NTNU – Trondheim
Norwegian University of
Science and Technology

Trade-offs between Performance and Robustness for Ultra Low Power/Low Energy Subthreshold D flip-flops in 65nm CMOS

Magne Værnes

Master of Science in Electronics
Submission date: June 2013
Supervisor: Snorre Aunet, IET
Co-supervisor: Anders Hagen, Q-Free ASA

Norwegian University of Science and Technology
Department of Electronics and Telecommunications

Abstract

The need for Ultra Low Power systems has increased with increasing number of portable devices. The maintenance costs of battery powered systems can be greatly reduced by improving the battery time, especially in places where battery replacement is hard or impossible. Implementation of subthreshold D flip-flops in layout is one step closer to having a subthreshold building block library. The task for this thesis is to implement D flip-flop blocks, which are highly suitable for subthreshold operation in layout. These are the PowerPC 603, C^2MOS, a Classic NAND-based D flip-flop, and two Minority3-based D flip-flops. The D flip-flops are first custom designed for $250mV$ in schematic at transistor level, and then implemented in layout. The implementation in layout focuses on high robustness against process variations, by using high regularity for the cost of area.

The D flip-flops are simulated in both schematic and layout, and the results are compared to each other and earlier results found in papers. The results show that the PowerPC 603 has the lowest PDP, the lowest power consumption, very low propagation delay, and an average relative standard deviation for delay. The C^2MOS has the lowest propagation delay, low power consumption and low PDP results. However, it has the highest relative standard deviation on delay. The Minority3-based D flip-flops have a very low relative standard deviation for delay, which makes them the most robust against process variations in this sense. However, they have the highest propagation delay, highest power consumption and PDP, and consumes the highest chip area. The Classic NAND-based D flip-flop has good PDP and power consumption results, but a high delay and average standard deviation for delay. Earlier papers show similar results for the C^2MOS and the PowerPC 603, but no results are found for the rest. Future work consists of implementing and testing forced-stacked blocks, body biasing, high threshold voltage transistors, and tape-out measurements.

Preface

This Thesis is the finishing part of the degree Master of Science in Circuit and System design at the Department of Electronics and Telecommunication, Faculty of Information Technology, Mathematics and Electrical Engineering, at the Norwegian University of Science and Technology (NTNU). It is done in cooperation with Q-free ASA Trondheim, with Professor Snorre Aunet at NTNU and Anders Hagen from Q-free as supervisors, and Professor Trond Ytterdal at NTNU as co-supervisor. I chose low power design for my Thesis because I find it interesting, and because I believe it will be an important part of everyday electronics in the future. The task was challenging, and gave me valuable knowledge in subthreshold operation, IC-design at schematic and layout, process variations, D flip-flop functionality and more.

I will firstly like to thank my supervisor Professor Snorre Aunet for his help and guidance through the project. He has shown great interest to the task, and always been available for help in his office, on the phone or through email. Secondly, I would like to thank co-supervisors Anders Hagen and Trond Ytterdal for the help provided by working with my Thesis.

Third, I would like to thank my fellow students Joacim Dybedal, Jonathan Bjerkedok and Lars-Frode Schjolden at the study room for the support and the RC-helicopter flying in the breaks.

Forth, I would like to thank my family for their support.

Last, but not least, I would like to thank my partner Tuva for supporting me through the whole process.

Contents

List of Figures

List of Tables

1 Introduction

Ultra low power systems and circuits are getting more and more desired for applications where the power supply is limited. Battery supplied systems like pacemakers and subsea electronic equipment, where battery charging or battery replacement is hard or impossible, could have a great economic saving from improving the battery life time. With ultra low power systems, arises the possibility to use energy harvesting to power the devices. Energy harvesting is a technique to extract energy from external sources like heat, vibration, electromagnetic radiation and more, which could remove the need of batteries and greatly reduce the maintenance and battery cost.

Today there are many integrated-circuit building blocks for circuits operating in the super-threshold region. Super-threshold building blocks are well-tested and developed by large companies, and used in systems and circuits for many years. Subthreshold operation has recently become more popular as the need for battery powered systems has increased. Since the use of subthreshold design started relatively recent, there are not many well-tested and developed building blocks. Building blocks like the D flip-flop are much used in larger systems, and contributes a lot to the total system area. By creating robust D flip-flop building blocks, which are simulated on schematic and layout, the path to well-tested and reliable subthreshold building blocks shortens.

Some papers has been published which looks into different subthreshold D flip-flop implementations, but only with simulations on schematics like [6], [13] and [14]. In this Thesis, five known D flip-flop implementations will be custom designed for subthreshold operation, and simulated on Schematic and Layout. The transition from schematic to layout introduces many new non-idealities like parasitic capacitances, electric fields, mismatch from process variations, and more which affects the functionality of the circuit.

1.1 Overview of the Thesis

The chapters and appendixes contain the following:

- Chapter 1 contains the Introduction and motivation towards subthreshold design.
- Chapter 2 presents the Problem Description, and the tools and technology used.
- Chapter 3 presents all the theoretical background needed to fully understand the implementation and results.
- Chapter 4 explains the process of choosing D flip-flop structures and the procedure of the schematic design.
- Chapter 5 presents the layout implementation method and steps towards a highly regular design.
- Chapter 6 explains the different testbenches used to simulate both schematic and layout design.
- Chapter 7 explains the different simulation methods, tests, and input signals used.
- Chapter 8 presents the simulation results based on the tests described in Chapter 7.
- Chapter 9 discuss and compare the results to each other and to earlier results found in papers.
- Chapter 10 gives a summary of the results and discuss future work and improvements.
- Appendix A presents all Monte Carlo results.
- Appendix B shows the layout view of the D flip-flops.
- Appendix C presents the source code used to initiate simulations, and process results data.

2 Problem Description

The task is to implement D flip-flop structures which are highly suitable for sub-threshold operation, and can be used as building block for greater design. The D flip-flops are custom designed in Schematics at transistor level, and implemented in layout by using techniques for high process variation robustness. The schematic and layout implementations are simulated to find delays, power consumption, PDP and susceptibility against process variations. These results are compared to find differences between schematic and layout implementations.

The tool used to implement both schematic and layout designs are Cadence Virtuoso Design Environment version IC6.1.5. The simulator used for both schematic and layout is the Cadence Virtuoso Spectre Circuit Simulator.

The transistor technology used is the STMircroelectronics 65nm SVTGP. The SVTGP is a standard threshold voltage general purpose transistor, as the name points out.

3 Theoretical Background

This Section contains the theory for understanding the basics in the design, layout, simulation and results chapters. First, subthreshold operation and difficulties around operating in that region, will be mentioned. Secondly, transistor matching, balancing and robustness will be described. Thirdly, the different D flip-flop functionality and design will be explained. At the end, timing and layout will be commented.

3.1 Subthreshold Operation

A MOS transistor is either n-channel or p-channel depending on the doping of the substrate and the doping of the Source and Drain terminals[18]. By applying a positive voltage to the gate of an n-channel MOS transistor, the gate attracts negative charge from the source and drain regions, creating a channel with mobile electrons connecting the source and the drain. By applying a sufficiently large gate-to-source voltage at the transistor, the p-region under the gate is changed to an n-channel, and is said to be inverted. The minimum gate-to-source voltage for this to happen is said to be the Threshold Voltage. The Threshold Voltage V_t of a MOS transistor is the gate-to-source Voltage where the concentration of electrons under the gate is equal to the concentration of holes in the substrate. Normally, a gate-to-source Voltage V_{GS} under the threshold voltage is said to cause the transistor to be turned off, since the current flow through the channel is significantly smaller than when applying a high voltage. However, this subthreshold current is still usable to create functional CMOS circuitry.
A MOS transistor is said to be in the subthreshold region or weak inversion when $V_{GS} - V_t < -100mV$. When operating in the subthreshold region, the current through the Drain terminal I_D should not be modelled by the square-law function. The subthreshold current is more accurately modelled by the exponential relationship. This approximation is shown in Equation 1 through Equation 4[31].

$$I_D = I_{D0} e^{\frac{V_G}{nU_T}} (e^{-\frac{V_S}{U_T}} - e^{-\frac{V_D}{U_T}}) \tag{1}$$

$$I_{D0} \sim \beta e^{\frac{V_{TH}}{nU_T}} \tag{2}$$

$$\beta = \mu C_{OX} \frac{W}{L} \tag{3}$$

The slope factor n is

$$n = 1 + \frac{\gamma}{2\sqrt{2\phi_F + V_S}} \tag{4}$$

U_T is the thermal voltage, V_{TH} is the threshold voltage, μ is the charge-carrier effective mobility, C_{OX} is the gate oxide capacitance per unit area, W and L is

the gate width and gate length of the MOS transistor. γ is the substrate factor, ϕ_F is the Fermi potential in the substrate.

As the supply voltage is reduced, the current charging the switching capacitances is also reduced, causing an increased propagation delay through the logic, so the maximum frequency of the circuit is reduced. This reduction in maximum switching frequency is one of the main drawbacks with subthreshold operation as it limits the usage area, since many applications have real-time demands which needs high frequencies to be met.

3.1.1 Subthreshold Delay

The delay of a logic gate in subthreshold operation is estimated by the time the subthreshold current uses to charge the output node. By using balanced blocks, the delay should be the same for both nMOS and pMOS. The expression for subthreshold delay can be seen in Expression 5[22], where K is a fitting parameter.

$$t_d = \frac{Q_{output}}{I_D} = \frac{KC_{output}V_{DD}}{I_D} \tag{5}$$

3.1.2 Subthreshold Power Consumption

The main purpose for operating a system in the subthreshold region is the significant decrease in power consumption. An electric system uses a combination of static and dynamic power which depends on many parameters, but especially the supply voltage. The expression for total Power consumption is

$$P_{TOT} = P_{DYNAMIC} + P_{STATIC} \tag{6}$$

where the dynamic part can be expressed as

$$P_{DYNAMIC} = \alpha C_O V_{DD}^2 f \tag{7}$$

and the static part as

$$P_{STATIC} = V_{DD} I_{OFF} \tag{8}$$

The α parameter in the expression for dynamic power consumption is the average activity factor for the system. C_O is the switched capacitance, V_{DD} is the supply voltage and f is the clock frequency of the system. The I_{OFF} parameter for the static power consumption is the average leakage current[31].

Expression 7 shows that the supply voltage has quadratic effect on the Dynamic Power Consumption, making it the most dominating factor. This means

that lowering the supply voltage to half will reduce the Dynamic power by a factor of four. In addition will the static power consumption decreases linearly with a decrease in the Supply Voltage. These are the main reasons why subthreshold design are preferred for some applications where low power is the most important factor.

3.1.3 Subthreshold Leakage Current

The subthreshold leakage currents affects the static power consumption, as seen in Expression 8. The leakage current is seen in Expression 9 and Expression 10[5].

$$I_{OFF} = \beta_2 e^{\lambda_{DS} V_{DD}/n \cdot U_t} \left(1 - e^{-V_{DD}}\right) \tag{9}$$

where

$$\beta_2 = I_0 \frac{W}{L} e^{-(V_{TH0} - \lambda_{BS} V_{BS})/nU_t} \tag{10}$$

λ_{DS} is the DIBL coefficient, λ_{BS} is the body effect coefficient, V_{TH0} is the zero-bias threshold voltage, n is the subthreshold factor and I_0 is the technology-dependent subthreshold current extrapolated for $V_{GS} = VTH$. The Threshold Voltage is affected by the drain-source voltage and the bulk-source voltage as seen in Expression 11.

$$V_{TH} = V_{TH0} - \lambda_{DS} V_{DS} - \lambda_{BS} V_{BS} \tag{11}$$

This means that keeping V_{BS} to a minimum reduces threshold voltage variations.

The Body effect occurs because the source-bulk voltage V_{SB} increases, and causes the Threshold voltage of the transistor to increase[18]. The body effect coefficient $\lambda_{BS} > 0$ is a technology dependent parameter.

Drain Induced Barrier Lowering occurs in short channel transistors, where the source and drain depletion width in the vertical direction and the source-drain potential have strong effect on the band bending over a significant portion of the device. This causes the threshold voltage and consequently the subthreshold current to vary with the drain bias[24].

3.2 Transistor Matching

It is desired to have equal drive strength for the nMOS and the pMOS transistors. Since the nMOS transistors charge mobility is naturally higher than pMOS, equal nMOS and pMOS transistors will have a different drain currents. The current through a transistor operating in the subthreshold region is described by Expression 1. The current is proportionally dependent on the β value as described in Expressions 1 to 4. Expression 3 shows that the mobility factor μ, the transistor width W, the transistor length L and the gate oxide capacitance C_{OX} determines the value of β. This means that increasing the width W or reducing the length L of the pMOS transistor can compensate for its lower mobility. One other possibility is to tune the bulk voltage V_{BS} of the nMOS or pMOS so that the drain currents are equal[5].

The nMOS/pMOS strength must be comparable to ensure a good noise margin and to achieve reasonably symmetric rise-fall times[5]. Imbalance in the system forces $V_{DD,min}$ to increase, and gives exponentially higher power consumption, if operating at $V_{DD,min}$.

In addition, the threshold voltages for an nMOS and a pMOS transistor of the same technology are usually different. For the transistors used in this Thesis, the STMicroelectronic SVTGP, the threshold voltage at $27°C$ is approximately $-315.6mV$ for pMOS, and $344.3mV$ for nMOS, but for some other technology the difference can be higher or lower.

3.3 Robustness

When operating in the subthreshold region, the channel under the gate is not inverted, so the tolerance of the transistor when it comes to temperature, process variations and mismatch are different than when operating in the superthreshold region[27]. An increase or decrease in the Drain current changes the drive of the transistor, making blocks and gates imbalanced. The exponential relationship between the subthreshold Drain current and threshold voltage increases the effect of threshold voltage variations[33] compared to superthreshold operation. The threshold voltage is affected by many variables, including temperature variations, local and global process variations and nWell distance to gate of transistor. These situations will be mentioned in this section.

3.3.1 Temperature Variations

Temperature variations affects CMOS circuits mainly for two reasons. As the temperature rises, the mobility factor μ decreases which gives lower Drain current, and an increased CMOS gate delay. However, the Threshold Voltage also decreases as the temperature rises, giving a higher Drain Current and a decreased CMOS gate delay. This can be seen in equation 12 and equation 13[9],

$$\mu(T) = \mu(T_0)(\frac{T}{T_0})^{-M} \tag{12}$$

$$V_T(T) = V_T(T_0) - KT \tag{13}$$

where T_0 is $300K$,K is the threshold voltage temperature coefficient (typical $2.4mV/K$) and M is the mobility-temperature exponent (typical 1.5).

For superthreshold operation, it is known that an increase in the temperature causes an increase in the CMOS gate delay, giving a slower circuit. This is because the decrease in the mobility factor μ dominates for superthreshold operation. For subthreshold operation it is opposite, the decrease in threshold voltage dominates. So an increase in the temperature causes a decrease in the CMOS gate delay, causing a faster circuit. In subthreshold operation, the increased temperature also gives an increased leakage current[33], as the threshold voltage rises.

3.3.2 Process Variations

Process variations can be split up into global process variations and local process variations.

Global Process Variations are variations which are equal over the die, like wafer-to-wafer misalignment or processing temperatures. These variations normally affects all transistors in the system in the same degree. However, some parts of the circuit can be more susceptible to process variations and cause threshold voltage variations.

Local Process Variations are variations which only affects parts of the die or circuit. The local process variations can consist of both systematic and random components. These can typically be aberrations in the processing equipment which can give systematic variations. Or placement and number of dopant atoms in the device which contributes to random variations[33].

3.3.3 Well-Proximity-Effect

The Well-Proximity-Effect or WPE is an effect that arises when the pMOS transistors are placed too close to the egde of the nWell. In the manufacturing process, high energy ions are scattered at the well photo resist edge and introduces extra dopant atoms in the silicon near the well edge[26]. The closer a transistor is to the nWell edge, the higher concentration of dopant in the n-channel. In [21] it is mentioned that the nWell should be placed $> 2\mu m$ from gate of the pMOS transistor to prevent the Well-Proximity-Effect (WPE), which could produce a threshold voltage increase.

3.4 The Building Blocks

The building blocks for the D flip-flops will be described in this Section. These are the inverter, Transmission Gate, the Clocked Inverter, the Minority3-gate, the D-latch and the D flip-flop.

3.4.1 Inverter

The inverter takes an input signal and inverts it to the output. It consists of one nMOS and one pMOS transistor.

3.4.2 Transmission Gate

The transmission gate consists of one nMOS and one pMOS transisitor connected in parallel as shown in Figure 1. The transmission gate functions as a switch where the s signal is either high or low[30]. When s is high, both nMOS and pMOS conducts, and while s is low, both nMOS and pMOS is off. The parallel connection makes the transmission gate conduct the whole voltage range from 0 to V_{DD}.

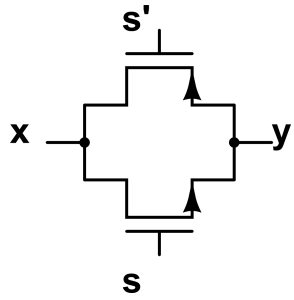


Figure 1: Transmission Gate

3.4.3 Clocked Inverter

The clocked inverter is like an ordinary inverter except that it is controlled by a set-signal. The schematic of the clocked inverter can be seen in Figure 2, where signals c and c' are the set-signals. These are normally connected to the clock signal so that the inverter either inverts while the clock is low or while it is high. The control transistors are placed between the input-signal transistors to remove the unwanted effects of charge sharing, which decreases the output swing and can cause instability[29].

3.4.4 Minority3 Gate

The Minority3 gate is a logic circuit with three inputs which outputs logic one if the minority of the inputs are logic one[20]. The truth table can be seen in

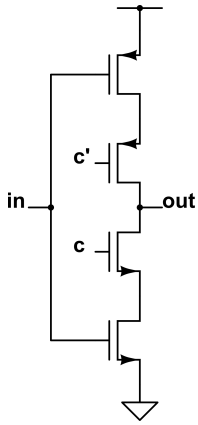


Figure 2: The Clocked Inverter

Table 1. In addition to the minority functionality, the minority-3 gate can take the form of a NAND by forcing one input to ground, it can get the functionality of a NOR by forcing one input to V_{DD}, and it can be an inverter by forcing one input to ground and one input to V_{DD}. This gives the opportunity to create a D flip-flop composed of only minority-3 gates and inverters as proposed in [8]. The 10 transistor Minority3-gate is chosen because it is relative reliable in subthreshold[8].The Minority3-gate schematic is shown in Figure 3.

X	Y	Z	Output
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 for the Minority-3 gate

The NAND-coupled Minority3-gate can be seen in Figure 4, with one input pin removed, and the involved transistors coupled to ground.

3.4.5 D-latches

A D-latch is a logic device that can hold the value of a single input bit[30]. A standard D-latch is transparent, which means that a change on the input D will be seen on the output Q and $\bar{Q}$ immediately after the logic delay time. To control

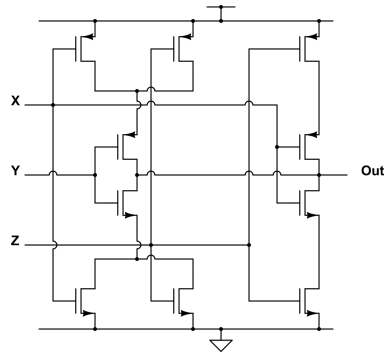


Figure 3: The Minority3 Gate

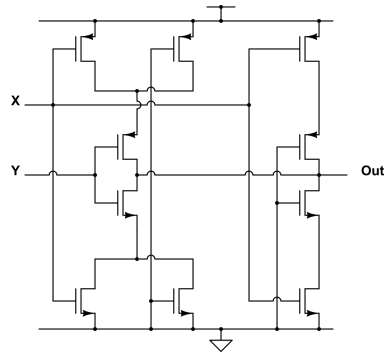


Figure 4: The NAND-coupled Minority3 Gate

when the D-latch should read a new input and propagate it to the output, an Enable signal is implemented. By setting a clock signal to the Enable input of the gated D-latch, the input value at D will be propagated to the output Q only while the clock signal is high.

The typical symbol for a gated D-latch is shown in Figure 5.

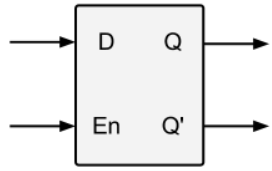


Figure 5: Gated D-latch symbol

The D-latch circuit can be constructed by four 2-input NAND-gates as seen in Figure 6. This configuration is based on an D-latch which is controlled by the Enable signal. By setting Enable to 0, the input values are blocked by setting the D-latch to a hold state.

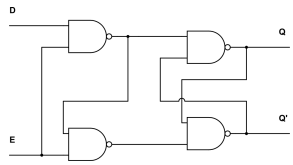


Figure 6: Logic Diagram for NAND-based D-latch

Another possibility is to construct D-latches with inverters, clocked inverters or transmission gates. This construction uses significantly less transistors and have a shorter critical path, so they should have a lower propagation delay.

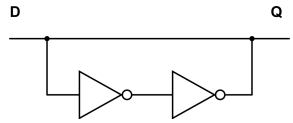


Figure 7: Logic Diagram for inverter-based latch

Figure 7 shows the idea of the inverter-based D-latch without Enable-input[30]. The inverters work as a feedback loop providing the desired latching, but need different sizing. It is also slow since the bistable circuit tries to hold onto the stored value when a new value is present on the input. A solution to this problem is to add either transmission gates, or replace the inverters with clocked inverters. This will give the ability to control the loading, and break the feedback loop while a value is stored.

A D-latch with transmission gates is shown in Figure 8. The transmission gates are oppositely clocked so that the feedback is open as the clock is high, and value D is stored in the inverters. As the clock goes low, the feedback is closed, and the value is stored.

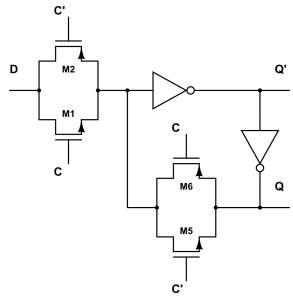


Figure 8: CMOS Diagram for inverter-based D-latch with Transmission Gates [30]

The same functionality can be made by using clocked inverters instead of

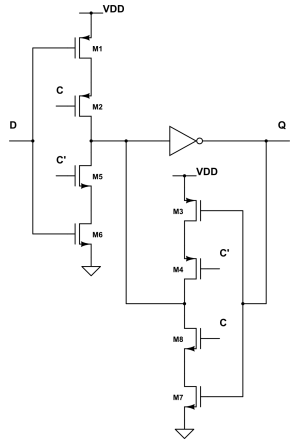


Figure 9: CMOS Diagram for clocked inverter-based D-latch[30]

transmission gates, as seen in Figure 9. This circuit is called the C^2MOS latch.

3.4.6 The D flip-flop

This section contains information about the basic functionality of a D flip-flop, and describes the different D flip-flop implementations used in this Thesis.

A D flip-flop holds the value of a single bit, as the latch, but is not transparent. For standard D flip-flops, the stored value is updated when a rising edge event occurs at the Clock input. The standard symbol for a D flip-flop is shown in Figure 10.

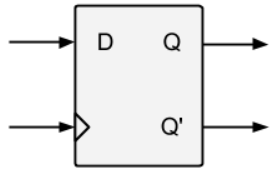


Figure 10: The D flip-flop symbol

A basic D flip-flop is designed by using two gated D-latches with opposite clock signals. The first is called the master latch, and the second is called slave latch. Figure 11 shows this design.

The master latch detects changes in the Data signal at the low clock level, and propagates the signal to the input of the slave latch. As the clock signal switches to high, the slave latch propagates this input value to the D flip-flop output Q. If the clock is non-overlapping, the D flip-flop will be non-transparent, which is required for normal functionality.

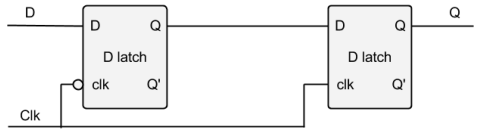


Figure 11: Master-slave D flip-flop, Master latch to the left, and Slave latch to the right

3.5 D flip-flop Design Structures

There are different ways to implement the functionality of a D flip-flop. Some have higher switching frequencies, some have a lower power consumption, and others may be more robust against process variations and mismatch. This section will describe the D flip-flops used in this Thesis.

3.5.1 The Classic NAND-based D flip-flop

The Classic NAND-based D flip-flop is shown in Figure 12. It consists of five two-input NANDs and one three-input NAND. The design can be split into a master latch and a slave latch to ease the delay measurements, and to get a better understanding of the functionality. The master latch output, and the slave latch input is node $P3$ in Figure 12.

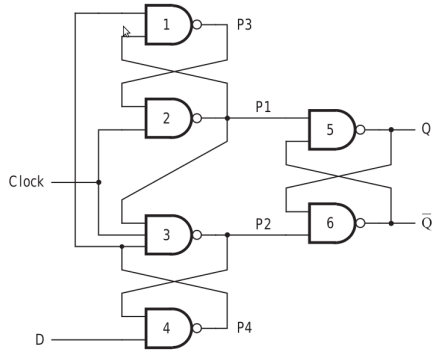


Figure 12: Classic NAND-based D flip-flop [12]

The transistor count for this design with standard CMOS logic is $5 \cdot 4$ for two-input NANDs, and $1 \cdot 6$ for the three-input NAND, which sums up to $5 \cdot 4 + 6 = 26$ transistors.

3.5.2 Minority3-based D flip-flop

The Minority3-based D flip-flop will be referred to as the the Min3, and is constructed by using the Minority3 gates as mentioned in Section 3.5.2. Two

Minority3-based D-latches as described in [8], are used as master and slave latch to create the Minority3 D flip-flop. The schematics can be seen in Figure 13, where the three-input objects are minority-3 gates.

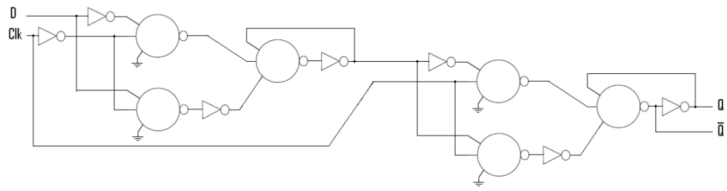


Figure 13: Min3-based D flip-flop Schematics [8]

3.5.3 C^{2}MOS D flip-flop

C^{2}MOS stands for clocked CMOS and is an inverter-based master-slave D flip-flop which uses clocked inverters to control the loading of a new value, and to break the feedback loop. It is based on using two clocked inverter-based D-latches, as described in Section 3.4.5. The schematic for the C^{2}MOS is shown in Figure 14.

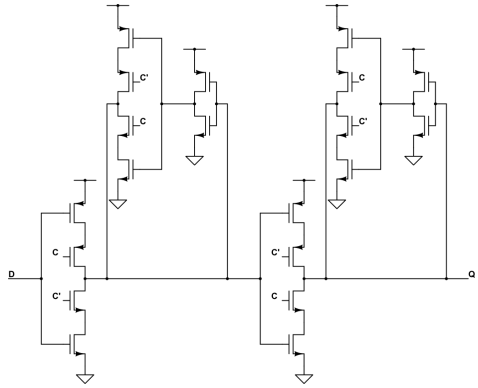


Figure 14: C^{2}MOS D flip-flop structure

3.5.4 PowerPC 603 D flip-flop

The PowerPC 603 D flip-flop were introduced in the PowerPC 603 RISC Microprocessor, designed for low power operation and battery-saving power management modes[15]. The structure is two identical D-latches which forms a master and a slave latch. The D-latches are similar to the clocked inverter-based D-latch as described in section 3.4.5 and Figure 9, but uses a transmission gate on the input to load a new value instead of a clocked inverter. This design is shown in Figure 15.

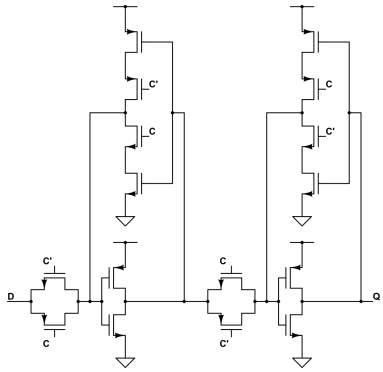


Figure 15: The PowerPC 603 D flip-flop structure

3.6 Timing and Delay

Both master and slave latches have propagation delays which must be considered before implementing them in designs. These delays limit the maximum switching frequency of the clock signal and input data signal. Two different delays are used for these D flip-flops:

The Setup Time t_{su} is the delay through the master latch, and is the minimum time needed for the data signal to be stable on the input D before the rising-edge clock event occurs[19]. For a master-slave D flip-flop, the input signal needs to be available at the slave latches input before the rising edge of the clock. A violation of the setup time can give incorrect data at the output, or set the D flip-flop to a metastable state.

The Propagation delay t_{co} is the delay through the slave latch. It is the time from a rising edge on the clock and to a new input value is stable on the output Q.

Figure 16 shows the two delays on a waveform.

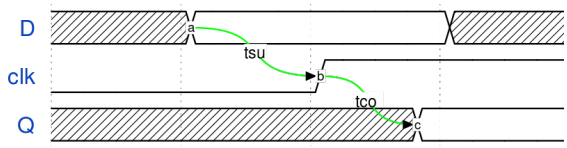


Figure 16: The two delays in a waveform, where t_{su} is the setup time, t_{co} is the propagation delay

3.7 Transistor Layout

The layout is the the physical drawing of the circuit. It is how the chip will be fabricated with different silicon layers, metal wires, contacts and vias, to get the desired functionality. The introduction of non-ideal metal wires, electric fields, parasitic capacitances, and process variations to the circuit typically affects its total functionality, increases the propagation delay, increases the power consumption, and could cause a non-functional physical circuit. However, there are some design procedures and techniques which could decrease these effects. These will be described in this Section.

3.7.1 Substrate Connection

The source of a MOS transistor should be connected to the bulk or substrate to prevent second-order effects like the body effect[18]. The body effect causes the threshold voltage for a transistor to increase given by the equation 14

$$V_{tn} = V_{tn0} + \gamma(\sqrt{V_{SB} + |2\phi_F|} - \sqrt{|2\phi_F|}) \tag{14}$$

where V_{tn0} is the threshold voltage with zero V_{SB}. γ is the body-effect constant and is given by

$$\gamma = \frac{\sqrt{2qN_A K_S \epsilon_0}}{C_{OX}} \tag{15}$$

Equation 14 shows that the threshold voltage increases for a given transistor as the source-to-substrate reverse-bias voltage increases[18]. A good technique to reduce the body effect is to ensure a low resistance between the source terminal and the substrate by placing multiple n-taps/p-taps close to the transistor. Missing substrate connection or high substrate-to-source resistance can cause transistor latch-up[16].

3.7.2 Dummy transistors

Transistors operating in the subthreshold region are very susceptible to mismatch from process variations and layout irregularity. At the end of a long row of active transistors in layout, the end-transistors does not see the same surroundings as the other active transistors. Higher regularity can be achieved by placing dummy transistors at the end of these transistor rows, so that all active transistor has a neighbour transistor at both sides. Figure 17 shows the pMOS row of a layout block with active transistors and dummy transistors at the end. Blocks like the inverter and the transmission gate, which has only one nMOS and one pMOS, can benefit of having dummy transistors on the top and bottom to increase the regularity and cause equal surroundings for the transistors.

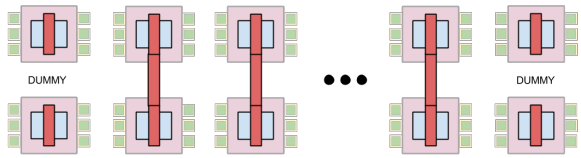


Figure 17: Dummy transistors on the end of a pMOS transistor row

3.7.3 Design Rules

For every transistor technology, there are design rules which must be satisfied to verify the correctness of the mask net. By violating these rules, close objects could be shorted, an object could be too thin or split, or an object could be misplaced[30]. These design rules are often published by the technology provider. The design rules for the STMicroelectronics $65nm$ technology can be seen in Table 2.

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
Via-x (nm)	210	100/110
M-x (nm)	210	100/110
PO-CO distance (nm)	210	100/110
N+/p+ distance (nm)	190	100/110

Table 2: Design Rules for $65nm$ STMicroelectronics [7]

3.7.4 Parasitic Extraction

The introduction of non-ideal metal wires, parasitic capacitances and electric fields to the circuit will affect the total functionality. The resistance in the poly-silicon and the metal wires can give small voltage drops, so small voltages differences are present at the terminals at matched transistors. The parasitic capacitances normally affects the circuit in a larger degree, since the capacitances causes an increase in both dynamic power consumption and propagation delay, which is seen in Expression 7 and Expression 5.

Cadence uses Mentor Graphics Calibre xRC or xACT 3D to extract post-layout parasitic data. These two tools use different procedures to extract the parasitics, so the results may differ from each other. From Mentor Graphics web page [3] [2], the xACT 3D should be the most accurate one, as it uses 3 Dimensional field models to calculate the parasitics.

4 Selecting D flip-flop Implementation and Design in Schematics

This section contains argumentation for the D flip-flop selection, it describes the building blocks and D flip-flops design procedure, and at the end explains D flip-flop modifications to improve the layout regularity.

4.1 Selecting D flip-flop Designs

There are static and dynamic D flip-flop designs, which has different advantages and disadvantages depending on the application.

Dynamic D flip-flops can be designed with less transistors, they typically have lower power consumption and lower propagation delay than the static designs. However, dynamic design uses the transistors parasitic capacitances to store a value for a short period of time, which requires the circuit to be clocked in a frequency such that the data is updated before it is lost due to leakage currents[30]. For low switching speeds, dynamic designs can output invalid data.

Static D flip-flops stores the data value in a buffer, and does not need updates regularly to keep the data. Because of the buffer, Static D flip-flops are typically larger, uses more transistors, they have a higher power consumption and a higher propagation delay. However, Static D flip-flops should be more robust, since they do not depend on a constant switching to maintain their functionality, and are less dependant on the propagation delay. In addition is the leakage current very sensitive to temperature variations in subthreshold operation[5], which makes Dynamic D flip-flops less suitable.

Based on the discussion above, only static D flip-flops are chosen for comparison, because of their higher reliability when it comes to timing and process variations. The chosen D flip-flops are as follows:

- Classic NAND-based D flip-flop
- Minority 3-based D flip-flop
- Minority 3-based D flip-flop without set input
- C^2MOS D flip-flop
- PowerPC 603 D flip-flop

4.1.1 Classic NAND-based D flip-flop

The Classic NAND-based D flip-flop has the well known structure as seen in Figure 12. It consists of seven two-input NANDs, and one inverter if the three-input NAND is converted to two-input. A three-input NAND should be avoided as it has larger fan-in, which has higher propagation delay, and causes larger nMOS/pMOS imbalance, which increases VDD, min[5].

This sums up to $7 \cdot 4$ transistors $+2$ transistors $= 30$ transistors, which is 4 transistors more than using a three-input NAND. The Classic NAND D flip-flop has a relatively high transistor count and should not compete with the best when it comes to performance and power consumption, but is used as a reference for the other D flip-flops.

4.1.2 Minority3-based D flip-flop

The Min3 D flip-flop is designed as in [8], with 6 Min3 gates where 4 are NAND coupled, and 7 inverters. In addition is a Set input implemented to gate the D flip-flop. The Set Input introduces one more NAND-coupled Min3 and two more inverters, which sums up to 7 Minority3-gates and 9 inverters. With 10 transistor Minority3-gates the total transistor count is $7 \cdot 10$ transistors $+9 \cdot 2$ transistors $= 88$ transistors. Earlier simulations of the Minority3-based SR-latch shows that it has a higher yield than a traditional SR-latch using NANDs and NORs at low Supply Voltages[8]. The Full Adders used to implement a Ripple-Carry Adder in [32] is constructed by Minority3 gates, and were able to function down to the supply voltage of $106mV$ with the help of body biasing. A Minority3-based modulation/demodulation system developed for Q-free ASA showed a correct functionality at $185mV$ at $6MHz$ without body biasing[32]. These results indicate that the Minority3 is relatively robust against process variations. The result of these simulation can help improving the Q-free modulation/demodulation system, and will be discussed in the Discussion and Conclusion Section.

4.1.3 Minority3-based D flip-flop without Set Input

A Minority3-based D flip-flop without the Set input is also designed, which gives a more fair compare to the other no-set D flip-flops. It is designed as the Minority3-based D flip-flop, but without the Set input. This reduces the Minority3 gate count to 6 and gives a total of 78 transistors.

4.1.4 C^{2}MOS D flip-flop

The C^{2}MOS D flip-flop is a well known design, used in several D flip-flop comparisons like [6] and [13]. Simulations from these papers show that the C^{2}MOS D flip-flop has good power consumption and delay results compared to other static D flip-flops. The structure is relatively simple, using only inverters and clocked inverters as seen in Figure 14. The total transistor count is 20, which is the second lowest of the chosen D flip-flop structures. The C^{2}MOS D flip-flop can also be found in IC's by Toshiba, like the TC7W74FU[4].

4.1.5 PowerPC 603 D flip-flop

The PowerPC 603 D flip-flop can also be found in several papers comparing different D flip-flops. In [6] it scores best of all static D flip-flops when it comes to delay, power consumption, PDP and EDP. [14] also compares the PowerPC 603 to other D flip-flops, with good results. It has the lowest transistor count of

all D flip-flops in this Thesis with a total of 16 transistors. The structure can be seen in Figure 15.

4.2 Designing Schematics for D flip-flop building blocks

The D flip-flop structures are designed in Cadence Virtuoso on transistor level. The design procedure starts with drawing the schematics for each building block like inverters, clocked inverters and NAND-gates, which are connected together to form the D flip-flops. Each of these building blocks are balanced so that both nMOS and pMOS have the same strength as mentioned in Section 3.2. The balancing procedure is a bit different for some building blocks, as mentioned later in this Section.

To able to measure the setup time t_{su} of the D flip-flops, the node X is inserted at the output of the master latch, as shown in Figure 18. The X node is placed between the master and slave latch for all D flip-flops. For the Classic NAND D flip-flop this point is node $P3$ in Figure 12. The measurement of the t_{su} and t_{co} can be seen in Figure 19.

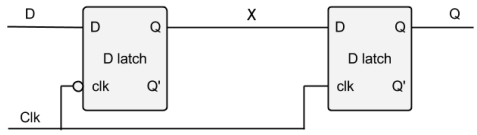


Figure 18: Master-Slave D flip-flop with X node

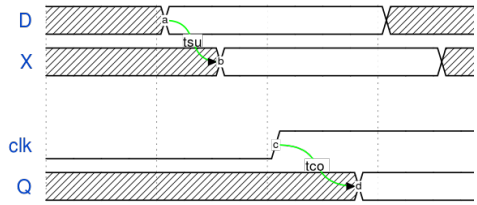


Figure 19: Measurement of t_{su} and t_{co}, with the X node

4.2.1 Sizing of Transistors

The transistors for the different D flip-flop block are sized using the same procedure. The length of both nMOS and pMOS transistors are set to $1.5L_{min} = 90nm$ to reduce the variation of propagation delay and other process variation consequences, as proposed in [11]. The nMOS width for all transistors are set to $200nm$, which is above twice the length of the transistor. The pMOS width is determined while balancing the the logic, so that the nMOS and pMOS transistor have the same drive strength.

The transistor size balancing procedure for a building block depends on its functionality. The idea is that an input signal of $V_{DD}/2$ should give an output signal of $V_{DD}/2$, if a change on that single bit would change the output of the block. For an inverter, a logic change on the input will always cause a change on the output. So by putting $V_{DD}/2$ on the input, the pMOS width can be sweeped to find the balance point where the block also outputs $V_{DD}/2$.

For more complicated blocks like the NAND-gate and the minority-3 gate, the procedure is a bit more complicated, and can have more than one balance points depending on the input values. To understand which input values that needs to be tested to balance the block, the blocks truth table must be examined.

4.2.2 Deciding the Supply Voltage

As the threshold voltage for SVTGP transistors is between $|314.6mV|$ and $|344.3mV|$, the supply voltage should be kept under $300mV$ to ensure subthreshold operation. All blocks and D flip-flops in this design are balanced for $250mV$ to make sure the transistors are in weak inversion, and at the same time function under normal circumstances.

4.2.3 Designing the Inverter

The Inverter is designed as a standard Inverter with one pMOS and one nMOS as shown in Figure 20.

By sweeping the pMOS width while having $V_{DD}/2$ on the input, the output shows $V_{DD}/2$ at $310nm$ as shown in Table 3

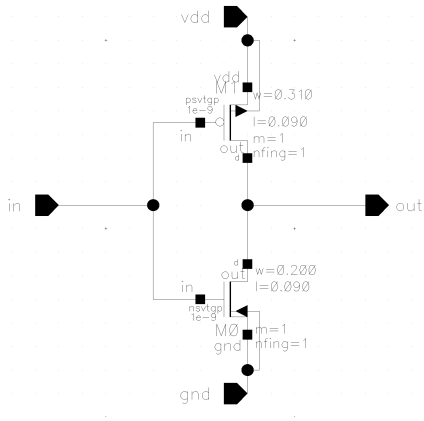


Figure 20: Clocked Inverter schematics

dimension	length
nmos width	200 nm
nmos length	90 nm
pmos width	310 nm
pmos length	90 nm

Table 3: Inverter dimensions

4.2.4 Designing the Clocked Inverter

The clocked inverter is designed as shown in Figure 21, with the dimensions as shown in Table 4. Figure 21 shows that the input transistors are placed closest to the rail, which prevents charge sharing, which will cause a more stable

operation[29]. The balancing procedure is the same as for the inverter. The pMOS width is sweeped from $100nm$ to $1\mu m$ to find the point where the clocked inverter outputs $V_{DD}/2$ while inputting $V_{DD}/2$.

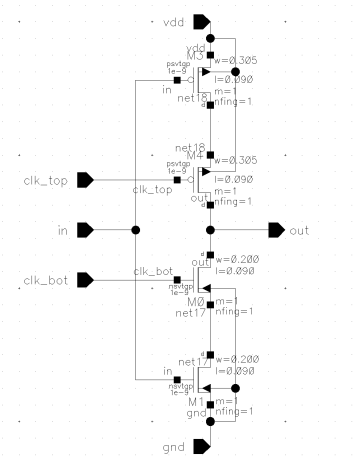


Figure 21: Clocked Inverter schematics

dimension	length
nmos width	200 nm
nmos length	90 nm
pmos width	305 nm
pmos length	90 nm

Table 4: Clocked inverter dimensions

4.2.5 Designing the Transmission Gate

The transmission gate can not be balanced in the same way as the other blocks, as the transistors are coupled in parallel. The size of the transistors should be set equal to the other designs to improve regularity. The transmission gate schematics can be seen in Figure 22.

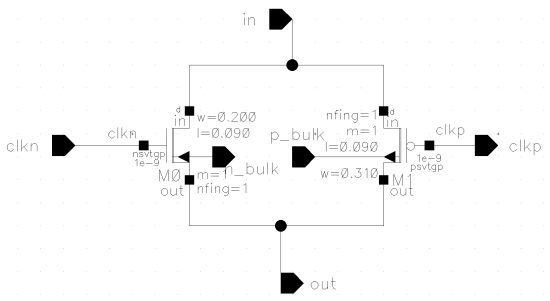


Figure 22: Transmission Gate Schematics

4.2.6 Designing the Minority3-Gate

The Minority3-Gate is designed as described in Section 3.4.4, with 10 transistors. The balancing procedure for this gate is a bit more complicated than the previous ones since there are more input pins, and most likely several balance points, depending on the input values. The procedure to find these balance points is

found by looking at the Minority3 truth table (Table 1). If one input is logic 0 and one input is logic 1, the third input will switch the output if itself is switched. Therefore, this third input must be set to $V_{DD}/2$ and the p-Width sweeped to find the balance point. Table 5 shows the different inputs, and the balance point for each input combination.

X	Y	Z	pmos bal. width
1	0	sweep	$313nm$
0	1	sweep	$313nm$
0	sweep	1	$304.5nm$
1	sweep	0	$307nm$
sweep	1	0	$288nm$
sweep	0	1	$312.5nm$

Table 5: Simulating the different Min3 inputs to find the pMOS balance point

The average of the Minority3 balance points are used as the final point and can be seen in Table 6.

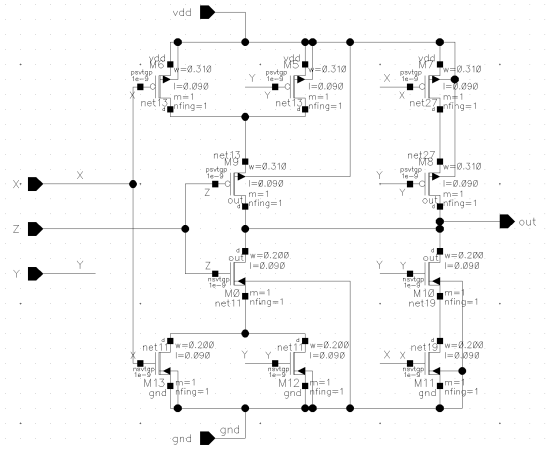


Figure 23: Minority3 Schematics

dimension	length
nmos width	200 nm
nmos length	90 nm
pmos width	306 nm
pmos length	90 nm

Table 6: Minority3 dimensions

4.2.7 Designing the Minority3-based NAND-gate

The Minority3-based NAND-gate is balanced for $250mV$, and the same procedure as for the Minority3-gate is used. The balanced pMOS width is $313nm$ and can be seen in Table 7. The Schematics for the Minority3-based NAND is shown in Figure 24.

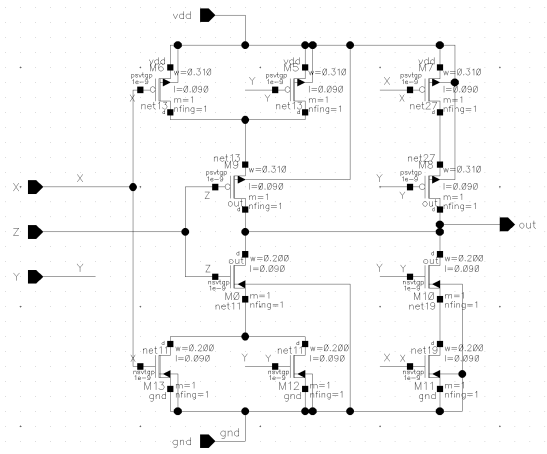


Figure 24: Minority3 Schematics

dimension	length
nmos width	$200nm$
nmos length	$90nm$
pmos width	$313nm$
pmos length	$90nm$

Table 7: Minority3 dimensions

4.2.8 Designing the two-input NAND

The two-input NAND-gate is used in the Classic NAND D flip-flop, and is a standard CMOS NAND as displayed in Figure 25. It is balanced like the Minority3 blocks, and gets a pMOS width value of $261nm$.

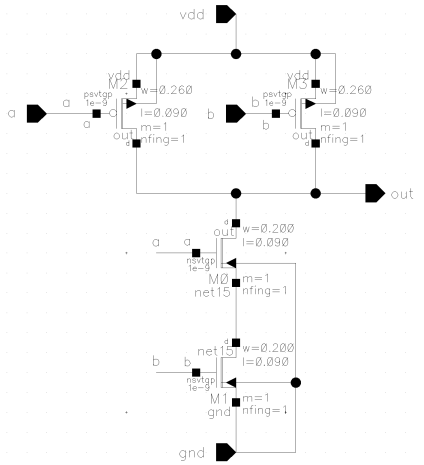


Figure 25: Two-input NAND Schematics

dimension	length
nmos width	200 nm
nmos length	90 nm
pmos width	261 nm
pmos length	90 nm

Table 8: Two-Input NAND dimensions

4.3 Designing the Schematics for the D flip-flops

The building blocks from Section 4.2 are used to design the D flip-flops mentioned in Section 4.1.

4.3.1 Designing the Classic NAND D flip-flop

The Classic NAND D flip-flop is designed as the one described in Section 3.5.1, using the two-input NAND gates as described in Subsection 4.2.8, and can be seen in Figure 12. The only exception is the 3-input NAND, which is transformed into two 2-input NANDs and one inverter, which can be done as seen in Expression 16. This will simplify the layout construction, since it only consists of 2-input NANDs and the already constructed inverter.

$$Y = \neg(A \wedge B \wedge C) = \neg((A \wedge B) \wedge C) \tag{16}$$

The complete Classic NAND D flip-flop design in Schematics is shown in Figure 26.

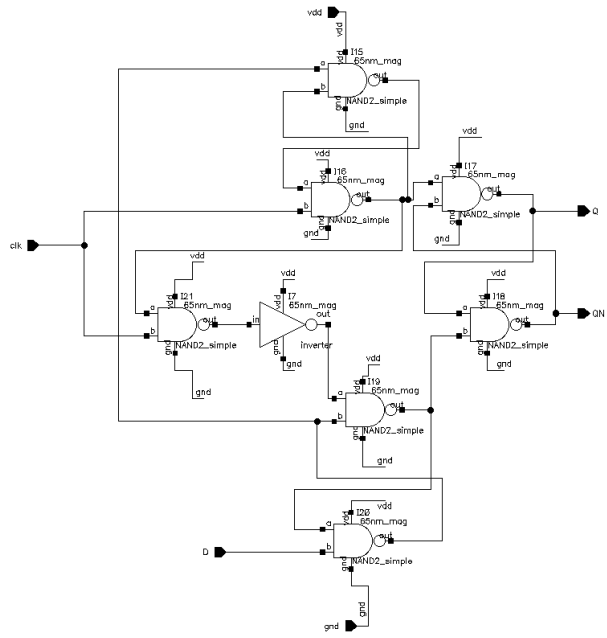


Figure 26: Classic NAND D flip-flop Schematics

4.3.2 Designing the Minority3 D flip-flop

The Minority3 D flip-flop is designed based on the construction described in Subsection 3.5.2, but with a *Set* input, as used in the circuits described in [32].

The *Set* input needs an additional Minority3-based NAND to control the D flip-flop. The Minority3-based D flip-flop schematics can be seen in Figure 27, and consists of two Minority3-based D-latches as seen in Figure 28.

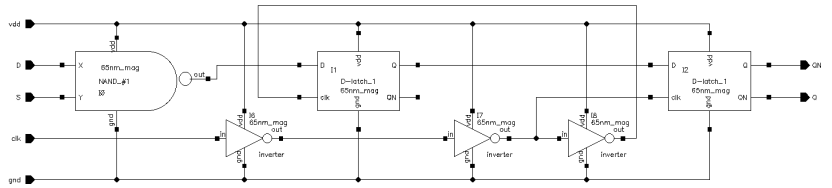


Figure 27: Minority3-based D flip-flop Schematics

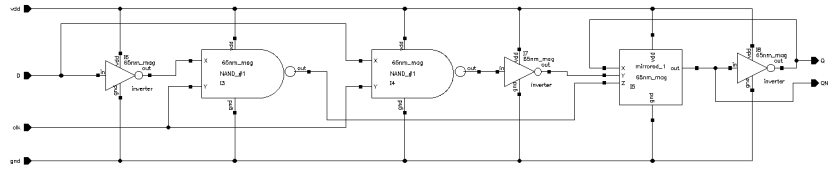


Figure 28: Minority3-based D-latch Schematics

4.3.3 Designing the Minority3 no-set D flip-flop

The Minority3 no-set D flip-flop or Min3ns, is basically the same as the Minority3 D flip-flop from Subsection 4.3.2, but without the *Set* input. The removal of the *Set* input reduces the total amount of transistors in the D flip-flop and should decrease the power consumption and the setup time. The schematics for the Minority3 no-set D flip-flop can be seen in Figure 29, and uses the same Minority3-based D-latches as the other Minority3 D flip-flop, which is described in the Section 4.3.2. Since the Minority3-based NAND is removed at the input, the signal through the D flip-flop will be inverted compared to the Minority3 D flip-flop with *Set* input. This is fixed by setting Q to QN and opposite instead of inserting an extra inverter at the input.

4.3.4 Designing the C^{2}MOS D flip-flop

The C^{2}MOS D flip-flop is designed exactly as the one described in Section 3.5.3. In addition, two inverters are added to create the two clock signal used to control the clocked inverters. The schematic for the C^{2}MOS D flip-flop can be seen in Figure 30.

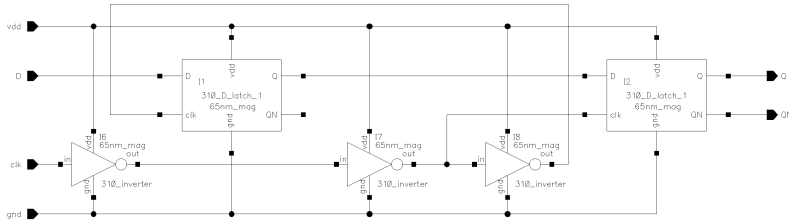


Figure 29: Minority3-based no-set D flip-flop Schematics

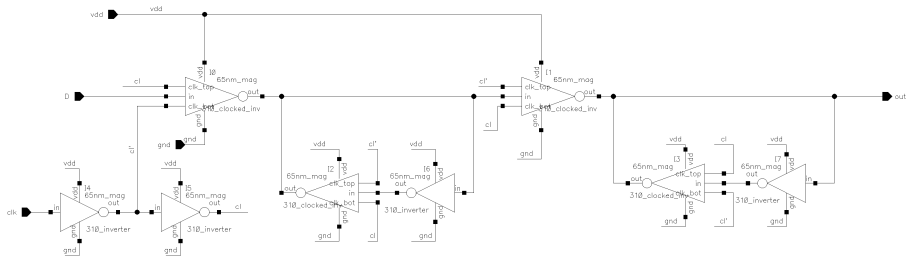


Figure 30: C^{2}MOS D flip-flop Schematics

4.3.5 Designing the PowerPC 603 D flip-flop

The PowerPC 603 is designed as described earlier in Subsection 3.5.4 and Figure 15. Like for the C^{2}MOS, two inverters are used to generate the clock signals used to control the transmission gates and clocked inverters. The schematic for the PowerPC 603 can be seen in Figure 31.

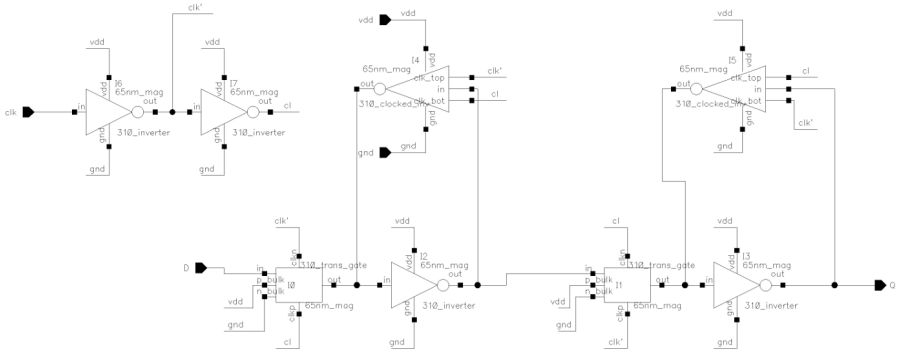


Figure 31: PowerPC 603 D flip-flop Schematics

4.3.6 D flip-flop Transistor Count

With the alterations done in this section, some D flip-flops have increased the transistor count. The total amount of transistors for each D flip-flop can be seen in Table 9.

	Transistors
Classic NAND	30
Min3	88
Min3ns	78
C^2MOS	24
PowerPC 603	20

Table 9: Transistor count for the different D flip-flops in Schematics

4.4 Modifying the Transistor Dimensions To Improve the Regularity

All building blocks except the simple four-transistor NAND have pMOS transistors widths close to $310nm$ when balancing them to $250mV$, as seen in Table 10. The minimum transistor gate width step size is $5nm$, which means that these pMOS widths must be set to either $305nm$, $310nm$ or $315nm$ when implementing them in layout. The similarity of these pMOS widths can be taken in advantage by setting all to the same dimension and achieve great improvement in regularity, and consequently improve the robustness against process variations. The average pMOS width for the Inverter, Clocked Inverter, Minority3, and Minority3-based NAND is $308.5nm$, which can be rounded off to $310nm$. The trade-off by rounding off to $310nm$ is that some block can be slightly imbalanced, but the improvement in regularity, and ease of construction, outweighs this small imbalance. However, the CMOS NAND has a balance point for 250mV at a pMOS width of $261nm$, which is too far away from the $310mV$ area of the other blocks to achieve an overall improvement of modifying the width. The CMOS NAND pMOS width is therefore set to $260nm$. The modified pMOS widths for all building blocks can be seen in Table 11.

Block	pMOS width
Inverter	$310nm$
Clocked Inverter	$305nm$
Minority3	$306nm$
Minority3-based NAND	$313nm$
CMOS NAND	$261nm$

Table 10: Block dimensions

The new pMOS dimensions for the blocks is shown in Table 11. It shows that the deviation for the every block is below $6nm$. The transmission gate is not balanced as the other blocks, so it is set to $310nm$ to improve regularity.

Block	pMOS width
Inverter	$310nm$
Clocked Inverter	$310nm$
Minority3	$310nm$
Minority3-based NAND	$310nm$
Transmission Gate	$310nm$
CMOS NAND	$260nm$

Table 11: Modified Block dimensions

The lower pMOS dimensions for the CMOS NAND can give the Classic NAND D flip-flop a lower power consumption than the others, based on the reduced gate capacitance on smaller gate areas[18]. However, the reduced pMOS width also reduces the subthreshold current, which can give an increased gate delay.

5 Implementation in Layout

This Section contains the techniques and procedures used to implement the design in layout.

5.1 Transistor Layout

Different procedures and techniques can be used to generate good layout with low parasitic capacitances, resistances and low threshold voltage variations. This is important to keep the propagation delay low, the power consumption low, and to make the system robust against process variations. In this section, different layout procedures, and design techniques will be described.

5.1.1 Substrate Connection

The substrate should be connected with multiple p-taps/n-taps placed close to the transistor as shown in Figure 32. This implementation causes low resistance to the substrate and should ensure a source-bulk voltage V_{SB} very close to zero. This is important to keep the threshold variations low, as explained in Subsection 3.1.3.

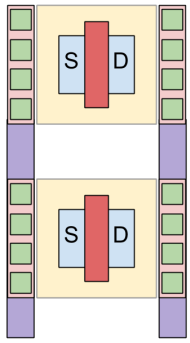


Figure 32: p-tap/n-tap connection

5.1.2 Parasitic Extraction

As mentioned in Section 3.7.4, Cadence uses Mentor Graphics Calibre xRC or xACT 3D to extract post-layout parasitic data. To find the difference between the xRC and the xACT3D parasitic extraction tools, an inverter is designed in layout and run through both extraction tools. The inverter circuit is two inverters in series and the layout can be seen in Figure 33. This simulation is described in Section 7.1.1.

5.1.3 nWell placement and sizing

As mentioned in Section 3.3.3, the distance from the gate of the pMOS transistor to the edge of the nWell can increase the threshold Voltage. This effect (WPE)

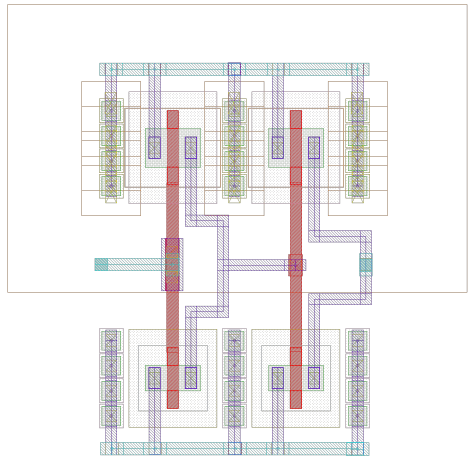


Figure 33: Test circuit for comparing xRC and xACT 3D Parasitic Extraction

is investigated with Cadence, by increasing the size of the nWell for an inverter circuit. It is a copy of an earlier drawn inverter layout, where the nMOS transistor and the pMOS transistor are placed $2\mu m$ from each other, so that the nWell can be increased from minimum size $160nm$ to $2\mu m$ distance from gate. As the nWell increases, its edge gets further away from the pMOS gate and the WPE should decrease (pMOS Threshold Voltage should decrease). However, as the nWell increases it gets closer to the nMOS, and its Threshold voltage could be affected. Figure 34 shows the layout and the nWell as the big the Orange rectangle at the top. The nWell distance is $1\mu m$ from both nMOS and pMOS in this Figure.

5.1.4 Regularity

Regular designs are important to maintain a high yield when it comes to printability. For sub-$100nm$ technology, the regularity becomes more important to maintain manufacturability[21]. [28] shows that the standard deviation of performance can improve by 16% by using regular poly patterns and dummy poly between blocks. The effect can be even greater by using more techniques described in this Subsection. However, some of these techniques causes the layout area to increase, so there is a trade-off for the improved yield.

Below are some techniques to improve the robustness and decrease some layout parasitics.

- Regular polysilicon patterns should be used for the nMOS and pMOS transistors. The geometry of these should be identical for high transistor matching. This also means that all transistors must be placed in the same direction[16]. All polysilicon connections to the metal wires are placed such that the distance to both nMOS and pMOS are the same.

- Place transistors in parallel orientation to reduce stress and tilt-induced variations[16]. Also keep the of layout transistors as compact as possible.

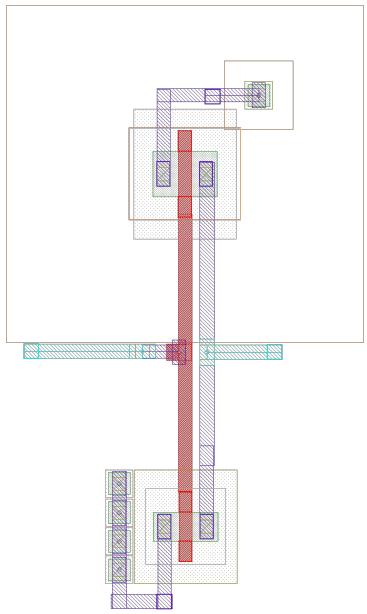


Figure 34: Inverter layout for WPE simulation, nWell edge distance is $1\mu m$ from nMOS and pMOS

The use of same-sized transistors, as mentioned in Section 4.4 eases this parallel orientation.

- Put dummy transistors at the end of transistor arrays to increase regularity for the active transistors as described in Section 3.7.2. Connect the dummy gates to the back-gate of the transistor (Ground for nMOS and V_{DD} for the pMOS)[16].
- Do not route metal or place contacts on top of active silicon area[16].
- Keep pMOS gate $> 2\mu m$ and nMOS gate $> 500nm$ from the nWell edge to prevent threshold voltage increase as mentioned is Subsection 5.1.3, and found in the results in Subsection 8.1.2.
- Keep metal wire layer 1 in vertical direction and metal wire layer 2 in horizontal direction to ease the routing of the D flip-flops and reduce the capacitance from parallel wires.

5.1.5 Design Rules

The design rules for the $65nm$ STMicroelectronic which are defined in Section 3.7.3 are used for the layout. In addition are these design rules checked by the Design Rule Check in Cadence Virtuoso. Beacuse of the minimum nWell edge distance of $> 2\mu m$ for the pMOS, there is much room between the nMOS and pMOS transistors that give lots of room for metal wires and vias to be placed

between the transistors. Figure 35 shows the transistor positions, distances, and rail placement for layout. This configuration is kept for all layouts, to achieve a good regularity. The $700nm$ distance between the transistors in the horizontal direction makes room for metal wires to go vertically between the transistors without using metal layer 2. Between the two rows of pMOS transistors and between the two row of nMOS transistors, there is $500nm$ space to route two wires horizontally, or three wires without room for vias.

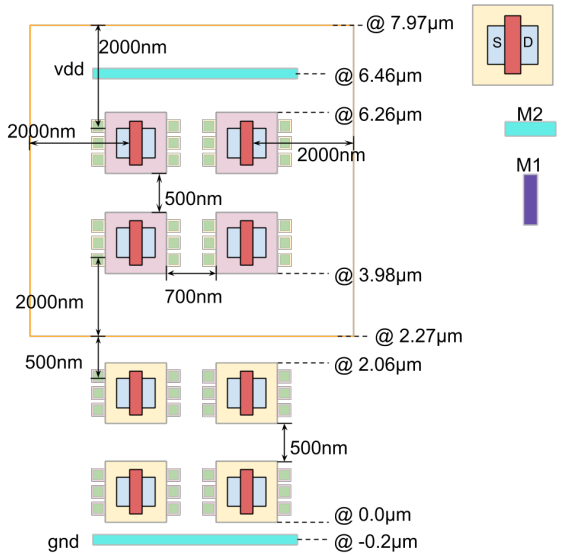


Figure 35: Layout Placement, Distances and Positions

Figure 36 shows the wire positions between the nMOS and pMOS transistors. With $170nm$ distance between each wire, there is room for vias to be placed on the same horizontal position without breaking any design rules. All horizontal wires (Metal layer 2) are set to a width of $100nm$, and all vertically wires (Metal layer 1) are set to a width of $90nm$, as described in the design rules. The only exception is the the vertically wires used for the substrate connection for both nMOS and pMOS. They are kept at a width of $170nm$ to match the guard ring thickness, and to minimize the source-bulk voltage V_{SB}. These wires can be seen in Figure 32.

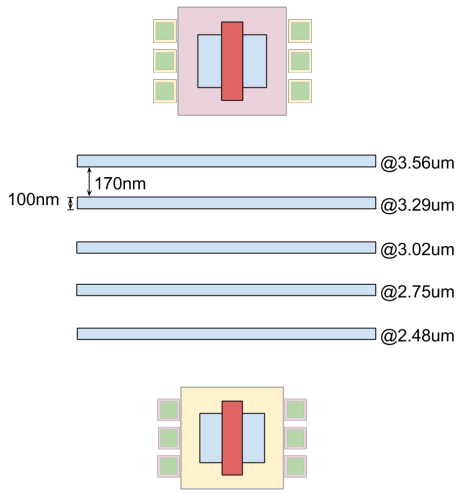


Figure 36: Wire positions in Layout

5.2 D Flip-Flop Implementation in Layout

This Section describes the procedure of implementing the D flip-flop designs from schematic to layout, and the layout design solutions for the building blocks and D flip-flops.

First, the D flip-flop building blocks described in Section 4.2 are implemented in layout. The implementation are based on the design rules and regularity rules described in Section 5.1. The D flip-flop layout is then implemented with the already designed building blocks. The layout for all building blocks and D flip-flops are shown in Appendix B.

The inverter and transmission gate are both two-transistor blocks, which means they must be implemented with dummy transistors to improve regularity. The dummy transistors are placed next to the rail on the top and the bottom, with all transistor terminals shorted to the closest rail. The Minority3-gate and the Minority3-based NAND-gate are also implemented with dummy transistors, to achieve regularity.

The Classic NAND Dff that uses CMOS NAND blocks will have a problem regarding regularity in layout because of different sized blocks. One solution is to use dummy transistors between the 260nm pMOS and the 310nm pMOS in the inverter, so that the active transistors only see same size transistors as their neighbours. This can also be done for the transistors at the end of the block row, so that the next blocks sees $310nm$ transistors. However, this is not done in these layouts and simulation, but should be noted for future improvements.

Table 12 shows the layout area of the D flip-flops and the total transistor count, with the number of dummy transistors. In addition, the area with the complete nWell in the horizontal direction is shown.

	Area	Area with nWell	Total Transistors (Dummy)
Classic NAND	$92.23\mu m^2$	$114.18\mu m^2$	32(2)
Min3	$345.40\mu m^2$	$367.35\mu m^2$	120(32)
Min3ns	$310.88\mu m^2$	$332.83\mu m^2$	108(30)
C^2MOS	$92.23\mu m^2$	$114.18\mu m^2$	32(8)
PowerPC 603	$92.23\mu m^2$	$114.18\mu m^2$	32(12)

Table 12: D flip-flop Area, Size and Transistor Count in Layout

6 Testbenches

The simulation of the D flip-flop schematics and layouts are done with Cadence Virtuoso Spectre Circuit Simulator. The simulations are executed on testbenches designed to give the correct input signals at the correct time, and to make the D flip-flops function as if they were implemented in a real system.

6.1 Balancing Testbench

The balancing testbench were made to find the nMOS-pMOS balance point for each D flip-flop at a specific supply voltage, as mentioned in Section 4.2.1. The testbench is displayed in Figure 37, with a Minority3 NAND gate being tested. Signal D is the input signal, set to $V_{DD}/2$, and Q is the output signal which can be plotted on a graph to find the nMOS-pMOS balance point. This testbench is also used to measure the effects of different parasitic extraction tools.

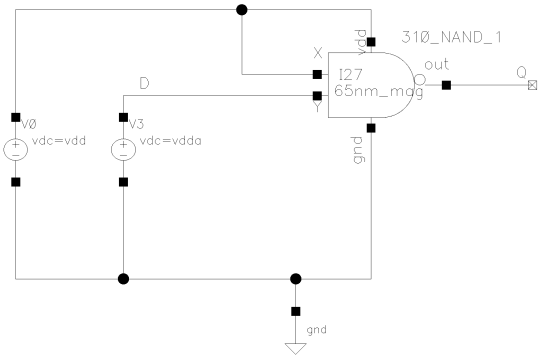


Figure 37: The Balancing Testbench

6.2 The Delay Testbench

The delay testbench is used to find the setup time and propagation delay for the D flip-flops at different temperatures and supply voltages. The testbench is equipped with two inverters as a driver at every D flip-flop input pin, to get a more natural input signal. This is shown in Figure 38, which shows the delay testbench. In addition to the drivers, the testbench has inverter loads at the output Q of each D flip-flop. The loads are suppose to give the D flip-flops a typical load capacitance, as if they where implemented in a real system.

The delay testbench is set up with one voltage pulse source for the D signal, and one for the clk signal. Both these pulse sources output a square signal modified by variables in the testbench.

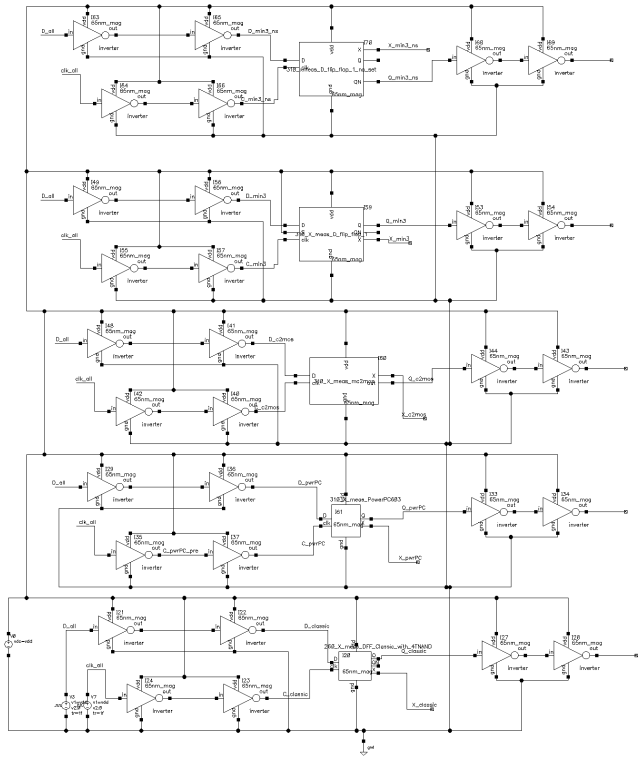


Figure 38: The Delay Testbench

6.3 The Power and PDP Testbench

The power and PDP testbench is used to measure the static power consumption, the total power consumption, and the PDP power data. It is equal to the delay testbench in many ways, but it has unique pulse sources for every D flip-flop and unique voltage supplies to each D flip-flop as seen in Figure 39. The unique pulse sources makes it possible to set a unique D signal and clk signal for each D flip-flops as they have different limits when it comes to switching speed. The unique voltage supplies make it possible to measure the exact power consumption for each D flip-flop, by multiplying the supply voltage with the current drawn from the D flip-flop.

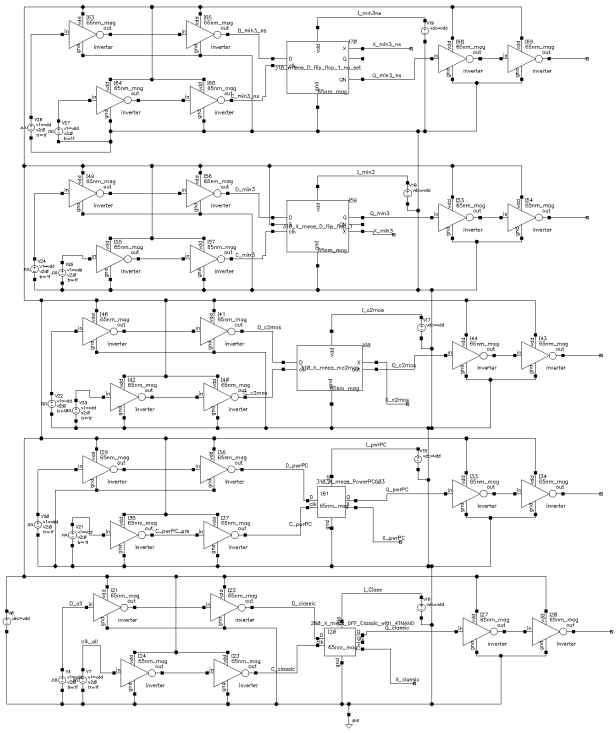


Figure 39: The power testbench

7 Simulations

This Section describes the different tests and simulations on the D flip-flops. These include a Basic Functionality test, delay measurements, power measurements, pdp-calculations and Monte Carlo analysis off the Delay.

All delay, power and pdp simulations are executed for three different temperatures. One is normal temperature condition at $27°C$, one is extreme cold temperature condition at $-40°C$, and the last is extreme warm temperature condition at $80°C$. These simulations are done for the supply voltages from $100mV$ to $300mV$ with voltage steps of $20mV$. All output and input pins on the D flip-flops in the testbenches has initial condition set to $0V$ to make sure they start at zero.

7.1 Transistor Layout

The simulation procedure for the Parasitic Extraction test is presented first in this Section. Secondly comes the nWell edge and placement simulation procedure.

7.1.1 Parasitic Extraction

The parasitic capacitance and resistance of a two-inverter buffer as shown in Figure 33 is extracted by both xRC and xACT3D tools, and compared directly, and through a testbench to find the delay and power consumption differences. The balancing testbench were used to test the circuit at $250mV$ and $27°C$.

7.1.2 nWell placement and sizing

The Well-Proximity-Effect is described in Section 3.3.3, and is measured as the function of distance from the pMOS-gate, and the nMOS-gate. The minimum nWell distance from the pMOS is $160nm$ as standard for this technology, so that distance will be the lowest for this simulation. By simulating the schematics for the circuit, the absolute threshold voltage is $|315.6mV|$ for the pMOS, and $|344.3mV|$ for the nMOS, which is the ideal values. The measurements are done by increasing the nWell size from the minimum distance $160nm$ from the pMOs active gate, to $2\mu m$, with $200nm$ steps.

7.2 D flip-flop functionality

The basic functionality of the different D flip-flops are first tested by inputting a clock signal clk and a data signal D. The output signal Q for all D flip-flops are plotted in a chart to confirm their functionality. The input signals can be seen in Table 14 in Section 7.11.

7.3 Delay Simulation

The Setup time t_{su} and the propagation delay t_{co} is measured by using the Delay Testbench from Section 6.2. The inputs used are the clock signal clk and the data signal D. The output signals used to measure are X, which is the output

from the master latch, and the D flip-flop output signal Q, which also is the slave latch output.

The waveforms for the input signals and the expected output signals can be seen in Figure 40. The clk period must be minimum 4 times the worst case delay for the slowest D flip-flop in the measurement, but should be even higher to ensure correct behaviour. The D signal is set to $2 \cdot clk$, and the D-delay is set to $-3/4 \cdot clk$.

Since all D flip-flops except the Classic NAND and the Min3 no-set inverts the data signal through the master latch, the X signal is inverted compared to the value it holds as seen in the waveform. As mentioned, the testbench is used to measure the setup time (tsu) and the propagation delay (tco). The setup time is measured as the delay between the D signal reaches 80% of its new value until the X signal reaches 80% of its new value. The propagation delay is measured as the delay between the the clk signal is 80% at rising edge until the output Q signal is 80% of its new value. These measurements are done for both falling and rising edge, as it could be differences in the delay.

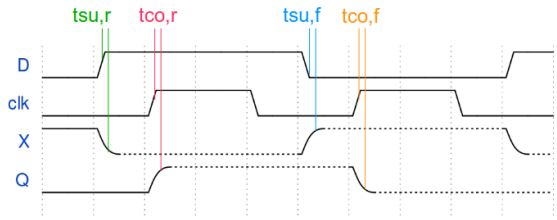


Figure 40: Delay testbench waveforms, tsu is setup time, tco is propagation delay, r is rising edge, f is falling edge

The Delay simulations will be presented as:

- Master Latch Delay (Setup time t_{su})
- Slave Latch Delay (Propagation delay t_{co})
- Total delay (Setup time + Propagation delay)
- Maximum Frequency ((2·Worst Case delay)$^{-1}$)

7.4 Maximum D flip-flop frequency based on maximum delay

The maximum delay is the worst case delay of the Setup time and Propagation delay for rising and falling edge. The absolute highest clock frequency the D flip-flops can operate under is 2· Maximum delay. This is because the clock needs to be stable while the master latch and slave latch propagates data, and the highest propagation time a D flip-flop can have is its maximum delay. The maximum delay will be calculated for all D flip-flops and plotted as a function of the supply voltage.

7.5 Static Power Simulation

The static power consumption is measured by having static signals at the inputs of the D flip-flops. Four different combinations of the input signals D and clk to will be simulated to check for differences, and to find the worst case scenario. These combinations can be seen in Table 13.

D	clk	Name
0	0	DL CL
0	1	DL CH
1	0	DH CL
1	1	DH CH

Table 13: Input combinations for static power consumption measurements

The Static Power Consumption simulations will be presented as:

- DL CL - D low, clk low
- DL CH - D low, clk high
- DH CL - D high, clk low
- DH CH - D high, clk high

7.6 Total Power Consumption

The total power consumption is the sum of static power consumption and dynamic power consumption. The total power consumption measurement is meant to be a fair comparison of all D flip-flops, so the switching frequency clk is set to the same value for all D flip-flops. This frequency is set to the maximum frequency of the slowest D flip-flop at layout, so that all D flip-flops give correct functionality. The input signal D is set so that the output Q switch value at every rising edge of the clock signal clk. Figure 41 shows the waveforms of the input signals and the output signals. Every change on X and Q will cause a current to charge/discharge the output capacitance of the switched latch. These current spikes causes the dynamic power consumption which usually dominates the total power consumption at high switching speeds. The simulations are set to run for exactly 15 clock cycles for each D flip-flop, and the average current over this period is multiplied with the supply voltage to find the total power consumption.

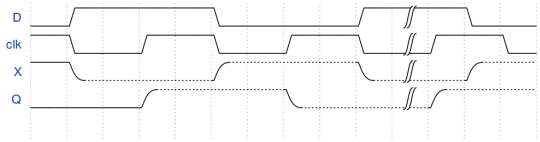


Figure 41: The power testbench waveforms

The *clk* signal period is set to twice the value of the highest Minority3 delay at layout, since the Minority3 is the slowest D flip-flop. Signal D period is set to twice the value of *clk*, and the D delay is set to $-clk$. This can also be seen in Table 14.

7.7 Maximum Power Consumption

For the Max Power measurement, each D flip-flop is clocked with a *clk* signal that match its maximum switching frequency. The maximum frequency is set to $2 \cdot t_{max}$, where t_{max} is the maximum delay of the setup times and propagation delays for that given temperature and supply voltage. The D signal period is set to $2 \cdot clk$, and D delay is set to $-clk$. By doing this the input signals are set as seen in Figure 41, and the D flip-flops should always be at its maximum frequency limit. This can also be seen in Table 14

7.8 Power-Delay-Product

The results from the Max Power measurements are multiplied with the delay used for the simulation to find the Power-Delay-Product. The Power-Delay-Product or PDP is the measure of Energy used in the circuit per switching operation. The PDP can be plotted as a function of the supply voltage to find the point where the circuit consumes its minimum power per transition. By operating at this supply voltage, the circuit is at its highest efficiency.

7.9 Monte Carlo Delay Simulation

Monte Carlo analysis are used to simulate the effects of process variations by using random statistical process variation data provided by the technology manufacturer. Multiple simulations are run with random process variations to simulate the yield of the design.

For this design, Monte Carlo simulations are run for delay measurements at $250mV$ for the temperatures $-40°C$, $27°C$ and $80°C$. All measured objects are simulated with a 300 run Monte Carlo Analysis to get accurate results. 100 Monte Carlo runs were first used when running Monte Carlo analysis, but the simulations showed that the standard deviation (sigma) still changed significantly around 100 runs. By using 300 runs, the sigma were more stable. The seed number is 200.

The analysis gives four results, the mean delay, the minimum delay, the maximum delay, and the standard deviation. The process variations are disabled for

the loads and drivers in the testbench, so that the D flip-flops process variations affect themselves.

7.9.1 Average Mean Delay and Standard Deviation for Schematic and Layout

The Mean Delay for the cases: Master latch rising edge, Master latch falling edge, Slave latch rising edge and Slave latch falling edge, for each D flip-flop are averaged to get an overview of each D flip-flop delay. The standard deviation for all cases are also averaged. These data will be plotted into charts to compare the schematic results with the layout results.

7.9.2 Average Mean Delay for Schematic and Layout at different Temperatures

The average mean delays are plotted into a chart to compare the results across the different temperatures. This will give a better perspective of how the temperature affects the mean delay.

7.9.3 Worst Case Mean Delay for Schematic and Layout at different Temperatures

Next, the worst case delay for all D flip-flops at schematic and layout are plotted, to show how these changes with the temperature. This plot should give a better comparison than the average delay since the maximum frequency is limited by the worst case delay, not the average delay.

7.9.4 Relative Standard Deviation Comparison

The relative standard deviation (relative sigma, $\sigma_{relative}$) indicates the size of the delay standard deviation compared to the size of the mean delay. It shows the percentage delay deviation for a D flip-flop, and should give a indication of the robustness. It is found by dividing the average sigma by the average delay for the given D flip-flop and temperature, as seen in Expression 17.

$$\sigma_{relative} = \frac{\sigma_{average}}{\mu_{average}} \tag{17}$$

This is done for all D flip-flops to find the one with the least delay variations.

7.10 Running Simulations with OCEAN scripts

The running and test setup of the Power-Delay-Product Simulations are time consuming and needs a lot of manual inputting of delay data from previous simulations. All D flip-flops need to be clocked with the maximum frequency clock signal for the given supply voltage and temperature. In addition must the D signal be set so that the D flip-flop switches the output signal Q at every rising edge of the clock. The results from the PDP simulations also need to be stored in data files, to ease the chart creation process.

OCEAN is a text-based process which allows the user to set up, simulate, and analyze circuit data. It can be run from the UNIX shell or from Cadence Virtuoso[1]. The produced results from a simulation can be exported into data files directly from the OCEAN-initiated test.

The OCEAN scripts are generated by a Python script that imports delay data from previous simulations and calculates clock period and input signal for each D flip-flop. The Python script generates multiple tests simultaneously and run them in parallel from the UNIX shell. The exported output data files from the OCEAN script, are read with a Python script which calculates the PDP. All these scripts can be seen in Appendix C.

7.11 Simulation Input Signals

The input signals for all D flip-flop simulations are presented in Table 14. D_p is the period of the D signal, D_d is the delay on the D signal, clk_p is period of the clock signal and clk_d is the delay on the clock signal.

Test	Sect.	Temp.	V_{DD} [mV]	D_p	D_d	clk_p	clk_d
Basic Func.	7.2	$27°C$	250	$240ns$	$-90ns$	$120ns$	$-2fs$
Delay	7.3	$-40°C$	$100-300$	$160\mu s$	$-60\mu s$	$80\mu s$	$-2fs$
&	7.3	$27°C$	$100-300$	$20\mu s$	$-7.5\mu s$	$10\mu s$	$-2fs$
Max. freq.	7.3	$80°C$	$100-300$	$8\mu s$	$-3\mu s$	$4\mu s$	$-2fs$
Static Pow.	7.5	All	$100-300$	-	-	-	-
Total Pow.	7.6	All	$100-300$	$2 \cdot clk_p$	$-clk_p$	$2{\cdot}\text{Min3}_{max}$	$-2fs$
Max. Pow.	7.7	All	$100-300$	$2 \cdot clk_p$	$-clk_p$	$2{\cdot}\text{Dff}_{max}$	$-2fs$
PDP	7.8	All	$100-300$	$2 \cdot clk_p$	$-clk_p$	$2{\cdot}\text{Dff}_{max}$	$-2fs$
Delay MC	7.9	$-40°C$	250	$1.2\mu s$	$-450ns$	$600ns$	$-2fs$
	7.9	$27°C$	250	$400ns$	$-150ns$	$200ns$	$-2fs$
	7.9	$80°C$	250	$200ns$	$-75ns$	$100ns$	$-2fs$

Table 14: Input Variables for the different Tests

8 Results from Simulations

The results from the simulations described in Section 7 is found in this Section.

8.1 Transistor Layout

The results for the parasitic extraction simulation, and the results for the nWell edge placement can be seen in this Section.

8.1.1 Parasitic Extraction

By running the extraction tools, parasitic data are extracted for the different nets in the layout. For the xRC tool, they are shown in Table 15. The parasitic data extracted by using the xACT 3D tool are shown in Table 16.

Nets	R Count	$C_{TOT}[F]$	$CC_{TOT}[F]$	$C + CC_{TOT}[F]$
in	14	$1.762 \cdot 10^{-19}$	$4.100 \cdot 10^{-16}$	$4.102 \cdot 10^{-16}$
2	27	$2.520 \cdot 10^{-19}$	$8.514 \cdot 10^{-16}$	$8.516 \cdot 10^{-16}$
gnd	45	$2.382 \cdot 10^{-19}$	$9.249 \cdot 10^{-16}$	$9.251 \cdot 10^{-16}$
Vdd	50	$7.575 \cdot 10^{-20}$	$6.247 \cdot 10^{-16}$	$6.248 \cdot 10^{-16}$
out	15	$2.382 \cdot 10^{-19}$	$4.366 \cdot 10^{-16}$	$4.368 \cdot 10^{-16}$

Table 15: Layout parasitics for test circuit using xRC extraction tool

Nets	R Count	$C_{TOT}[F]$	$CC_{TOT}[F]$	$C + CC_{TOT}[F]$
in	14	$4.048 \cdot 10^{-18}$	$7.252 \cdot 10^{-16}$	$7.292 \cdot 10^{-16}$
2	27	$3.817 \cdot 10^{-17}$	$1.233 \cdot 10^{-15}$	$1.271 \cdot 10^{-15}$
gnd	45	$3.883 \cdot 10^{-18}$	$6.638 \cdot 10^{-16}$	$6.676 \cdot 10^{-16}$
Vdd	50	$4.734 \cdot 10^{-18}$	$7.715 \cdot 10^{-16}$	$7.762 \cdot 10^{-16}$
out	15	$3.284 \cdot 10^{-17}$	$5.895 \cdot 10^{-16}$	$6.223 \cdot 10^{-16}$

Table 16: Layout parasitics for test circuit using xACT 3D extraction tool

The test bench results for propagation delay and power consumption are shown in Table 17.

	xRC	xACT 3D
Delay	$1.419ns$	$1.860ns$
Power	$20.04nW$	$31.86nW$

Table 17: Layout parasitics Simulation Results comparing the xRC and xACT 3D extraction tools

8.1.2 nWell Placement and Sizing

Figure 42 shows the threshold voltages for both nMOS and pMOS transistors as the nWell distance to the active gate area increases, and also shows the schematic threshold voltage.

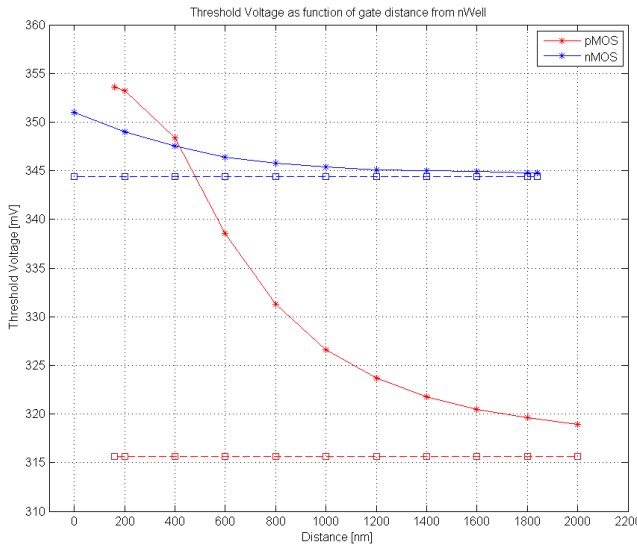


Figure 42: WPE chart for inverter circuit, Dashed lines are the schematic threshold voltage

8.2 D flip-flop functionality

The D flip-flops are tested at $27°C$ with $250mV$ supply voltage to confirm the functionality. The results can be seen in Figure 43.

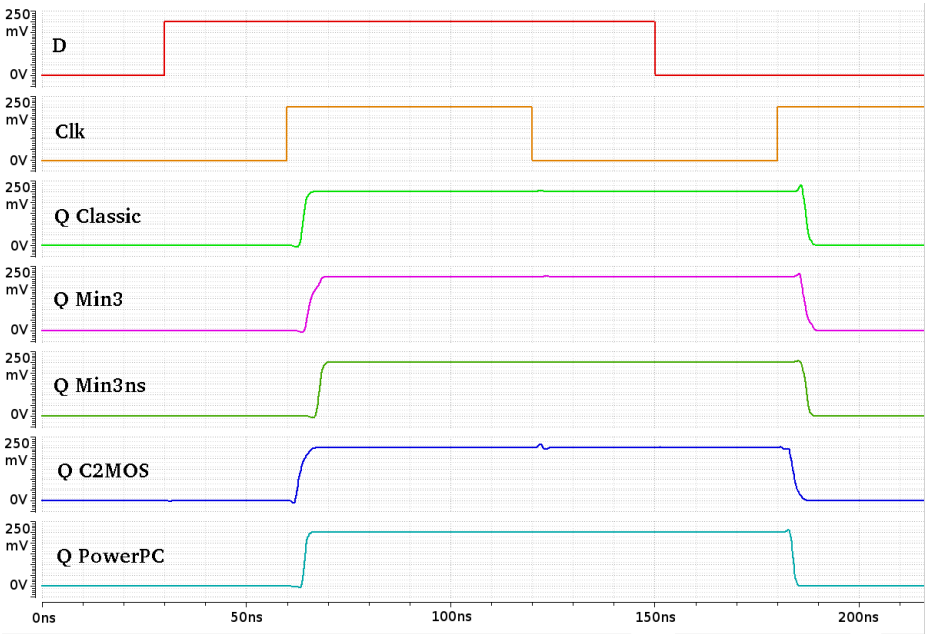
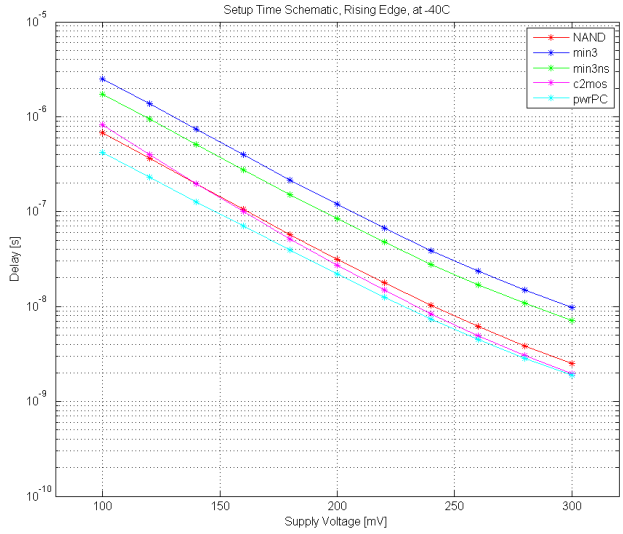


Figure 43: The confirmed functionality of the D flip-flops at $250mV$,$27°C$

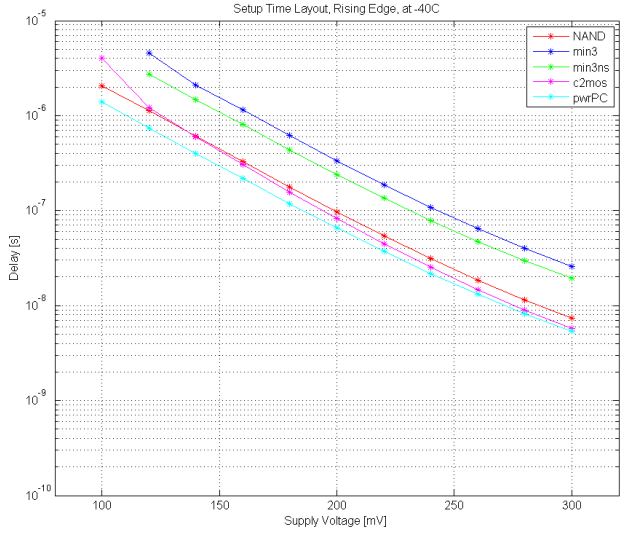
8.3 Delay Comparison of D flip-flops

The different D flip-flop delays will be presented in this Section.

8.3.1 Master latch delay

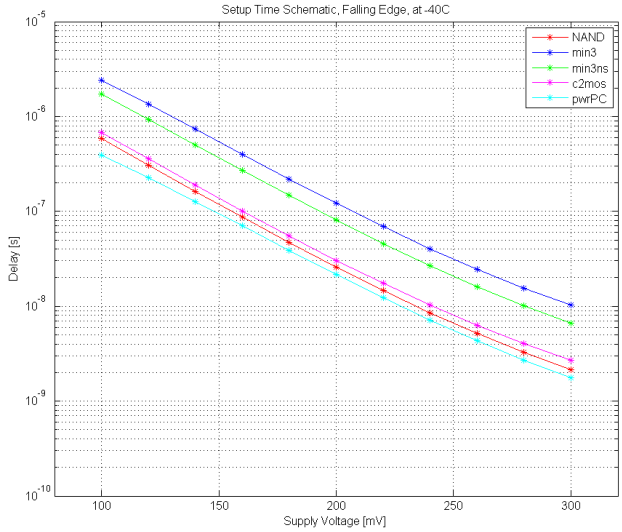


(a) Rising edge Schematic at $-40°C$

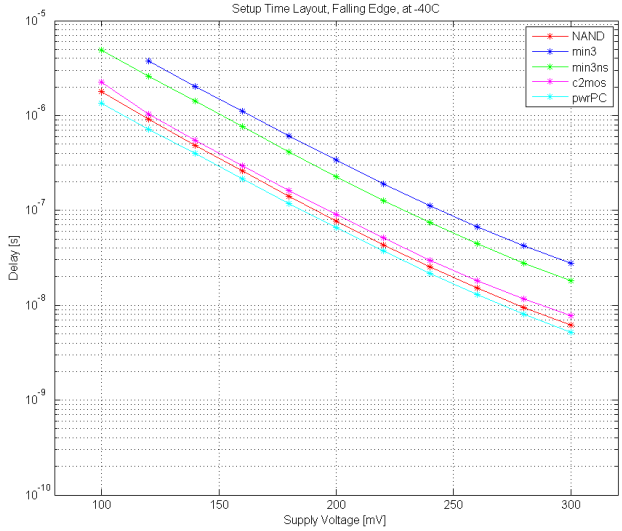


(b) Rising edge Layout at $-40°C$

Figure 44: Setup delay for schematic and layout at $-40°C$, Rising edge

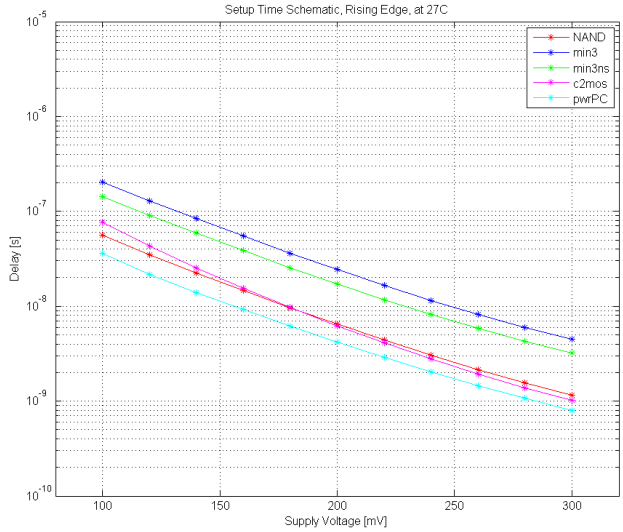


(a) Falling edge Schematic at $-40°C$

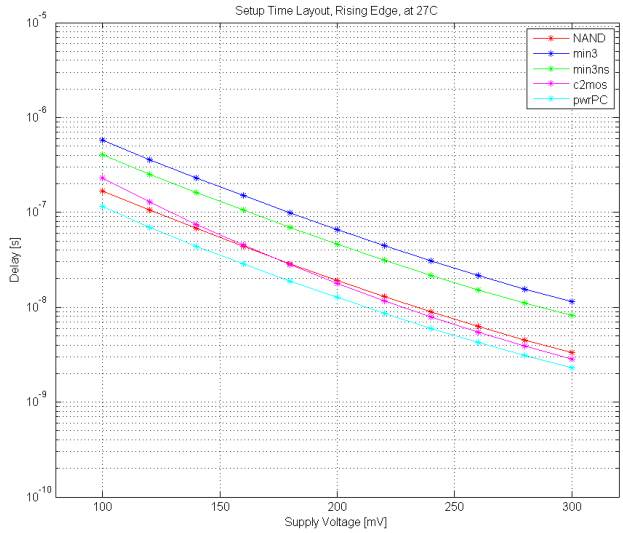


(b) Falling edge Layout at $-40°C$

Figure 45: Setup delay for schematic and layout at $-40°C$, Falling edge

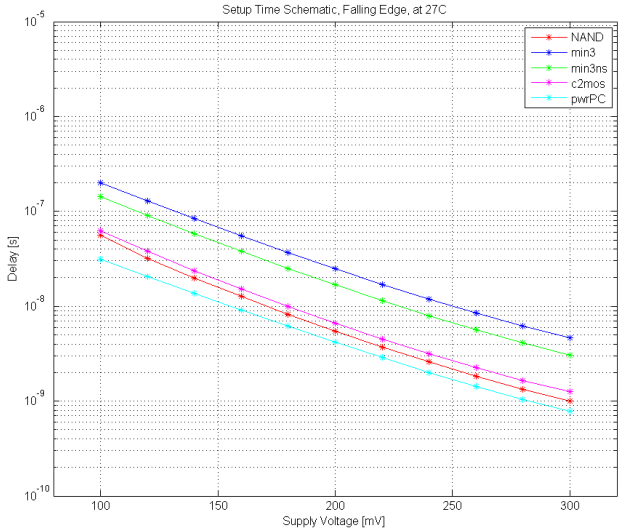


(a) Rising edge Schematic at $27°C$

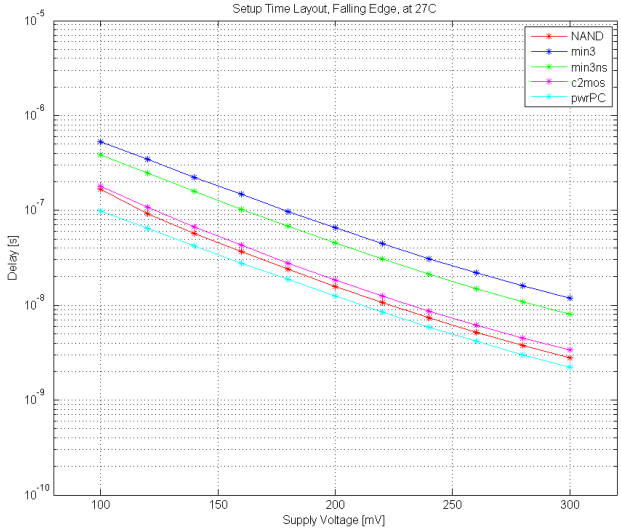


(b) Rising edge Layout at $27°C$

Figure 46: Setup delay for schematic and layout at $27°C$, Rising edge

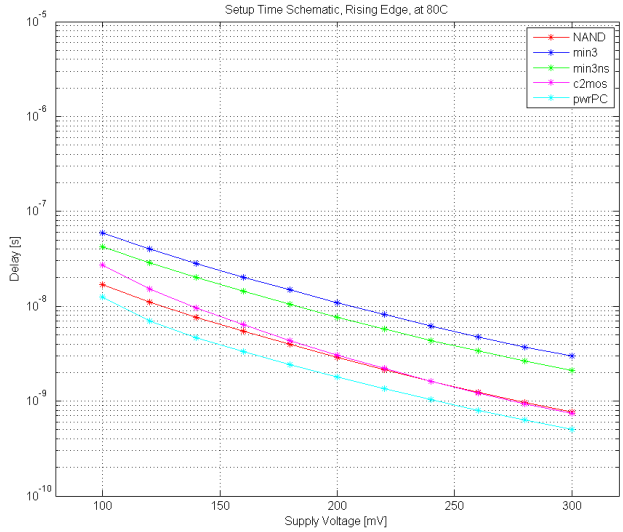


(a) Falling edge Schematic at $27°C$

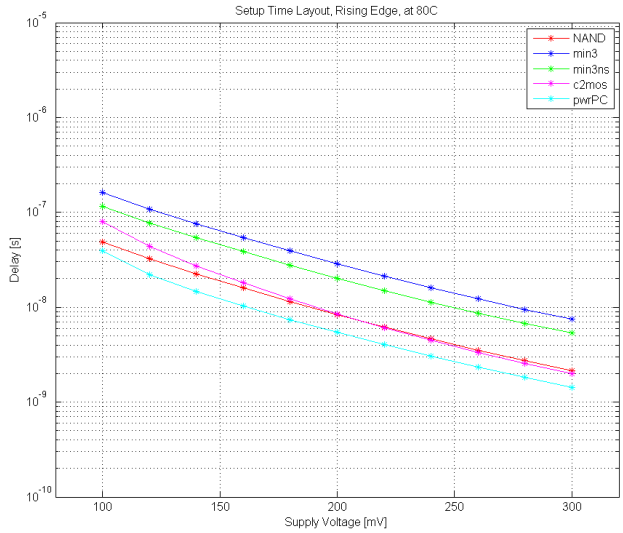


(b) Falling edge Layout at $27°C$

Figure 47: Setup delay for schematic and layout at $27°C$, Falling edge

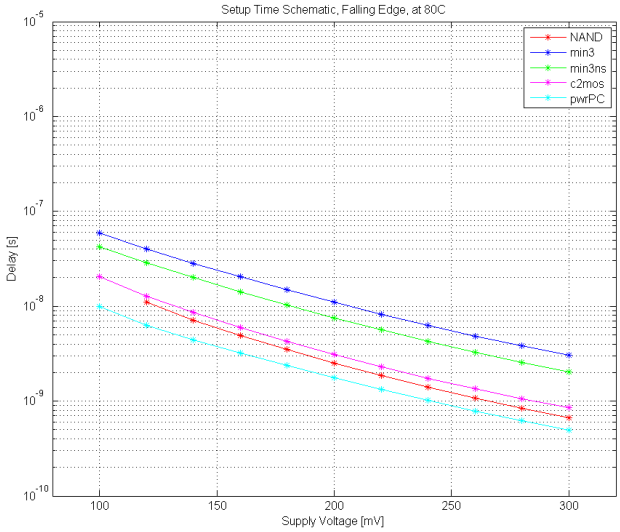


(a) Rising edge Schematic at $80°C$

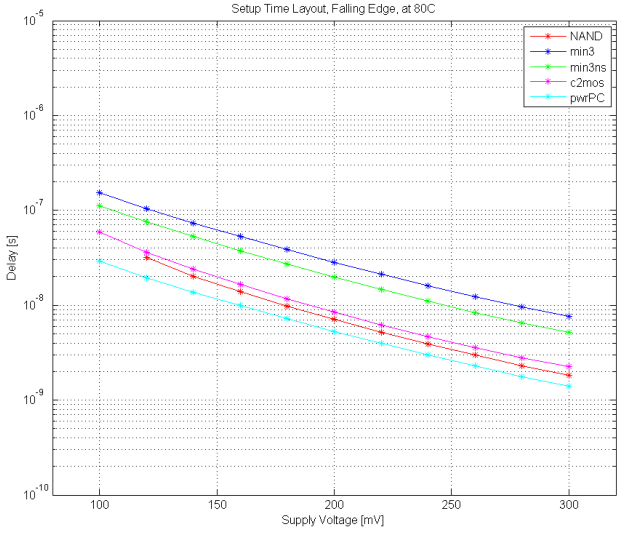


(b) Rising edge Layout at $80°C$

Figure 48: Setup delay for schematic and layout at $80°C$, Rising edge



(a) Falling edge Schematic at $80°C$

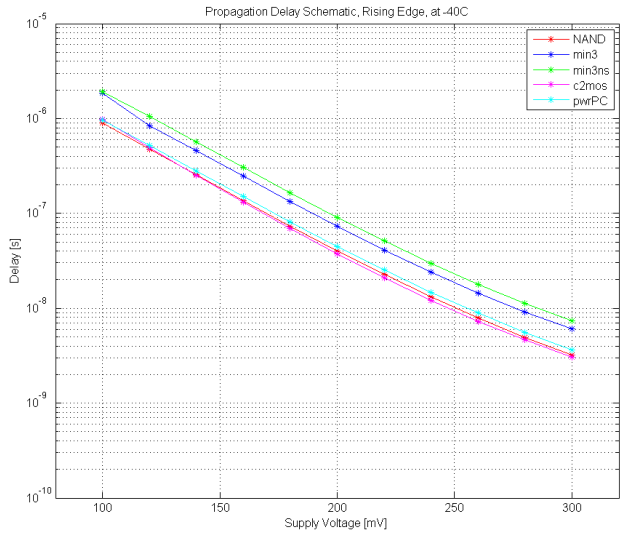


(b) Falling edge Layout at $80°C$

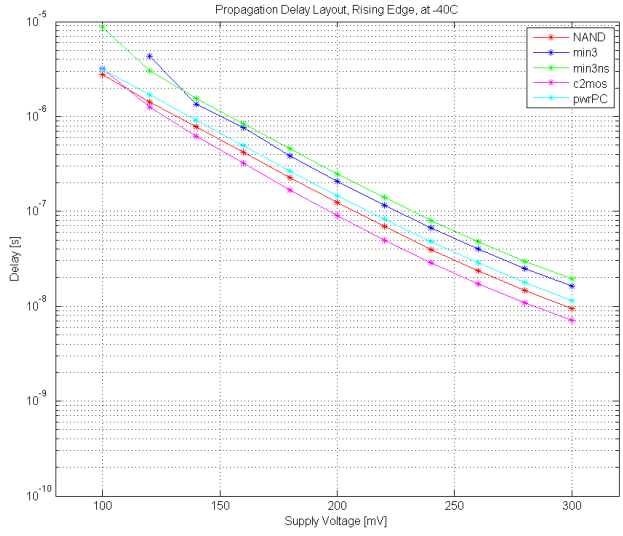
Figure 49: Setup delay for schematic and layout at $80°C$, Falling edge

8.3.2 Slave latch delay

The slave latch delay or propagation delay t_{co} are presented here.

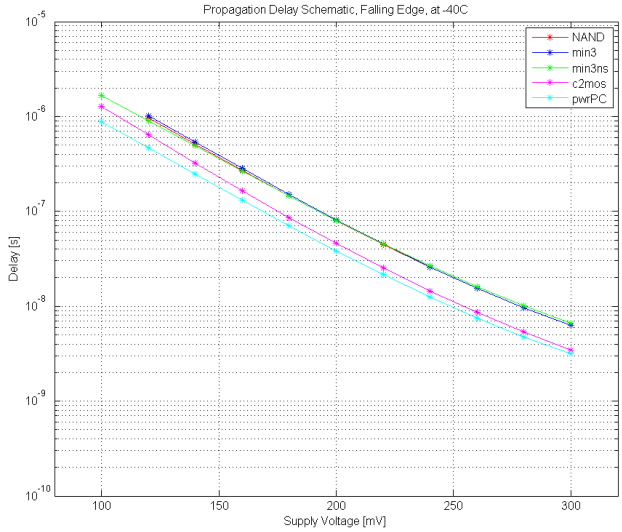


(a) Rising edge Schematic at $-40°C$

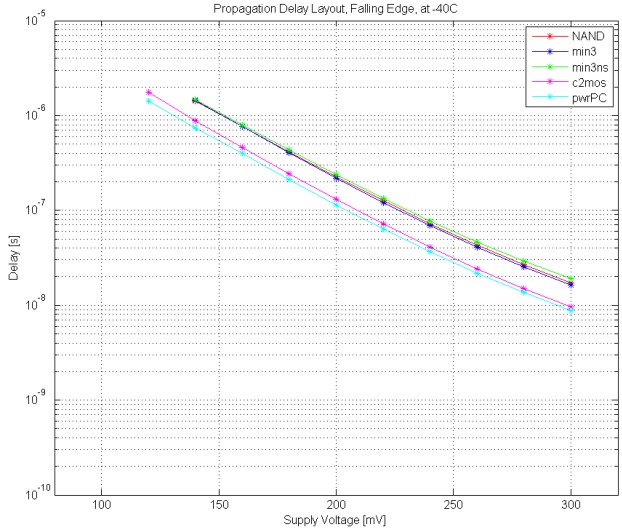


(b) Rising edge Layout at $-40°C$

Figure 50: Propagation delay for schematic and layout at $-40°C$, Rising edge

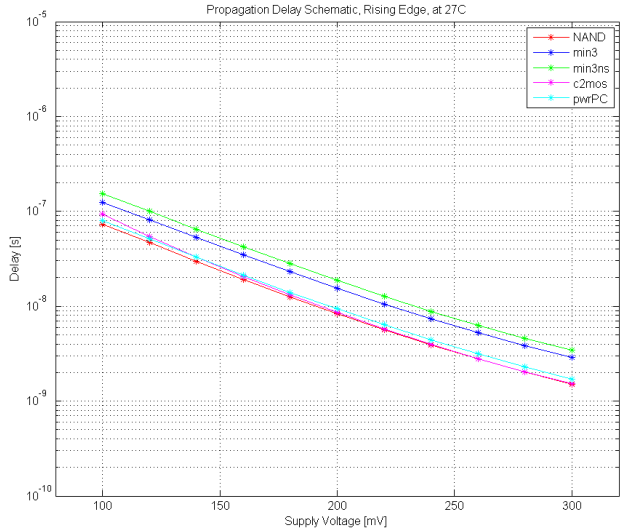


(a) Falling edge Schematic at $-40°C$

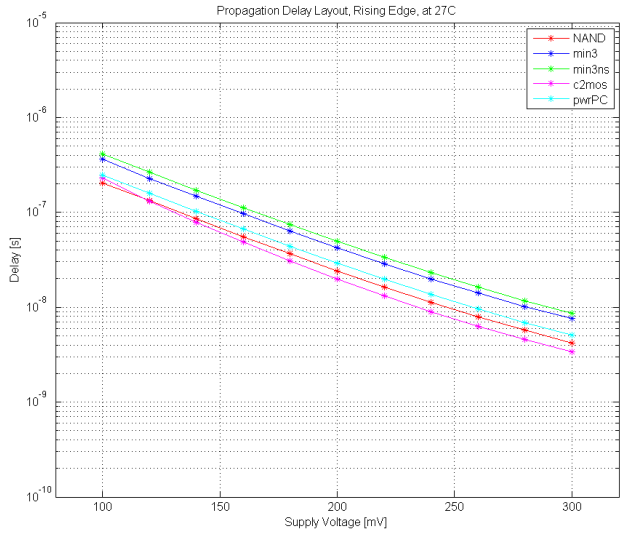


(b) Falling edge Layout at $-40°C$

Figure 51: Propagation delay for schematic and layout at $-40°C$, Falling edge

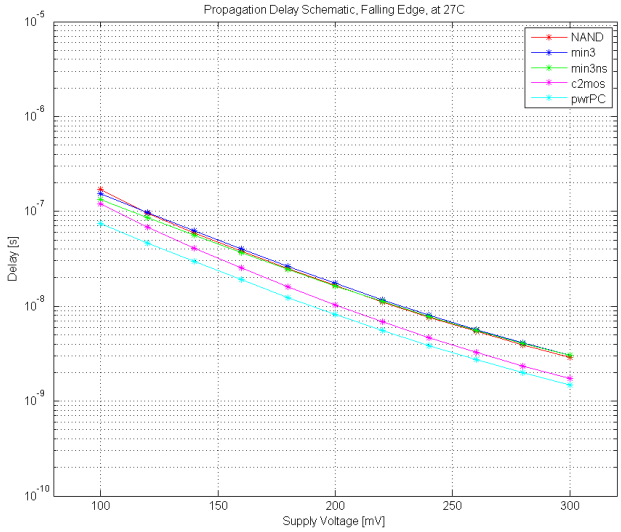


(a) Rising edge Schematic at $27°C$

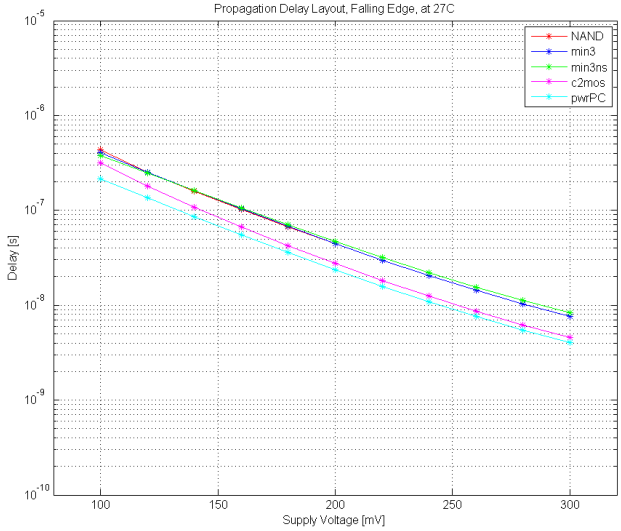


(b) Rising edge Layout at $27°C$

Figure 52: Propagation delay for schematic and layout at $27°C$, Rising edge

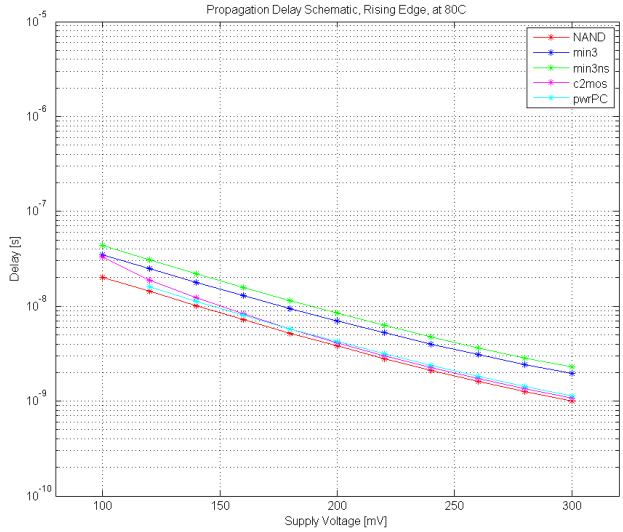


(a) Falling edge Schematic at $27°C$

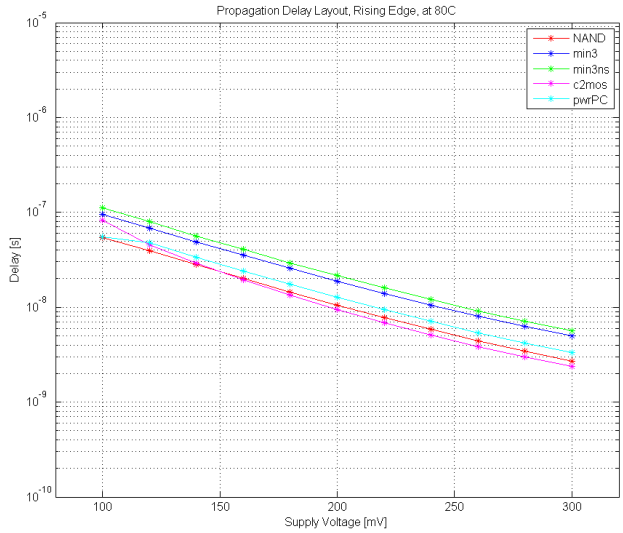


(b) Falling edge Layout at $27°C$

Figure 53: Propagation delay for schematic and layout at $27°C$, Falling edge

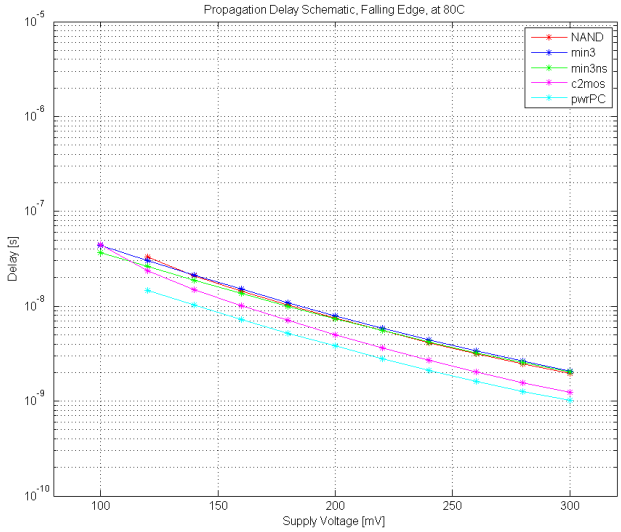


(a) Rising edge Schematic at $80°C$

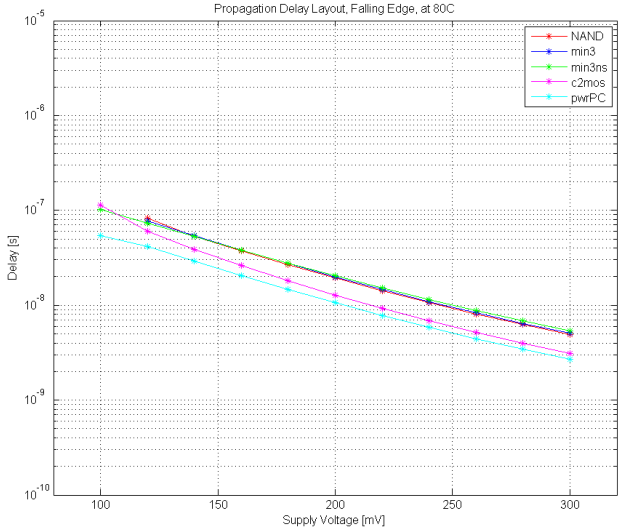


(b) Rising edge Layout at $80°C$

Figure 54: Propagation delay for schematic and layout at $80°C$, Rising edge



(a) Falling edge Schematic at $80°C$

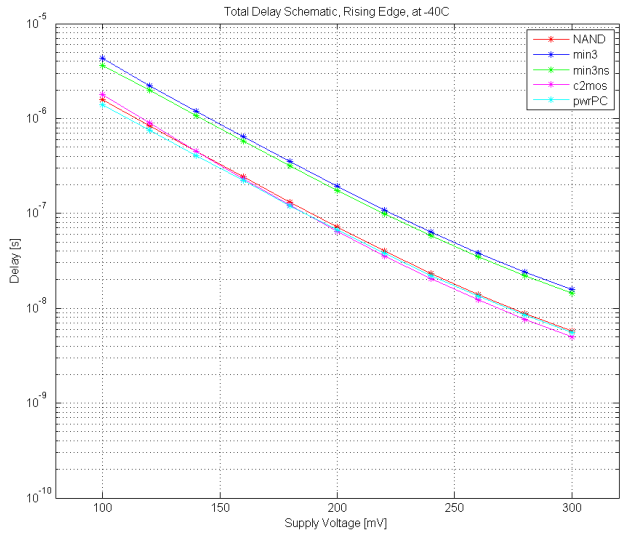


(b) Falling edge Layout at $80°C$

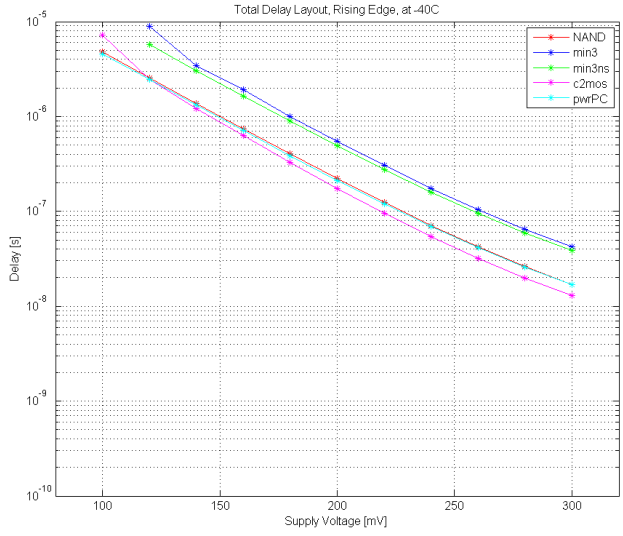
Figure 55: Propagation delay for schematic and layout at $80°C$, Falling edge

8.3.3 Total D flip-flop delay

The total delay is the sum of setup time t_{su} and propagation delay t_{co}.

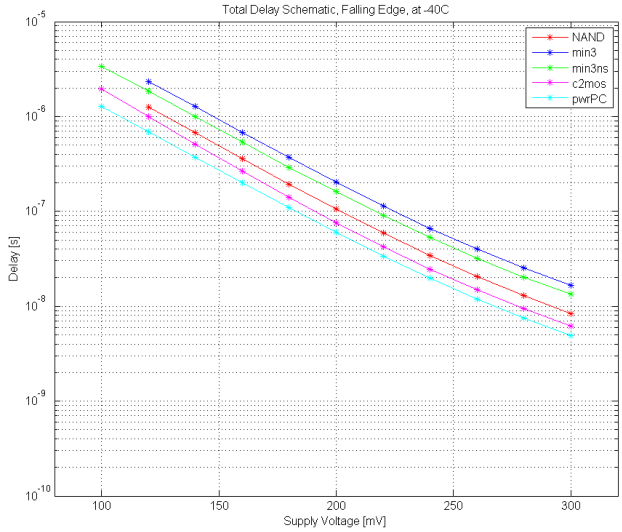


(a) Rising edge Schematic at $-40°C$

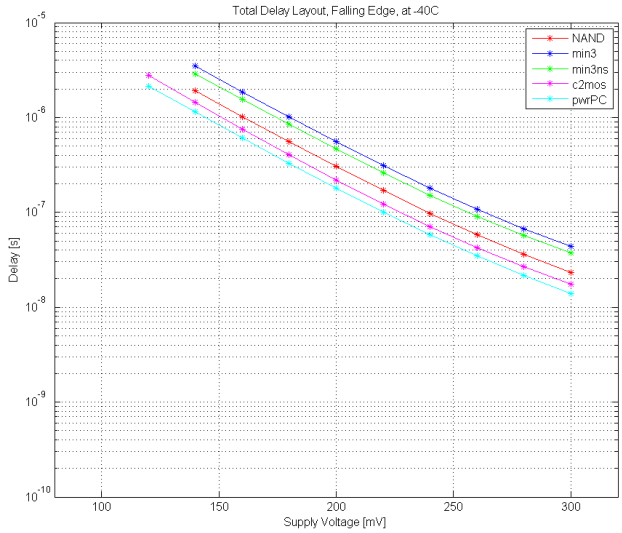


(b) Rising edge Layout at $-40°C$

Figure 56: Total delay for schematic and layout at $-40°C$, Rising edge

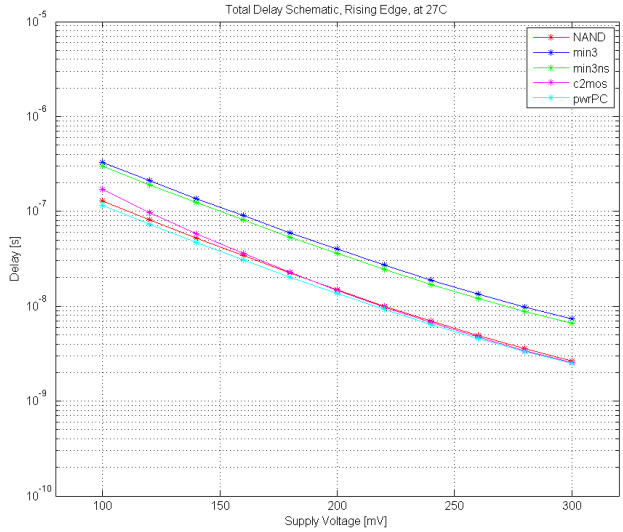


(a) Falling edge Schematic at $-40°C$

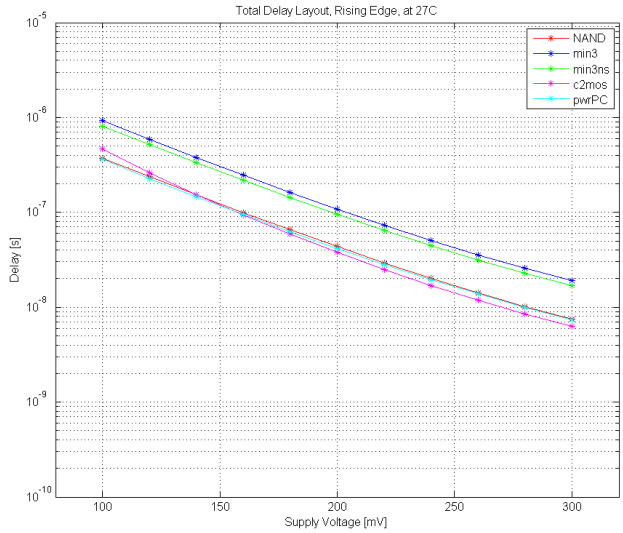


(b) Falling edge Layout at $-40°C$

Figure 57: Total delay for schematic and layout at $-40°C$, Falling edge

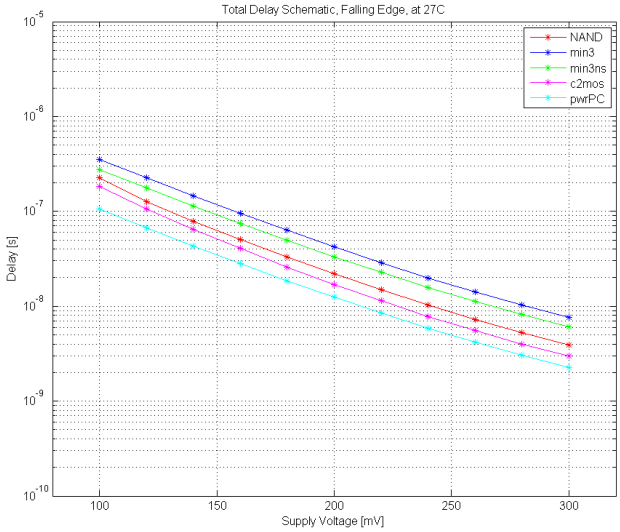


(a) Rising edge Schematic at $27°C$

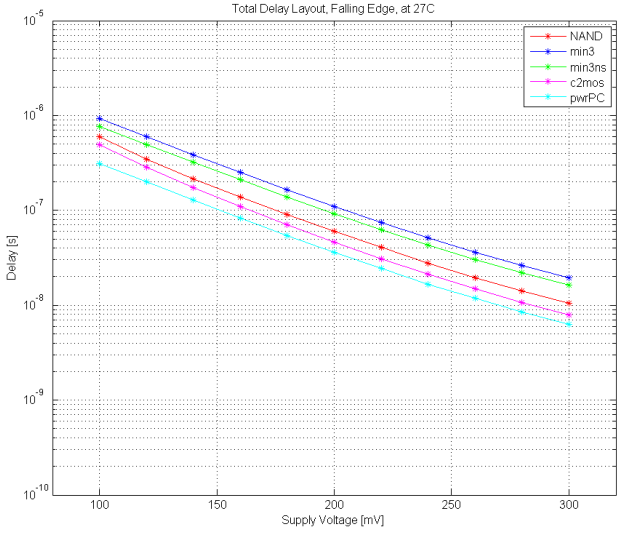


(b) Rising edge Layout at $27°C$

Figure 58: Total delay for schematic and layout at $27°C$, Rising edge

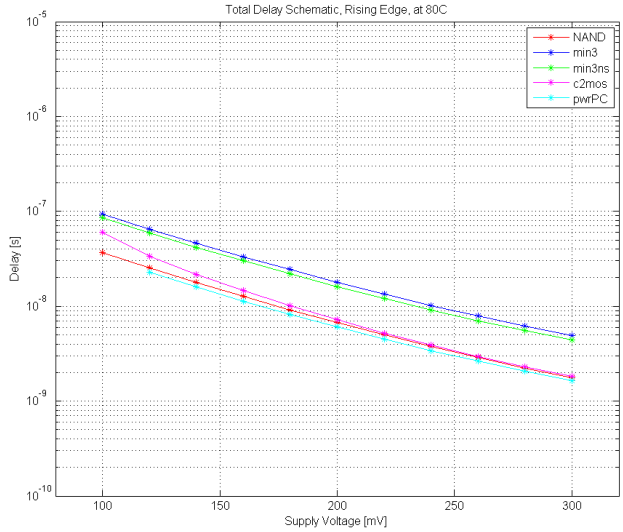


(a) Falling edge Schematic at $27°C$

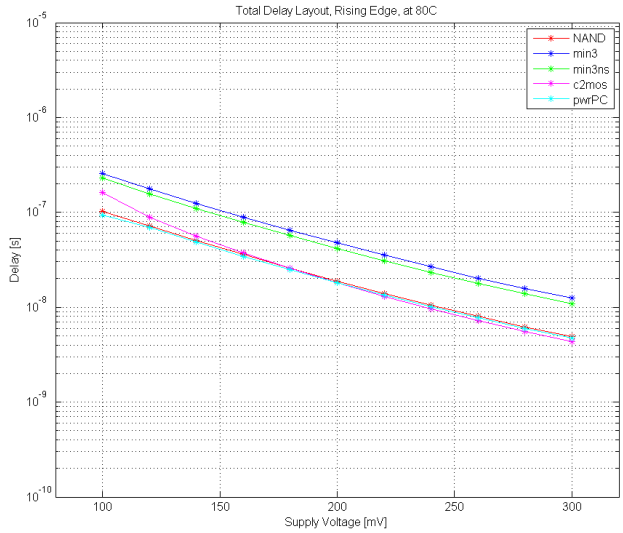


(b) Falling edge Layout at $27°C$

Figure 59: Total delay for schematic and layout at $27°C$, Falling edge

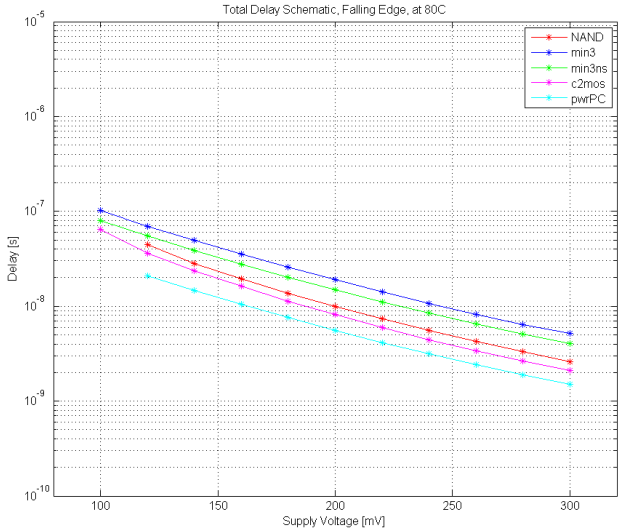


(a) Rising edge Schematic at $80°C$

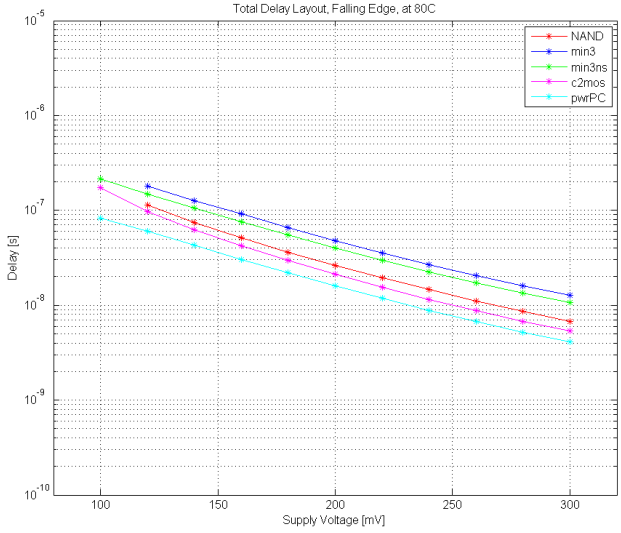


(b) Rising edge Layout at $80°C$

Figure 60: Total delay for schematic and layout at $80°C$, Rising edge



(a) Falling edge Schematic at $80°C$

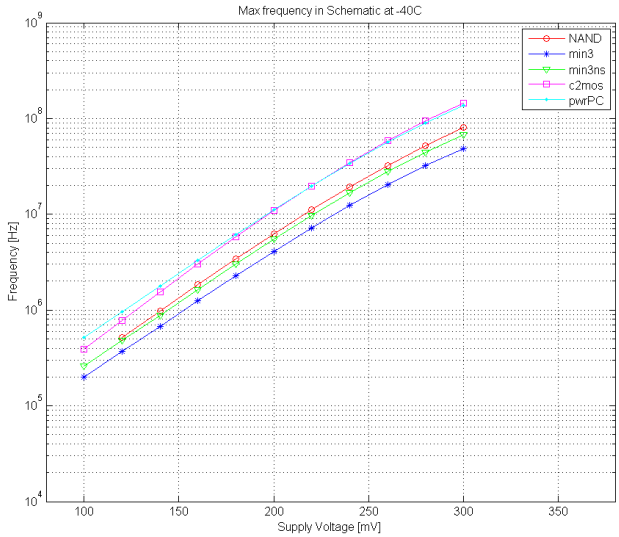


(b) Falling edge Layout at $80°C$

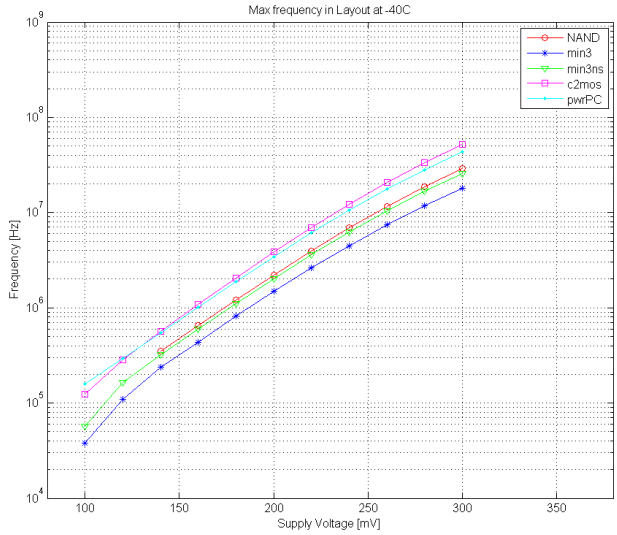
Figure 61: Total delay for schematic and layout at $80°C$, Falling edge

8.4 Maximum D flip-flop frequency based on maximum delay

The maximum delay for all D flip-flops are used to find the maximum frequency. The maximum delay is multiplied with 2 to find the maximum clock period. The results can be seen in this Section.

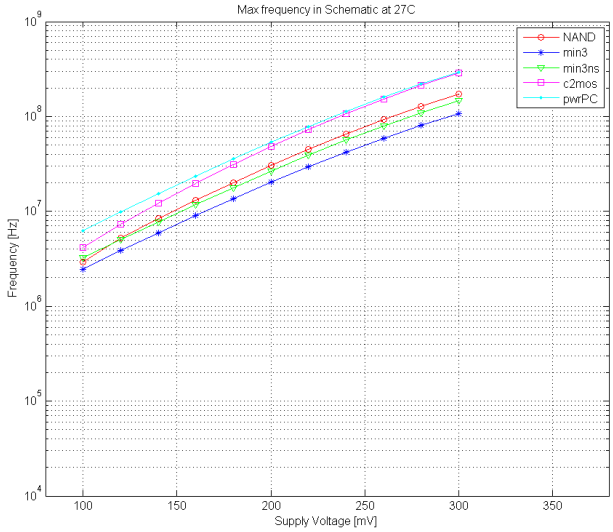


(a) Schematic

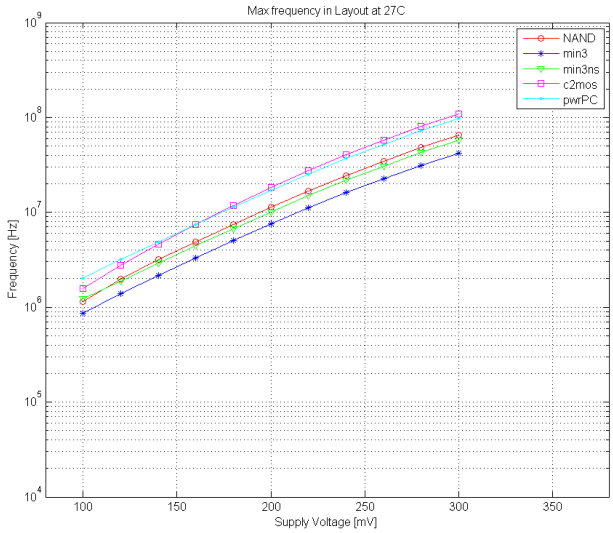


(b) Layout

Figure 62: Maximum frequency for all D flip-flops at $-40°C$

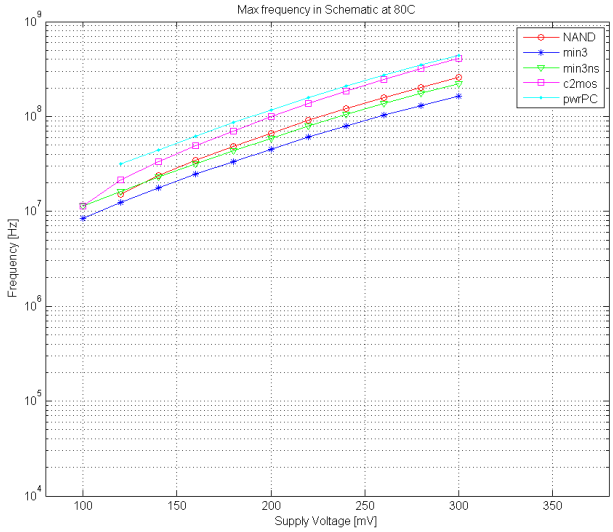


(a) Schematic

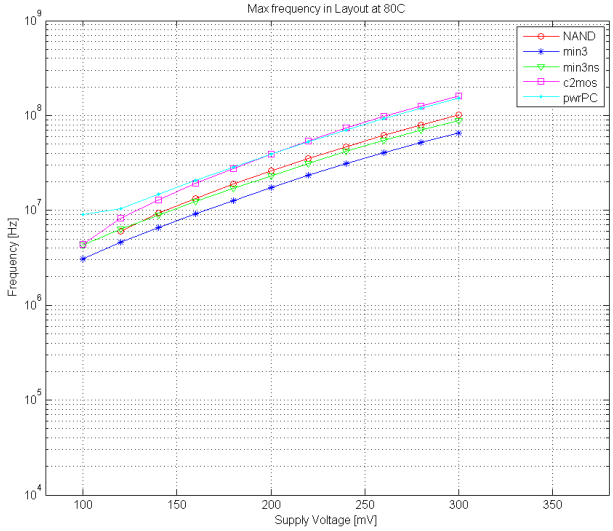


(b) Layout

Figure 63: Maximum frequency for all D flip-flops at $27°C$

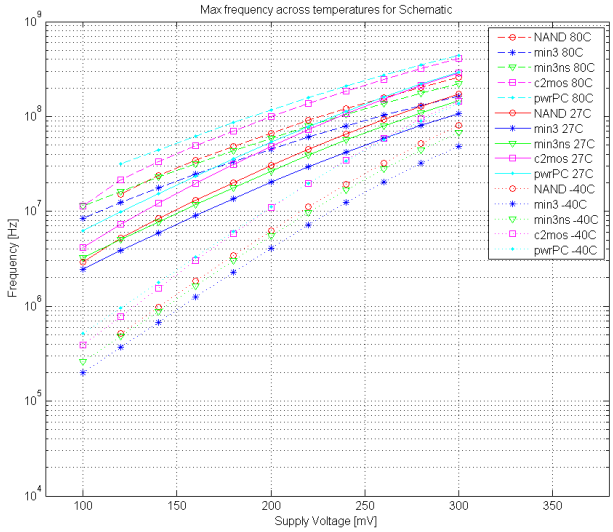


(a) Schematic

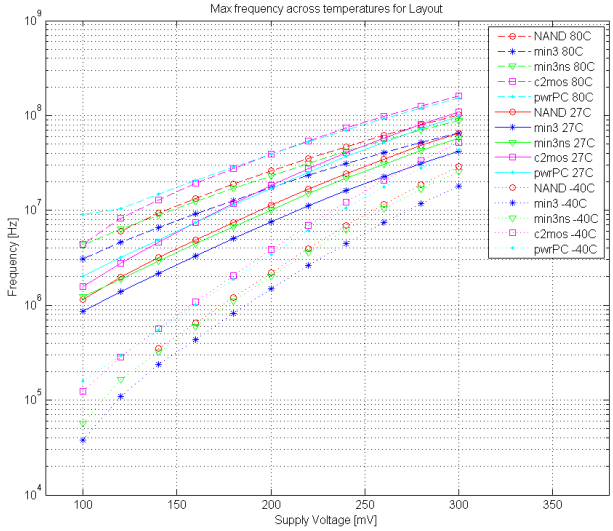


(b) Layout

Figure 64: Maximum frequency for all D flip-flops at $80°C$



(a) Schematic

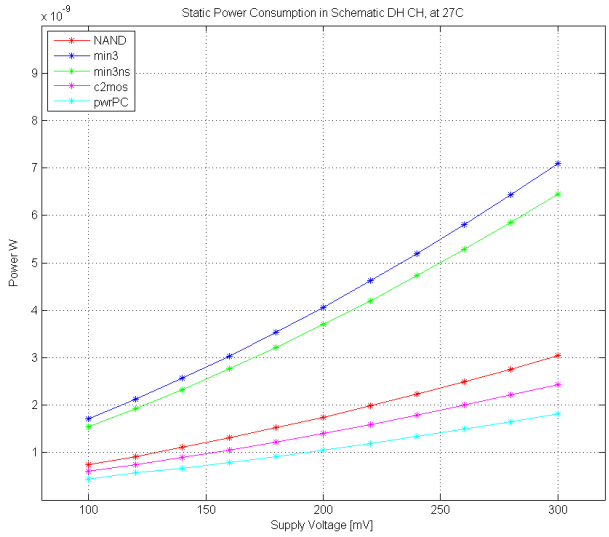


(b) Layout

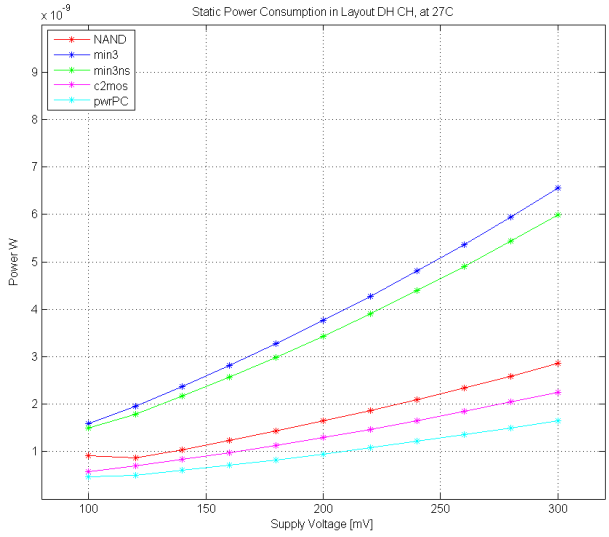
Figure 65: Maximum frequency for all D flip-flops across temperatures

8.5 Static Power Consumption

The static power consumption is found by having stable values at the inputs.

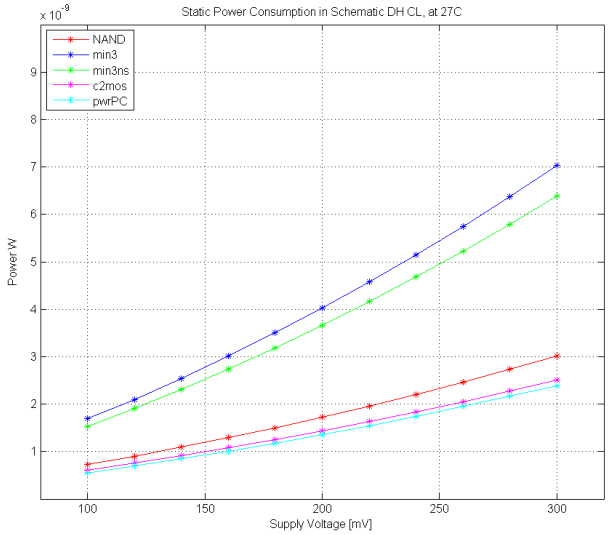


(a) Schematic with D high and Clk high

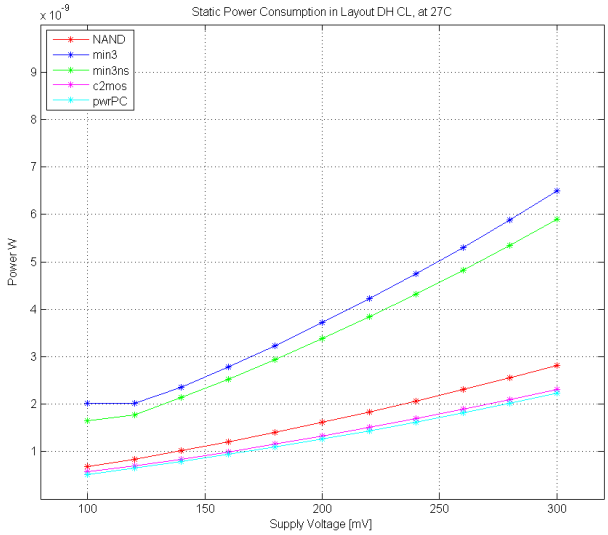


(b) Layout with D high and Clk high

Figure 66: Static Power Consumption for layout and schematic with D high, Clk high

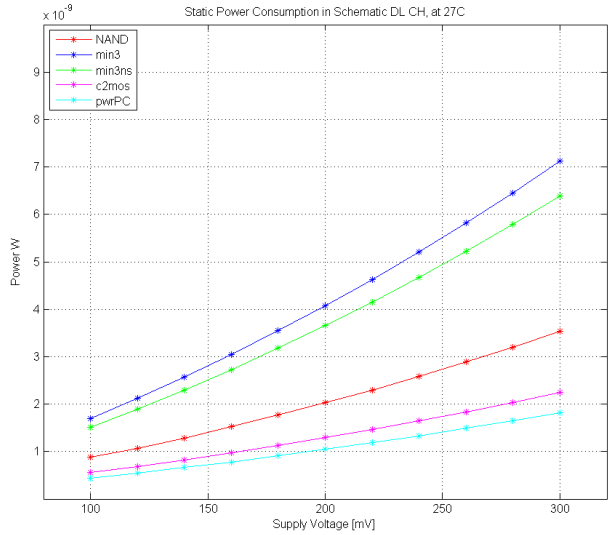


(a) Schematic with D high and Clk low

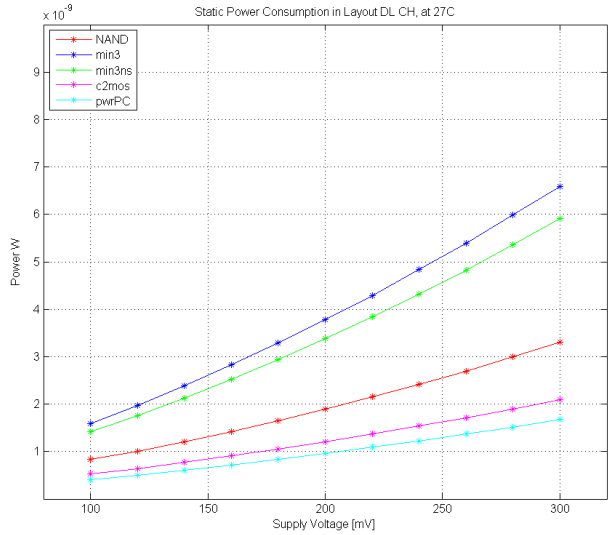


(b) Layout with D high and Clk low

Figure 67: Static Power Consumption for layout and schematic with D high, Clk low

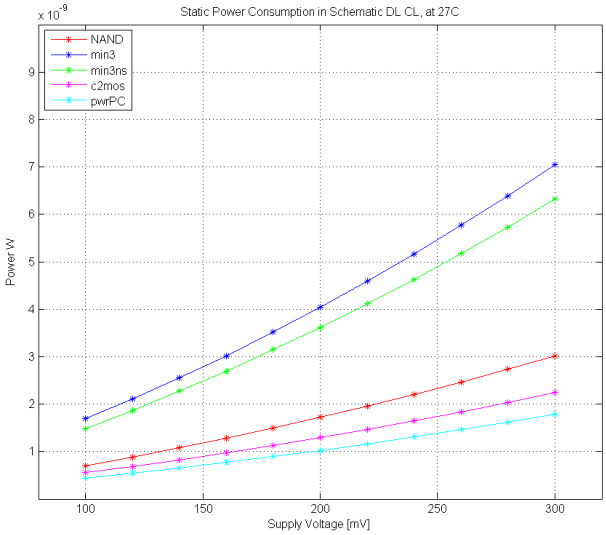


(a) Schematic with D low and Clk high

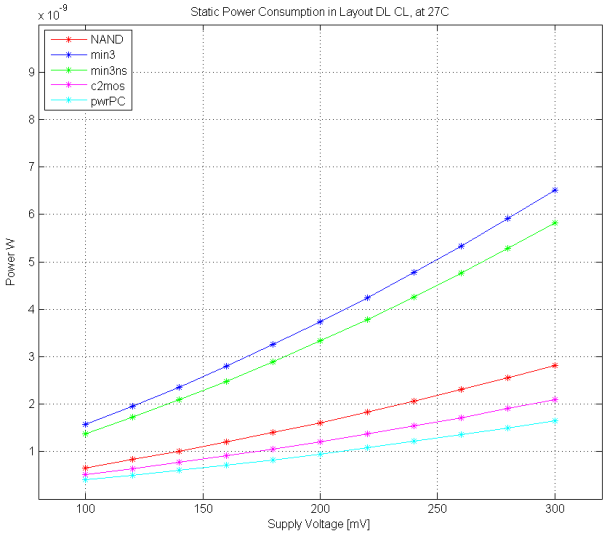


(b) Layout with D low and Clk high

Figure 68: Static Power Consumption for layout and schematic with D low, Clk high



(a) Schematic with D low and Clk low

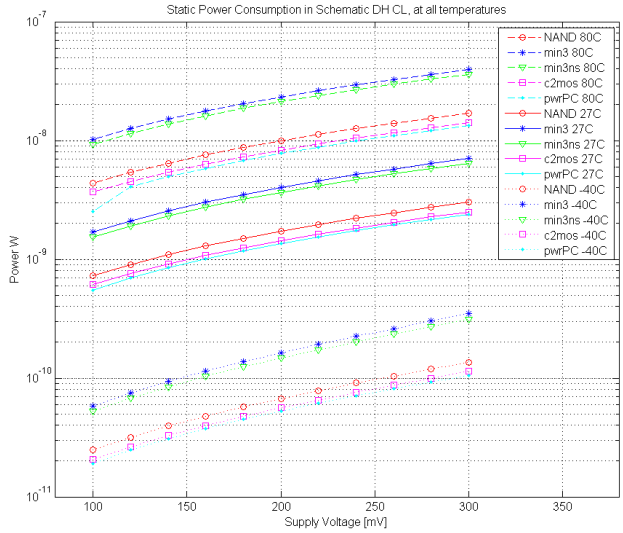


(b) Layout with D low and Clk low

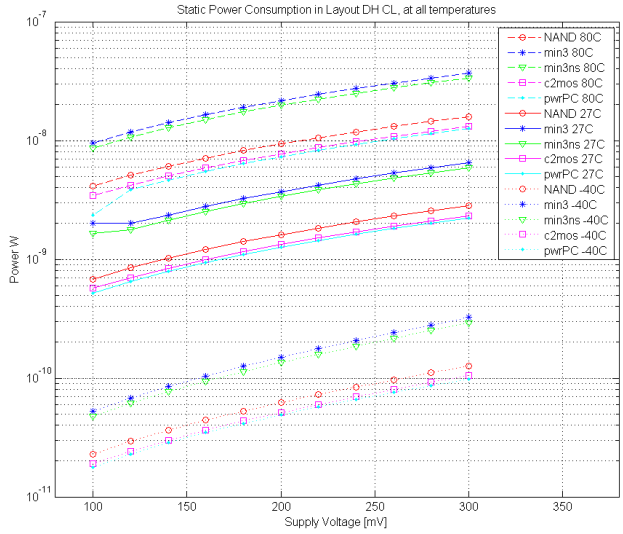
Figure 69: Static Power Consumption for layout and schematic with D low, Clk low

8.5.1 Static Power Comparison at Different Temperatures

Since the static power consumption with D high and Clk low, had the overall highest results, these inputs are also tested at $-40°C$ and $80°C$. The results are below.



(a) Schematic

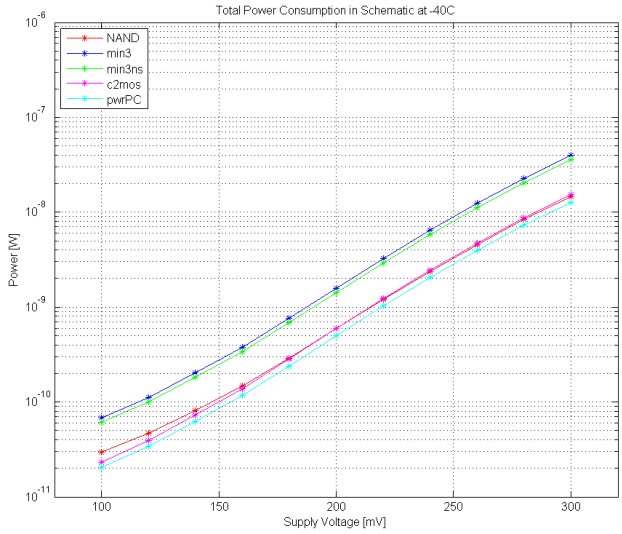


(b) Layout

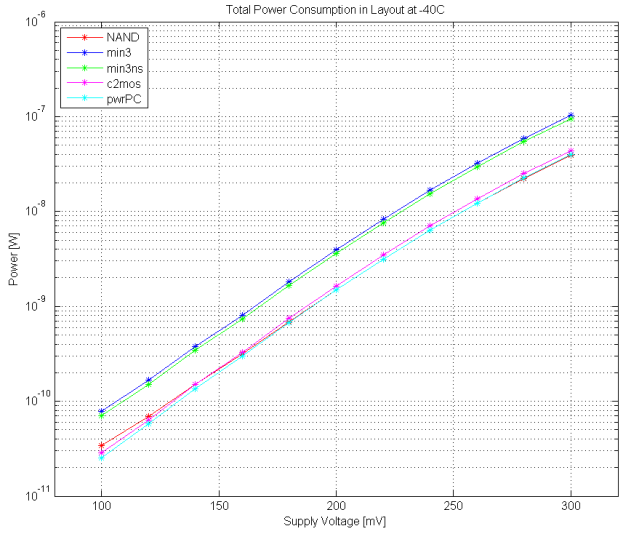
Figure 70: Static Power Consumption DH CL at all temperatures

8.6 Total Power Consumption

The Total Power Consumption is measured at the highest *clk* frequency where all Dffs function normally. This frequency is the Min3 D flip-flops maximum frequency in layout1.

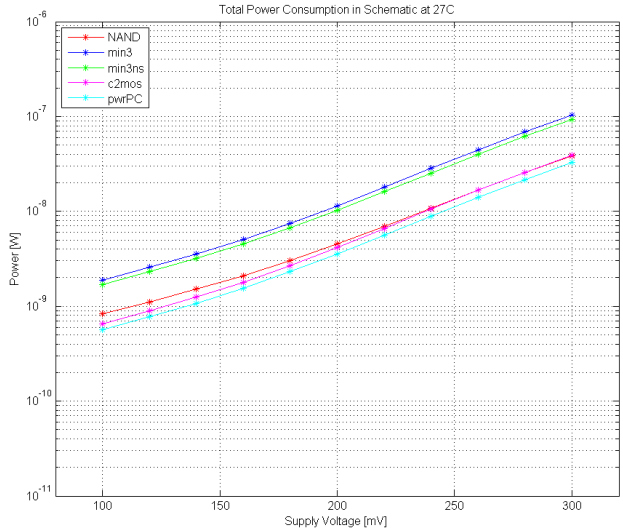


(a) Schematic

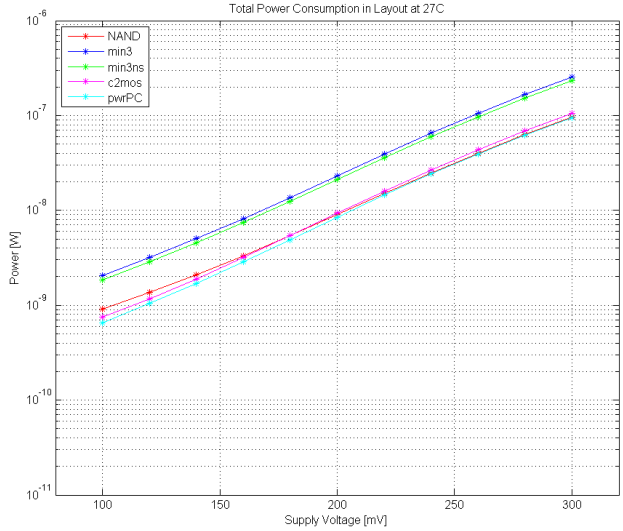


(b) Layout

Figure 71: Total Power Consumption at $-40°C$

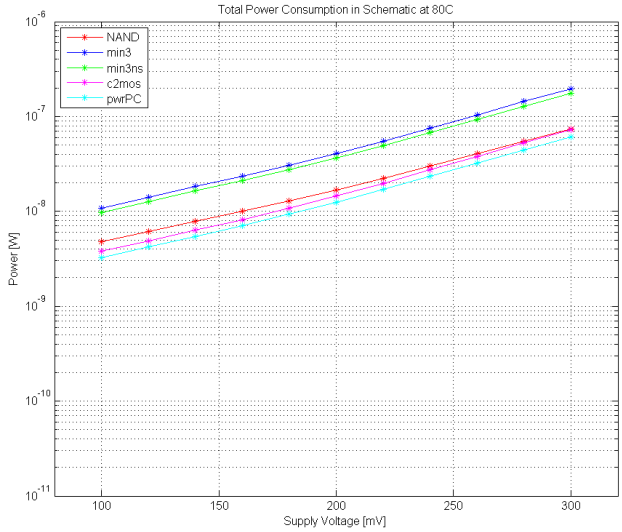


(a) Schematic

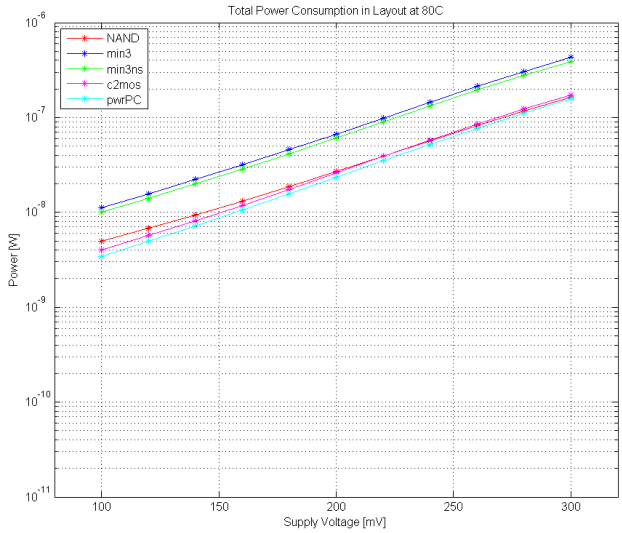


(b) Layout

Figure 72: Total Power Consumption at $27°C$



(a) Schematic



(b) Layout

Figure 73: Total Power Consumption at $80°C$

8.6.1 Total Power Consumption Schematic versus Layout Comparison

These charts show the total power consumption for schematic and layout compared.

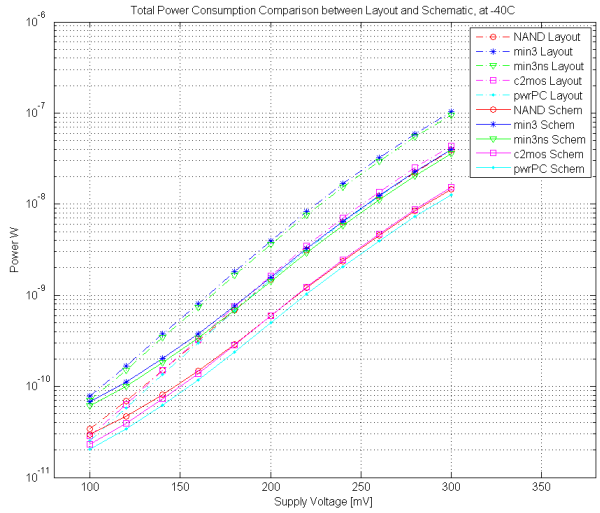


Figure 74: Total Power Consumption Comparison between schematic and layout at $-40°C$

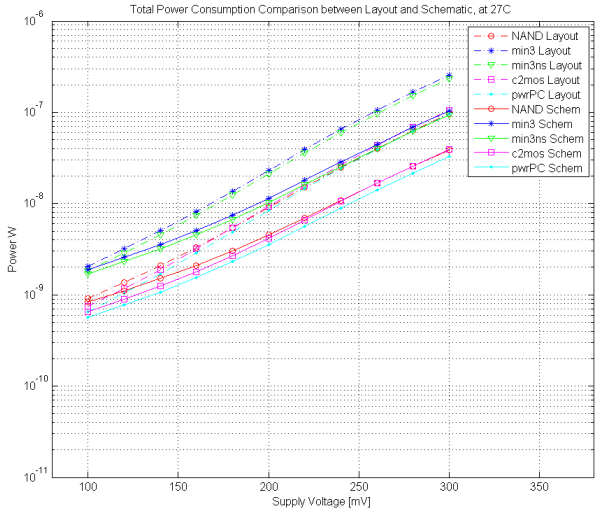


Figure 75: Total Power Consumption Comparison between schematic and layout at $27°C$

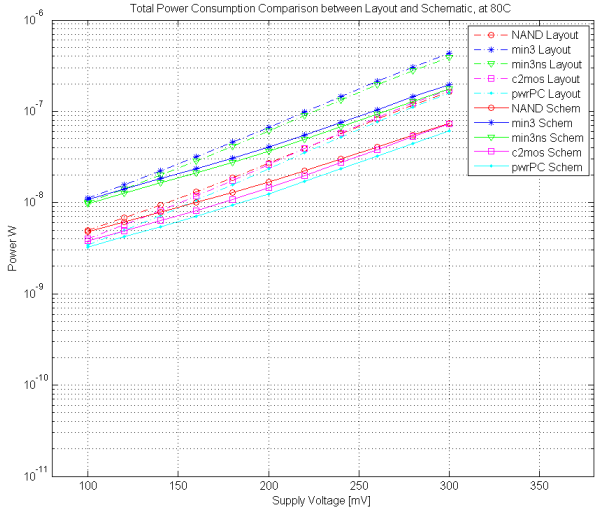


Figure 76: Total Power Consumption Comparison between schematic and layout at $80°C$

8.6.2 Total Power Consumption Temperature Comparison

These charts show how the total power consumption changes with temperature.

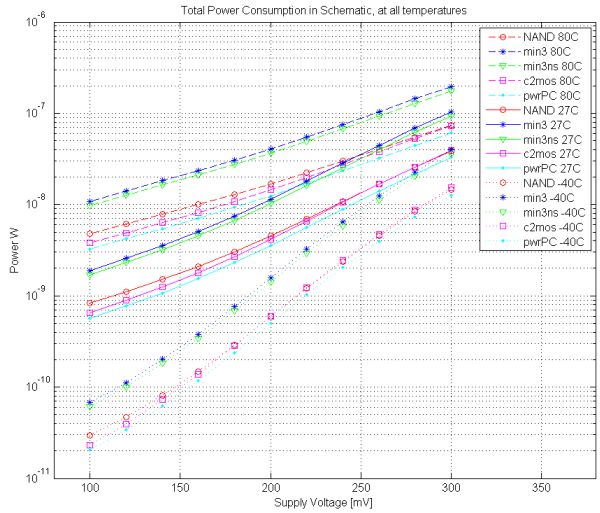


Figure 77: Total Power Consumption Comparison between Temperatures for schematic

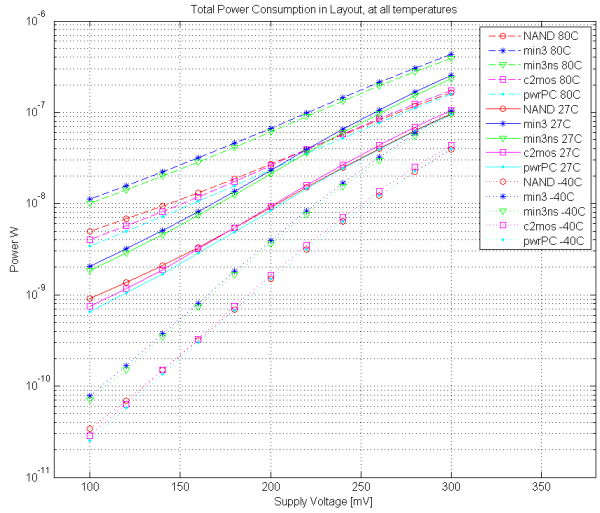
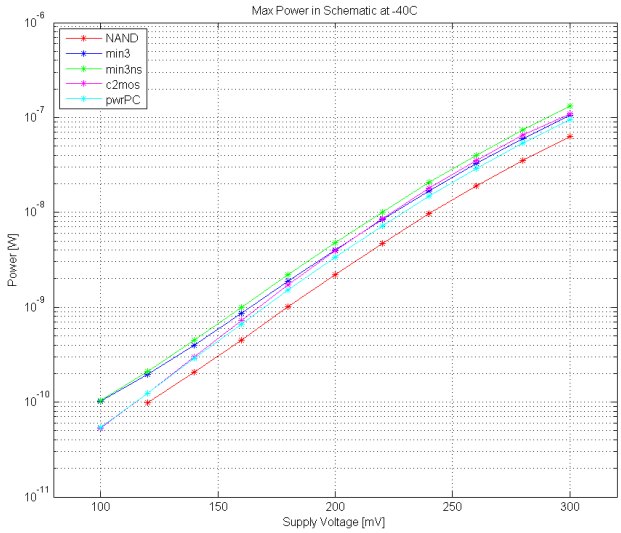


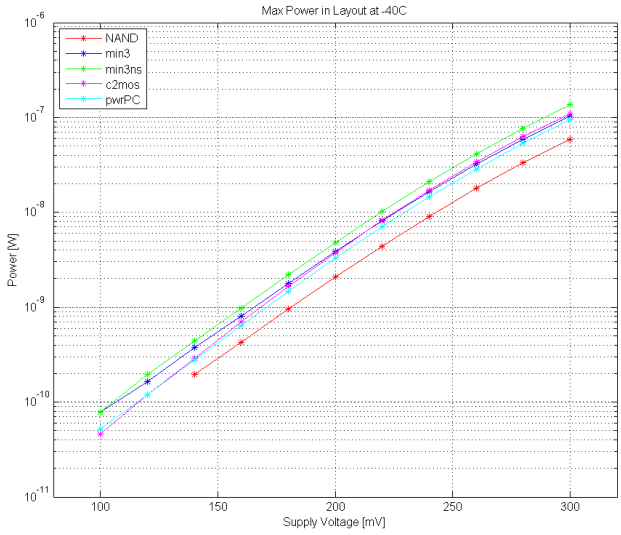
Figure 78: Total Power Consumption Comparison between Temperatures for layout

8.7 Maximum Power Consumption

The D flip-flops consumes its maximum power when operating at its maximum frequency and switching the output at every clock cycle. These results are presented here.

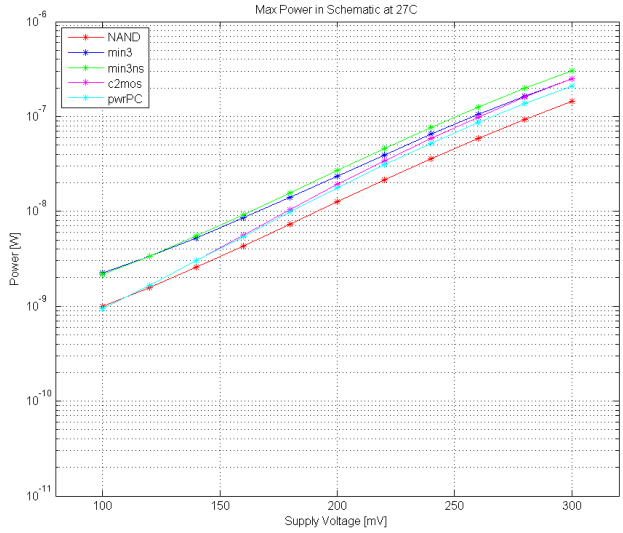


(a) Schematic

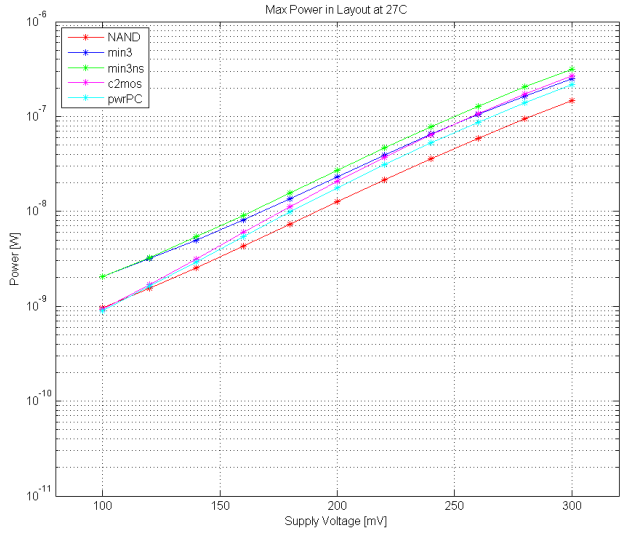


(b) Layout

Figure 79: Maximum Power Consumption at $-40°C$

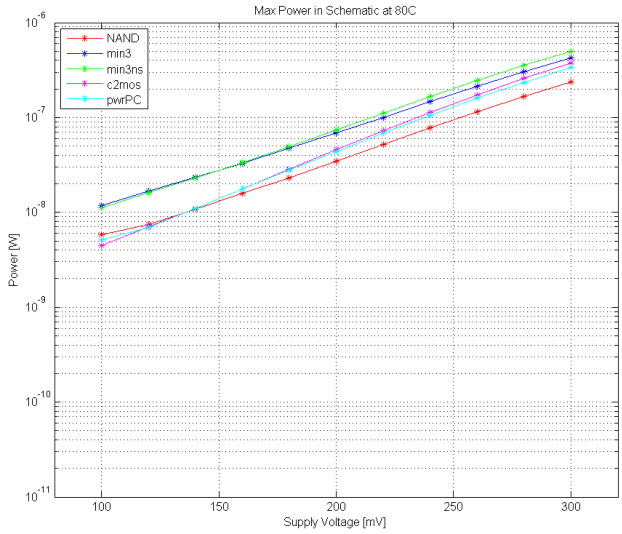


(a) Schematic

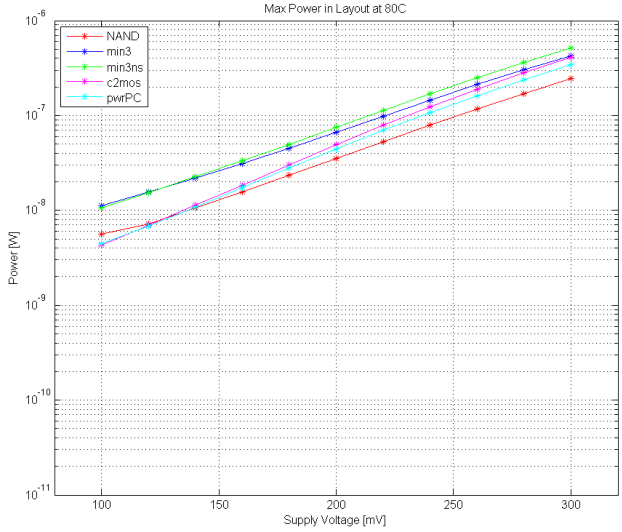


(b) Layout

Figure 80: Maximum Power Consumption at $27°C$



(a) Schematic

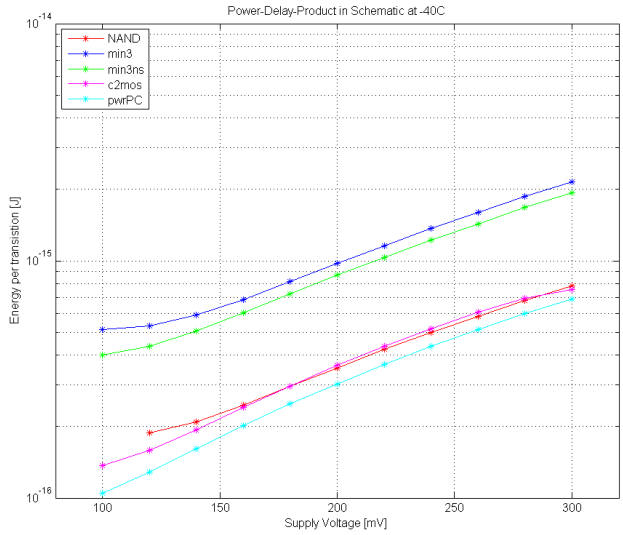


(b) Layout

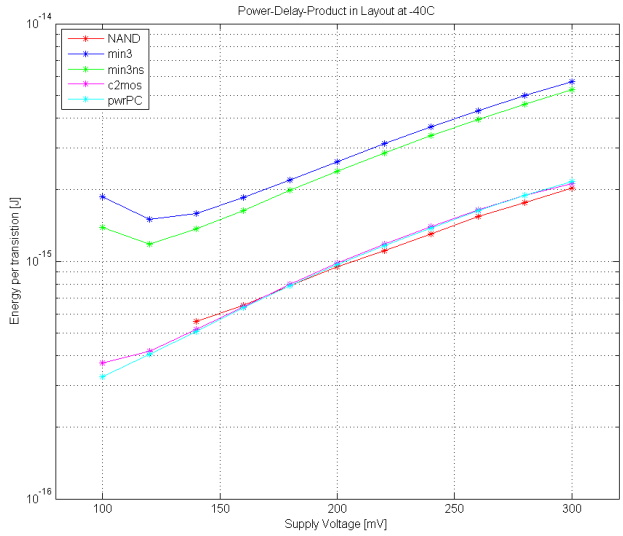
Figure 81: Maximum Power Consumption at $80°C$

8.8 Power-Delay-Product

The Power-Delay-Product results are presented here.

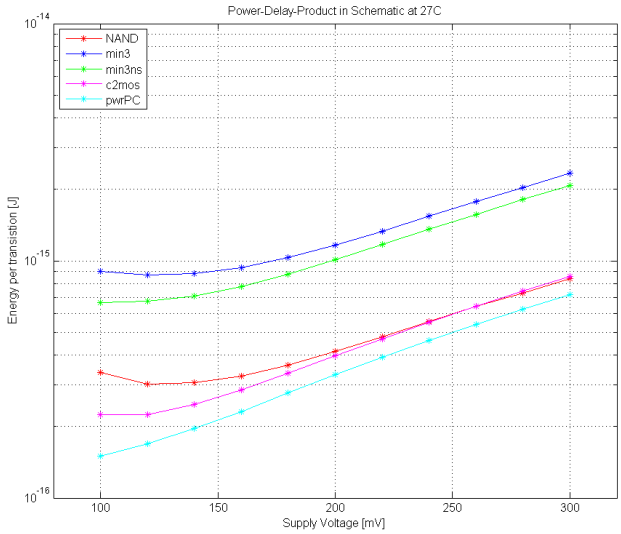


(a) Schematic

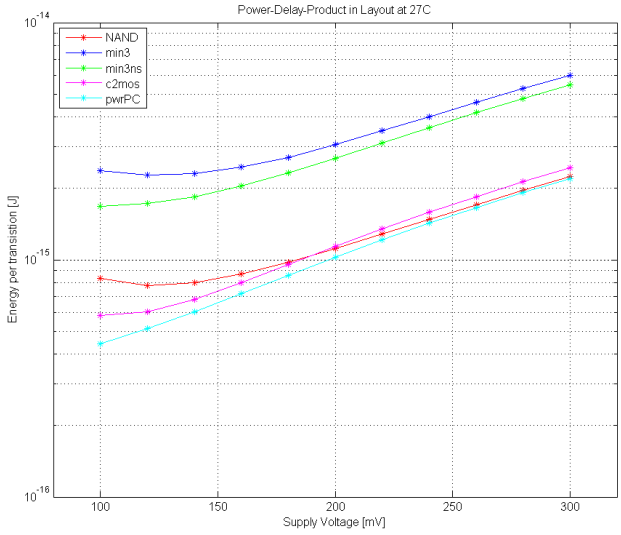


(b) Layout

Figure 82: Power-Delay-Product at $-40°C$

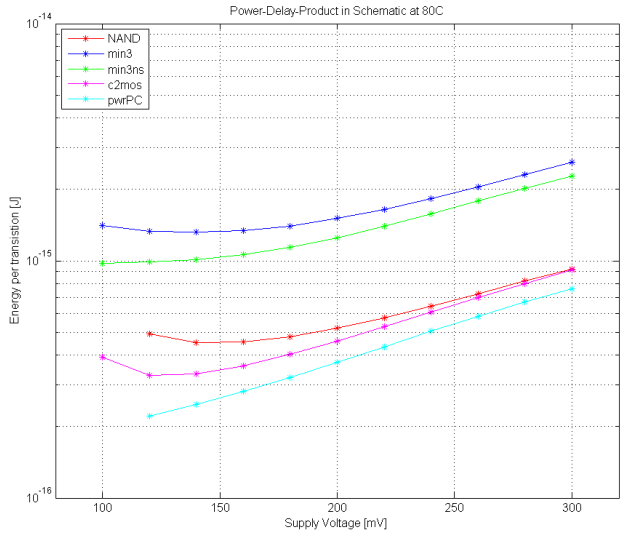


(a) Schematic

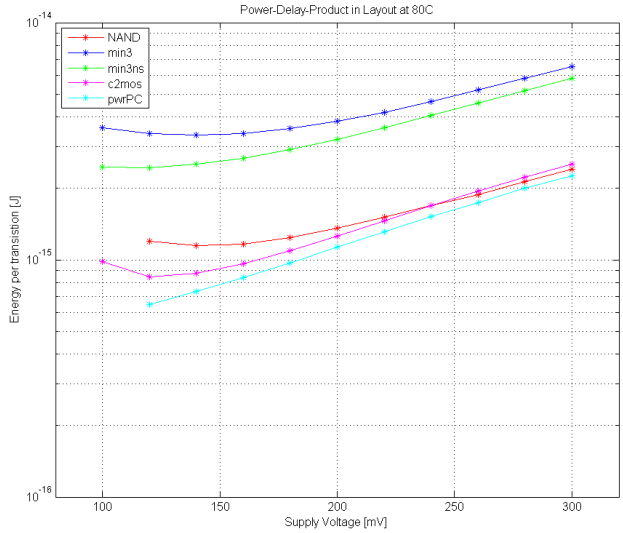


(b) Layout

Figure 83: Power-Delay-Product at $27°C$



(a) Schematic



(b) Layout

Figure 84: Power-Delay-Product at $80°C$

8.9 Monte Carlo Delay Simulation

Monte Carlo analysis are run for delay measurements using statistical process variation data. The results can be seen in this Subsection. The Statistical data can be seen in Appendix A. The supply voltage for all results are $250mV$.

8.9.1 Average Mean Delay and Standard Deviation for Schematic and Layout

The Average Mean Delay and Standard Deviation Results for $-40°C$ can be seen in Figure 85. The Results for $27°C$ in Figure 86, and the Results for $80°C$ in Figure 87.

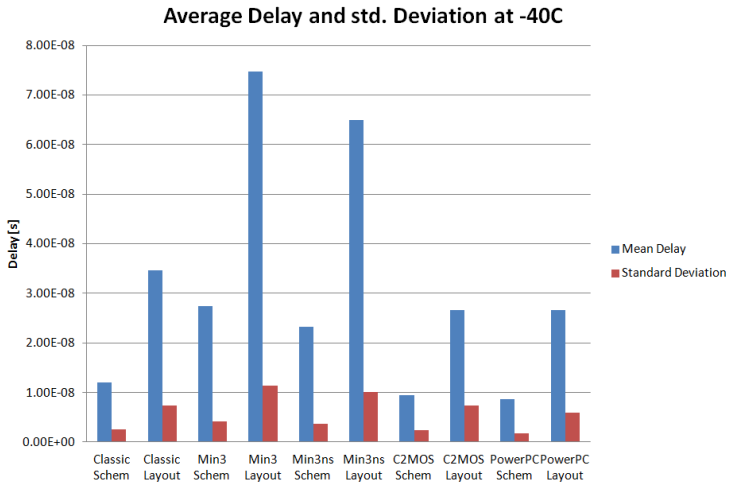


Figure 85: Monte Carlo Analysis Average Results and std. Deviation for schematic and layout at $-40°C$

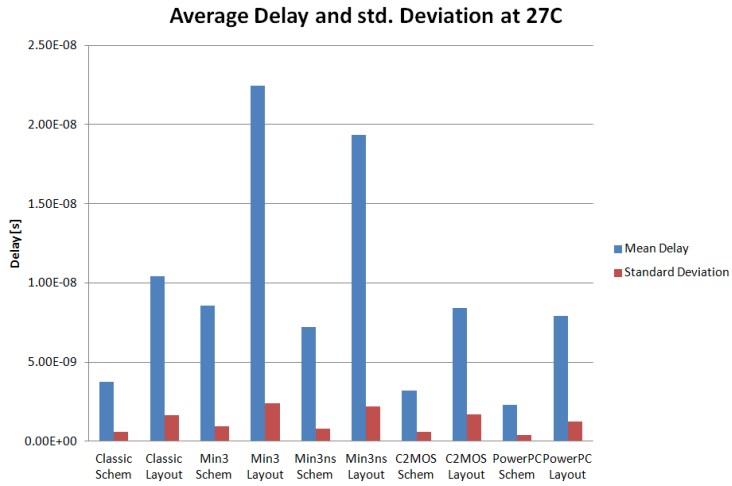


Figure 86: Monte Carlo Analysis Average Results and std. Deviation for schematic and layout at $27°C$

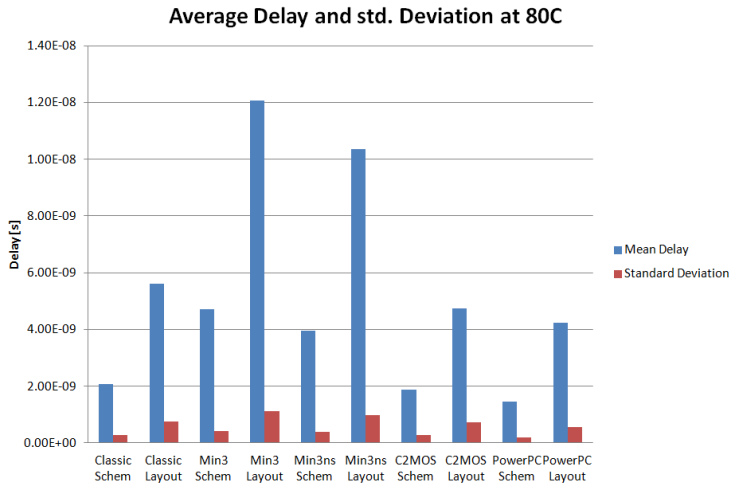


Figure 87: Monte Carlo Analysis Average Results and std. Deviation for schematic and layout at $80°C$

8.9.2 Average Mean Delay for Schematic and Layout at different temperatures

Figure 88 shows the average mean delays for all D flip-flops at schematic and layout, and how these changes with the temperature.

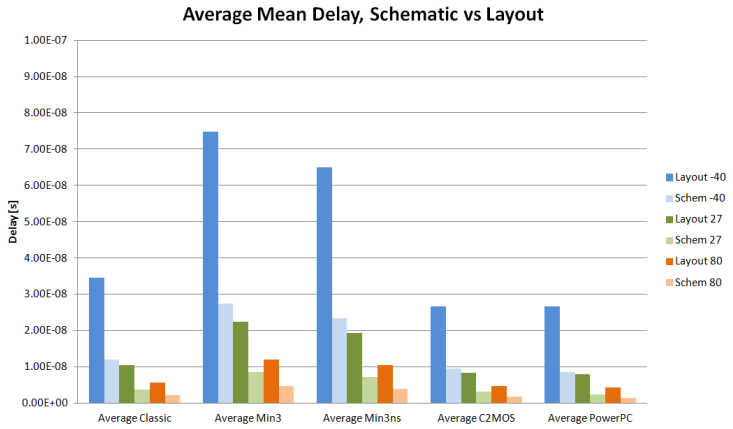


Figure 88: Monte Carlo Analysis Average Results for schematic and layout

8.9.3 Worst Case Mean Delay for Schematic and Layout at different temperatures

Figure 89 shows the worst case delay for all D flip-flops at schematic and layout, and how these changes with the temperature.

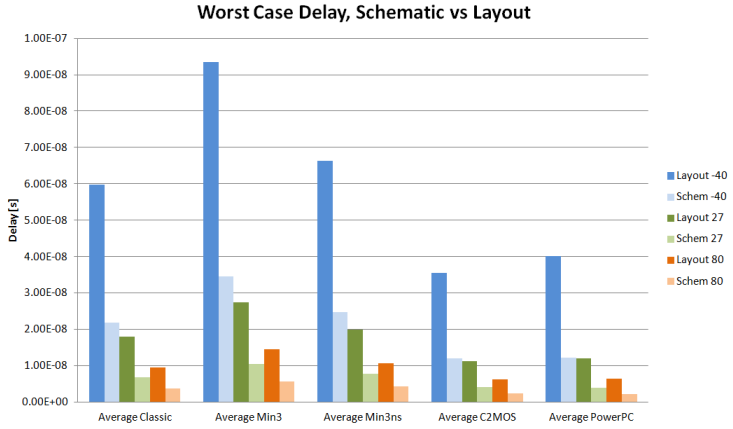


Figure 89: Monte Carlo Analysis Worst Case Results for schematic and layout

8.9.4 Relative Standard Deviation Comparison

Figure 90 shows the relative standards deviation for all D flip-flops at schematic and layout for all temperatures. These results can also be seen in Appendix A.3.

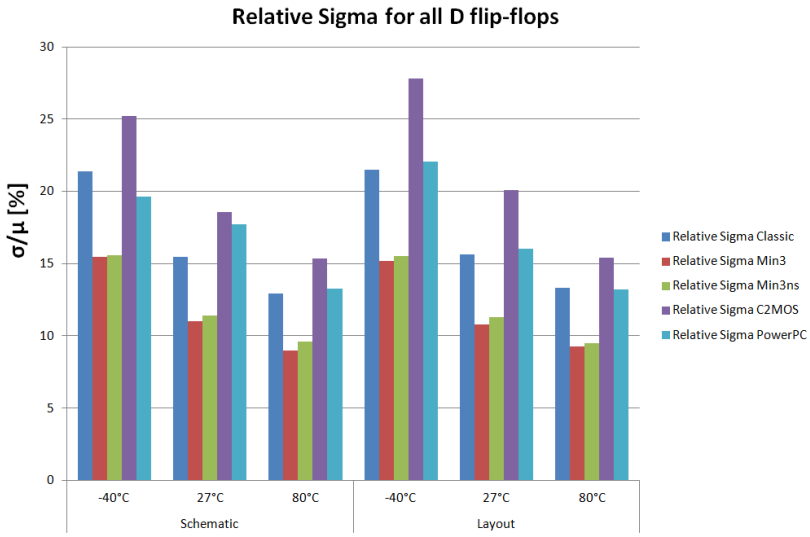


Figure 90: Relative Standard Deviation for all D flip-flops. The Y-value is average Standard Deviation divided by the average Mean Delay

9 Discussion

This Section contains the discussion of the results from Section 8.

9.1 Transistor Layout

The transistor layout results will be discussed in this Section. First the parasitic extraction, and then the nWell edge placement.

9.1.1 Parasitic Extraction

Table 15 and Table 16 shows that the xACT3D extraction gives higher capacitances for most nets, and significantly higher for net *in* and net 2. The layout with xRC parasitic extraction, and xACT3D extraction are run through the same test bench to calculate the delay and power consumption differences. The results are displayed in Table 17, and shows that the difference is significant when it comes to both propagation delay and power consumption. The propagation delay increases with over 31% and the power consumption increases with approximately 59% at the same time. The xACT3D extraction is more time consuming than the xRC, but since the creator of the extraction tools claim that xACT3D is the most accurate one[2], all layout parasitics extraction are done by xACT3D.

9.1.2 nWell Placement and Sizing

The results in Figure 42 shows that the threshold voltage of the pMOS increases with $38mV$, or 12% from the schematic value when using the minimum nWell size of $160nm$. An increase in the threshold voltage by $38mV$ causes a large increase in the Drain current, and consequently increases the nMOS/pMOS imbalance. As the nWell size increases, and the distance from the nWell edge to the active gate area increases, the Figure shows the Threshold value for the pMOS transistor approaches its schematic value, but at $2\mu m$ distance there is still around $3mV$ deviation. Because of the exponential relationship between threshold voltage and drain current in subthreshold operation, the threshold voltage variation should be kept to a minimum. So the distance from the nWell edge to the pMOS transistors active gate area are kept to minimum $2\mu m$ for all D flip-flop layouts. The nMOS transistors are also affected by the nWell edge distance to its active gate area, but not in the same degree. As the distance passes $1\mu m$, the threshold curve starts to flatten and slowly approach its schematic value. The nMOS active gate area distance to the nWell is therefore kept to $1\mu m$.

The trade-off for this relatively high nMOS to pMOS distance is the chip area, since it increases with the distance between the transistors, but also for the increased nWell area above the pMOS transistors.

One other problem with the nWell is the relative distance for each transistor. If one transistor is closer to the edge of the nWell than others, its threshold voltage will be different and can cause mismatch between balanced transistors. The WPE could also be higher for transistors at the end of transistor rows, as

they are surrounded by the nWell edge at one side, in contrast to transistors in the middle of the row.

9.2 D flip-flop functionality

Figure 43 shows that the D flip-flops function with $250mV$ supply voltage at $27°C$. The C^{2}MOS and the PowerPC 603 D flip-flops seems to be the fastest, atleast at the falling edge of Q.

9.3 Delay Comparison of D flip-flops

The Setup time, the propagation delay and the total delay measurements will be discussed in this section.

9.3.1 Master latch delay

The master latch delay (Setup time tsu) results are shown in Section 8.3.1 and shows the delay as a function of the supply voltage.

The $-40°C$ results are shown in Figure 44 and Figure 45, and shows four figures which compares the Setup time results for schematic and layout. In all sub-figures, the PowerPC 603 gives the best result for all supply voltage values. The C^{2}MOS and the Classic NAND D flip-flop shows similar results, where the C^{2}MOS is a bit faster at rising edge, and the Classic NAND is a bit faster at falling edge. The Minority3 D flip flop is the overall slowest one, with the Minority3 no-set D flip-flop a bit faster.

The $27°C$ results can be seen in Figure 46 and Figure 47, and shows much the same as the $-40°C$ results. The PwrPC 603 is still the fastest for all supply voltages, with C^{2}MOS and the Classic NAND D flip-flop a bit slower. The Minority3 is the slowest, with the Minority3 no-set D flip-flop a bit faster.

The $80°C$ results are shown in Figure 48 and Figure 49, with the PowerPC 603 as the fastest for all supply voltages. The Classic NAND D flip-flop is still a bit faster than the C^{2}MOS at the falling edge condition, and is also faster at voltages lower than $200mV$ at rising edge condition. The Minority3 D flip-flop is the slowest, with the Minority3 no-set D flip-flop is a bit faster.

The delay increases by roughly 2.8 times from schematic to layout, which should be expected with the introduction of parasitics.

9.3.2 Slave latch delay

The slave latch delay (propagation delay tco) will be discussed in this Subsection. The results are shown in Section 8.3.2.

The $-40°C$ results are shown in Figure 50 and Figure 51. The C^{2}MOS has the lowest propagation delay at rising edge, especially in layout. At falling edge, the PowerPC 603 is fastest for both schematic and layout. One thing that is noticed is that the Classic NAND is among the fastest at rising edge, but is among the slowest at falling edge. The reason for this is the increased critical path for the signal, at this exact condition. The clock signal needs to propagate through two NAND gates and one inverter before changing the last D-latch, where the signal must propagate through the two last NANDs. This sums up to 4 NAND gates and on inverter, in contrast to two NAND gates for the other conditions. The C^{2}MOS seems to do be less affected by layout non-idealities at rising edge, than for falling edge.

The $27°C$ results can be seen in Figure 52 and Figure 53, and show much the same as for $-40°C$. The Classic NAND has still a much higher propagation delay for falling edge.

The $80°C$ results are displayed in Figure 54 and Figure 55, and shows that C^{2}MOS is the fastest D flip-flop at rising edge, and PowerPC 603 is fastest at falling edge. The Classic NAND is still very slow at falling edge compared to rising edge.

The total results, show that the propagation delays across the D flip-flops are closer to each other than for the setup time. The Minority3 D flip-flop has lower propagation delay than the Minority3 no-set D flip flop, in contrast to the setup delay. This can be explained by the change in structure because of the removal of the Set input. The Minority3 no-set has less transistors in the master latch, but one more inverter in the slave latch.

9.3.3 Total D flip-flop delay

The total delay is the sum of propagation delay and total delay and should show the overall fastest D flip-flops. The results are presented in Section 8.3.3.

The $-40°C$ results are shown in Figure 56 and Figure 57, and shows that the Classic NAND, the C^{2}MOS and the PowerPC 603 have very similar delay at rising edge schematics. At layout rising edge, the C^{2}MOS is fastest. At falling edge the differences are more clear, the PowerPC 603 is fastest at both schematics and layout. The C^{2}MOS and Classic NAND are number two and three, while the Minority3 is the slowest one.

The $27°C$ results are shown in Figure 58 and Figure 59, and shows exactly the same as the $-40°C$ results.

The $80°C$ results also gives much the same results as for the lower temperatures, as seen in Figure 60 and Figure 61.

The total results for total delay shows that the PowerPC 603 and C^{2}MOS is the fastest D flip-flops. Even though the total delay shows which D flip-flop that

has the lowest total delay, it does not necessarily means that D flip-flop is the fastest. The worst case delay of all delays should set the maximum frequency, to be sure that the D flip-flop functions under all circumstances. The Maximum total delay, the lowest delay, and the highest delay for all D flip-flops can be seen in Table 18.

Dff	Max. Total Delay	Min. Delay	Max. Delay
Classic NAND	$10.44ns$	$2.798ns$	$7.645ns$
Minority3	$19.45ns$	$7.568ns$	$11.93ns$
Minority3 no-set	$16.90ns$	$8.030ns$	$8.652ns$
C^2MOS	$7.950ns$	$2.832ns$	$4.560ns$
PowerPC 603	$7.391ns$	$2.234ns$	$5.102ns$

Table 18: Delay Comparison at 300mV, $27°C$, Layout

9.4 Maximum D flip-flop frequency based on maximum delay

The D flip-flop maximum frequency is as mentioned in Section 7.7, the frequency based on the maximum delay. The calculated maximum frequency for all D flip-flops can be seen in Figure 62, Figure 63 and Figure 64. The PowerPC is the fastest D flip-flop at Schematic at all temperatures, just in front of the C^2MOS. The Classic NAND D flip-flop and the Min3 no-set D flip-flop shows similar results, but with the Classic NAND a bit faster. The Min3 D flip-flop is the slowest for all temperatures. At layout, the results are the same, except that the C^2MOS is faster than the PowerPC D flip-flop at higher supply voltages. Figure 65 shows how the maximum frequencies changes with temperature.

These results agrees with previous results in [6], where the PowerPC 603 showed the best delay results of the Static D flip-flops on schematic. It also compares a NAND-based D flip-flop to the PowerPC 603 and the C^2MOS, but the structure and functionality is different. The C^2MOS results consists with the results in [6], with a schematic delay a little higher than the PowerPC 603. [14] compares PowerPC 603 with C^2MOS at schematic level and shows that the PowerPC 603 is slightly faster than the C^2MOS. However, [17] shows that the C^2MOS is faster than the PowerPC 603 at schematic simulations. The Classic NAND results were meant to be compared to the results of the NAND-based D flip-flop implementation presented in [6], but the basic NAND cell presented as a figure in the paper appeared to be non-functional, and the reference shows a NANDNOR D flip-flop.

9.5 Static power consumption at different inputs

The static power consumption results are shown in Figure 66, Figure 67, Figure 68 and Figure 69. The figures show that different input values affects the static power consumption of the D flip-flops. The PowerPC 603 has an increase in static power

consumption when D is high and clk is low. The C^{2}MOS has a bit higher static power consumption when D is high, and Classic NAND has its highest when D is low and clk is high. The Minority3-based D flip-flops are not affected much by the change in input values.

By comparing the schematic results with the layout results, all D flip-flops seems reduce their static power consumption by around 5% when moving from schematic to layout. This reduction in the static power consumption is a result by a reduction in the leakage current of the transistors. By looking at the leakage current Expression 9 and Expression 10 in Subsection 3.1.3, the reduced leakage current most likely comes from increased threshold voltage when moving from schematics to layout.

9.5.1 Static power comparison at different temperatures

The results in Figure 70 shows the static power consumption for D high and Clk low for the temperatures $-40°C$, $27°C$ and $80°C$.

In Subsection 3.3.1, Expression 13 shows that the threshold voltage decreases as the temperature rises for subthreshold operation. This threshold voltage decrease causes the leakage current to rise as seen in Figure 70. It can also be seen that the static power consumption in layout is slightly lower than for schematic.

9.6 Total power consumption

The results for the total power consumption measurements can be seen in Figure 71, Figure 72 and Figure 73. The results show that the PowerPC 603 has the lowest total power consumption at all temperatures for both schematic and layout. Right behind the PowerPC 603 is C^{2}MOS and the Classic NAND D flip-flop, which shows very similar results. The Minority3-based D flip-flops stands out by having a significantly higher total power consumption, as seen in the Figures. The no-set D flip-flop has still a slightly lower total power consumption than the Minority3 D flip-flop at all temperatures.

[6] compares the PowerPC 603 and C^{2}MOS D flip-flops total power consumption in the same way as done in this thesis. It also shows that the PowerPC 603 has a slightly lower power consumption while operating at the same frequency. [14] shows the same results.

9.6.1 Total Power Consumption Schematic vs Layout Comparison

Figure 74, Figure 75 and Figure 76 compares the schematic and layout total power consumption as the supply voltage changes. The results show that the total power consumption is close to equal for the schematic and layout at $100mV$, but as the supply voltage increases, the layout gradually consumes more power than the schematic. The equal total power consumption at $100mV$ can be explained by the domination of static power consumption, which is around 5% lower for layout than schematic as explained in Section 9.5. But as the supply voltage increases, the dynamic power consumption starts to dominate, and the increase in parasitic output capacitance causes the layout to have a higher total power

consumption.

9.6.2 Total Power Consumption Temperature Comparison

By comparing the total power consumption over different temperatures as seen in Figure 77 and Figure 78, the effect of the threshold voltage increase at low temperatures becomes visible. At $100mV$ the total power consumption is close to the static power consumption, since it dominates at this supply voltage. As the supply voltage increases, the total power consumption starts to dominate. This causes the total power consumption differences between the temperatures to decrease, since the dynamic power consumption is not as temperature dependent as the static.

9.7 Maximum Power Consumption

For the Maximum Power Consumption Results in Figure 79, Figure 80 and Figure 81, all D flip-flops operates at its maximum frequency. The results show that the Classic NAND consumes the least power while operating at maximum speed. The PowerPC has the second lowest consumption, followed by the C^2MOS. The Min3 no-set D flip-flop has the highest maximum power consumption, and the Min3 is a little lower.

The main reason for the low Maximum Power Consumption for the Classic NAND D flip-flop is its maximum frequency. The maximum frequency is limited to its worst case delay at falling edge of the slave latch, and is around twice as high than its other delays. This causes the Classic NAND to spend much of the working waiting, and not consuming dynamic power. A second reason can be the lower pMOS transistor area of the NAND-gates used in the Classic NAND D flip-flop, which gives lower Capacitances, and can lower the subthreshold current (Drive).

9.8 Power-Delay-Product

The results for the Power-Delay-Product simulation can be seen in Figure 82, Figure 83 and Figure 84.

At $-40°C$, the PowerPC has the lowest Energy per switching at schematic, but for layout the Classic NAND has a lower Energy per switching for supply voltages above $180mV$. The C^2MOS has similar results to the Classic NAND in schematic and PowerPC in layout. The Minority3-based D flip-flop has the highest Energy per switching, with the Minority3-based no-set D flip-flop a little lower. While no D flip-flop meet its minimum point of Energy used per transition at schematic, both Min3 and Min3ns meet their minimum at $120mV$ in layout. This is because the layout adds parasitic capacitances to the circuit, which gives a higher propagation delay and pushes the minimum point to a higher supply voltage.

At $27°C$, the PowerPC D flip-flop has the lowest Energy per transition at both schematic and layout. The C^2MOS is the second best at supply voltages under $180mV$, but over $180mV$, the Classic NAND is the second best. The Min3 has the highest energy per transition, with the Min3ns as the second highest. All D flip-flops except the PowerPC D flip-flop shows a minimum Energy per transition point, or a point close to minimum. For the Min3 and the Classic NAND D flip-flop, the minimum Energy per transition point is around $120mV$. The C^2MOS and Min3ns shows a minimum Energy per transition point around $100mV$.

At $80°C$, the PowerPC has still the lowest Energy per transition, with the C^2MOS as the second best for supply voltages under $240mV$. The Classic NAND has the second lowest Energy per transition for supply voltages over $240mV$.

By comparing the results across temperatures, the Energy per transition increases as the temperature rises for low supply voltages. This happens because the static power consumption starts to dominate at this point, and the static power consumption is dependent on the temperature, because of the change in threshold voltage. At higher supply voltages, the difference is minimal since the dynamic power dominates.

The overall PDP results show that the PowerPC 603 is the D flip-flop with the lowest Energy consumption per transition. The Classic NAND has the second lowest Energy consumption per transition for supply voltages over around $240mV$. The C^2MOS is the second best for supply voltages under $160mV$. For the voltages between $160mV$ and $240mV$, the C^2MOS and the Classic NAND are very similar. The Min3 D flip-flop has the highest Energy consumption at all supply voltages and temperatures, with the Min3ns a little lower.

The Classic NAND D flip-flop has a significantly higher worst case delay than the C^2MOS and the PowerPC D flip-flop, because of reasons mentioned in Section 9.3.2. However, it shows very good results in PDP, which is connected to the low Maximum Power Consumption. This may also be connected to the reduced gate area of the Classic NAND D flip-flop.

[6] compares C^2MOS and PowerPC 603 PDP results on schematic, with results showing that the PDP of the C^2MOS is a little higher than for PowerPC 603, which consists with the PDP results in this thesis. [14] gives the same conclusions.

9.9 Monte Carlo Delay Simulation

This Section will be used to discuss the Monte Carlo results seen in Section 8.9.

9.9.1 Average Mean Delay and Standard Deviation for Schematic and Layout

Figure 85, Figure 86 and Figure 87 shows the average mean delay and average standard deviation for the D flip-flops at schematic and layout. The results show that the Min3 has the highest average delay at both schematic and layout. The

Min3 no-set D flip-flop has a bit lower average mean delay than the Min3. The Classic NAND, the C^{2}MOS and the PowerPC D flip flop are relatively close in both average delay and standard deviation, though the PowerPC has a marginally lower delay and standard deviation. The average data can be seen in Table 35 in Appendix A.2. As the temperature rises the

9.9.2 Average Mean Delay for Schematic and Layout at different temperatures

Figure 88 shows the average mean delay for each D flip-flop at the different temperatures. The results show that an increase in temperature from $-40°C$ to $27°C$ reduces the relative delay by approximately the same for all D flip-flops. The average relative reduction is 3.29, which means that the delay at $27°C$ is 3.29 times lower than for $-40°C$. With an increase in temperature from $27°C$ to $80°C$ the relative reduction in delay is 1.79.

9.9.3 Worst Case Mean Delay for Schematic and Layout at different temperatures

The Worst Case Results can be seen in Figure 89, and shows the worst case delay for the D flip-flops for schematic and layout at all temperatures. As mentioned earlier, the worst case delay limits the maximum D flip-flop frequency, so this results give a more realistic delay comparison. The results show that the PowerPC and the C^{2}MOS gives the best and very similar results. The Classic NAND has fallen behind, because of its high propagation delay at falling edge. The Min3 and Min3 no-set is still the slowest.

9.9.4 Relative Standard Deviation Comparison

Figure 90 shows the relative standard deviation for the different D flip-flops at the different temperatures. It shows that the Min3 D flip-flop has the lowest relative standard deviation for all temperatures, at both schematic and layout. It goes from around 15% at $-40°C$, to around 11% at $27°C$, and around 9% at $80°C$. The Min3 no-set D flip-flop is very close to the Min3, with an average of 0.36 percentage points above. The PowerPC and the Classic NAND shows very similar results at layout with results around 22%, 16% and 13% for the temperatures $-40°C$, $27°C$ and $80°C$, respectively. Their differences are a bit larger in schematic. The C^{2}MOS D flip-flop shows the overall highest relative standard deviation at all temperatures. At layout the results are approximately 27.8%, 20.1% and 15.4% at the temperatures $-40°C$, $27°C$ and $80°C$, respectively.

The low relative standard deviation delay results for the Minority-3 gate makes it the most robust to process variations in this sense. The low supply voltage results in [32], can be linked to this, where the Minority-3 based Ripple-Carry Adder showed correct functionality down to $106mV$ with tuning the body bias, and $119mV$ without the body voltage set to V_{DD}. This very low supply voltage indicates a good nMOS/pMOS balance as mentioned in Section 3.2.

[10] proposes a 8-transistor logic configuration which will be called the 8T-gate here. It has a forced stacking of 2, and it can take the form of a NAND or

NOR gate. [25] compares the 8T-gate configuration to the Minority3-gate, and the 8T-gate shows a lower PDP and a lower delay, but the static leakage is higher. The most important results for the 8T-gate is that the relative standard deviation is significantly lower than for the Minority3, for all results. The advantages of this will be mentioned in the future work Section 10.1.

9.10 The Total Results

By looking at the results in total, the D flip-flops have different usage advantages and disadvantages.

The fastest D flip-flop were found by looking at the delay measurements, and the Monte Carlo analysis. Early in the simulations, the C^{2}MOS, the PowerPC and the Classic NAND stood out as the best for most of the simulations. By using the highest delay measurement as the maximum frequency limit, the Classic NAND D flip-flop falls behind, because of its high propagation delay at falling edge. By looking at Figure 65 in Section 8.4, the PowerPC shows the highest maximum frequency at schematic, at all temperatures, with the C^{2}MOS right behind. The Layout results show that the C^{2}MOS is the fastest at supply voltages above around $200mV$, which makes it the fastest at the balance voltage $250mV$. The PowerPC is the second fastest and the Classic NAND is number three, right in front of the Min3 no-set D flip-flop. The Min3 D flip-flop has the lowest maximum frequency for all temperatures. Table 19 shows the maximum frequency at $250mV$ supply voltage at $27°C$. The $250mV$ data is interpolated by the values for $240mV$ and $260mV$.

Dff	Maximum Frequency
Classic NAND	$28.61MHz$
Minority3	$18.87MHz$
Minority3 no-set	$25.48MHz$
C^{2}MOS	$47.42MHz$
PowerPC 603	$43.20MHz$

Table 19: Maximum Frequency of D flip-flops, $250mV$ supply voltage at $27°C$ Layout

The Static Power Consumption results show that the PowerPC 603 D flip-flop consumes the least power when there is no switching on the inputs. The C^{2}MOS is second and the Classic NAND is third. The Min3 and the Min3 no-set D flip-flops consumes over twice as much static power than the rest.

The D flip-flop with the lowest total power consumption is the PowerPC, which is marginally better than the Classic NAND and the C^{2}MOS at all supply voltages and temperatures. At low supply voltage values, the C^{2}MOS has the second lowest power consumption, but the Classic NAND is the second lowest at higher supply voltages. The Min3 D flip-flop has the highest total power

consumption, with the Min3 no-set a little better. The interpolated total power consumption for Layout operating at $250mV$ with the temperature $27°C$ can be seen in Table 20.

Dff	Total Power Consumption
Classic NAND	$65.55nW$
Minority3	$165.10nW$
Minority3 no-set	$142.18nW$
C^2MOS	$67.84nW$
PowerPC 603	$61.35nW$

Table 20: Total Power Consumption of D flip-flops, $250mV$ supply voltage at $27°C$ Layout

The Power-Delay-Product simulations shows that the PowerPC 603 overall uses the least Energy per transition at all supply voltages and temperatures, except at $-40°C$, where the Classic NAND is slightly better at higher supply voltages. The Classic NAND has the second lowest Energy per transition at higher supply voltages (over $240mV$). The C^2MOS is second best at low supply voltages (under $180mV$). The Min3 and the Min3 no-set shows a significantly higher PDP than the rest of the D flip-flops. The interpolated PDP on Layout at $250mV$, $27°C$ is shown in Table 21.

Dff	Total Power Consumption
Classic NAND	$1.591fJ$
Minority3	$4.314fJ$
Minority3 no-set	$3.895fJ$
C^2MOS	$1.716fJ$
PowerPC 603	$1.545fJ$

Table 21: Power-Delay-Product of D flip-flops, $250mV$ supply voltage at $27°C$ Layout

The Monte Carlo Delay Simulations showed that Min3 and the Min3 no-set had the lowest relative standard deviation, at all temperatures. The Classic NAND and the PowerPC had similar results some percentage points above the Min3 and the Min3 no-set. The C^2MOS gave the highest relative standard deviation results at all temperatures.

10 Concluding Remarks

The results show that the PowerPC 603 D flip-flop has the lowest Static Power Consumption, the lowest Total Power Consumption, and the overall best Power-Delay-Product results in layout simulations. It also has the second best switching speed, right behind the C^2MOS D flip-flop. It is well suited for high-speed low-power systems, and is already used in the PowerPC 603 low-power microprocessors. The C^2MOS D flip-flop has the highest switching speed, good power consumption and PDP results, but is the most susceptible to propagation delay variations because of process variations. When it comes to delay and power consumption, [6], [14] and [17] show similar results for the C^2MOS and PowerPC 603 for simulations based on the schematic.

For subthreshold systems where the yield is of the highest priority, the Minority3 D flip-flop is the best choice. It is relatively slow compared to the other D flip-flops, it has a higher power consumption and higher Energy consumption per transition, but has a low relative standard deviation on delay measurements compared to the other D flip-flops. The Minority3 no-set D flip-flop shows similar results to the Minority3. It is a bit faster, has a slightly lower power consumption and PDP results. However, the relative standard deviation is little higher.

The Classic NAND has a relative high propagation delay compared to the best. Static power consumption is average, but the total power consumption and maximum power consumption is among the best. The PDP results are second best, right behind the PowerPC. The relative standard deviation when it comes to delay is average.

10.1 Improvements of the D flip-flops and Future Work

There are some methods that can improve the D flip-flops further. [23] describes how forced stacking of devices can reduce the leakage current, which can give large power savings for devices with low switching activity. Some blocks are already stacked, like the clocked inverter, so the forced stacking is mainly ment for the inverter gate, Minority3-gate and the CMOS NAND-gate.

By using the 8T-gates, the Minority3-based NAND-gates in the Minority3 D flip-flop can be implemented with less transistors, and at the same using more robust building blocks, which can improve the total robustness against process variations. The current Minority3 D flip-flop has a total transistor count of 88, but with the 8T-improvement, the total transistor count will be 78, which is a 11.36% reduction.

This 8T-gate transistor reduction technique can also be used with the high regularity layout structure used in this Thesis. The total transistor count for the Minority3 D flip-flop will be reduced from 120 to 100 transistors in this case, which is a 16.67% reduction. This reduction is in addition to keeping a highly regular layout, which should improve the yield, and by using more robust building blocks, which can further improve the yield.

A possible improvement of the RX/TX-module made by Q-free ASA men-

tioned in [32], is to replace the Minority3-based NANDs and NORs with the 8T-gates, which is shown to have a better robustness against process variations. In addition will the 8T-gates reduce the total transistor count by reducing a NAND-gate from 12 transistors to 8 transistors. In addition does the RX/TX-module consists of 2 XOR-gates constructed by 3 Minority3-gates and two inverters. The XOR-configuration can be simplified to a 8-transistor CMOS XOR-gate with two inverters, and reduce the transistor count from 34 to 12 for each XOR.

By quick calculations, the total block count of the RX/TX-module is 140 inverters, 24 Min3, 67 NAND-coupled Min3, 1 NOR-coupled Min3 and 2 Min3-based XOR-gate. The transistor count for each block for the current design and the new proposed design can be seen in Table 22.

	Inv.	Min3	NAND	NOR	XOR	Tot.
Current Design	280	240	670	10	68	1268
New Design	280	240	536	8	24	1088

Table 22: Transistor count in the current design, and the transistor count in the new proposed design, for the RX/TX-module

The new proposed design reduces the total transistor count from 1268 to 1088 transistors, which is a 14.2% reduction. In addition should the new design show a low relative standard deviation, because of the 8T-gates, and could cause a better robustness against process variations, and therefore provide an improved yield.

Body biasing can be used to tune the threshold voltage of the transistors to achieve better nMOS/pMOS balance. It can also reduce the leakage currents by increasing the threshold voltage. Pins to the bulk-gate of the transistors can be implemented to statically or dynamically tune the V_{SB} to achieve the desired functionality.

By switching to high-threshold voltage transistors, the total power consumption can be greatly reduced, for the cost of reduced switching speed. This Thesis mainly compares D flip-flop structures, but most results should be comparable in other transistor technologies.

The high regularity should make the designs more robust against manufacturing process-created process variations. A tape-out of the D flip-flops and measurements on chip can show if the designs functions, and the results can be compared to the layout results simulated in Cadence Virtuoso.

Papers regarding the results found in this Thesis are in the making.

References

[1] Cadence ocean referance, product version 4.4.6. http://www.ece.gatech.edu/academic/courses/ece4430/ECE4430/Unit1/EKV_Model_Extraction/oceanref.pdf. Accessed: 2013-06-01.

[2] Mentor Graphics Calibre xACT3D. http://www.mentor.com/products/ic_nanometer_design/verification-signoff/circuit-verification/calibre-xact/. Accessed: 2013-04-21.

[3] Mentor Graphics Calibre xRC. http://www.mentor.com/products/ic_nanometer_design/verification-signoff/circuit-verification/calibre-xrc/. Accessed: 2013-04-21.

[4] Tc7w74fu datasheet (pdf) - toshiba semiconductor - d-type flip flop with preset and clear. http://www.alldatasheet.com/datasheet-pdf/pdf/32075/TOSHIBA/TC7W74FU.html. Accessed: 2013-05-20.

[5] M. Alioto. Ultra-low power vlsi circuit design demystified and explained: A tutorial. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 59(1):3–29, 2012.

[6] H.P. Alstad and S. Aunet. Seven subthreshold flip-flop cells. In *Norchip, 2007*, pages 1–4, 2007.

[7] 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.

[8] S. Aunet and Amir Hasanbegovic. Memory elements based on minority-3 gates and inverters implemented in 90 nm cmos. In *Design and Diagnostics of Electronic Circuits and Systems (DDECS), 2010 IEEE 13th International Symposium on*, pages 267–272, 2010.

[9] A. Bellaouar, A. Fridi, M.I. Elmasry, and K. Itoh. Supply voltage scaling for temperature insensitive cmos circuit operation. *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, 45(3), 1998.

[10] Jonathan Bjerkedok. Real Time Clock in Subthreshold CMOS. Master's thesis, Department of Electronics and Telecommunications, Norwegian University of Science and Technology.

[11] M. Blesken, S. Lutkemeier, and U. Ruckert. 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, 2010.

[12] Stephen Brown and Zvonko Vranesic. *Fundamentals of Digital Logic with VHDL Design*. The McGraw-Hill Companies, Inc, 2009.

[13] A. Chavan, G. Dukle, B. Graniello, and E. MacDonald. Robust ultra-low power subthreshold logic flip-flop design for reconfigurable architectures. In *Reconfigurable Computing and FPGA's, 2006. ReConFig 2006. IEEE International Conference on*, pages 1–7, 2006.

[14] Bo Fu and P. Ampadu. Comparative analysis of ultra-low voltage flip-flops for energy efficiency. In *Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on*, pages 1173–1176, 2007.

[15] G. Gerosa, S. Gary, C. Dietz, Dac Pham, K. Hoover, J. Alvarez, H. Sanchez, P. Ippolito, Tai Ngo, S. Litch, J. Eno, J. Golab, N. Vanderschaaf, and J. Kahle. A 2.2 w, 80 mhz superscalar risc microprocessor. *Solid-State Circuits, IEEE Journal of*, 29(12):1440–1454, 1994.

[16] Alan Hastings. *The Art of Analog Layout*. Prentice Hall, 2001.

[17] Wei Jin, Sheng Lu, Weifeng He, and Zhigang Mao. Robust design of subthreshold flip-flop cells for wireless sensor network. In *VLSI and System-on-Chip (VLSI-SoC), 2011 IEEE/IFIP 19th International Conference on*, pages 440–443, 2011.

[18] David Johns and Ken Martin. *Analog Integrated Circuit Design*. Wiley, 1996.

[19] William Kleitz. *Digital Electronics, A Practical Approach with VHDL, Ninth Edition*. Pearson Education, Inc., 2012.

[20] 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.

[21] 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.

[22] Tong Lin, Kwen-Siong Chong, Bah-Hwee Gwee, Joseph S. Chang, and Zhao-Xiang Qiu. Analytical delay variation modeling for evaluating sub-threshold synchronous/asynchronous designs. In *NEWCAS Conference (NEWCAS), 2010 8th IEEE International*, pages 69–72, 2010.

[23] S. Narendra, S. Borkar, V. De, D. Antoniadis, and A. Chandrakasan. Scaling of stack effect and its application for leakage reduction. In *Low Power Electronics and Design, International Symposium on, 2001.*, pages 195–200, 2001.

[24] K. Roy, S. Mukhopadhyay, and H. Mahmoodi-Meimand. Leakage current mechanisms and leakage reduction techniques in deep-submicrometer cmos circuits. *Proceedings of the IEEE*, 91(2):305–327, 2003.

[25] Lars-Frode Schjolden. Low Energy Implementation of Robust Digital Arithmetic in Sub/Near-Threshold Nanoscale CMOS. Master's thesis, Department of Electronics and Telecommunications, Norwegian University of Science and Technology.

[26] Yi-Ming Sheu, KeWei Su, Sheng-Jier Yang, Hsien-Te Chen, Chih-Chiang Wang, Ming-Jer Chen, and S. Liu. Modeling well edge proximity effect on highly-scaled mosfets. In *Custom Integrated Circuits Conference, 2005. Proceedings of the IEEE 2005*, pages 831–834, 2005.

[27] 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.

[28] H. Sunagawa, H. Terada, A. Tsuchiya, K. Kobayashi, and Hidetoshi Onodera. Erect of regularity-enhanced layout on printability and circuit performance of standard cells. In *Quality of Electronic Design, 2009. ISQED 2009. Quality Electronic Design*, pages 195–200, 2009.

[29] V. Suzuki, K. Odagawa, and T. Abe. Clocked cmos calculator circuitry. *Solid-State Circuits, IEEE Journal of*, 8(6):462–469, 1973.

[30] John P. Uyemura. *Introduction to VLSI Circuits and Systems*. Wiley, 2002.

[31] Eric A. Vittoz. Design of vlsi circuits for telecommunication and signal processing, micropower techniques. 1994.

[32] Magne Værnes. Ultra low power/low energy cmos/sub-threshold asic characterization. Semester report, Department of Electronics and Telecommunications, Norwegian University of Science and Technology, December 2012.

[33] Calhoun Benton Highsmith Chandrakasan Anantha P. Wang, Alice. *Subthreshold Design for Ultra Low-Power Systems*. Springer, 2006.

A Monte Carlo Results

A.1 Monte Carlo Delay Data

		Min. Delay	Max. Delay	Mean Delay	Sigma
Classic NAND	Schematic	4.058ns	25.53ns	8.374ns	2.329ns
	Layout	11.09ns	63.54ns	25.25ns	6.828ns
Minority3	Schematic	22.05ns	49.86ns	33.13ns	5.042ns
	Layout	63.91ns	132.7ns	92.2ns	13.43ns
Minority3 no-set	Schematic	15.65ns	35.87ns	23.01ns	3.796ns
	Layout	43.46ns	94.3ns	63.67ns	9.715ns
C^2MOS	Schematic	3.035ns	15.19ns	7.107ns	2.439ns
	Layout	9.781ns	66.26ns	21.96ns	8.197ns
PowerPC 603	Schematic	3.764ns	12.6ns	6.109ns	1.274ns
	Layout	9.437ns	35.16ns	17.81ns	4.341ns

Table 23: Monte Carlo Delay Results for Master latch, rising edge at $-40°C$

		Min. Delay	Max. Delay	Mean Delay	Sigma
Classic NAND	Schematic	4.067ns	13.03ns	7.048ns	1.581ns
	Layout	11.32ns	39.3ns	21.13ns	4.941ns
Minority3	Schematic	20.44ns	48.44ns	34.47ns	4.429ns
	Layout	64.23ns	144.1ns	93.54ns	11.7ns
Minority3 no-set	Schematic	16.01ns	35.03ns	23.15ns	3.285ns
	Layout	42.23ns	105.9ns	66.12ns	10.06ns
C^2MOS	Schematic	4.573ns	16.49ns	8.565ns	1.983ns
	Layout	12.22ns	49.6ns	24.31ns	5.401ns
PowerPC 603	Schematic	3.534ns	9.434ns	5.851ns	1.147ns
	Layout	9.512ns	32.44ns	17.72ns	3.904ns

Table 24: Monte Carlo Delay Results for Master latch, falling edge at $-40°C$

		Min. Delay	Max. Delay	Mean Delay	Sigma
Classic NAND	Schematic	5.87ns	23.50ns	10.80ns	2.860ns
	Layout	14.28ns	62.81ns	32.26ns	8.239ns
Minority3	Schematic	12.82ns	31.66ns	19.83ns	3.300ns
	Layout	33.26ns	83.61ns	56.01ns	9.632ns
Minority3 no-set	Schematic	15.82ns	42.94ns	24.73ns	3.805ns
	Layout	43.88ns	96.58ns	66.26ns	9.432ns
C^2MOS	Schematic	5.265ns	19.19ns	10.15ns	2.225ns
	Layout	10.48ns	60.25ns	24.43ns	6.780ns
PowerPC 603	Schematic	7.834ns	21.31ns	12.17ns	2.170ns
	Layout	24.28ns	69.19ns	40.03ns	7.722ns

Table 25: Monte Carlo Delay Results for Slave latch, rising edge at $-40°C$

		Min. Delay	Max. Delay	Mean Delay	Sigma
Classic NAND	Schematic	13.71ns	40.83ns	21.87ns	3.508ns
	Layout	41.33ns	97.95ns	59.78ns	9.727ns
Minority3	Schematic	13.64ns	39.28ns	21.95ns	4.153ns
	Layout	37.09ns	94.12ns	57.30ns	10.60ns
Minority3 no-set	Schematic	13.39ns	36.97ns	23.37ns	3.647ns
	Layout	40.50ns	105.7ns	64.04ns	11.13ns
C^2MOS	Schematic	5.885ns	21.11ns	12.03ns	2.908ns
	Layout	17.23ns	82.20ns	35.58ns	9.152ns
PowerPC 603	Schematic	5.618ns	19.63ns	10.19ns	2.144ns
	Layout	15.62ns	67.77ns	31.06ns	7.545ns

Table 26: Monte Carlo Delay Results for Slave latch, falling edge at $-40°C$

		Min. Delay	Max. Delay	Mean Delay	Sigma
Classic NAND	Schematic	1.514ns	5.656ns	2.624ns	508.5ps
	Layout	4.123ns	14.43ns	7.628ns	1.471ns
Minority3	Schematic	7.656ns	14.04ns	10.24ns	1.126ns
	Layout	21.23ns	34.81ns	27.27ns	2.798ns
Minority3 no-set	Schematic	5.346ns	9.716ns	7.092ns	830.5ps
	Layout	14.22ns	25.31ns	18.81ns	2.048ns
C^2MOS	Schematic	1.285ns	4.316ns	2.441ns	622.2ps
	Layout	3.899ns	16.26ns	7.144ns	1.943ns
PowerPC 603	Schematic	1.146ns	3.262ns	1.786ns	329.1ps
	Layout	3.198ns	8.860ns	5.172ns	1.006ns

Table 27: Monte Carlo Delay Results for Master latch, rising edge at 27°C

		Min. Delay	Max. Delay	Mean Delay	Sigma
Classic NAND	Schematic	1.506ns	3.600ns	2.245ns	370.7ps
	Layout	4.035ns	10.36ns	6.471ns	1.112ns
Minority3	Schematic	7.208ns	13.24ns	10.51ns	958.2ps
	Layout	20.88ns	37.01ns	27.48ns	2.488ns
Minority3 no-set	Schematic	5.311ns	9.911ns	7.106ns	765.6ps
	Layout	14.09ns	27.67ns	19.41ns	2.235ns
C^2MOS	Schematic	1.727ns	4.449ns	2.762ns	470.2ps
	Layout	4.503ns	12.54ns	7.539ns	1.243ns
PowerPC 603	Schematic	1.134ns	2.504ns	1.744ns	276.5ps
	Layout	3.331ns	8.319ns	5.115ns	807.5ps

Table 28: Monte Carlo Delay Results for Master latch, falling edge at 27°C

		Min. Delay	Max. Delay	Mean Delay	Sigma
Classic NAND	Schematic	2.161ns	6.113ns	3.390ns	655.0ps
	Layout	5.240ns	16.30ns	9.711ns	1.817ns
Minority3	Schematic	4.824ns	9.008ns	6.395ns	737.0ps
	Layout	11.92ns	23.05ns	17.36ns	2.069ns
Minority3 no-set	Schematic	5.556ns	11.48ns	7.730ns	842.9ps
	Layout	14.88ns	26.01ns	19.98ns	2.034ns
C^2MOS	Schematic	2.223ns	5.392ns	3.462ns	536.1ps
	Layout	4.023ns	14.78ns	7.770ns	1.507ns
PowerPC 603	Schematic	2.784ns	5.710ns	3.838ns	490.9ps
	Layout	8.270ns	17.71ns	11.92ns	1.623ns

Table 29: Monte Carlo Delay Results for Slave latch, rising edge at 27°C

		Min. Delay	Max. Delay	Mean Delay	Sigma
Classic NAND	Schematic	4.791ns	10.76ns	6.802ns	655.0ps
	Layout	13.87ns	25.83ns	17.91ns	2.118ns
Minority3	Schematic	4.939ns	10.61ns	7.066ns	946.3ps
	Layout	12.99ns	24.97ns	17.75ns	2.344ns
Minority3 no-set	Schematic	4.684ns	9.963ns	6.929ns	848.3ps
	Layout	13.62ns	27.40ns	19.16ns	2.427ns
C^2MOS	Schematic	2.420ns	6.587ns	4.073ns	738.2ps
	Layout	6.626ns	20.72ns	11.16ns	2.068ns
PowerPC 603	Schematic	2.123ns	5.436ns	3.323ns	518.6ps
	Layout	5.725ns	16.22ns	9.493ns	1.638ns

Table 30: Monte Carlo Delay Results for Slave latch, falling edge at 27°C

		Min. Delay	Max. Delay	Mean Delay	Sigma
Classic NAND	Schematic	905.9ps	2.681ns	1.448ns	230.6ps
	Layout	2.731ns	6.556ns	4.191ns	673.8ps
Minority3	Schematic	4.455ns	7.348ns	5.615ns	512.9ps
	Layout	10.66ns	20.56ns	14.39ns	1.328ns
Minority3 no-set	Schematic	3.071ns	5.029ns	3.892ns	373.0ps
	Layout	7.393ns	13.02ns	10.03ns	910.2ps
C^2MOS	Schematic	845.8ps	2.394ns	1.457ns	310.1ps
	Layout	2.324ns	8.442ns	4.077ns	829.0ps
PowerPC 603	Schematic	598.4ps	1.630ns	935.4ps	167.1ps
	Layout	1.651ns	4.029ns	2.691ns	460.1ps

Table 31: Monte Carlo Delay Results for Master latch, rising edge at 80°C

		Min. Delay	Max. Delay	Mean Delay	Sigma
Classic NAND	Schematic	897.1ps	1.905ns	1.258ns	173.9ps
	Layout	2.229ns	5.190ns	3.482ns	504.3ps
Minority3	Schematic	4.186ns	6.881ns	5.713ns	426.9ps
	Layout	11.67ns	18.07ns	14.51ns	1.147ns
Minority3 no-set	Schematic	3.029ns	5.250ns	3.902ns	365.8ps
	Layout	8.049ns	13.86ns	10.36ns	993.3ps
C^2MOS	Schematic	1.061ns	2.324ns	1.573ns	220.7ps
	Layout	2.936ns	5.856ns	4.180ns	550.0ps
PowerPC 603	Schematic	607.6ps	1.272ns	913.5ps	133.6ps
	Layout	1.797ns	3.650ns	2.655ns	352.4ps

Table 32: Monte Carlo Delay Results for Master latch, falling edge at 80°C

		Min. Delay	Max. Delay	Mean Delay	Sigma
Classic NAND	Schematic	1.289ns	3.100ns	1.887ns	303.9ps
	Layout	3.133ns	8.571ns	5.243ns	861.9ps
Minority3	Schematic	2.825ns	4.714ns	3.563ns	335.5ps
	Layout	6.991ns	12.29ns	9.466ns	964.8ps
Minority3 no-set	Schematic	3.231ns	5.874ns	4.260ns	380.8ps
	Layout	8.050ns	14.37ns	10.70ns	1.046ns
C^2MOS	Schematic	1.434ns	2.914ns	2.028ns	250.5ps
	Layout	3.216ns	7.078ns	4.598ns	657.5ps
PowerPC 603	Schematic	1.626ns	2.929ns	2.128ns	252.2ps
	Layout	4.627ns	8.882ns	6.362ns	718.2ps

Table 33: Monte Carlo Delay Results for Slave latch, rising edge at 80°C

		Min. Delay	Max. Delay	Mean Delay	Sigma
Classic NAND	Schematic	2.785ns	5.526ns	3.758ns	369.0ps
	Layout	7.309ns	13.00ns	9.580ns	957.1ps
Minority3	Schematic	2.956ns	5.455ns	3.979ns	424.6ps
	Layout	7.628ns	13.15ns	9.878ns	1.034ns
Minority3 no-set	Schematic	2.731ns	5.183ns	3.819ns	401.5ps
	Layout	7.726ns	13.32ns	10.35ns	985.3ps
C^2MOS	Schematic	1.569ns	3.603ns	2.399ns	362.4ps
	Layout	4.193ns	8.917ns	6.151ns	898.8ps
PowerPC 603	Schematic	1.281ns	2.883ns	1.884ns	251.4ps
	Layout	3.669ns	7.776ns	5.234ns	709.3ps

Table 34: Monte Carlo Delay Results for Slave latch, falling edge at 80°C

A.2 Average Monte Carlo Delay Data

		μ $-40°C$	σ $-40°C$	μ $27°C$	σ $27°C$	μ $80°C$	σ $80°C$
Classic NAND	Schematic	$12.01ns$	$2.570ns$	$3.765ns$	$582.4ps$	$2.088ns$	$269.4ps$
	Layout	$34.61ns$	$7.434ns$	$10.43ns$	$1.630ns$	$5.624ns$	$749.3ps$
Minority3	Schematic	$27.35ns$	$4.231ns$	$8.553ns$	$941.9ps$	$4.718ns$	$424.9ps$
	Layout	$74.76ns$	$11.34ns$	$22.47ns$	$2.423ns$	$12.06ns$	$1.118ns$
Minority3 no-set	Schematic	$23.32ns$	$3.633ns$	$7.214ns$	$821.8ps$	$3.968ns$	$380.3ps$
	Layout	$65.02ns$	$10.08ns$	$19.34ns$	$2.186ns$	$10.36ns$	$983.7ps$
C^2MOS	Schematic	$9.463ns$	$2.389ns$	$3.185ns$	$591.7ps$	$1.864ns$	$285.9ps$
	Layout	$26.67ns$	$7.383ns$	$8.396ns$	$1.688ns$	$4.752ns$	$731.6ps$
PowerPC 603	Schematic	$8.580ns$	$1.684ns$	$2.278ns$	$403.8ps$	$1.465ns$	$194.3ps$
	Layout	$26.66ns$	$5.878ns$	$7.925ns$	$1.269ns$	$4.236ns$	$560.0ps$

Table 35: Average Monte Carlo Delay Results, μ is mean delay, σ is standard deviation

A.3 Relative Sigma Results

		σ/μ $-40°C$	σ/μ $27°C$	σ/μ $80°C$
Classic NAND	Schematic	21.40%	15.47%	12.90%
	Layout	21.48%	15.62%	13.32%
Minority3	Schematic	15.47%	11.01%	9.00%
	Layout	15.17%	10.78%	9.27%
Minority3 no-set	Schematic	15.58%	11.39%	9.58%
	Layout	15.50%	11.30%	9.50%
C^2MOS	Schematic	25.24%	18.57%	15.33%
	Layout	27.79%	20.11%	15.40%
PowerPC 603	Schematic	19.62%	17.72%	13.26%
	Layout	22.05%	16.00%	13.22%

Table 36: Relative Sigma Results

B Layout

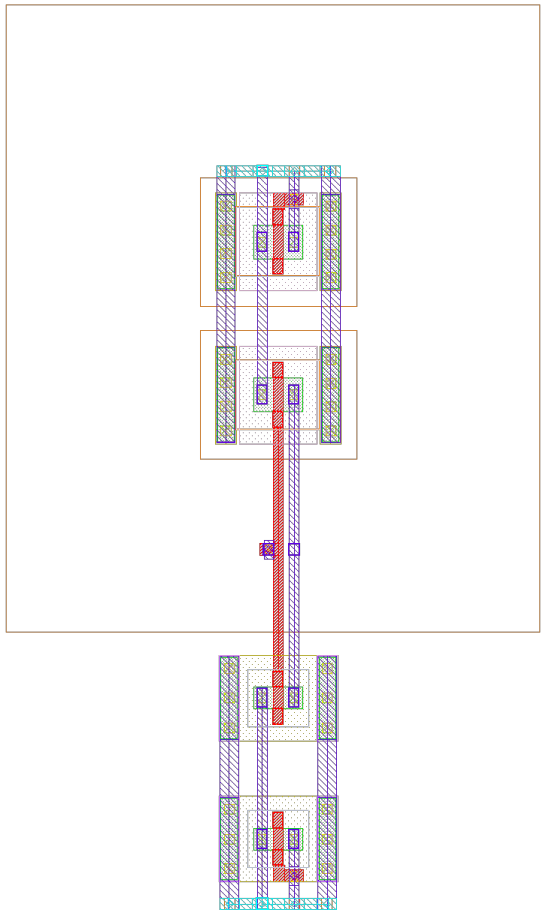


Figure 91: The Inverter Layout

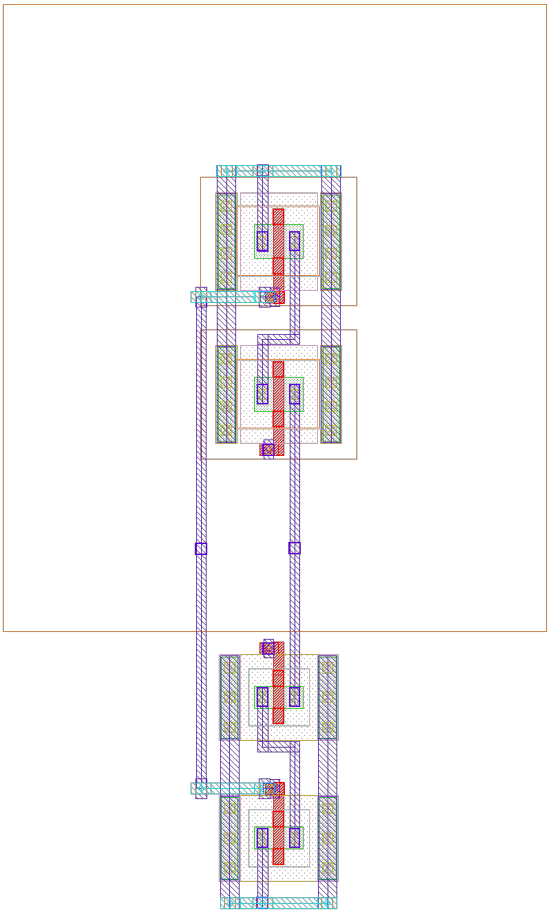


Figure 92: The Clocked Inverter Layout

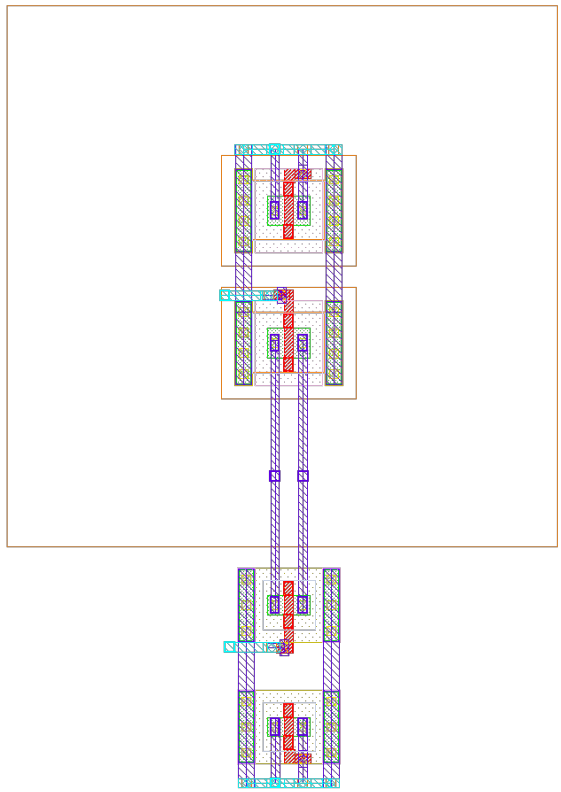


Figure 93: The Transmission Gate Layout

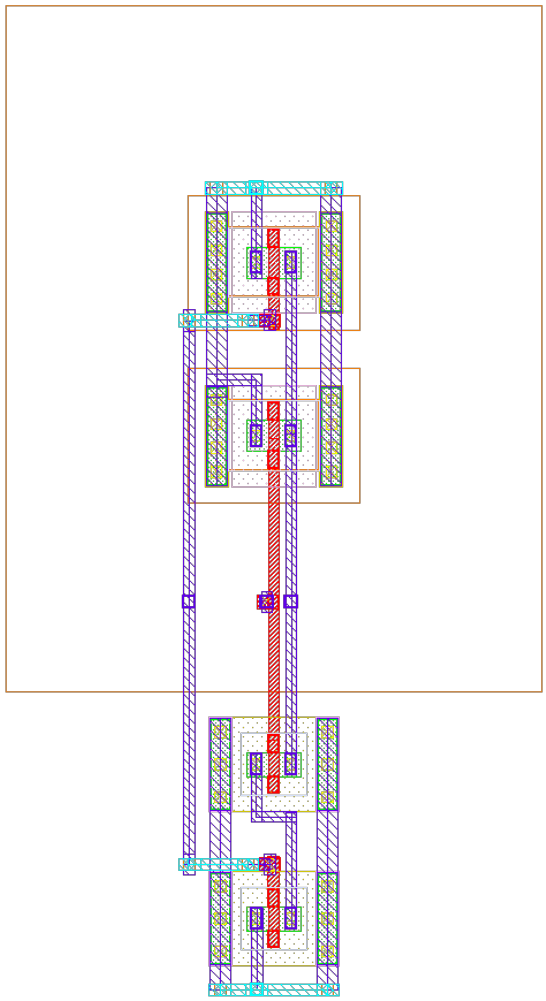


Figure 94: The Simple NAND Layout

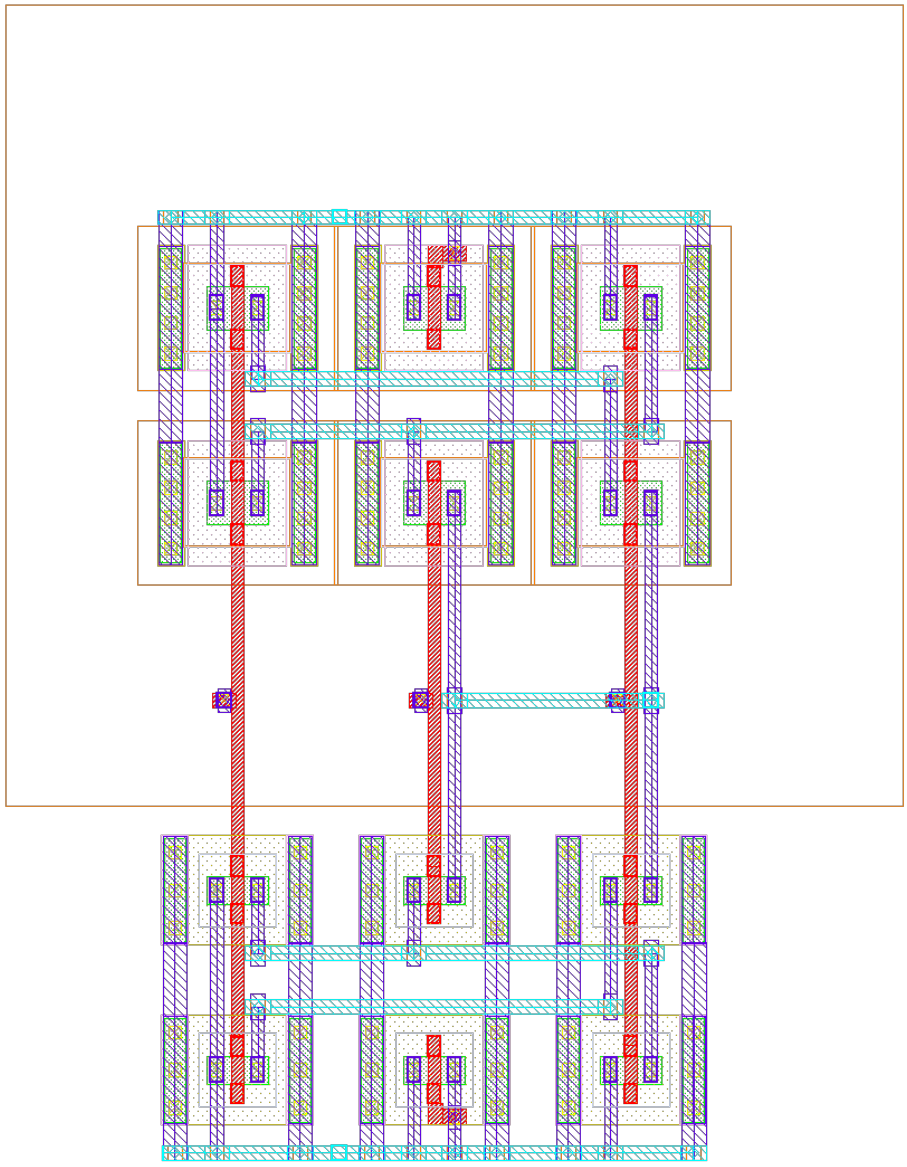


Figure 95: The Minority3-gate Layout

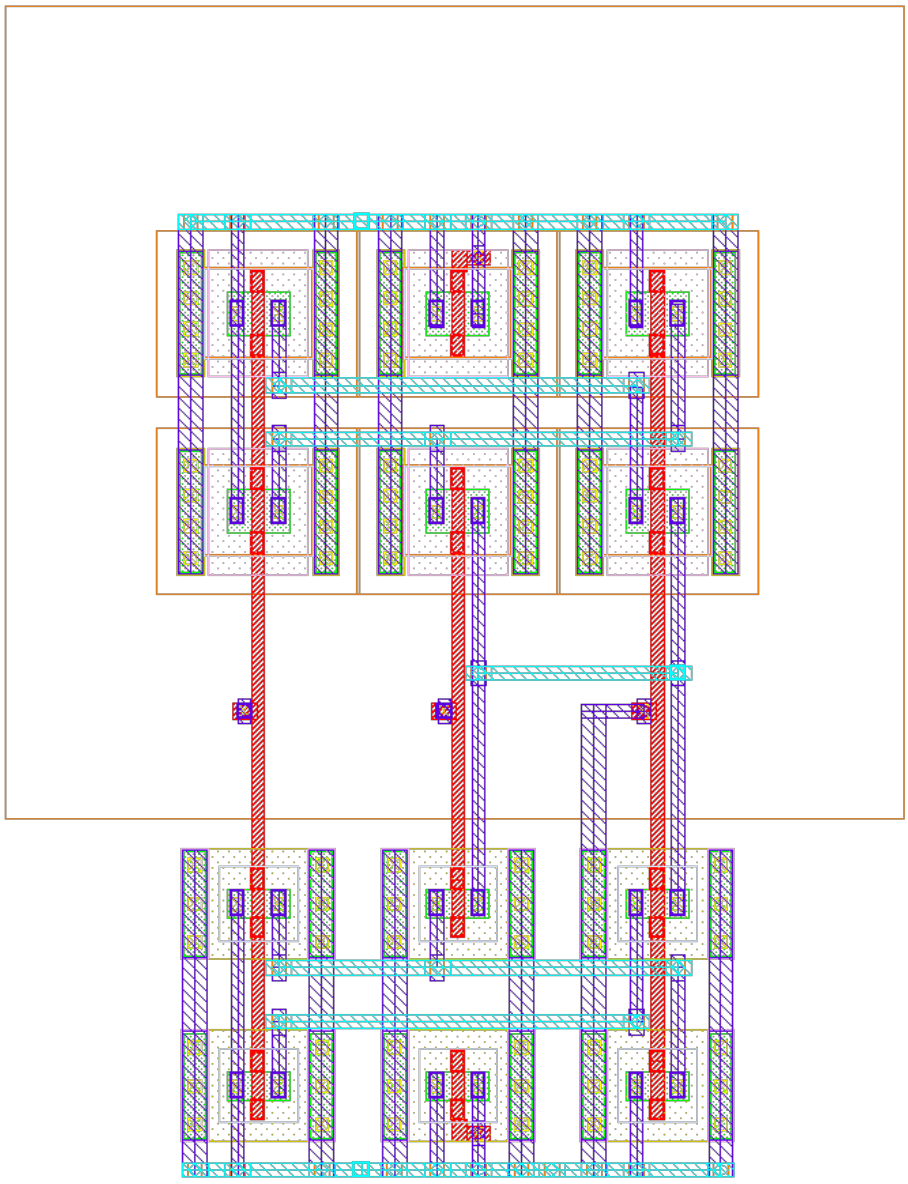


Figure 96: The Minority-based NAND-gate Layout

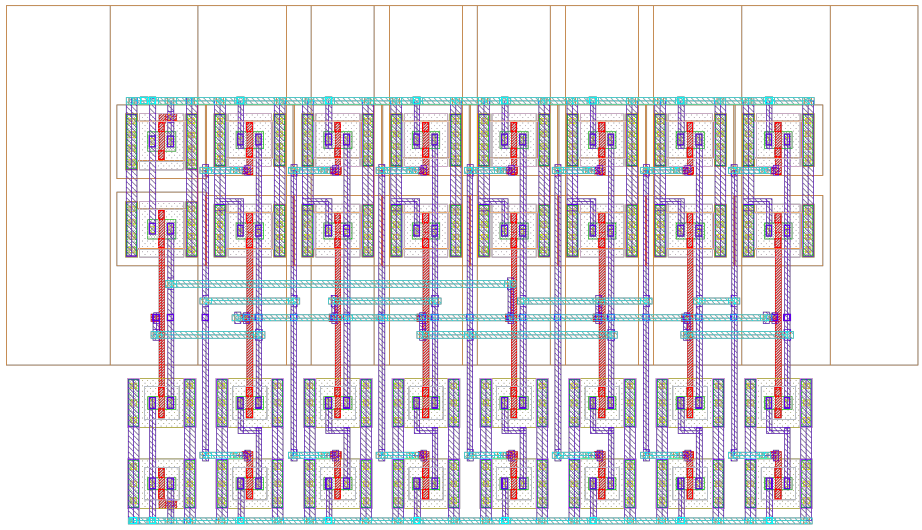


Figure 97: The Classic NAND D flip-flop Layout

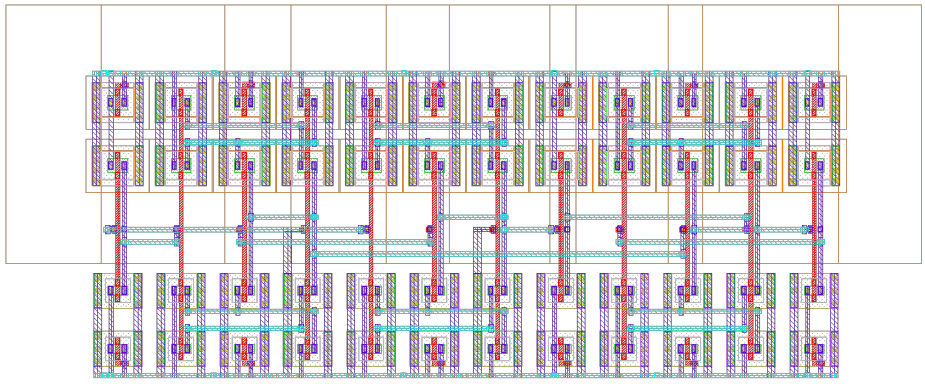


Figure 98: The Minority-based D-latch Layout

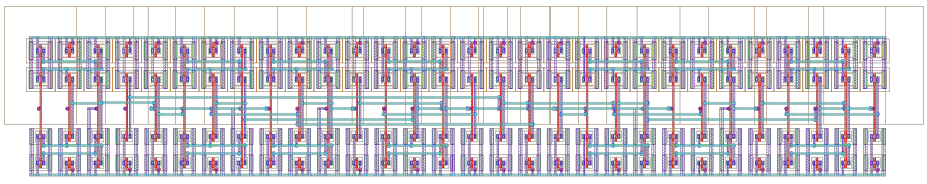


Figure 99: The Minority3-based D flip-flop Layout

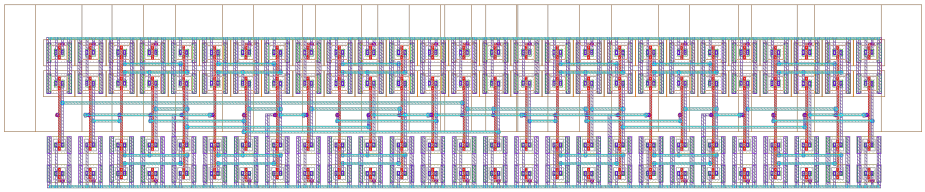


Figure 100: The Minority3-based D flip-flop no-set Layout

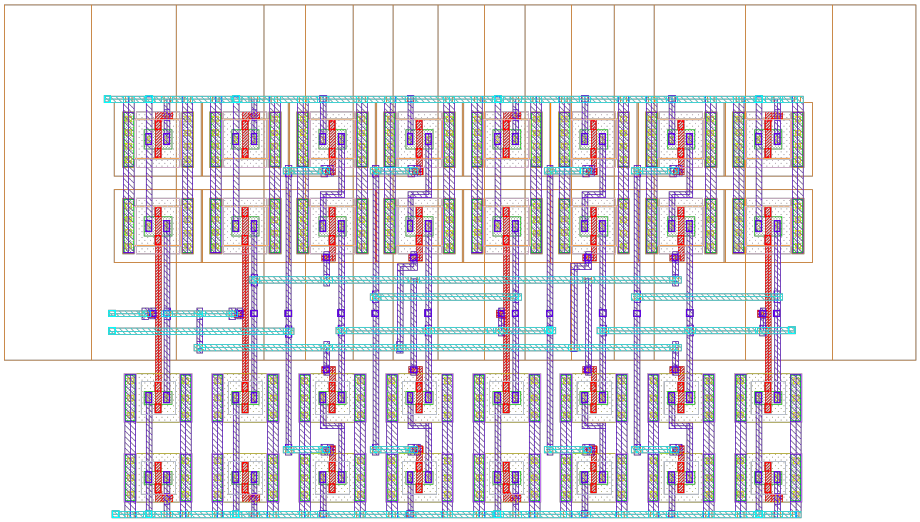


Figure 101: The C^2MOS D flip-flop Layout

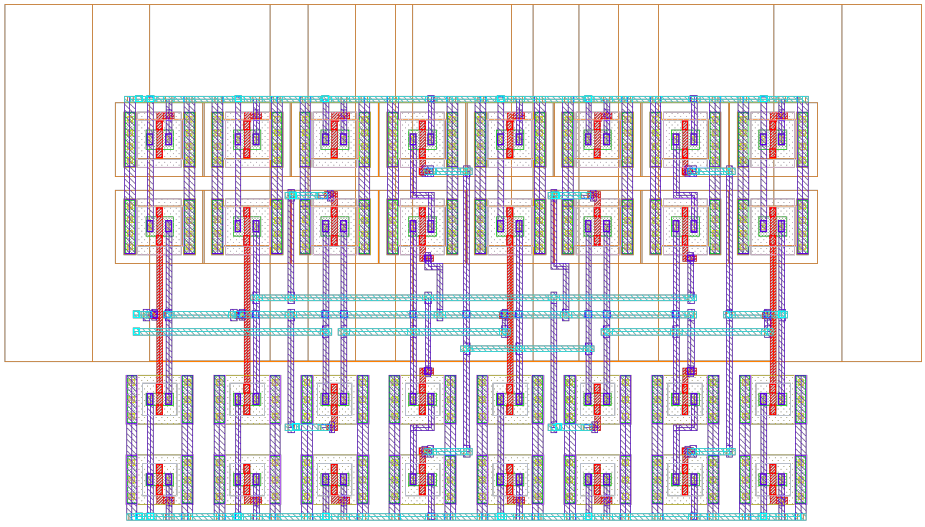


Figure 102: The PowerPC 603 D flip-flop Layout

C Source Code

C.1 Python Scripts

```
# Imported files
import multiprocessing
import shlex, subprocess
from subprocess import call, Popen, PIPE

# Function to run subprocess call in the UNIX shell
def work(cmd):
    return subprocess.call(cmd, shell=True)

# Varibles
output_string = []
Temperature = '80'
Temp_write = '80'

# Output file name
out_name = "pdp_schematic_" + Temp_write

# Run file name, must be run to start simulation
run_file_name = "dyn_pow_schematic_" + Temp_write


# Open the Input data file, must be changed for each test
i = open('python/inputs/dyn_power/max_delay_Schematic_80.txt', 'r')

# Skip first line
temp = i.readline()

# Second line is Classic NAND Data
# Third line is Min3 Data
# Forth line is Min3ns Data
# Fifth line is C2mos Data
# Sixth line is pwrPC Data

# Import and generate data arrays
temp = i.readline()
Class_in = temp.split()
temp = i.readline()
Min3_in = temp.split()
temp = i.readline()
Min3ns_in = temp.split()
temp = i.readline()
C2mos_in = temp.split()
temp = i.readline()
PwrPC_in = temp.split()

i.close()

# Define the D signal array, and D delay array
Class_in_D = []
Class_in_Dd = []
Min3_in_D = []
Min3_in_Dd = []
Min3ns_in_D = []
Min3ns_in_Dd = []
```

```
C2mos_in_D = []
C2mos_in_Dd = []
PwrPC_in_D = []
PwrPC_in_Dd = []
Time_d = []
max_step = []

# Creates the input signals for the tests, for the supply voltage
    range 100mV - 300mV
for num in range(2,13):

  Class_in[num] = float(Class_in[num])
  Class_in[num] *= 1e9
  Class_in[num] *= 2
  Class_in_D.append(str(2*Class_in[num]) + 'n')   # D in
  Class_in_Dd.append(str(-1*Class_in[num]) + 'n')   # D delay in
  Class_in[num] = str(Class_in[num]) + 'n'    # Clk in

  Min3_in[num] = float(Min3_in[num])
  Min3_in[num] *= 1e9
  Min3_in[num] *= 2
  Min3_in_D.append(str(2*Min3_in[num]) + 'n')   # D in
  Min3_in_Dd.append(str(-1*Min3_in[num]) + 'n')   # D delay in
  Min3_in[num] = str(Min3_in[num]) + 'n'      # Clk in

  Min3ns_in[num] = float(Min3ns_in[num])
  Min3ns_in[num] *= 1e9
  Min3ns_in[num] *= 2
  Min3ns_in_D.append(str(2*Min3ns_in[num]) + 'n')   # D in
  Min3ns_in_Dd.append(str(-1*Min3ns_in[num]) + 'n') # D delay in
  Min3ns_in[num] = str(Min3ns_in[num]) + 'n'    # Clk in

  C2mos_in[num] = float(C2mos_in[num])
  C2mos_in[num] *= 1e9
  C2mos_in[num] *= 2
  C2mos_in_D.append(str(2*C2mos_in[num]) + 'n')   # D in
  C2mos_in_Dd.append(str(-1*C2mos_in[num]) + 'n')   # D delay in
  C2mos_in[num] = str(C2mos_in[num]) + 'n'    # Clk in

  PwrPC_in[num] = float(PwrPC_in[num])
  PwrPC_in[num] *= 1e9
  PwrPC_in[num] *= 2
  PwrPC_in_D.append(str(2*PwrPC_in[num]) + 'n')   # D in
  PwrPC_in_Dd.append(str(-1*PwrPC_in[num]) + 'n')   # D delay in
  PwrPC_in[num] = str(PwrPC_in[num]) + 'n'    # Clk in

  # Run the test for 15*Min3 clock period
  Time_d.append(str(Min3_in[num] * 15) + 'n')

# Supply Voltage array
Vdd_in = ['100m', '120m', '140m', '160m', '180m', '200m', '220m', '
    240m', '260m', '280m', '300m']

# Creates the OCEAN script with the data arrays above
for tall in range(len(Class_in_D)):

  Time_in = Time_d[tall]

  # Create the OCEAN script name
```

```
open_string = 'test_'+ run_file_name + "_" + str(tall) + '.ocn'

# Open the Ocean file and starts writing commands
f = open( open_string , 'w+')
f.write(';====================== Set to XL mode
    ========================================\nocnSetXLMode()\
    nocnxlProjectDir( "/work_local/magneva/simulation" )\
    nocnxlTargetCellView( "65nm_mag" "testbenk_power" "adex3_' +
    Temp_write + '_' + str(tall) + '" )\nocnxlResultsLocation( "" )
    \nocnxlSimResultsLocation( "" )\n;====================== Tests
    setup ========================================\n\n')

# Ocean test data
f.write(';---------- Test "65nm_mag:testbenk_power:1"
    ------------\n\nocnxlBeginTest("65nm_mag:testbenk_power:1")\
    nsimulator( \'spectre )\ndesign( "65nm_mag" "testbenk_power" "
    config")\nanalysis(\'tran ?stop "VAR(\\"Time\\")"  ?method "
    euler" )\ndesVar(    "Time" ' + Time_in + '  )\ndesVar( "
    C_in_c2mos" ' + C2mos_in[tall+2] + '  )\ndesVar(  "c_in_classic
    " ' + Class_in[tall+2] + ' )\ndesVar(    "C_in_min3" ' +
    Min3_in[tall+2] + ' )\n')
f.write('desVar(    "C_in_min3ns" ' + Min3ns_in[tall+2] + ' )\
    ndesVar(    "C_in_pwrPC" ' + PwrPC_in[tall+2] + ' )\ndesVar(
      "D_in_c2mos" '  + C2mos_in_D[tall] + '  )\ndesVar(    "
    D_in_min3" ' + Min3_in_D[tall] + ' )\ndesVar(    "
    D_in_min3_delay" ' + Min3_in_Dd[tall] + '  )\ndesVar(    "
    D_in_min3ns" ' + Min3ns_in_D[tall] + ' )\ndesVar(    "
    D_in_min3ns_delay" ' + Min3ns_in_Dd[tall] + '  )\ndesVar(    "
    D_in_pwrPC" ' + PwrPC_in_D[tall] + ' )\n')
f.write('desVar(    "D_in_pwrPC_delay" ' + PwrPC_in_Dd[tall] + '  )
    \ndesVar(    "vdd" ' + Vdd_in[tall] + '  )\ndesVar(    "
    D_in_classic_delay" ' + Class_in_Dd[tall] + '  )\ndesVar(    "
    D_in_classic" '  + Class_in_D[tall] + ' )\ndesVar(   "
    D_in_c2mos_delay" ' + C2mos_in_Dd[tall] + '  )\n')

f.write('envOption(\n \'analysisOrder  list("tran")\n)\noption( \'
    temp  "' + Temperature + '"\n)\noption( ?categ \'turboOpts\n
    \'uniMode  "APS"\n)\n')

f.write('save( \'i "/V17/PLUS" "/V19/PLUS" "/V18/PLUS" "/V15/PLUS"
    "/V16/PLUS" )\n  converge( \'ic "/D_min3" "0" )\n  converge( \'
    ic "/C_min3" "0" )\n  converge( \'ic "/D_c2mos" "0" )\n
    converge( \'ic "/C_c2mos" "0" )\n converge( \'ic "/D_min3_ns"
    "0" )\n converge( \'ic "/C_min3_ns" "0" )\n converge( \'ic "/
    D_pwrPC" "0" )\n converge( \'ic "/C_pwrPC" "0" )\n converge( \'
    ic "/D_classic" "0" )\n converge( \'ic "/C_classic" "0" )\n
    converge( \'ic "/Q_c2mos" "0" )\n converge( \'ic "/Q_pwrPC" "0"
     )\n converge( \'ic "/Q_classic" "0" )\n converge( \'ic "/
    Q_min3" "0" )\n  converge( \'ic "/Q_min3_ns" "0" )\n temp( ' +
    Temperature + ' )\n')
f.write(' ocnxlOutputSignal( "/D_all")\n  ocnxlOutputSignal( "/
    clk_all")\n  ;ocnxlOutputSignal( "/Q_classic" ?plot t)\n ;
    ocnxlOutputSignal( "/Q_pwrPC" ?plot t)\n ;ocnxlOutputSignal( "/
    Q_c2mos" ?plot t)\n ;ocnxlOutputSignal( "/Q_min3" ?plot t)\n  ;
    ocnxlOutputSignal( "/Q_min3_ns" ?plot t)\n ocnxlOutputTerminal(
     "/V17/PLUS" ?save t)\n ocnxlOutputTerminal( "/V19/PLUS" ?save
    t)\n ocnxlOutputTerminal( "/V18/PLUS" ?save t)\n
    ocnxlOutputTerminal( "/V15/PLUS" ?save t)\n ocnxlOutputTerminal
    ( "/V16/PLUS" ?save t)\n ocnxlOutputSignal( "/X_classic")\n
```

```
    ocnxlOutputSignal( "/X_pwrPC")\n  ocnxlOutputSignal( "/X_c2mos
    ")\n  ocnxlOutputSignal( "/X_min3")\n ocnxlOutputSignal( "/
    X_min3_ns")\n  ocnxlOutputSignal( "/D_pwrPC")\n
    ocnxlOutputSignal( "/C_pwrPC")\n  ocnxlOutputSignal( "/D_c2mos
    ")\n  ocnxlOutputSignal( "/C_c2mos")\n  ocnxlOutputSignal( "/
    D_min3")\n ocnxlOutputSignal( "/C_min3")\n ocnxlOutputSignal(
    "/D_min3_ns")\n  ocnxlOutputSignal( "/C_min3_ns")\n')

f.write('ocnxlSweepVar("Time" "' + Time_in + '")\nocnxlEndTest()\n'
    )
f.write(';====================== Model Group setup
    ========================================\n ocnxlModelGroup( "
    default"\n   \'(\n     ( "/home/magneva/master/65nm_mag/
    testbenk_power/adex3/modelgroups/spectre/default.scs" ?section
    "default")\n    )\n  )\n\n ;====================== Corners
    setup ========================================\n  ocnxlCorner
    ( "default"\n   \'(\n      ("modelGroup" "default")\n   )\n)\n'
    )

f.write(';====================== Job setup
    ==========================================\n
    ocnxlJobSetup( \'(\n  "blockemail" "1"\n  "configuretimeout"
    "300"\n  "distributionmethod" "Local"\n  "lingertimeout" "300"\
    n   "maxjobs" "8"\n "name" "ADE XL Default"\n "preemptivestart"
     "1"\n   "reconfigureimmediately" "1"\n  "runtimeout" "-1"\n "
    showerrorwhenretrying" "1"\n "showoutputlogerror" "0"\n  "
    startmaxjobsimmed" "1"\n "starttimeout" "300"\n  ) )\n')
f.write(';====================== Run Mode Options
    ======================================\n
    ocnxlMonteCarloOptions( ?dutSummary "65nm_mag:testbenk_power
    :1%/I69, /I68, /I66, /I65, /I63, /I64, /I57, /I55, /I49, /I56,
    /I53, /I54, /I43, /I44, /I40, /I41, /I42, /I48, /I37, /I35, /
    I29, /I36, /I33, /I34, /I28, /I27, /I23, /I22, /I24, /I21%
    Schematic%Schematic#" ?ignoreFlag "1" ?mcMethod "all" ?
    mcNumPoints "100" ?mcNumBins "" ?mcStopEarly "0" ?mcYieldTarget
     "99.73" ?mcYieldAlphaLimit "0.05" ?samplingMode "random" ?
    saveProcess "1" ?saveMismatch "1" ?useReference "0" ?donominal
    "1" ?saveAllPlots "1" ?monteCarloSeed "1" ?mcStartingRunNumber
    "" )\n')
f.write(';====================== Run command
    ==========================================\n ;out = outfile("
    myResults/testout.ocn" "a")\n ocnxlRun( ?mode \'
    sweepsAndCorners ?nominalCornerEnabled nil ?allCornersEnabled t
     ?allSweepsEnabled t)\n  ocnxlOutputSummary(?exprSummary t) ;?
    specSummary t)\n\n')

f.write(' of1 = outfile("~/master/out/' + out_name + "_" + str(tall
    ) + '.out" "w+")\n\n openResults()\n selectResult(\'tran)\n\n
    fprintf(of1 "FirstLine")\n\n  close(of1)\n\n  of1 = outfile("~/
    master/out/' + out_name + "_" + str(tall) + '.out" "a")\n\n')
f.write(' openResults()\n selectResult(\'tran)\n  declare( out_arr
    [5] )\n out_arr[0] = (average(clip(IT("/V16/PLUS") (VAR("Time")
     * 0.1) (VAR("Time") * 0.999))) * average(clip(VT("/I_Class") (
    VAR("Time") * 0.1) (VAR("Time") * 0.999))))\n out_arr[1] = (
    average(clip(IT("/V18/PLUS") (VAR("Time") * 0.1) (VAR("Time") *
     0.999))) * average(clip(VT("/I_min3") (VAR("Time") * 0.1) (VAR
    ("Time") * 0.999))))\n ')
f.write(' out_arr[2] = (average(clip(IT("/V19/PLUS") (VAR("Time") *
     0.1) (VAR("Time") * 0.999))) * average(clip(VT("/I_min3ns") (
```

```
        VAR("Time") * 0.1) (VAR("Time") * 0.999))))\n  out_arr[3] = (
        average(clip(IT("/V17/PLUS") (VAR("Time") * 0.1) (VAR("Time") *
         0.999))) * average(clip(VT("/I_c2mos") (VAR("Time") * 0.1) (
        VAR("Time") * 0.999))))\n out_arr[4] = (average(clip(IT("/V15/
        PLUS") (VAR("Time") * 0.1) (VAR("Time") * 0.999))) * average(
        clip(VT("/I_pwrPC") (VAR("Time") * 0.1) (VAR("Time") * 0.999)))
        ) \n\n')

    f.write(' ocnPrint((out_arr[0]) ?output of1 ?format "engineering")\
        n  ocnPrint((out_arr[1]) ?output of1 ?format "engineering")\n
        ocnPrint((out_arr[2]) ?output of1 ?format "engineering")\n
        ocnPrint((out_arr[3]) ?output of1 ?format "engineering")\n
        ocnPrint((out_arr[4]) ?output of1 ?format "engineering")\n\n
        close(of1)\n  ocnPrint("Test number' + str(tall) + ' is
        Finished")\n')


    f.write(' ocnPrint("Simulation Finished!")\n;======================
         End XL Mode command ==================================\
        nocnxlEndXLMode()\n\nexit() ')
    f.close()

    # Create the OCEAN script to be run
    output_string.append("ocean -replay " + open_string + " &")

# Runs all tests in parallel for the given temperature
pool = multiprocessing.Pool(None)
r = pool.map_async(work, output_string)
r.wait() # Wait on the results
```

Listing 1: The Python Script for importing delay data and generating the Ocean scripts and initite the simulations for PDP

```
# Imported files
import multiprocessing
import shlex, subprocess
from subprocess import call, Popen, PIPE

# Function to run subprocess call in the UNIX shell
def work(cmd):
    return subprocess.call(cmd, shell=True)

# Variables
output_string = []
Temperature = '27.0'
Temp_write = '27'

# Output file name
out_name = "dyn_pow_schematic_" + Temp_write

# Run file name, must be run to start simulation
run_file_name = "dyn_pow_schematic_" + Temp_write

# Open the Input data file, must be changed for each test
i = open('python/inputs/dyn_power/max_delay_Schematic_27.txt', 'r')

# Skip first line
temp = i.readline()
```

```
# Second line is Classic NAND Data
# Third line is Min3
# Forth line is Min3ns
# Fifth line is C2mos
# Sixth line is pwrPC

# Import and generate data arrays
temp = i.readline()
Class_in = temp.split()
temp = i.readline()
Min3_in = temp.split()
temp = i.readline()
Min3ns_in = temp.split()
temp = i.readline()
C2mos_in = temp.split()
temp = i.readline()
PwrPC_in = temp.split()

i.close()

# Define the D signal array, and D delay array
Min3_in_D = []
Min3_in_Dd = []

Time_d = []

# Creates the input signals for the tests, for the supply voltage
    range 100mV - 300mV
# Here, all is set to the Min3 frequency.
for num in range(2,13):

  Min3_in[num] = float(Min3_in[num])
  Min3_in[num] *= 2e9
  Min3_in_D.append(str(2*Min3_in[num]) + 'n')
  Min3_in_Dd.append(str(-1*Min3_in[num]) + 'n')
  Min3_in[num] = str(Min3_in[num]) + 'n'

  # Run the test for 15*Min3 clock period
  Time_d.append(str(Min3_in[num] * 15) + 'n')

# Supply Voltage array
Vdd_in = ['100m', '120m', '140m', '160m', '180m', '200m', '220m', '
    240m', '260m', '280m', '300m']

# Creates the OCEAN script with the data arrays above
for tall in range(len(Class_in_D)):

  Time_in = Time_d[tall]

  # Create the OCEAN script name
  open_string = 'test_'+ run_file_name + "_" + str(tall) + '.ocn'

  # Open the Ocean file and starts writing commands
  f = open( open_string , 'w+')
  f.write(';====================== Set to XL mode
      ========================================\nocnSetXLMode()\
      nocnxlProjectDir( "/work_local/magneva/simulation" )\
      nocnxlTargetCellView( "65nm_mag" "testbenk_power" "adex3_' +
```

```
        Temp_write + '_' + str(tall) + '" )\nocnxlResultsLocation( "" )
        \nocnxlSimResultsLocation( "" )\n;====================== Tests
        setup ==========================================\n\n')

    # Ocean test data
    f.write(';---------- Test "65nm_mag:testbenk_power:1"
        ------------\n\nocnxlBeginTest("65nm_mag:testbenk_power:1")\
        nsimulator( \'spectre )\ndesign( "65nm_mag" "testbenk_power" "
        config")\nanalysis(\'tran ?stop "VAR(\\"Time\\")"  ?method "
        euler"  )\ndesVar(   "Time" ' + Time_in + '  )\ndesVar( "
        C_in_c2mos" ' + Min3_in[tall+2] + ' )\ndesVar(  "c_in_classic"
        ' + Min3_in[tall+2] + '  )\ndesVar(    "C_in_min3" ' + Min3_in[
        tall+2] + ' )\n')
    f.write('desVar(    "C_in_min3ns" ' + Min3_in[tall+2] + ' )\ndesVar
        (    "C_in_pwrPC" ' + Min3_in[tall+2] + '  )\ndesVar(    "
        D_in_c2mos" '  + Min3_in_D[tall] + ' )\ndesVar(    "D_in_min3"
        ' + Min3_in_D[tall] + ' )\ndesVar(    "D_in_min3_delay" ' +
        Min3_in_Dd[tall] + '  )\ndesVar(    "D_in_min3ns" ' + Min3_in_D
        [tall] + ' )\ndesVar(    "D_in_min3ns_delay" ' + Min3_in_Dd[
        tall] + '  )\ndesVar(    "D_in_pwrPC" ' + Min3_in_D[tall] + '
        )\n')
    f.write('desVar(    "D_in_pwrPC_delay" ' + Min3_in_Dd[tall] + ' )\
        ndesVar(    "vdd" ' + Vdd_in[tall] + '  )\ndesVar(    "
        D_in_classic_delay" ' + Min3_in_Dd[tall] + ' )\ndesVar(    "
        D_in_classic" '  + Min3_in_D[tall] + ' )\ndesVar(    "
        D_in_c2mos_delay" ' + Min3_in_Dd[tall] + ' )\n')

    f.write('envOption(\n \'analysisOrder  list("tran")\n)\noption( \'
        temp  "' + Temperature + '"\n)\noption( ?categ \'turboOpts\n
        \'uniMode  "APS"\n)\n')

    f.write('save( \'i "/V17/PLUS" "/V19/PLUS" "/V18/PLUS" "/V15/PLUS"
        "/V16/PLUS" )\n  converge( \'ic "/D_min3" "0" )\n  converge( \'
        ic "/C_min3" "0" )\n  converge( \'ic "/D_c2mos" "0" )\n
        converge( \'ic "/C_c2mos" "0" )\n converge( \'ic "/D_min3_ns"
        "0" )\n converge( \'ic "/C_min3_ns" "0" )\n converge( \'ic "/
        D_pwrPC" "0" )\n converge( \'ic "/C_pwrPC" "0" )\n converge( \'
        ic "/D_classic" "0" )\n converge( \'ic "/C_classic" "0" )\n
        converge( \'ic "/Q_c2mos" "0" )\n converge( \'ic "/Q_pwrPC" "0"
         )\n converge( \'ic "/Q_classic" "0" )\n converge( \'ic "/
        Q_min3" "0" )\n  converge( \'ic "/Q_min3_ns" "0" )\n temp( ' +
        Temperature + ' )\n')
    f.write(' ocnxlOutputSignal( "/D_all")\n  ocnxlOutputSignal( "/
        clk_all")\n  ;ocnxlOutputSignal( "/Q_classic" ?plot t)\n ;
        ocnxlOutputSignal( "/Q_pwrPC" ?plot t)\n ;ocnxlOutputSignal( "/
        Q_c2mos" ?plot t)\n ;ocnxlOutputSignal( "/Q_min3" ?plot t)\n  ;
        ocnxlOutputSignal( "/Q_min3_ns" ?plot t)\n ocnxlOutputTerminal(
         "/V17/PLUS" ?save t)\n ocnxlOutputTerminal( "/V19/PLUS" ?save
        t)\n ocnxlOutputTerminal( "/V18/PLUS" ?save t)\n
        ocnxlOutputTerminal( "/V15/PLUS" ?save t)\n ocnxlOutputTerminal
        ( "/V16/PLUS" ?save t)\n ocnxlOutputSignal( "/X_classic")\n
        ocnxlOutputSignal( "/X_pwrPC")\n  ocnxlOutputSignal( "/X_c2mos
        ")\n  ocnxlOutputSignal( "/X_min3")\n ocnxlOutputSignal( "/
        X_min3_ns")\n  ocnxlOutputSignal( "/D_pwrPC")\n
        ocnxlOutputSignal( "/C_pwrPC")\n  ocnxlOutputSignal( "/D_c2mos
        ")\n  ocnxlOutputSignal( "/C_c2mos")\n  ocnxlOutputSignal( "/
        D_min3")\n ocnxlOutputSignal( "/C_min3")\n ocnxlOutputSignal(
        "/D_min3_ns")\n  ocnxlOutputSignal( "/C_min3_ns")\n')
```

```
f.write('ocnxlSweepVar("Time" "' + Time_in + '")\nocnxlEndTest()\n'
    )
f.write(';====================== Model Group setup
    ============================================\n ocnxlModelGroup( "
    default"\n   \'(\n     ( "/home/magneva/master/65nm_mag/
    testbenk_power/adex3/modelgroups/spectre/default.scs" ?section
    "default")\n    )\n  )\n\n ;====================== Corners
    setup ========================================\n  ocnxlCorner
    ( "default"\n   \'(\n      ("modelGroup" "default")\n   )\n)\n'
    )

f.write(';====================== Job setup
    ============================================\n
    ocnxlJobSetup( \'(\n  "blockemail" "1"\n  "configuretimeout"
    "300"\n  "distributionmethod" "Local"\n  "lingertimeout" "300"\
    n   "maxjobs" "8"\n "name" "ADE XL Default"\n "preemptivestart"
     "1"\n   "reconfigureimmediately" "1"\n  "runtimeout" "-1"\n "
    showerrorwhenretrying" "1"\n "showoutputlogerror" "0"\n  "
    startmaxjobsimmed" "1"\n "starttimeout" "300"\n  ) )\n')
f.write(';====================== Run Mode Options
    ====================================\n
    ocnxlMonteCarloOptions( ?dutSummary "65nm_mag:testbenk_power
    :1%/I69, /I68, /I66, /I65, /I63, /I64, /I57, /I55, /I49, /I56,
    /I53, /I54, /I43, /I44, /I40, /I41, /I42, /I48, /I37, /I35, /
    I29, /I36, /I33, /I34, /I28, /I27, /I23, /I22, /I24, /I21%
    Schematic%Schematic#" ?ignoreFlag "1" ?mcMethod "all" ?
    mcNumPoints "100" ?mcNumBins "" ?mcStopEarly "0" ?mcYieldTarget
     "99.73" ?mcYieldAlphaLimit "0.05" ?samplingMode "random" ?
    saveProcess "1" ?saveMismatch "1" ?useReference "0" ?donominal
    "1" ?saveAllPlots "1" ?monteCarloSeed "1" ?mcStartingRunNumber
    "" )\n')
f.write(';====================== Run command
    ============================================\n ;out = outfile("
    myResults/testout.ocn" "a")\n ocnxlRun( ?mode \'
    sweepsAndCorners ?nominalCornerEnabled nil ?allCornersEnabled t
     ?allSweepsEnabled t)\n  ocnxlOutputSummary(?exprSummary t) ;?
    specSummary t)\n\n')

f.write(' of1 = outfile("~/master/out/' + out_name + "_" + str(tall
    ) + '.out" "w+")\n\n openResults()\n selectResult(\'tran)\n\n
    fprintf(of1 "FirstLine")\n\n  close(of1)\n\n  of1 = outfile("~/
    master/out/' + out_name + "_" + str(tall) + '.out" "a")\n\n')
f.write(' openResults()\n selectResult(\'tran)\n  declare( out_arr
    [5] )\n out_arr[0] = (average(clip(IT("/V16/PLUS") (VAR("Time")
     * 0.1) (VAR("Time") * 0.999))) * average(clip(VT("/I_Class") (
    VAR("Time") * 0.1) (VAR("Time") * 0.999))))\n out_arr[1] = (
    average(clip(IT("/V18/PLUS") (VAR("Time") * 0.1) (VAR("Time") *
     0.999))) * average(clip(VT("/I_min3") (VAR("Time") * 0.1) (VAR
    ("Time") * 0.999))))\n ')
f.write(' out_arr[2] = (average(clip(IT("/V19/PLUS") (VAR("Time") *
     0.1) (VAR("Time") * 0.999))) * average(clip(VT("/I_min3ns") (
    VAR("Time") * 0.1) (VAR("Time") * 0.999))))\n  out_arr[3] = (
    average(clip(IT("/V17/PLUS") (VAR("Time") * 0.1) (VAR("Time") *
     0.999))) * average(clip(VT("/I_c2mos") (VAR("Time") * 0.1) (
    VAR("Time") * 0.999))))\n out_arr[4] = (average(clip(IT("/V15/
    PLUS") (VAR("Time") * 0.1) (VAR("Time") * 0.999))) * average(
    clip(VT("/I_pwrPC") (VAR("Time") * 0.1) (VAR("Time") * 0.999)))
    ) \n\n')
```

```
    f.write(' ocnPrint((out_arr[0]) ?output of1 ?format "engineering")\
        n  ocnPrint((out_arr[1]) ?output of1 ?format "engineering")\n
        ocnPrint((out_arr[2]) ?output of1 ?format "engineering")\n
        ocnPrint((out_arr[3]) ?output of1 ?format "engineering")\n
        ocnPrint((out_arr[4]) ?output of1 ?format "engineering")\n\n
        close(of1)\n  ocnPrint("Test number' + str(tall) + ' is
        Finished")\n')


    f.write(' ocnPrint("Simulation Finished!")\n;====================
         End XL Mode command ================================\
        nocnxlEndXLMode()\n\nexit() ')
    f.close()

    # Create the OCEAN script to be run
    output_string.append("ocean -replay " + open_string + " &")

  # Runs all tests in parallel for the given temperature
  pool = multiprocessing.Pool(None)
  r = pool.map_async(work, output_string)
  r.wait() # Wait on the results
```

Listing 2: The Python Script for importing delay data and generating the Ocean scripts and initite the simulations for Total Power Consumption

```
  # Function for data exported from the OCEAN script
  # The first 4 lines are not used
  def get_res():
    # Read 4 headliners for each data
    for num in range(0,4):
      temp = i.readline()

    # Split the line
    temp = temp.split()

    return temp[1]

  # String Varibles for output and input files
  Test = 'dyn_pow'
  Type = 'layout'
  Temp = '40'

  # The output file
  o = open('out/file/'+ Test + '_' + Type + '_' + Temp + '.txt', 'w+')

  # Arrays to store input data
  Data_in = ['','','','','']

  # Sorts the Total Power Consumption data for the supply voltage range
       100mV -300mV
  for num in range(0,11):

    # Open the Input data file
    i = open('out/' + Test + '_' + Type + '_' + Temp + '_' + str(num) +
         '.out' , 'r')

    # Stores data in arrays
    Data_in[0] += (get_res() + ' ') # Classic
```

```
    Data_in[1] += (get_res() + ' ') # Min3
    Data_in[2] += (get_res() + ' ') # Min3ns
    Data_in[3] += (get_res() + ' ') # C2mos
    Data_in[4] += (get_res() + ' ') # PwrPc

# Print Total Power Consumption data to output file, ready for Matlab
    .
for dff in range(0,5):

    o.write(Data_in[dff] + '\n')

o.close()
i.close()
```

Listing 3: The Python Script for sorting and exporting Total Power Consumption Data

```
# Function for data exported from the OCEAN script
# The first 4 lines are not used
def get_res():
    # Read 4 headliners for each data
    for num in range(0,4):
        temp = i.readline()

    # Split the line
    temp = temp.split()

    return float(temp[1])

# String Varibles for output and input files
Test = 'pdp'
Type = 'Schematic'
Type2 ='schematic'
Temp = '80'

# Open output and input file
o = open('out/file/'+ Test + '_' + Type + '_' + Temp + '.txt', 'w+')
d = open('python/inputs/dyn_power/max_delay_' + Type + '_' + Temp + '
    .txt', 'r')

# Arrays to store input and output data
Data_in = ['','','','','']
Delay_in = ['','','','','']

# Skip first line
temp = d.readline()

# Read delay data
temp = d.readline()
Delay_in[0] = temp.split()
temp = d.readline()
Delay_in[1] = temp.split()
temp = d.readline()
Delay_in[2] = temp.split()
temp = d.readline()
Delay_in[3] = temp.split()
temp = d.readline()
Delay_in[4] = temp.split()
```

```
# Multiply delay data with power consumption for the supply voltage
    range 100mV - 300mV
for num in range(0,11):

  # Open the Power Consumption data file
  i = open('out/' + Test + '_' + Type2 + '_' + Temp + '_' + str(num)
      + '.out' , 'r')

  # Multiplies Power with Delay
  Data_in[0] += (str(get_res() * float(Delay_in[0][num+2]) * 2 ) + '
      ' ) # Classic
  Data_in[1] += (str(get_res() * float(Delay_in[1][num+2]) * 2 ) + '
      ' ) # Min3
  Data_in[2] += (str(get_res() * float(Delay_in[2][num+2]) * 2 ) + '
      ' ) # Min3ns
  Data_in[3] += (str(get_res() * float(Delay_in[3][num+2]) * 2 ) + '
      ' ) # C2mos
  Data_in[4] += (str(get_res() * float(Delay_in[4][num+2]) * 2 ) + '
      ' ) # PwrPc

# Print PDP data to output file, ready for Matlab.
for dff in range(0,5):

  o.write(Data_in[dff] + '\n')

o.close()
i.close()
d.close()
```

Listing 4: The Python Script for calculating and exporting PDP data

C.2 OCEAN Scripts

```
;====================== Set to XL mode
    ========================================
ocnSetXLMode()
ocnxlProjectDir( "/work_local/magneva/simulation" )
ocnxlTargetCellView( "65nm_mag" "testbenk_power" "adex3_80_1" )
ocnxlResultsLocation( "" )
ocnxlSimResultsLocation( "" )
;====================== Tests setup
    ==========================================

;---------- Test "65nm_mag:testbenk_power:1" -------------

ocnxlBeginTest("65nm_mag:testbenk_power:1")
simulator( 'spectre )
design( "65nm_mag" "testbenk_power" "config")
analysis('tran ?stop "VAR(\"Time\")"  ?method "euler" )
desVar(   "Time" 1202.7n  )
desVar( "C_in_c2mos" 46.88n )
desVar( "c_in_classic" 66.56n )
desVar(   "C_in_min3" 80.18n  )
desVar(   "C_in_min3ns" 61.46n  )
desVar(   "C_in_pwrPC" 31.78n )
desVar(   "D_in_c2mos" 93.76n )
```

```
desVar(   "D_in_min3" 160.36n )
desVar(   "D_in_min3_delay" -80.18n )
desVar(   "D_in_min3ns" 122.92n )
desVar(   "D_in_min3ns_delay" -61.46n )
desVar(   "D_in_pwrPC" 63.56n )
desVar(   "D_in_pwrPC_delay" -31.78n  )
desVar(   "vdd" 120m  )
desVar(   "D_in_classic_delay" -66.56n  )
desVar(   "D_in_classic" 133.12n )
desVar(   "D_in_c2mos_delay" -46.88n  )
envOption(
  'analysisOrder  list("tran")
)
option( 'temp  "80"
)
option( ?categ 'turboOpts
  'uniMode  "APS"
)
save( 'i "/V17/PLUS" "/V19/PLUS" "/V18/PLUS" "/V15/PLUS" "/V16/PLUS"
    )
  converge( 'ic "/D_min3" "0" )
  converge( 'ic "/C_min3" "0" )
  converge( 'ic "/D_c2mos" "0" )
  converge( 'ic "/C_c2mos" "0" )
  converge( 'ic "/D_min3_ns" "0" )
  converge( 'ic "/C_min3_ns" "0" )
  converge( 'ic "/D_pwrPC" "0" )
  converge( 'ic "/C_pwrPC" "0" )
  converge( 'ic "/D_classic" "0" )
  converge( 'ic "/C_classic" "0" )
  converge( 'ic "/Q_c2mos" "0" )
  converge( 'ic "/Q_pwrPC" "0" )
  converge( 'ic "/Q_classic" "0" )
  converge( 'ic "/Q_min3" "0" )
  converge( 'ic "/Q_min3_ns" "0" )
  temp( 80 )
  ocnxlOutputSignal( "/D_all")
  ocnxlOutputSignal( "/clk_all")
  ;ocnxlOutputSignal( "/Q_classic" ?plot t)
  ;ocnxlOutputSignal( "/Q_pwrPC" ?plot t)
  ;ocnxlOutputSignal( "/Q_c2mos" ?plot t)
  ;ocnxlOutputSignal( "/Q_min3" ?plot t)
  ;ocnxlOutputSignal( "/Q_min3_ns" ?plot t)
  ocnxlOutputTerminal( "/V17/PLUS" ?save t)
  ocnxlOutputTerminal( "/V19/PLUS" ?save t)
  ocnxlOutputTerminal( "/V18/PLUS" ?save t)
  ocnxlOutputTerminal( "/V15/PLUS" ?save t)
  ocnxlOutputTerminal( "/V16/PLUS" ?save t)
  ocnxlOutputSignal( "/X_classic")
  ocnxlOutputSignal( "/X_pwrPC")
  ocnxlOutputSignal( "/X_c2mos")
  ocnxlOutputSignal( "/X_min3")
  ocnxlOutputSignal( "/X_min3_ns")
  ocnxlOutputSignal( "/D_pwrPC")
  ocnxlOutputSignal( "/C_pwrPC")
  ocnxlOutputSignal( "/D_c2mos")
  ocnxlOutputSignal( "/C_c2mos")
  ocnxlOutputSignal( "/D_min3")
  ocnxlOutputSignal( "/C_min3")
```

```
  ocnxlOutputSignal( "/D_min3_ns")
  ocnxlOutputSignal( "/C_min3_ns")
ocnxlSweepVar("Time" "1202.7n")
ocnxlEndTest()
;====================== Model Group setup
    ========================================
  ocnxlModelGroup( "default"
   '(
     ( "/home/magneva/master/65nm_mag/testbenk_power/adex3/
         modelgroups/spectre/default.scs" ?section "default")
     )
  )

  ;====================== Corners setup
      ========================================
  ocnxlCorner( "default"
   '(
      ("modelGroup" "default")
   )
)
;====================== Job setup
    ==============================================
  ocnxlJobSetup( '(
  "blockemail" "1"
  "configuretimeout" "300"
  "distributionmethod" "Local"
  "lingertimeout" "300"
  "maxjobs" "8"
  "name" "ADE XL Default"
  "preemptivestart" "1"
  "reconfigureimmediately" "1"
  "runtimeout" "-1"
  "showerrorwhenretrying" "1"
  "showoutputlogerror" "0"
  "startmaxjobsimmed" "1"
  "starttimeout" "300"
  ) )
;====================== Run Mode Options
    ======================================
  ocnxlMonteCarloOptions( ?dutSummary "65nm_mag:testbenk_power:1%/I69
      , /I68, /I66, /I65, /I63, /I64, /I57, /I55, /I49, /I56, /I53, /
      I54, /I43, /I44, /I40, /I41, /I42, /I48, /I37, /I35, /I29, /I36
      , /I33, /I34, /I28, /I27, /I23, /I22, /I24, /I21%Schematic%
      Schematic#" ?ignoreFlag "1" ?mcMethod "all" ?mcNumPoints "100"
      ?mcNumBins "" ?mcStopEarly "0" ?mcYieldTarget "99.73" ?
      mcYieldAlphaLimit "0.05" ?samplingMode "random" ?saveProcess
      "1" ?saveMismatch "1" ?useReference "0" ?donominal "1" ?
      saveAllPlots "1" ?monteCarloSeed "1" ?mcStartingRunNumber "" )
;====================== Run command
    ============================================
  ;out = outfile("myResults/testout.ocn" "a")
  ocnxlRun( ?mode 'sweepsAndCorners ?nominalCornerEnabled nil ?
      allCornersEnabled t ?allSweepsEnabled t)
  ocnxlOutputSummary(?exprSummary t) ;?specSummary t)

  of1 = outfile("~/master/out/pdp_schematic_80_1.out" "w+")

  openResults()
  selectResult('tran)
```

```
  fprintf(of1 "FirstLine")

  close(of1)

  of1 = outfile("~/master/out/pdp_schematic_80_1.out" "a")

  openResults()
  selectResult('tran)
  declare( out_arr[5] )
  out_arr[0] = (average(clip(IT("/V16/PLUS") (VAR("Time") * 0.1) (VAR
      ("Time") * 0.999))) * average(clip(VT("/I_Class") (VAR("Time")
      * 0.1) (VAR("Time") * 0.999))))
  out_arr[1] = (average(clip(IT("/V18/PLUS") (VAR("Time") * 0.1) (VAR
      ("Time") * 0.999))) * average(clip(VT("/I_min3") (VAR("Time") *
       0.1) (VAR("Time") * 0.999))))
  out_arr[2] = (average(clip(IT("/V19/PLUS") (VAR("Time") * 0.1) (VAR
      ("Time") * 0.999))) * average(clip(VT("/I_min3ns") (VAR("Time")
       * 0.1) (VAR("Time") * 0.999))))
  out_arr[3] = (average(clip(IT("/V17/PLUS") (VAR("Time") * 0.1) (VAR
      ("Time") * 0.999))) * average(clip(VT("/I_c2mos") (VAR("Time")
      * 0.1) (VAR("Time") * 0.999))))
  out_arr[4] = (average(clip(IT("/V15/PLUS") (VAR("Time") * 0.1) (VAR
      ("Time") * 0.999))) * average(clip(VT("/I_pwrPC") (VAR("Time")
      * 0.1) (VAR("Time") * 0.999))))

  ocnPrint((out_arr[0]) ?output of1 ?format "engineering")
  ocnPrint((out_arr[1]) ?output of1 ?format "engineering")
  ocnPrint((out_arr[2]) ?output of1 ?format "engineering")
  ocnPrint((out_arr[3]) ?output of1 ?format "engineering")
  ocnPrint((out_arr[4]) ?output of1 ?format "engineering")

  close(of1)
  ocnPrint("Test number1 is Finished")
  ocnPrint("Simulation Finished!")
;====================== End XL Mode command
    ===================================
ocnxlEndXLMode()

exit()
```

Listing 5: A typical generated OCEAN script