = 0;
26     else if(tempA > 0.200)
27       a=1;
28     else
29       a=a;
30
31     if(tempB < 0.050)
32       b = 0;
33     else if(tempB > 0.200)
34       b=1;
35     else
36       b=b;
37
38     if(tempC < 0.050)
39       ci = 0;
40     else if(tempC > 0.200)
41       ci=1;
42     else
43       ci=ci;
44
45     //logical varibles for the FullAdder
46     x = a ^ b;
47     y = x & ci;
48     z = a & b;
49     sum = x ^ ci;
50     co = y | z;
```

```
51
52      //sets the outputs
53      if(sum == 1)
54        V(Sum) <+ 0.250;
55      if(sum == 0)
56        V(Sum) <+ 0;
57
58      if(co == 1)
59        V(Co) <+ 0.250;
60      if(co == 0)
61        V(Co) <+ 0;
62    end
63
64  endmodule
```

Listing 3: FullAdder

A.2 Layout

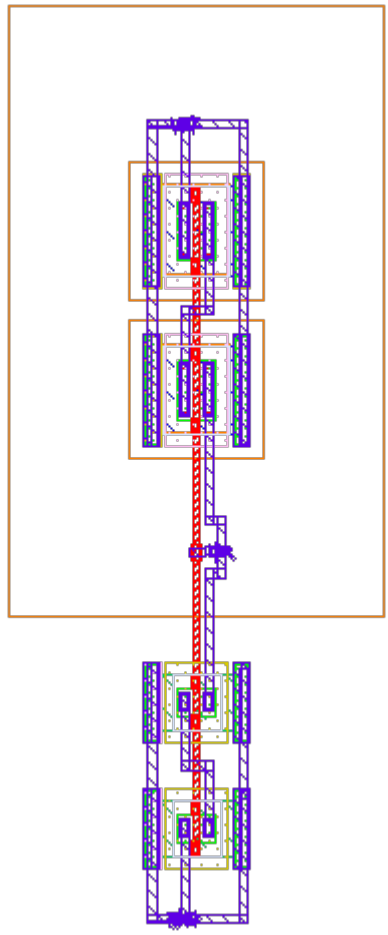


Figure 42: 4T inverter layout

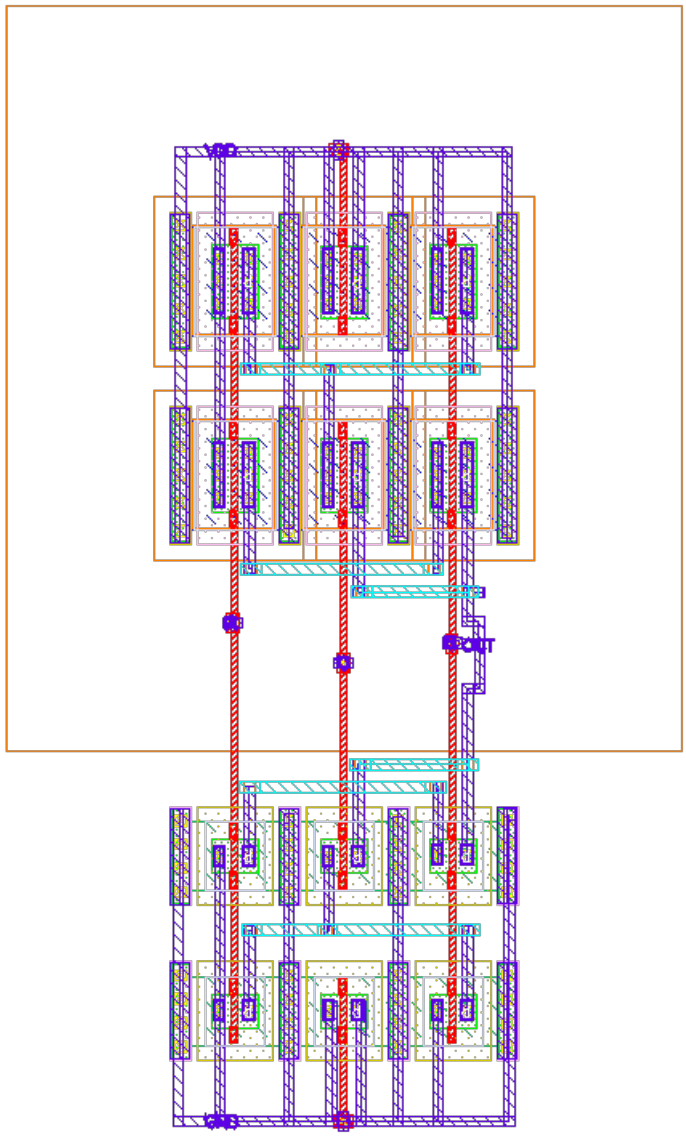


Figure 43: Minority3 layout

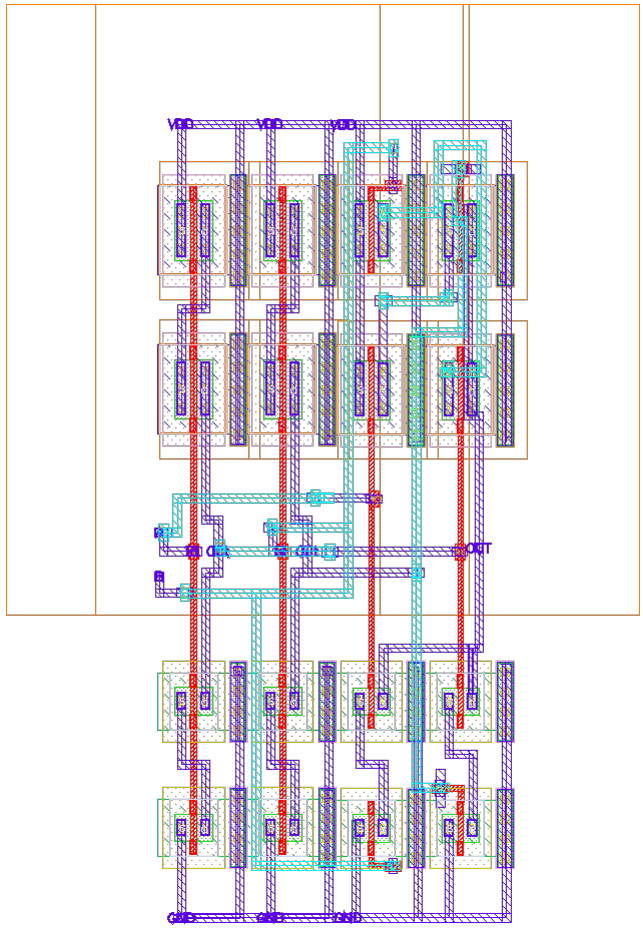


Figure 44: Xor layout

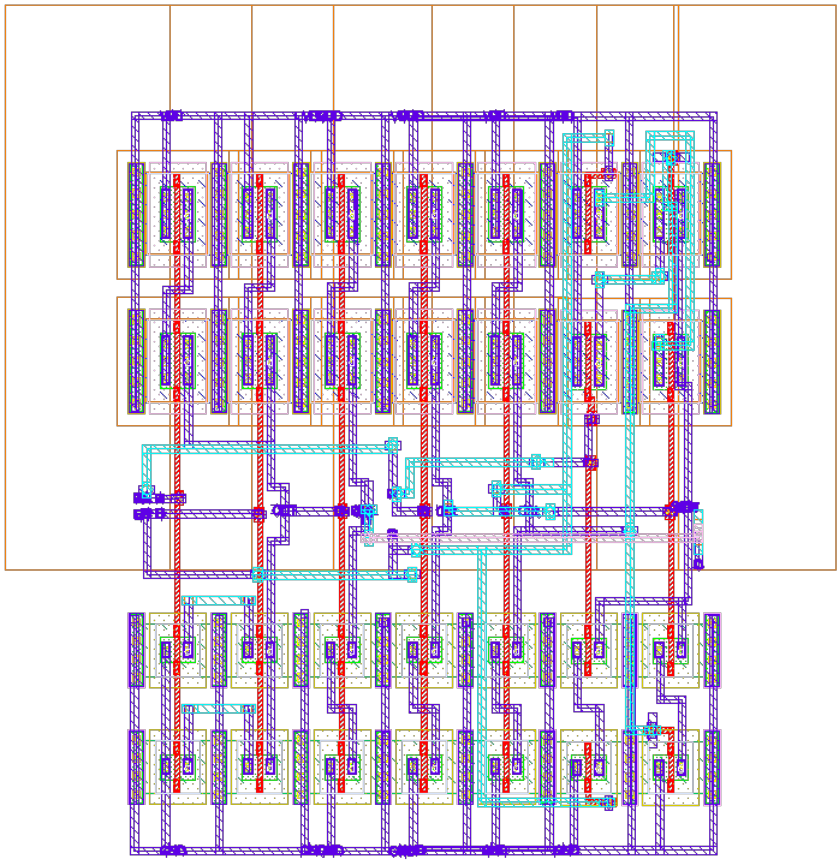


Figure 45: HalfAdder layout

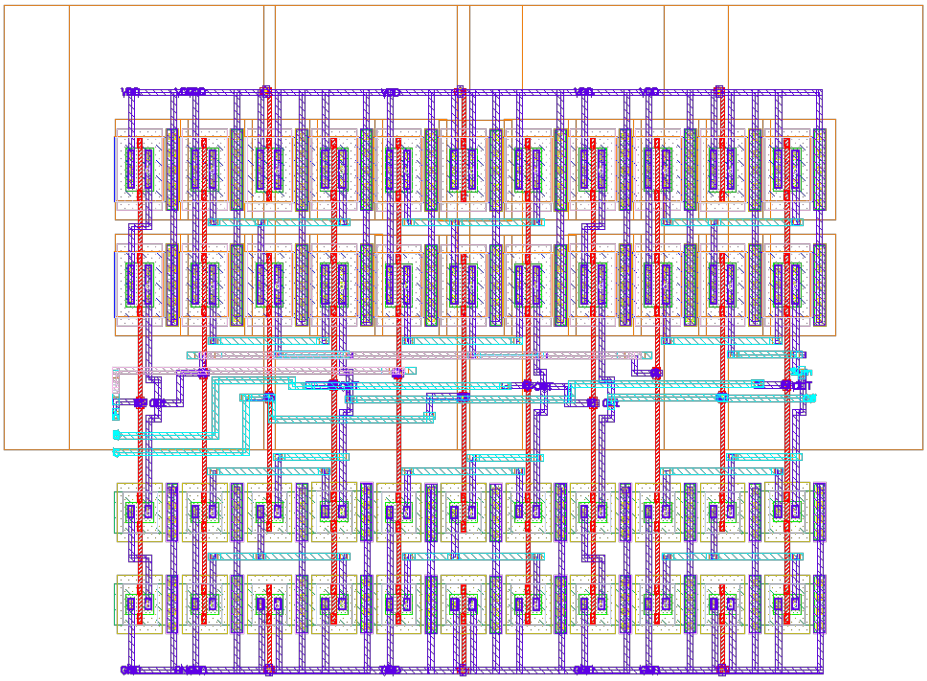


Figure 46: FullAdder layout

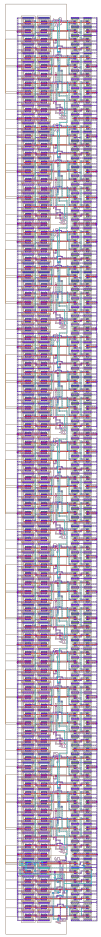


Figure 47: 9BitAdder layout

Figure 48: 16By9Bit Adder layout

A.3 Schematic

Each block in figure 49 are full adder Blocks.

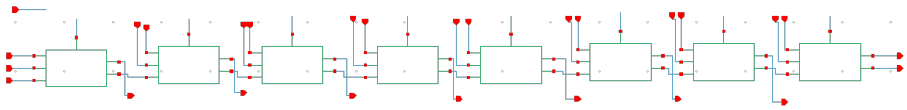


Figure 49: 8Bit Ripple-Carry Adder Schematic

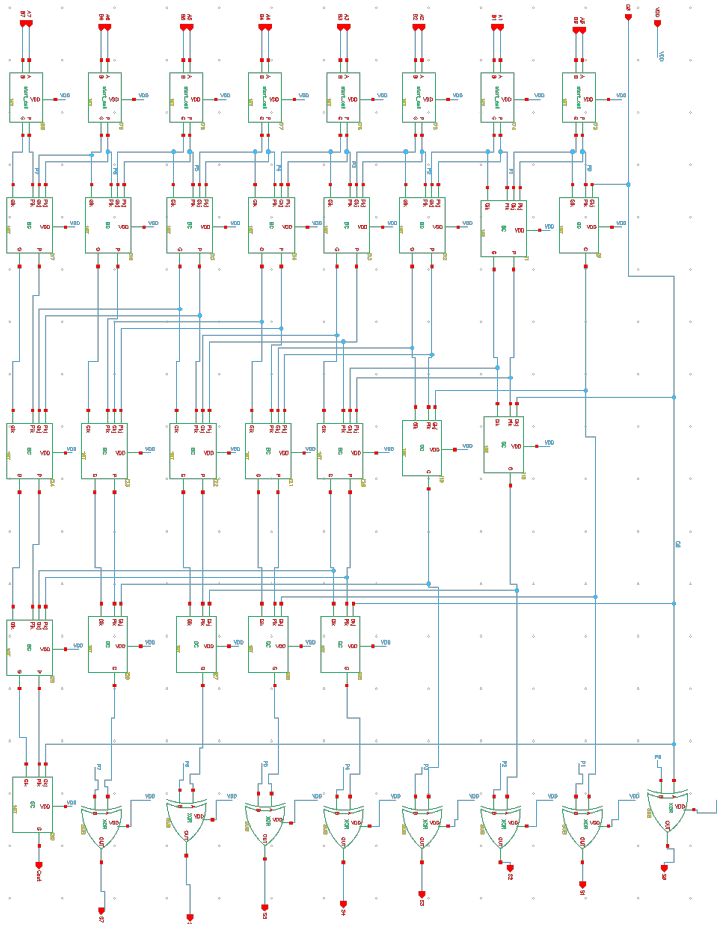


Figure 50: 8Bit Kogge-Stone Adder Schematic