NTNU – Trondheim
Norwegian University of
Science and Technology

Soft-Supply Inverter

A New Power-Saving Logical Gate Applied in Several Sub-Modules of a 9-bit 1kS/s SAR ADC

Carl Fredrik Hellwig

Electronics System Design and Innovation
Submission date: March 2014
Supervisor: Trond Ytterdal, IET
Co-supervisor: Snorre Aunet, IET

Norwegian University of Science and Technology
Department of Electronics and Telecommunications

Norwegian University of Science and Technology

Soft-Supply Inverter

A New Power-Saving Logical Gate Applied in Several Sub-Modules of a 9-bit 1kS/s SAR ADC

Author: Carl Fredrik Hellwig

Supervisor: Trond Ytterdal
Co-Supervisor: Snorre Aunet

March 2014

Abstract

A new double-input-single-output logical gate called soft-supply inverter (SSI) has been presented here in this report. Simulations of the gate reveals a 25% lower static power consumption and a 100% higher dynamic power consumption, compared to standard logical equivalent design. Thus, the design was chosen to be a simulated as a part of the digital control logic in a 1kS/s 9-bit successive approximation register analog-to-digital converter (SAR ADC).

New versions of the D-latch, D-Flip Flop (DFF), 4-1 mux and additional reset circuitry has been designed using the SSI. Also, the C^2MOS and PowerPC 603 latches were used as both latches and flip-flops in order to compare with the SSI as alternative solutions of the same design.

A literature study was performed where the designs of [4] and [11] were considered the current state-of-the-art SAR ADCs for group-A converters. Because [4] aims at the comparator, which is outside the scope of this paper, [11] was chosen as the basis for the digital part of this work.

Simulations revealed a large reduction in power consumption, compared to standard logic design. The state machine, which was mostly DFF-based, revealed a 40% reduction in power consumption with the use of the SSI latch and DFF. However, the C^2MOS and PowerPC 603 versions revealed an even further reduction by as much as 64%. Regardless, the reset circuitry for these alternate designs showed a 14% and 20% decrease in power consumption, respectively, when the SSI was applied.

Preface

This report is the final thesis in the master study in electronic engineering at the Department of Electronics and Telecommunication, Faculty of Information and Technology, at the Norwegian University of Science and Technology (NTNU).

Trying to figure out applications for the soft-supply inverter has not been easy. I spent a lot of my time trying to figure out which basic operations could be redesigned using the new gate. Much of this pre-work has not been included in this report because it would become irrelevant to the converter.

I would like to thank both of my supervisors for their enthusiasm about working with the soft-supply inverter, especially Trond Ytterdal who have been available for questions all hours of the week, and his assistance when facing technical difficulties with some of the tools.

Finally, I would like to thank my friends and family who have supported me greatly throughout this year.

List of Contents

Page

List of Figures Page

List of Tables

Page

Abbreviations

ADC	Analog-to-Digital Converter
DAC	Digital-to-Analog Converter
SSI	Soft-Supply Inverter
SAR	Successive Approximation Register
SNR	Signal-to-Noise Ratio
ENOB	Effective Number Of Bits
FOM	Field Of Merit
DFF	Data-Flip Flop

1. Introduction

SAR ADCs at very low frequency are often used in biomedical, sensory and wireless applications. These converters called group-A converters[1]. Area and power consumption are very important factors for these applications, especially in biomedical devices. The problem is that now as the ADCs have reached the noise floor [2], it becomes increasingly challenging to continue reducing the power consumption without the risk of degrading the quality of the converter. Several attempts include digital error correction [1], energy scavenging [3] and majority voting [4][5].

In an attempt to reduce transistor count, thereby reducing cost, power consumption and possibly area, a new logical gate called the soft-supply inverter (SSI) will be presented here. The design will be simulated and compared to the standard version of equivalent logical circuit. Based on these results, which are presented in appendix A, a set of design rules will be made to take better advantage of its advantages.

A literature study will be presented to get an overview of the current state-of-the-art group-A SAR ADCs, where field-of-merit (FOM) and power consumption are key parameters. From these findings, a digital design will be chosen and new designs using the SSI will be designed. Alternative solutions to the same designs will also be simulated for comparison.

Finally, a 1kS/s 9-bit SAR ADC will be implemented using the SSI designs in an attempt to see what kind of improvements the new gate can yield. Since the focus lies on the performance of logical designs, only the control logic will be implemented on transistor level. Both the comparator and DAC will be ideal. The VerilogA codes can be found in appendix B.

1.1 Overview

The report is structured as follows:

Chapter 1 presents the assignment, motivation for energy-efficient converter design and the overview of this paper.

Chapter 2 describes the theoretical background to understand converters, with focus on the SAR ADCs.

Chapter 3 presents the results of the literature study, a discussion of the findings, and the conclusive choice of architecture.

Chapter 4 contains a top-down overview of sub-modules in the chosen architecture of the SAR ADC's control logic.

Chapter 5 explains the idea behind the SSI, discusses its properties based on the gate's simulation results, and finds areas of application for the design.

Chapter 6 discusses some detail adjustments to the SAR ADC modules from chapter 3 based on soft-supply inverter properties.

Chapter 7 presents the test procedures and simulation results.

Chapter 8 discusses the results from the SAR ADC, its sub-modules, as well as the SSI in general.

Chapter 9 concludes the discussion from chapter 8 and summarizes the work.

Chapter 10 suggests areas of future work.

Chapter 11 contains a list of references used in this paper.

Appendix A contains the simulations for identifying the properties of the soft-supply inverter.

Appendix B contains the source code of modules that weren't implemented on transistor level (comparator, DAC, ideal DAC and ideal ADC).

2. Theoretical Background

This chapter will explain the binary search algorithm used by the SAR ADC, and the functionality of converters with focus on the SAR ADC.

2.1 Binary Search Algorithm

This algorithm finds a number in an array through position-tracking. It checks whether the desired number is higher than (or lower than) the middle value of an array. The answer will make it possible to disregard half of the array in a single step, thus reducing the computation time considerably compared to the case if one would check every single variable in the array.

As an example, consider an algorithm looking for the number 5 in an array 1 through 8. The first question would be "is the number larger than 4". The answer would be "yes", and the numbers 1-4 are discarded immediately, and only the new array 5 through 8 is left. Next, the number half between 5 and 8 is chosen as the new middle value, thus the question would be "is the number larger than 6". The answer is "no", and finally there would be a similar question of whether or not the number is larger than 6.

Since the algorithm always cuts the arrays by half in each step, the number of iterations is about $O(\log_2 N)$.

2.2 Converters

Converters are systems that translate a signal from one domain to another. This makes it possible for electronic devices to communicate with the outside world, which is analog. Analog-to-digital converters (ADCs) translate analog signals into the digital domain. Digital-to-analog converters (DACs) translate digital signals into the analog domain.

Analog signals are time-varying continuous signals that carry an amplitude, a frequency and a phase margin. To store them requires a capacitor, and leakage reduces their accuracy greatly.

Digital signals makes it possible to perform complex calculations at high speed and good accuracy. They are discrete binary values represented as either '0' (zero) or '1' (one). In CMOS technology, a digital value '0' or '1' is represented by a voltage being either at ground (usually at 0V) or at supply, respectively. They can be stored through capacitor-less logical modules, thus information is not lost upon storage. However, the usage of digital modules requires converters which spend both power and area.

Signal sampling must be done according to Nyquist rate in order for the result to be unambiguous. It means that the sampling frequency fs must have the following criteria:

(Eq. 1) $$fs \geq 2\,fin$$

where fin is the frequency of the input signal.

2.2.1 DAC

The DAC converts digital bits into an analog signal as shown in figure 2.1.

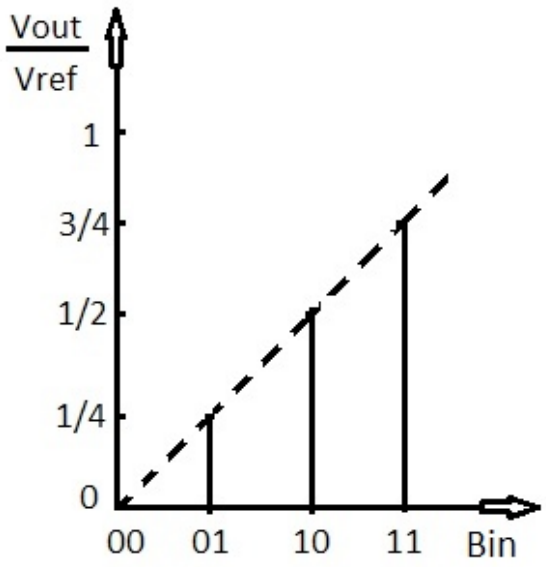

Figure 2.1: Ideal DAC [6]

This behavior can also be described mathematically in Eq. 2. The *Input* is a binary number with the digits being either high or low depending on the different input bits of the converter.

(Eq. 2) $$\frac{(Input)_2}{2^N} = \frac{Vout}{Vref}$$

As can be seen by the graph in figure 2.1, the different bits have different impact on, or significance to, the output value. Thus, the bit on the far left in the input is called the "most significant bit" (MSB). The next bit is called MSB-1, etc. The bit on the far right is called the "least significant bit" (LSB).

By changing equation 2, it becomes

(Eq. 3) $$(Input)_2 \frac{Vref}{2^N} = Vout$$

The smallest possible output voltage change is when the input changes by a single LSB.

(Eq. 4) $$(Input)_2 V_{LSB} = Vout$$

Thus we define this voltage change as follows:

(Eq. 5) $$V_{LSB} = \frac{Vref}{2^N}$$

For converters with high resolution, this voltage becomes very small, and thus accuracy becomes an increasingly important attribute.

2.2.2 ADC

The ADC converts data by quantifying the analog input signal into 2^N digital values, where N is the resolution of the converter. Figure 2.2 shows the relation between input and output on an ideal 2-bit ADC.

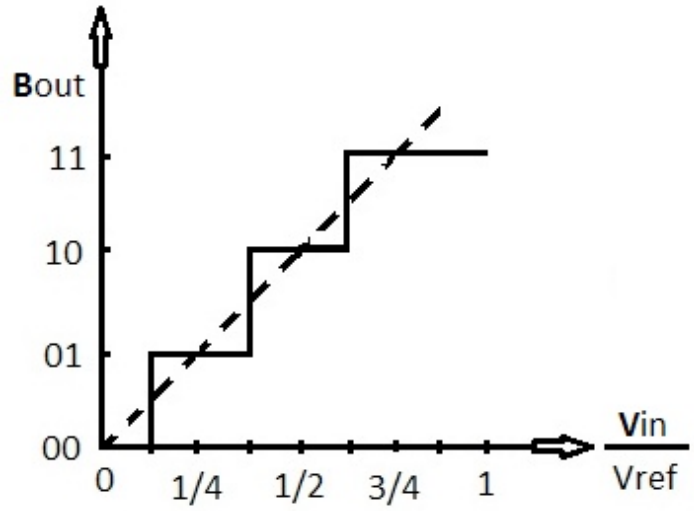

Figure 2.2: Ideal ADC [6]

Note that this quantization becomes a source of error when one attempts to regain the original signal, as will be discussed later.

2.3 SAR ADC

The successive approximation register (SAR) ADC uses a binary search algorithm to calculate the digital output bits from a sampled analog input signal. Figure 2.3, showing a binary search with a resolution of 3 bits, is drawn from [1]. Vcm (common mode) is half the voltage of Vref.

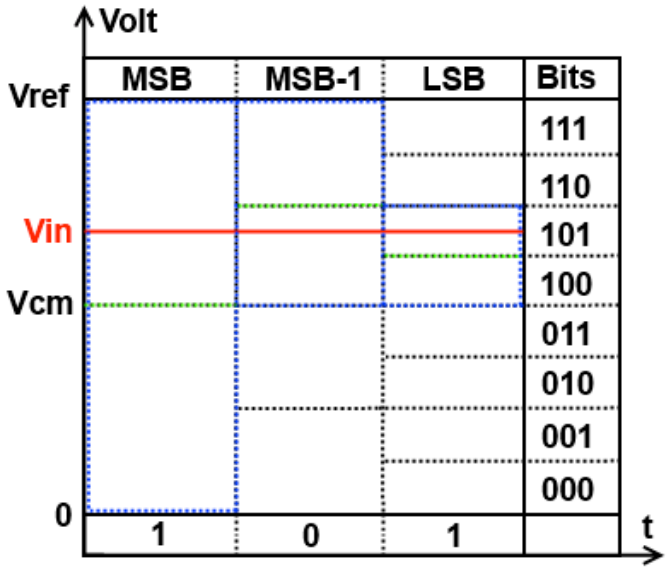

Figure 2.3: Binary Search, 3 Bits

This example performs just like the example given in section 2.1, except this range is from 0 to 7. Also, the iterations are made trying to find out where the analog input voltage lies in comparison to the voltage array from 0 (ground) to Vref (supply voltage). The blue boxes show the arrays where the algorithm calculates each bit; the result is '1' (yes) if the input signal is above the green line, and '0' (no) if it is equal to or below.

The SAR ADC output becomes the results of each iteration made to find out where the input lies in comparison to the green line. The bit array on the right side of the figure shows the range of digital representations based on possible input voltages.

Figure 2.4 shows the system blocks of a general SAR ADC. It uses a

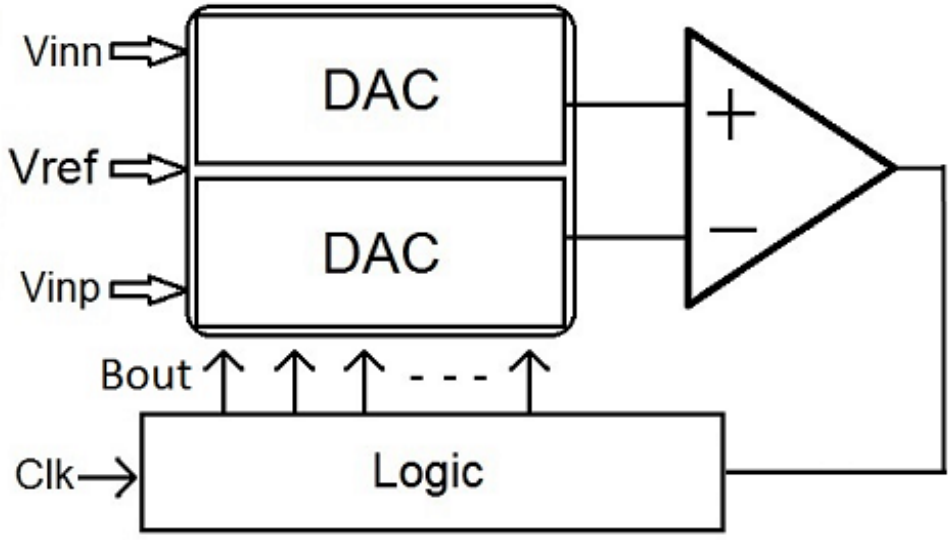

Figure 2.4: General SAR ADC

For a SAR ADC, the input is first sampled on a capacitive DAC, which removes the need for a separate sample-and-hold circuit. The stored input will be compared to another voltage through a comparator, akin to the green line in figure 2.3, and the result of this comparison decides the first bit. The bit is stored in a register, and the next comparison can begin. Each time a comparison is made, the voltage in the DAC is altered using switches controlled by the logic module. After a series of N comparisons, N being the resolution of the SAR ADC, the conversion is over, and another sampling can begin for the next signal conversion.

2.3.1 Comparator

The comparator is a circuit that compares the voltage of two input signals. Depending on which of them is larger than the other, the output goes either high or low.

2.3.2 DAC

As already mentioned, the DAC for the SAR ADC is capacitive. The conventional DAC array[6] is shown in figure 2.5, where N is the resolution of the converter and Cu is the unit capacitance.

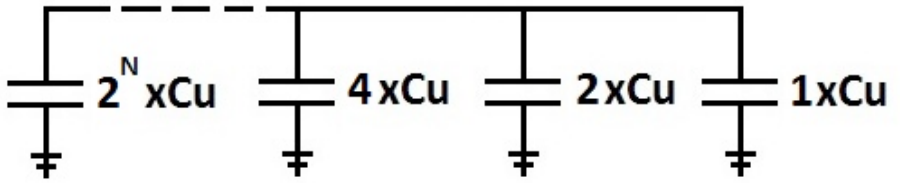

Figure 2.5: Conventional DAC array

Because the relationship between the voltage and charge stored on a capacitor can be described as

(Eq. 6) $$\Delta V * C = \Delta Q$$

the individual charge on the capacitors will be proportional to the size of the capacitor even if the voltage stored across them is the same. This means that the charge seen by the comparator will change differently based on which one of the capacitors receives a change in voltage. The stored bits from earlier comparisons can be used to adjust the voltage with switches. Hence, with a binary-weighted capacitive DAC like in figure 2.5, it will be possible to perform a binary search.

2.3.3 Control Logic

To control both the DAC and comparator correctly in order to perform the binary search algorithm used by the SAR ADC, certain digital logic-blocks are needed: a state machine to keep track of the calculations, a module to control the DAC switches during sampling and signal conversion, a system to reset the circuit and a memory to store the calculated bits. The specifics of these modules will be explained further in chapter 6.

2.3.4 Architecture

The way the binary algorithm is carried out relies heavily on the architecture. The two most commonly mentioned architectures will be presenter here: the conventional and monotonic methods.

The conventional architecture uses a trial-and-failure procedure. For each bit, it assumes a switch in the voltage of the DAC, and after the voltage has settled and been compared, that switch decision is either confirmed or disproved (see figure 2.6).

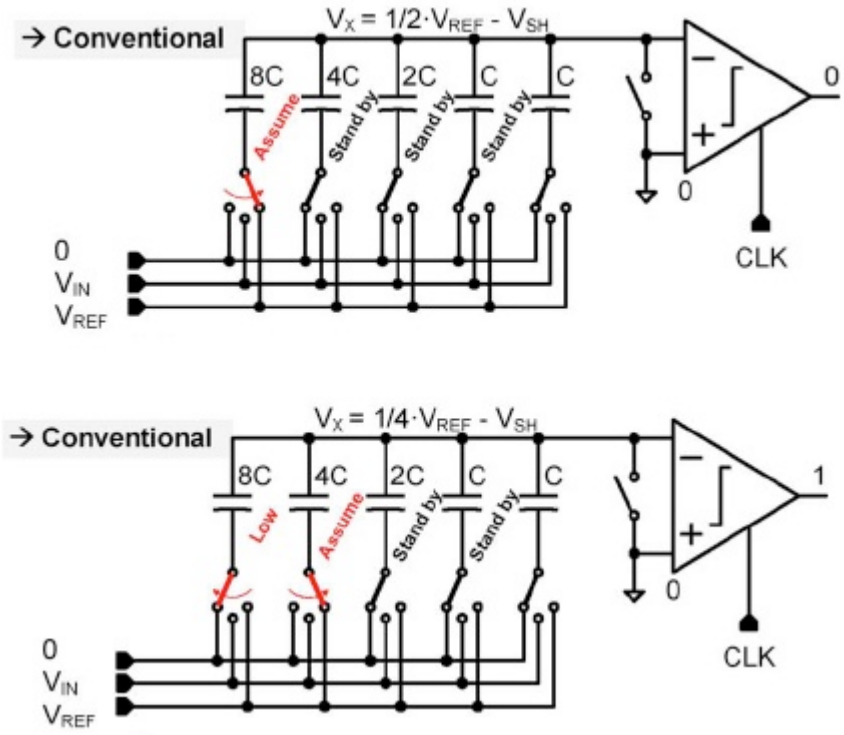

Figure 2.6: SAR ADC, conventional method [1]

In a worst case scenario, each bit calculation switches three times. This extra switcching increases the power consumption of the circuit.

The monotonic method uses a slightly different approach. Instead of testing each bit, the comparisons can be performed right after the voltage has settled without making any assumptions. Also, since one does not need to switch the MSB, that whole capacitor is not needed and can be removed, thus removing half of the capacitance in the DAC (see figure 2.7).

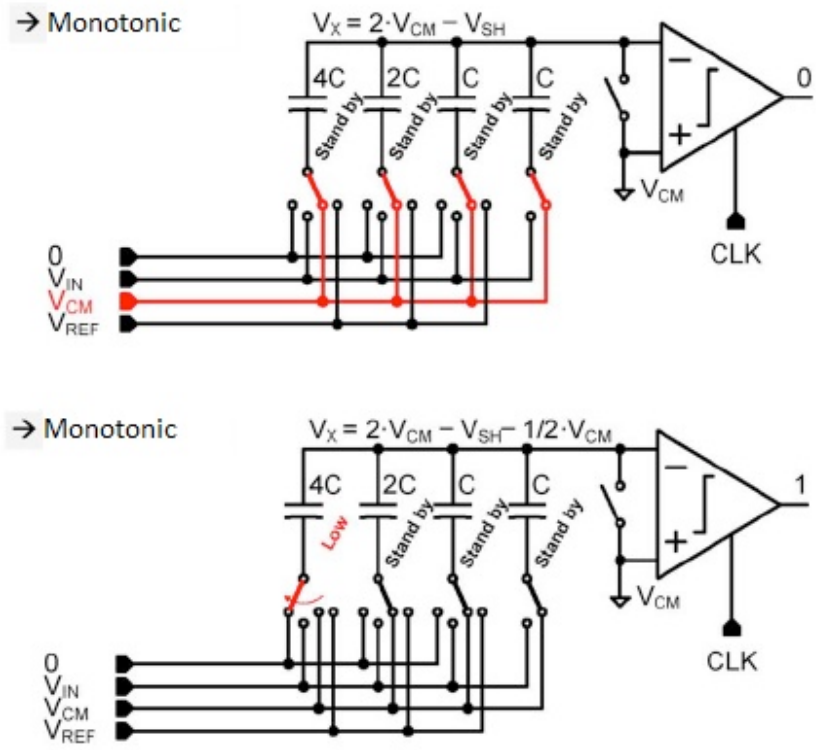

Figure 2.7: SAR ADC, monotonic method [1]

Another benefit is that the need for less switching reduces the power consumption further. However, the inclusion of the signal Vcm means an increase in the number of switches.

2.4 Quantization Error

Because of a limited resolution in the converters, it is not possible to fully restore the original analog signal. The difference between the original signal and the restored signal is called the quantization error (see figure 2.8).

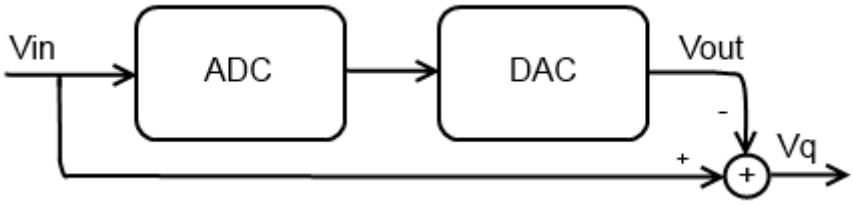

Figure 2.8: Quantization Noise

As mentioned in section 2.3.2, there voltage value between two digital representations is called V_{LSB}. The boarder limit between two signals are therefore half of that:

(Eq. 7)
$$-\frac{1}{2}V_{LSB} \leq V_Q \leq \frac{1}{2}V_{LSB}$$

With a linear increasing Vin, the quantization noise changes as can be seen in figure 2.9.

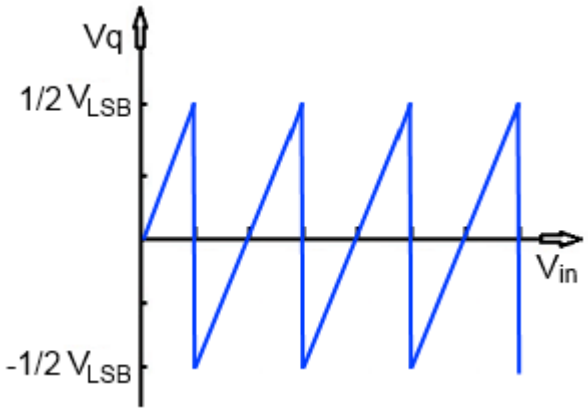

Figure 2.9: Quantization Sawtooth Graph

Johns & Martin in [6] shows that rms signal of the quantization noise can be expressed as

(Eq. 8)
$$V_Q = \frac{V_{LSB}}{\sqrt{12}}$$

This noise affects the general performance of the conversion.

Note: the quantization noise goes toward zero when the resolution goes toward infinity.

2.5 ENOB

The Effective Number Of Bits (ENOB) is a type of measurement that calculates the actual efficiency of the ADC. Different sources of error degrades performance of the converter to a point where it may not deliver the full resolution it was intended for.

Eq 9 shows how the signal-to-noise ratio is calculated.

(Eq. 9) $$SNR = 10\log\left(\frac{Vin^2_{(rms)}}{Vnoise^2_{(rms)}}\right)$$

The best possible achievable SNR is when the only source of error is the quantization noise, which has been argued cannot be avoided. Eq 9 can therefore be described as follows

(Eq. 10) $$SNR = 20\log\left(\frac{Vin_{(rms)}}{Vq_{(rms)}}\right)$$

Assuming that the input is sinusoidal [6], we get the following

(Eq. 11) $$SNR = 20\log\left(\frac{Vref/(2/\sqrt{2})}{V_{LSB}/(\sqrt{12})}\right)$$

(Eq. 12) $$SNR = 20\log\left(\sqrt{\frac{3}{2}}2^N\right)$$

(Eq. 13) $$SNR = 6.02N + 1.76dB$$

This is the highest possible SNR that can be achieved with a converter with a resolution of N bits without oversampling[6]. Naturally, other sources of error, such as charge-injection[7], kickback[8] and mismatch[9], will degrade this even further. Since most these errors reduce the accuracy, and therefore the resolution of the converter, in the DAC and comparator they will not be discussed here.

Rewriting Eq 13 gives

(Eq. 14) $$ENOB = \frac{SNR - 1.76}{6.02}$$

which describes the ENOB with respect to the signal-to-noise ratio.

2.6 Power Consumption

The total power consumption is the sum of the static and dynamic power consumption:

(Eq. 15) $$P = P_{Stat} + P_{DYN}$$

Static power consumption is the current leakage from supply to ground when there are no voltage change in a circuit, i.e. during a steady state. This leakage is caused by mismatches and non-idealities in the circuits, and is linearly proportional to the supply voltage. The static power dissipation increases as technology gets smaller and the number of transistors keeps growing.

Dynamic power consumption is dominating part of power consumption. Whenever there is a signal transition in a circuit, energy dissipate during the charge/discharge.

(Eq. 16) $$P_{Dyn} = \alpha C_{node} V_{DD}^{2} f$$

Vdd is the supply voltage, f is the switching frequency, C_{node} is the capacitance in the switching circuit and α is the activity factor. The dynamic power dissipation per component is reduced with smaller technology, but with the increase of transistors the total contribution is almost constant[10].

2.7 Figure-of-Merit

The figure-of-merit (FOM) is an quantity measure to used compare performance of electronic devices based on its power consumption per operation. For SAR ADCs, it takes into consideration the ENOB as well as the sampling frequency to determine the FOM.

The formula for the Walden FOM [2] is

(Eq. 17) $$FOM = \frac{P}{2^{ENOB} * fs}$$

where P is the power consumption, and fs is the Nyquist rate sample frequency.

The FOM will not be calculated for the results of this report, as it only includes the digital part of a SAR ADC, but it will be used to discuss the performance of previous work to determine current state-of-the-art.

3. Literature Study – SAR ADC

In this chapter, previous work will be discussed in order to find a suitable design for the control logic module.

Table 3.1 shows a performance overview of SAR ADCs considered to be among the state-of-the-art designs.

Table 3.1: SAR ADC Performance Overview

	[4]	[11]	[5]	[12]	[3]	[13]
Technology	65nm	65nm	65nm	40nm	65nm	90nm
Resolution	10b/12b	10b	10b	9b	10b	10b
Sampling Rate	40kS/s	1kS/s	250kS/s	1.1MS/s	1MS/s	100kS/s
Supply Voltage	0.6V	0.7V	0.4V – 1V	0.5V	1V	0.35V
ENOB	9.4 – 10.1	9.1	7.1 – 9.1	7.5	8	9.05
Power (nW)	72 – 97	3	290 – 510	1200	1900	170
FOM (fJ/c-s)	2.7 – 2.2	5.5	3.3 – 6.8	6.3	4.4	3.2

[4] features a data-driven noise-reduction method designed to assist the comparator. First of all, it uses a type of majority voting that helps the comparator to make the correct bit decision when faced with a high noise wall. The number of votes made are adjusted according to the amplitude of the sampled input signal to compensate for possibly low SNR, but to also stop the voting early if the SNR is good. In addition, logical gates have been added to the comparator which implements an internal self-oscillating clock for controlling the timing of the logic.

[11] uses an "ADC architecture with maximal simplicity". In addition to use a split-capacitive DAC array and bottom-plate sampling with only two voltage sources, it also uses a multi-V_t design for its switches to reduce current leakage. The register is designed with simple latches to further reduce power dissipation and the transistor count. Since the sampling frequency is so low, the logic circuits are design at minimum size and the latches are static, instead of dynamic (the difference will be explained later).

[5], much like [4], also uses a flexible majority vote comparison. Here, the first bit calculations have a reduced number of votes as this redundancy is not needed for the first low-noise comparisons. In addition, it also incorporates a non-binary redundant DAC with reduced capacitance. The error it induces is digitally corrected during each cycle.

[12] features a tri-level comparator and a DAC calibration technique. The tri-level comparator focuses to handle the issue of meta-stability which drastically reduces comparator output decision. By including a meta-state detector that is calibrated to the decision time, it can trigger a bit decision by knowing that the comparator has entered a meta-stable state. As this improves the DAC's resolution by one bit, this can reduce the capacitance by half, thus reduce power consumption.

[3] is based on a split-capacitive DAC array with step-wise charging for energy scavenging purposes. When a capacitor is discharged, parts of the charge is left in intermediate levels on other capacitors only to be harvested once the capacitor needs to be charged again. This results in a lot less total charge/discharge which lowers overall energy consumption.

[13] uses a non-binary technique; the DAC has an arbitrary-weighted capacitance array where each of the bits are weighted with a radix less than 2. It uses also a clever switching algorithm that alters the Vcm through a DAC common mode level shift, which improves comparator operation at low voltages by increasing the Vgs of the comparator inputs. Furthermore, they introduce a leakage reduction sample switch and a charge-pump circuit that decrease input disturbance and leakage further.

3.1 Conclusion

During this literature study, there was found no other SAR ADC that came close to the low FOM presented in [4]. Same goes for [11] in terms of total power consumption. Unless there are other designs with better performance in these areas that were neglected in this literature study, [4] and [11] are here considered the current state-of-the-art SAR ADCs.

Most of the designs listed here improve either the DAC or the comparator architectures in one way or another. Since the work in this report is focusing on the digital part, most of the designs are a little outside the scope of this work. However, [11] stands out as a SAR ADC which design is mostly concerned with reducing transistor count by simplifying the digital part. Therefore, this report will be based on the designs of [11], albeit with a few exceptions.

First of all, it was discovered a way to realize 4-1 muxes using the soft-supply inverter (which will be explained later). Since this mux can be used outside the scope of a SAR ADC, thus would be interesting to investigate further, it was decided that the design used in this work will use a tri-level based architecture [14].

Finally, with the choice of using 4-1 muxes as switches, the switch control blocks in the digital part will be quite different from the work in [11].

4. Control Logic - Design

The design implementation will be described in a top-down perspective which should make it more clear what modules are needed to make the system behave as intended, as well as which sub-modules must be designed. The goal is to use this knowledge later to easier find out where the soft-supply inverter can be applied.

A block diagram showing the synchronous control logic can be seen figure 4.1.

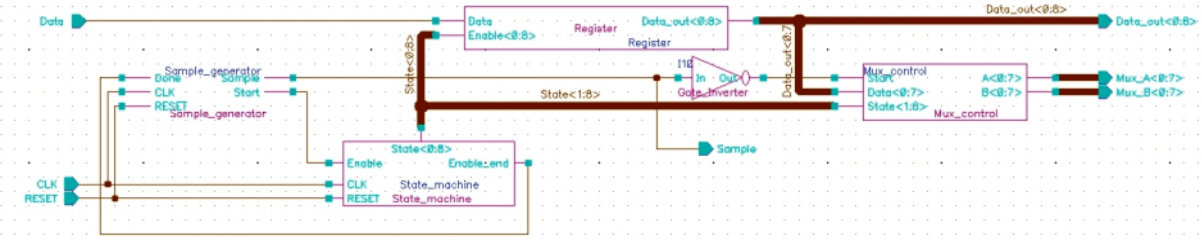

Figure 4.1: Block Diagram Control Logic

The clock-driven sample generator and state machine control the rest of the logic. The sample signal toggles DAC sampling while the state signals enables writing to the register for bit storage. The mux control module toggle the 4-1 muxes in the DAC.

4.1 State Machine

The state machine is a series of D-flip flops (DFFs) as shown in figure 4.2.

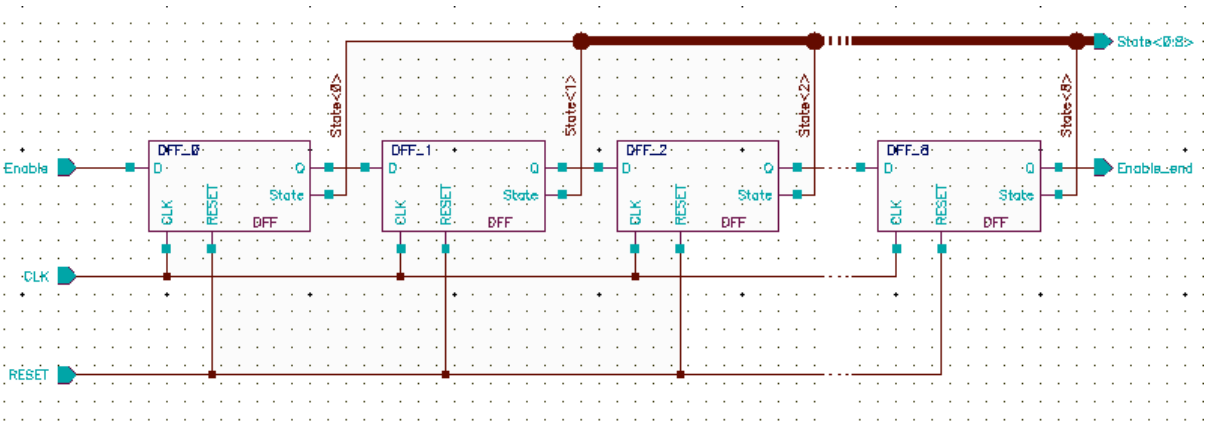

Figure 4.2: Block Diagram, State Machine

With each clock period, every D-flip flop will store their inputs for a single clock period before it is overwritten by the next signal. Thus, the input of the module will propagate through the module as the clock continues its swings. If said signal is a pulse not longer than a clock period, only a single D-flip flop will store said pulse at any given time. This means that one can use this circuitry to know how far the converter has come in its calculations (figure 4.3).

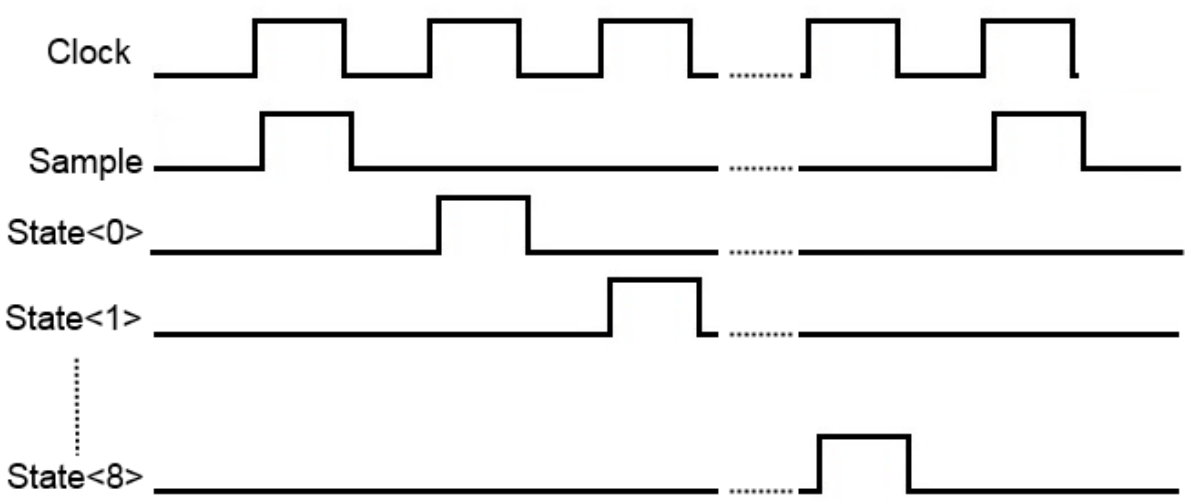

Figure 4.3: Signal Diagram State Machine

4.2 Memory Register

The calculated results of the comparator must be stored after every clock cycle during conversion. This can be done through the use of a register that consists of a memory latch for each bit. These memory latches must be enabled for writing seperately so that one does not overwrite earlier results. A way to realize this is to use the state signals from the state machines, since only one of them is high at any given moment. A block diagram of the memory register is shown in figure 4.4.

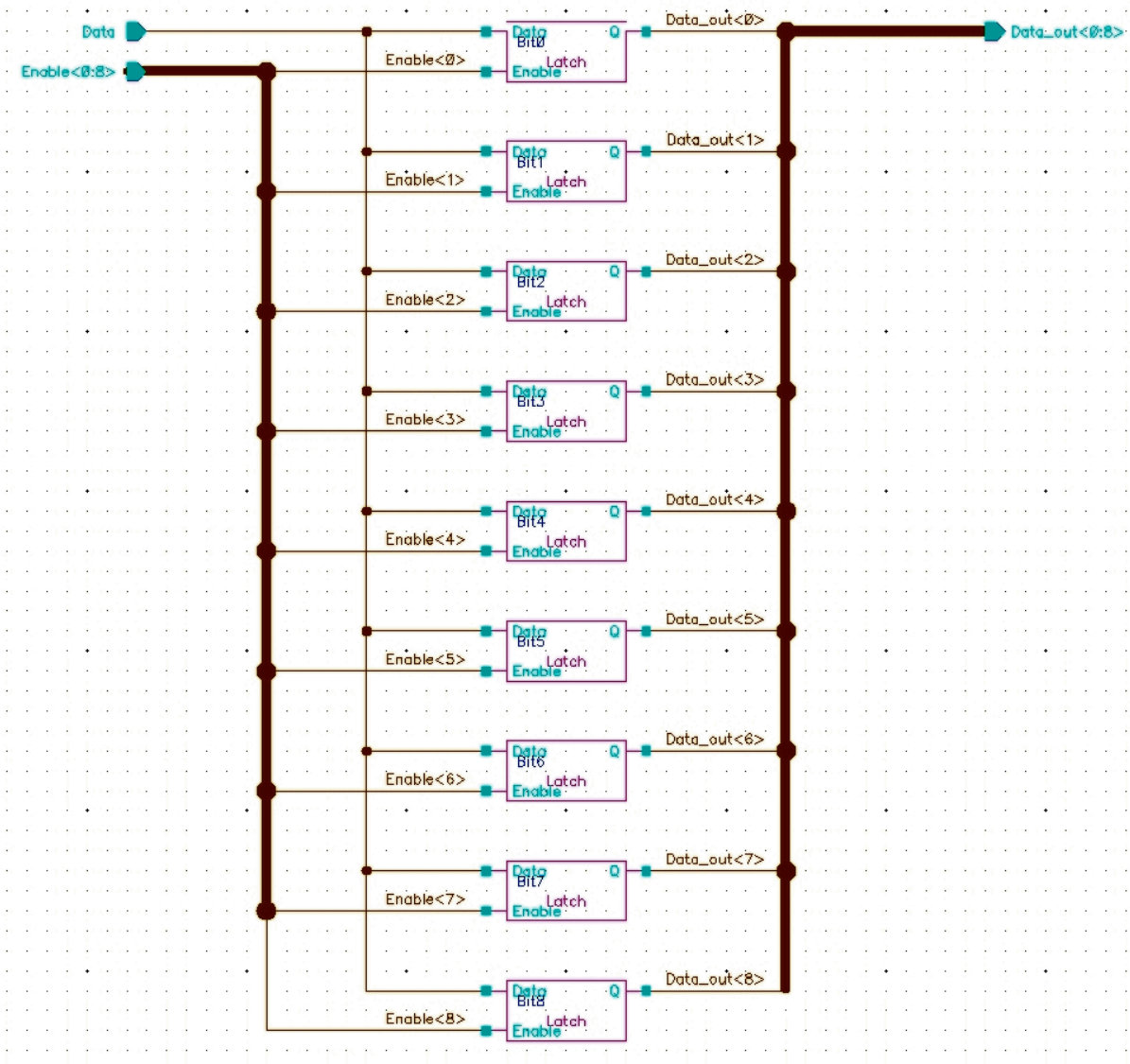

Figure 4.4: Block Diagram, Memory Register

The Enable bus is from the state machine, while the Data signal is the output of the comparator. The bit will be applied at every memory latch, but only stored on the one that the state machine enables.

It is very important that the clock does not reset the comparator before the state machine has been able to store the bit.

4.3 Switch Controls

In order for the DAC to properly calculate each digital bit, the switches that control the charge on the sampling capacitors must be switched correctly depending on several factors: whether or not the converter is sampling or conversion, the stored bits from previous calculations in the same conversion, and how far during the calculations it has come. The implementation of this logic circuit will be explained later in this report.

4.4 Mux, 4-1

This mux must be controlled by at least n controls signals, where n is

(Eq. 18) $2^n = 4$

This gives us that n = 2.

Figure 4.5 shows the conventional implementation for a 4-1 mux.

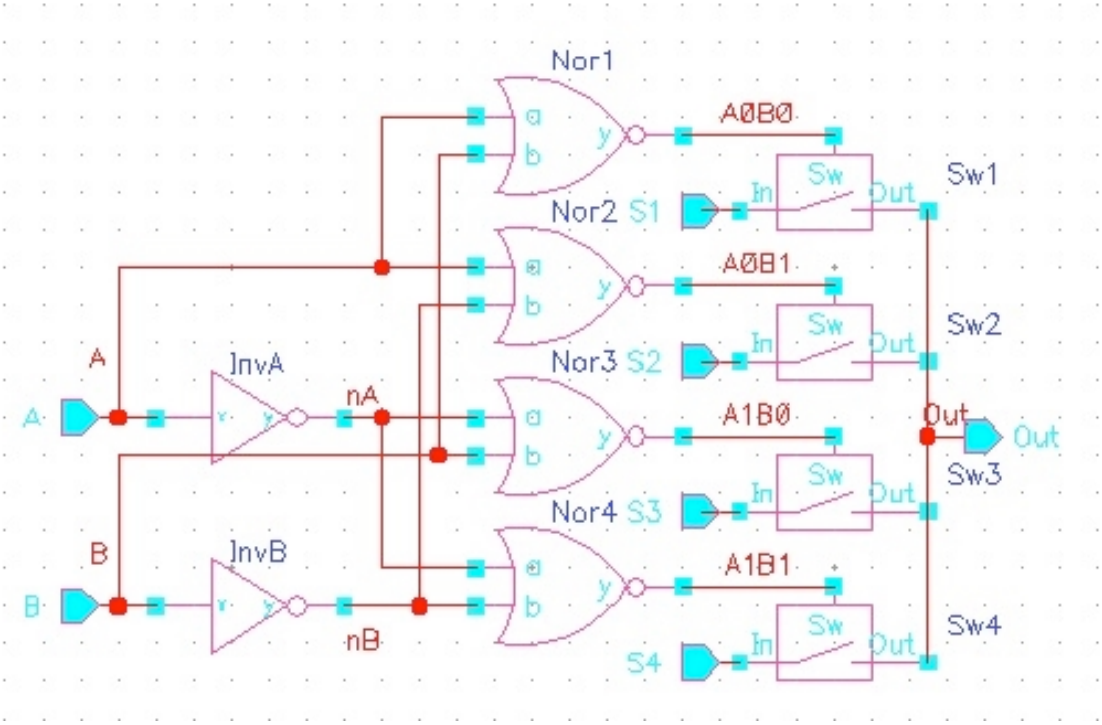

Figure 4.5: Schematic, Conventional 4-1 Mux

These two control signals must be logically generated from the three previously mentioned factors in the switch controls block.

4.5 Reset Circuitry

When the SAR ADC's reset signal goes high, the circuit must immediately stop whatever it is doing, and be ready to start from the beginning the moment the reset signal goes low again. In practice, that means that the state machine needs to force the stored values on the DFFs low, and the sample generator must force its value high. The state machine signals will propagate through the switch controls block which will set the DAC in its sampling state. When reset goes low again, the state machine will begin to propagate the high signal from the sample generator through the state machine, thus a new signal conversion begins.

4.6 Memory Latch

There are two types of latches: dynamic and static. Dynamic latches store bits on a capacitance, and therefore depends on continous switching to retain the information due to current leakage. It requires less transistors than static designs and can also operate at much higher frequencies. However, due to the current leakage, it cannot operate under low frequencies. Yeo and Burm in [15] experienced a bit rate lower than 1 Gbit/s would ruin the information entirely.

Static latches are larger than dynamic latches, and therefore consume more power and have slower signal propagation. However, since they store bits in a recursive buffer instead of on capacitance, they don't depend on continous switching to keep the bit value. Since the frequency for this SAR ADC is only 1kHz, it becomes the obvious choice between the two.

The latches presented here are all static latches. used in the DFF and the register is shown here.

4.6.1 SR-Latch

A way to store a bit signal is to connect to NAND-gates as seen in figure 4.6.

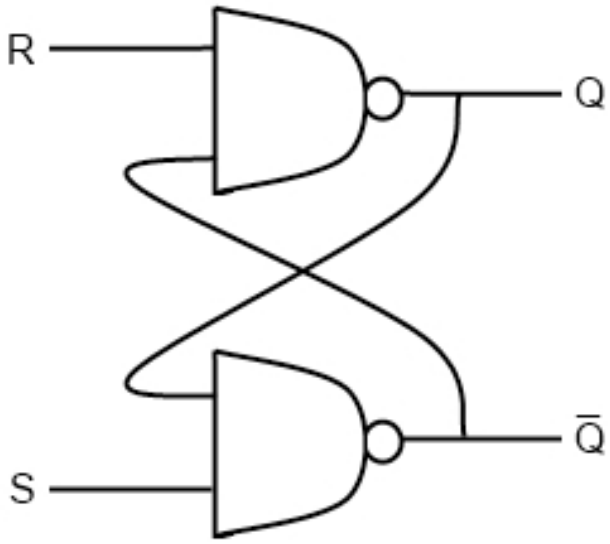

Figure 4.6: SR-Latch, Standard

The circuit has a truth given in table 4.1.

Table 4.1: Truth Table for SR-Latch, Standard

R	S	Q	Q_{NEXT}
0	0	0	Reset*
0	0	1	Reset*
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

***Reset means that both Q and Q(not) will have the same value which, in this case, is 1.**

Notice the value of Q is irrelevant for the value of Q_{NEXT}, except in the situation where both S and R equals 1. In this case, the value of Q remains the same regardless of its previous value. This allows the possibility to store a value on the latch for memory purposes.

From the truth table above, one can derive the following behavior in table 4.2.

Table 4.2: Behavior of SR-Latch, Standard

R	S	Q_{NEXT}
0	0	Reset
0	1	1
1	0	0
1	1	Q

4.6.2 D-Latch

Knowing that the SR-Latches can be used for memory purposes, it still needs some extra logic, to avoid the undesireable state where S and R are both low, before it can be safely used as a circuit to store data.

Figure 4.7 shows the extra logic used to make a standard version D-Latch.

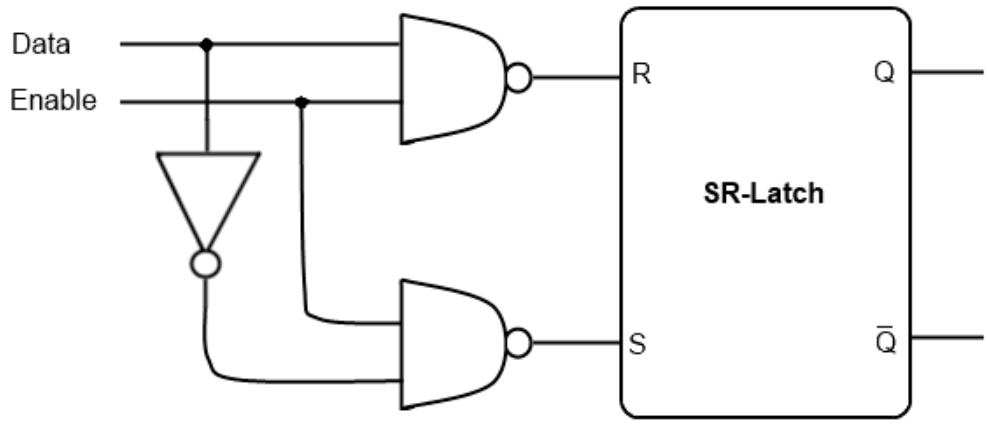

Figure 4.7: D-Latch, Standard

The truth table is shown in table 4.3.

Table 4.3: Truth Table for D-Latch, Standard

E	D	R	S	Q
0	0	1	1	Q
0	1	1	1	Q
1	0	1	0	0
1	1	0	1	1

The Data signal D is written onto the D-Latch when the Enable signal E is high. When Enable goes low, the value of D will be stored on Q until Enable goes high again.
Alternately, the same functionality can be made by removing the inverter and connect the R signal to where the inverter were connected previously. Thus, the circuit contains fewer transistors, and should consume less power.

The implementation of the entire transparent D-Latch can be seen in figure 4.8.

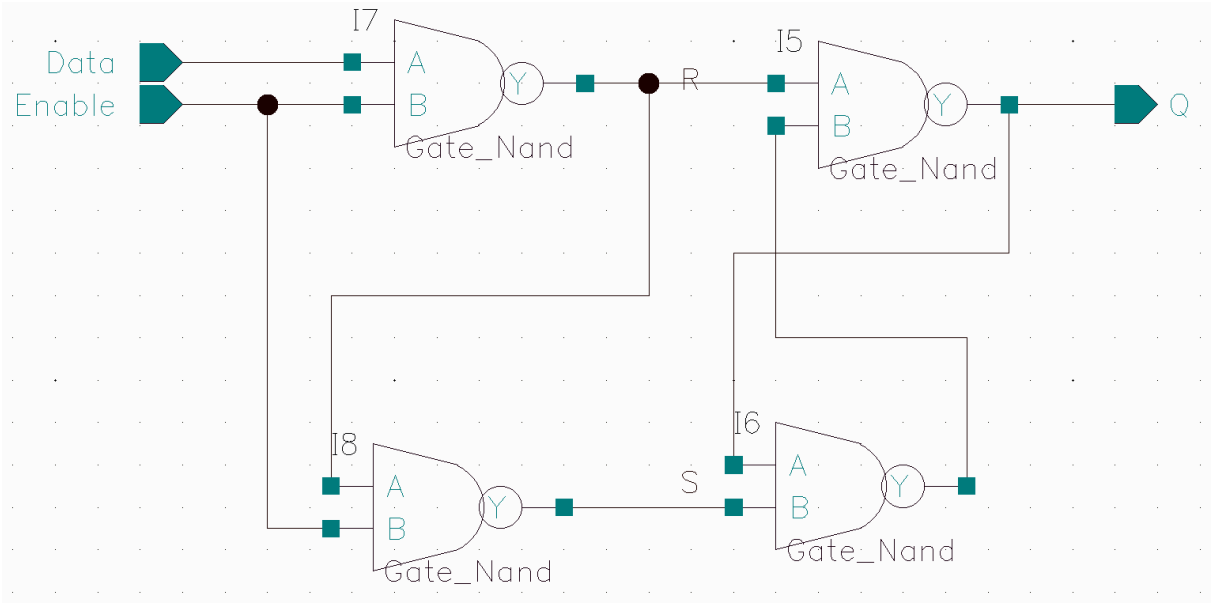

Figure 4.8: Complete D-Latch, Transparent

4.6.3 Inverter-Based Latches

Another way to store a bit value in a recursive buffer is to use a circle of two inverters coupled as shown in figure 4.9.

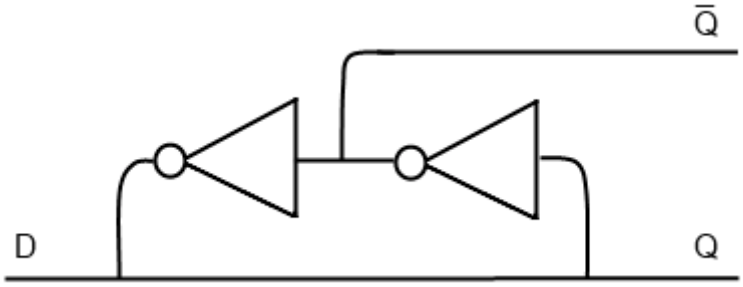

Figure 4.9: Inverter-Based Latch

The figure is simply for illustrating the design, as a complete latch would also need an enable signal, as well as a way to reset the circuit.

4.6.4 C^{2}MOS Latch

The C^{2}MOS, which stands for Clocked CMOS, is a pseudo-static inverter-based latch[16], as shown in figure 4.10.

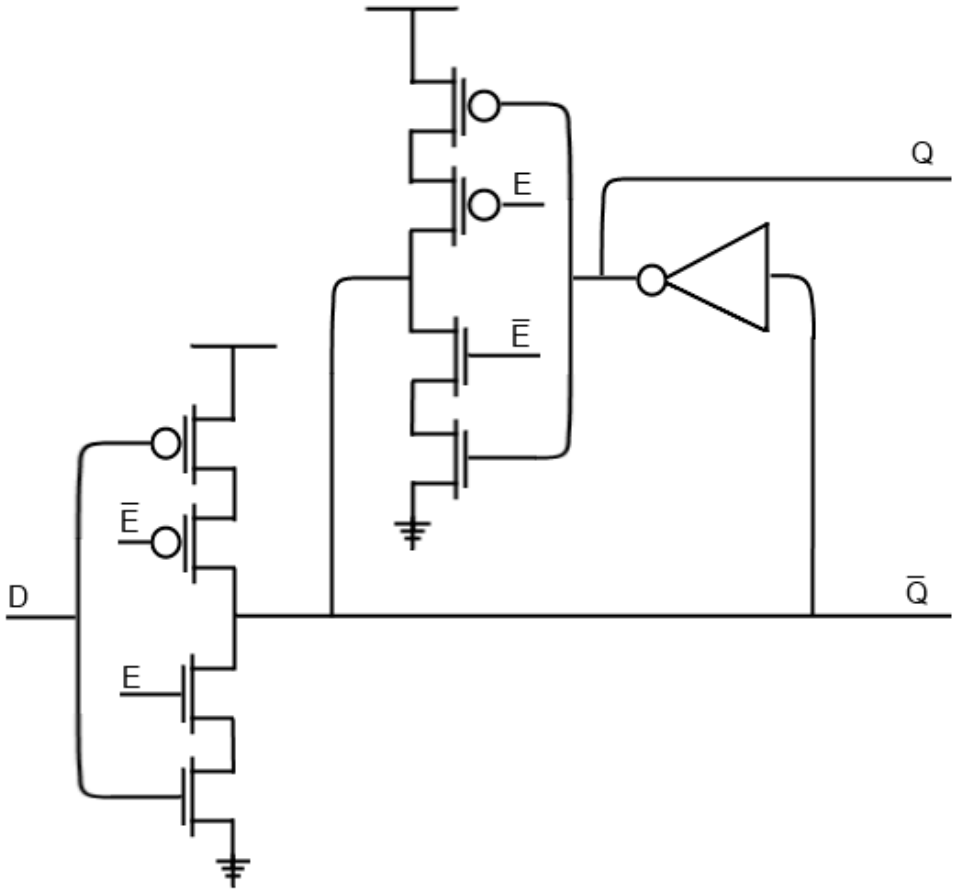

Figure 4.10: C^{2}MOS Latch

The circuit works similar to that of a standard D-latch. When the enable signal E is high, the data signal D is written onto the latch by enabling the clocked inverter at the input. At the same time, it cuts the inverter circle so that the stored signal doesn't intervene with the new one. When E goes low again, the latch gets cut off from the input, and the data is stored by the recursive buffer.

4.6.5 PowerPC 603

The PowerPC 603 latch is similar to the C^2MOS by the fact that it is also pseudo-static, and that is uses a recursive buffer consisting of two inverters. The term *603* is a reference to the second microprocessor from the PowerPC Architecture family[17]. Figure 4.11 shows the schematic of the circuit.

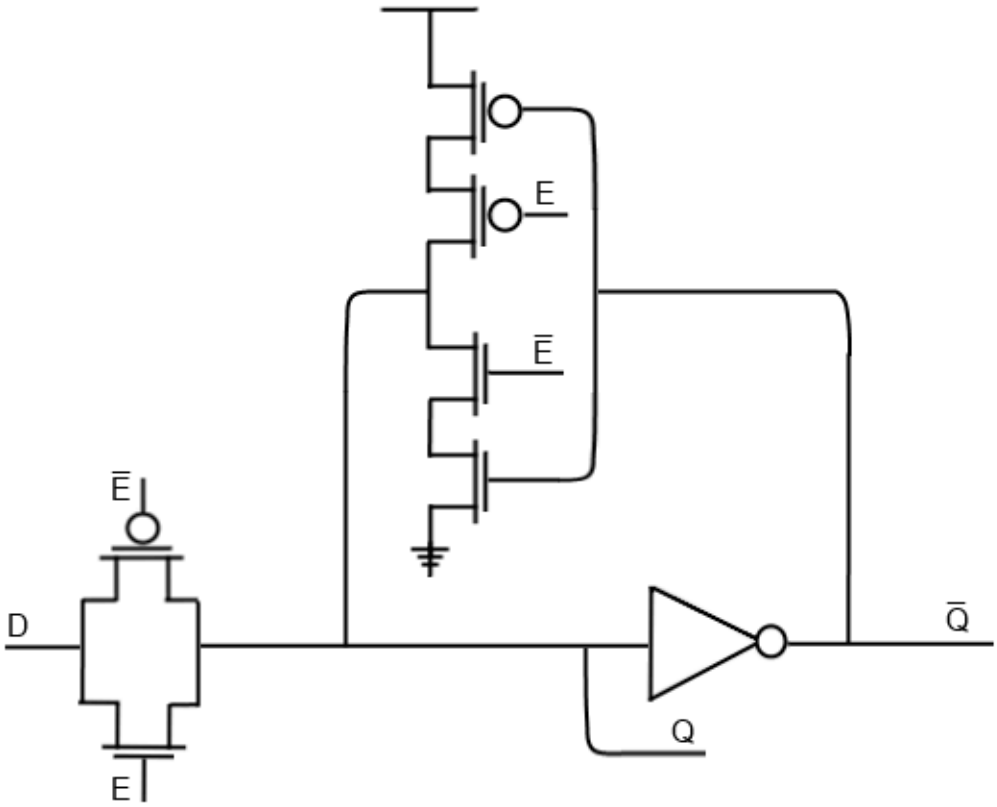

Figure 4.11: PowerPC 603 Latch

Here, the data signal is enabled through a CMOS transmission gate and applied on the first inverter while the clocked inverter is disabled. As the enable signal E goes low again, the inverter circle is re-enabled, and the input is stored on the latch.

4.7 D-Flip Flop

A D-Flip Flop (DFF) consists of a master latch and a slave latch. They are both controlled by a clock signal that enables each latch (high clock for master, and low clock for slave) to store the input data signal (figure 4.12).

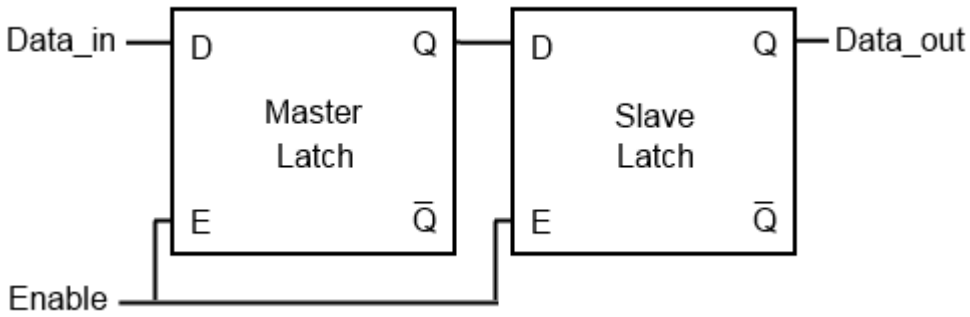

Figure 4.12: Block diagram, DFF

As mentioned earlier, the DFF of the state machine needs to be forced low when the reset signal goes high. A modified version of the DFF for the state machine can be seen in figure 4.13.

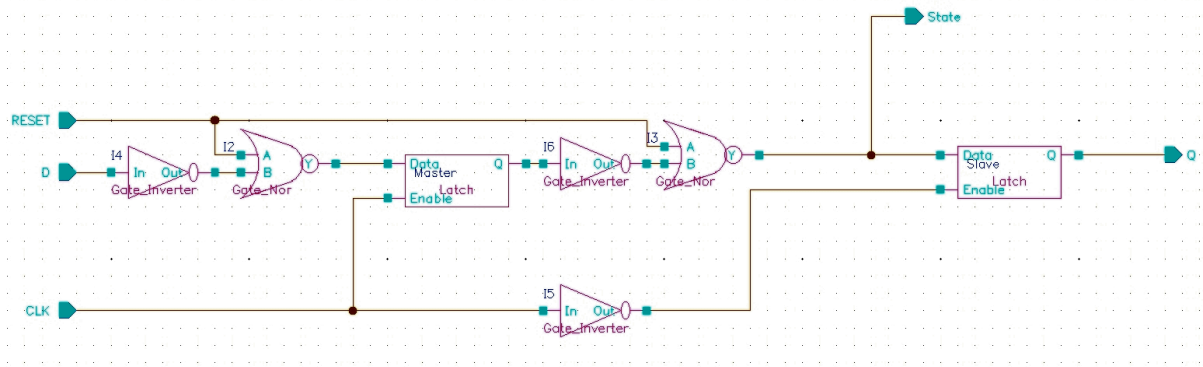

Figure 4.13: Block Diagram, DFF with Reset

A similar modification is needed for the sample generator, although it will be force high instead of low. The sample generator design is shown in figure 4.14.

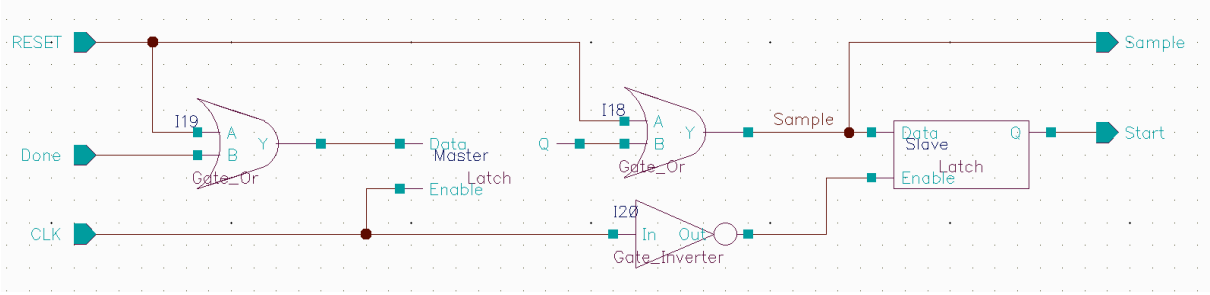

Figure 4.14: Sample generator

4.7.1 C²MOS DFF

Figure 4.15 shows the C²MOS design implemented as a DFF.

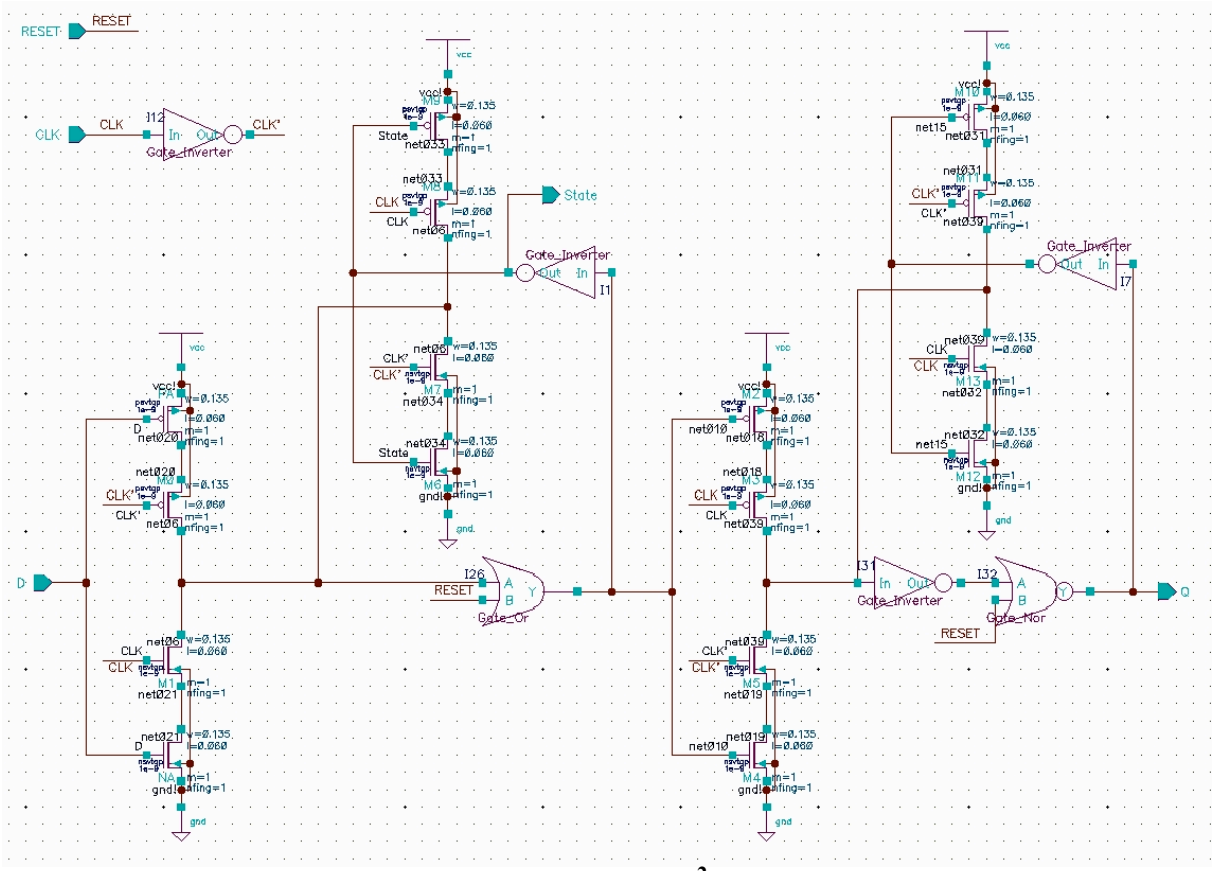

Figure 4.15: Schematic, C²MOS Latch

Extra gates have been added for reset purposes. Since each of the two latches invert the signal, the reset functionalities have to be different. In the first latch (master) the rest must be forced high, while on the second latch (slave) the reset must be forced low.

4.7.2 PowerPC 603

Figure 4.16 shows the PowerPC 603 design implemented as a DFF.

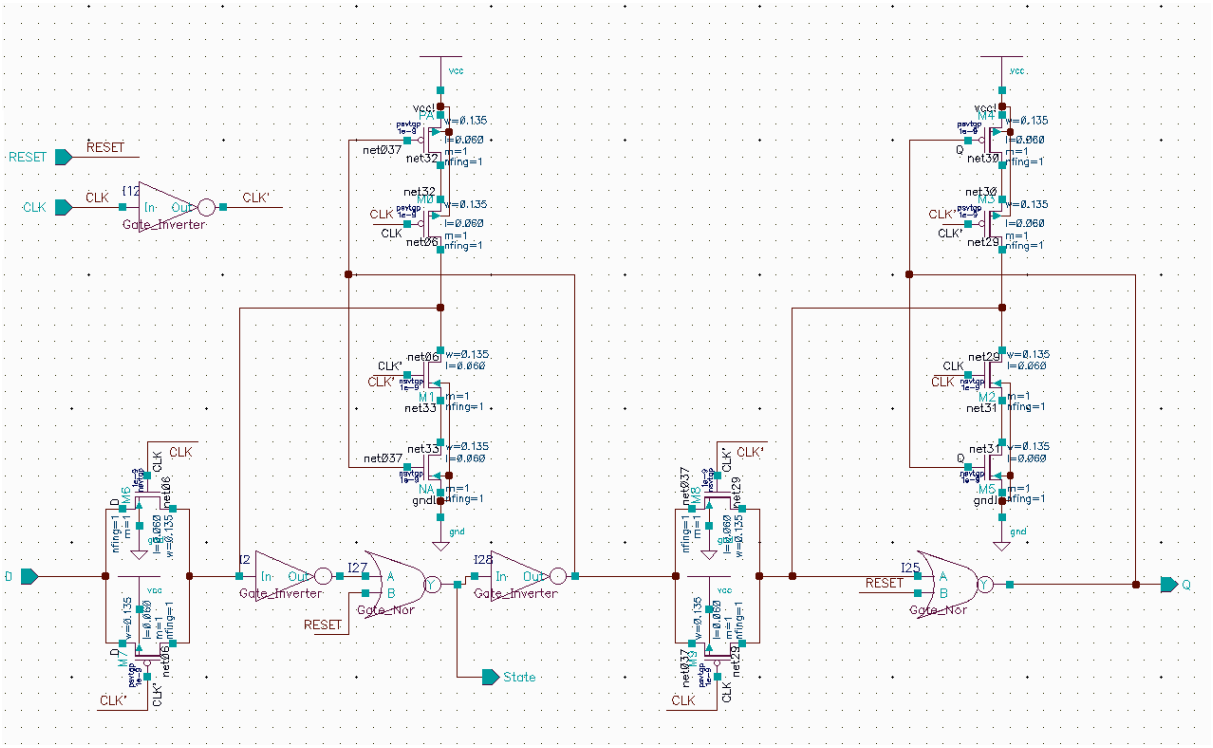

Figure 4.16: Schematic, PowerPC 603 Latch

The reset circuitry is implemented somewhat different from the C^2MOS, but he logical functionality is exactly the same.

4.8 Logical Gates

Logical gates are circuits that exhibit a very simple logical behavior consisting of only a few transistors. With them, one can design larger and more complex modules and systems. It is very important to understand them in order to know where the soft-supply inverter can be utilized.

4.8.1 Inverter

The inverter has one input and one output. A schematic on transistor level is shown in figure 4.17.

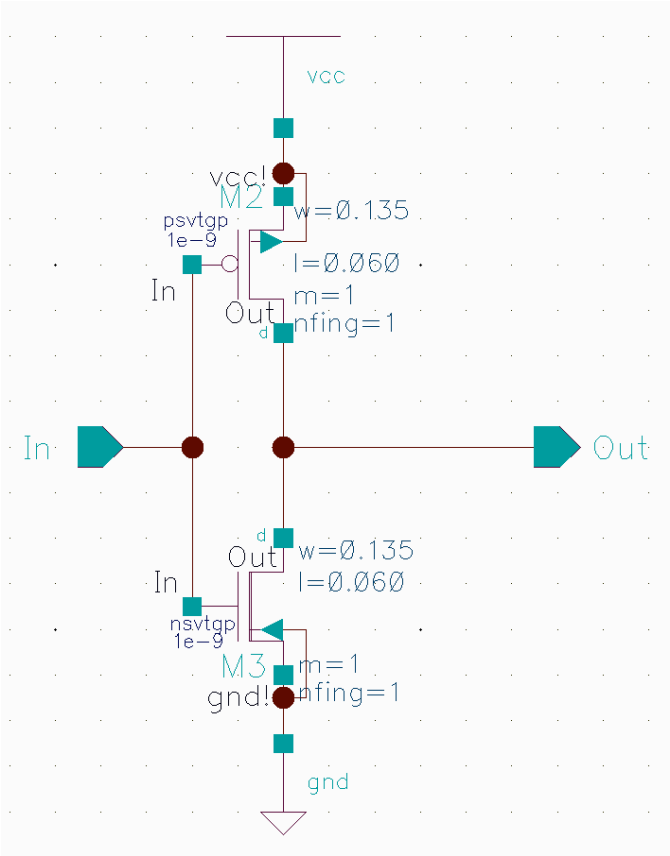

Figure 4.17: Schematic, Inverter

The logical behavior given in table 4.4 shows that the input signal is “inverted” meaning that if it is either set low-to-high or high-to-low.

Table 4.4: Behavior of the Inverter Gate

In	Out
0	1
1	0

4.8.2 Clocked Inverter

The clocked inverter works almost the same as the standard inverter, except it has an enable signal that either enables a regular inversion or disables the gate entirely, leaving only a high resistance to the output.

This version has two inputs and one output. A schematic on transistor level is shown in figure 4.18.

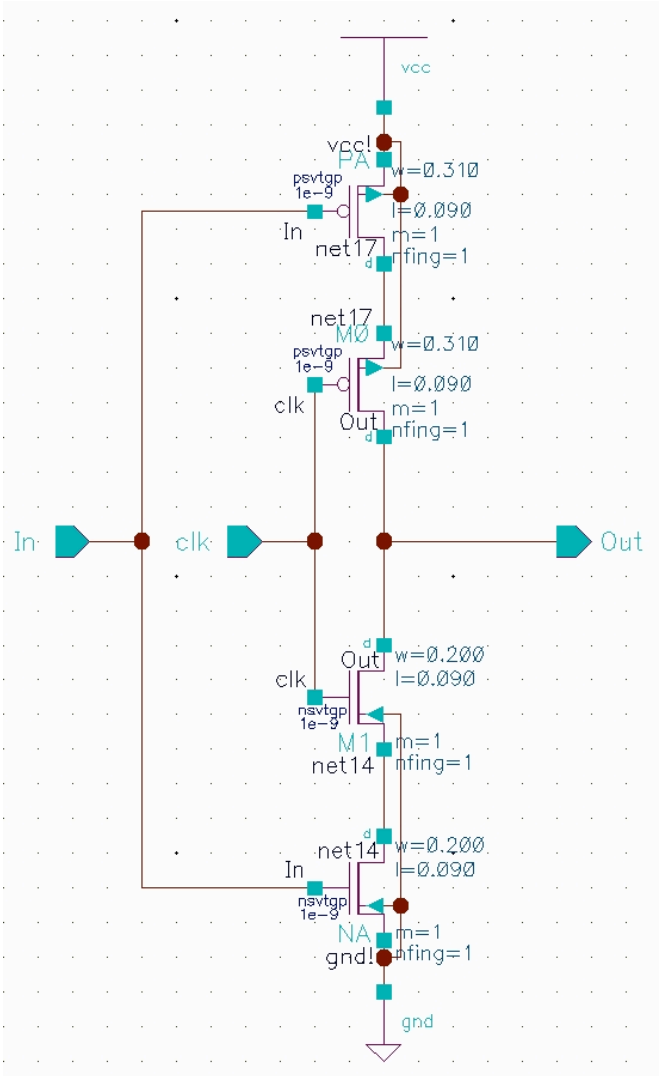

Figure 4.18: Schematic, Clocked Inverter

The logical behavior given in table 4.5.

Table 4.5: Behavior of the Clocked Inverter Gate

Clk	In	Out
0	0	-
0	1	-
1	0	1
1	1	0

4.8.3 NOR-Gate

A NOR-gate has two inputs and one output. A schematic on transistor level is shown in figure 4.19.

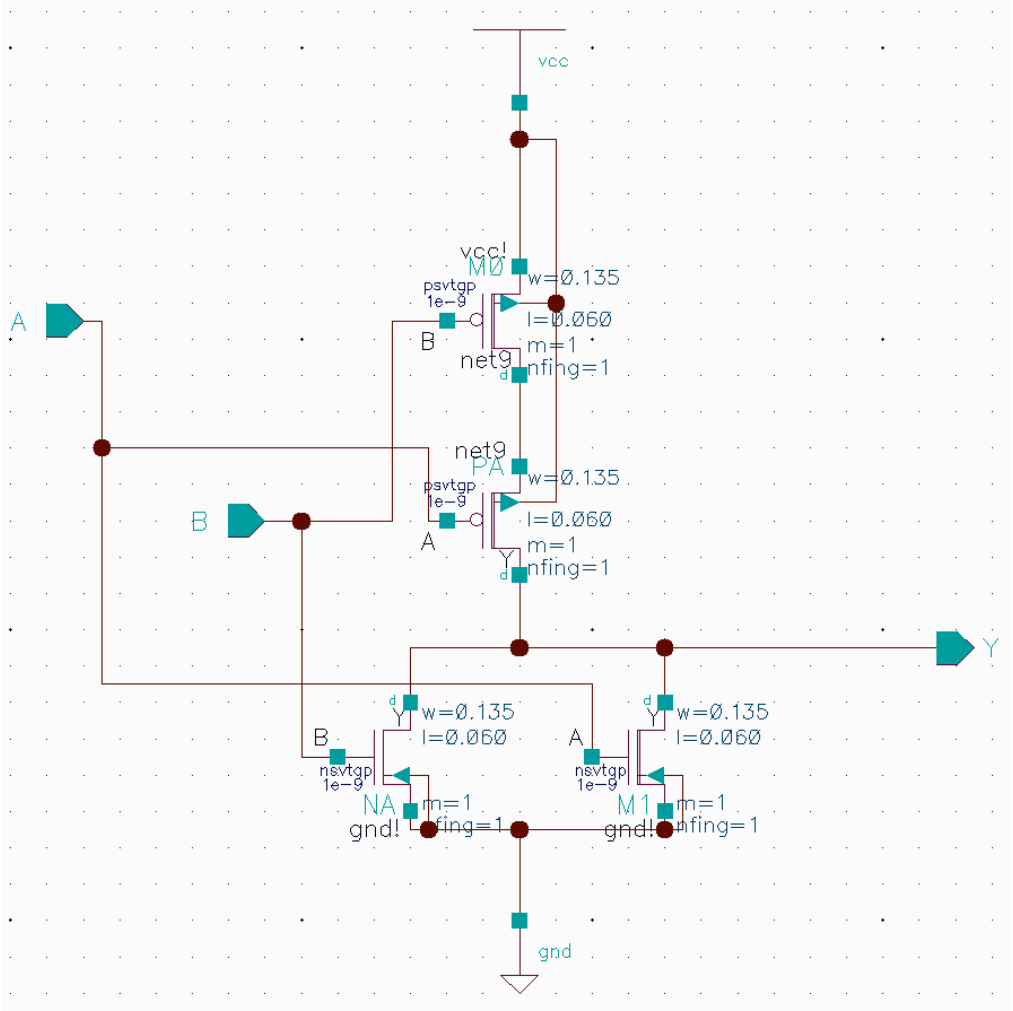

Figure 4.19: Schematic, NOR-Gate

The logical behavior given in table 4.6 shows that how the output changes depending on the inputs.

Table 4.6: Behavior of the NOR-Gate

A	B	Out
0	0	1
0	1	0
1	0	0
1	1	0

The name “NOR” is a shortening of Not-OR, as in how the output is not high whenever either A or B is high. It can also refer to how the output is high only when neither A NOR B are high.

4.8.4 NAND-Gate

A NOR-gate has two inputs and one output. A schematic on transistor level is shown in figure 4.20.

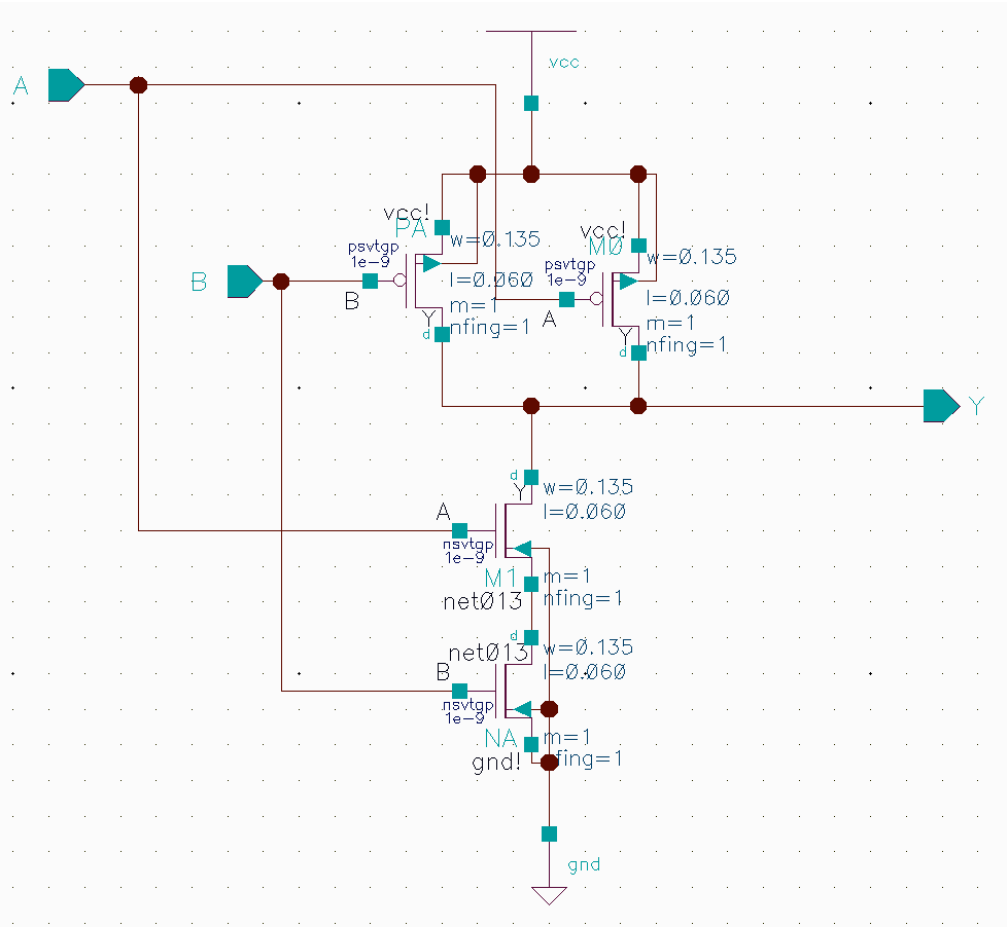

Figure 4.20: Schematic, NAND-Gate

The logical behavior given in table 4.7 shows that how the output changes depending on the inputs.

Table 4.7: Behavior of the NAND-Gate

A	B	Out
0	0	1
0	1	1
1	0	1
1	1	0

The name “NAND” is a shortening of Not-AND, as in how the output is not high whenever both A and B are both are high.

4.8.5 Other Gates

Other logical behaviors can be designed out of these three basic gates. The most common among them are AND and OR gates. They can be used by taking a NAND or and NOR gate, respectively, and placing an inverter at the output.

As can be seen from the tables 4.6 and 4.7, the AND output will be high when both A *and* B are high, while the OR output will be high when either A *or* B is high.

4.9 CMOS Transmission Gates

The CMOS transmission gate is a switch consisting of a NMOS and PMOS in parallell, and an inverter to make the two transistors turn on or off simultanously (figure 4.21).

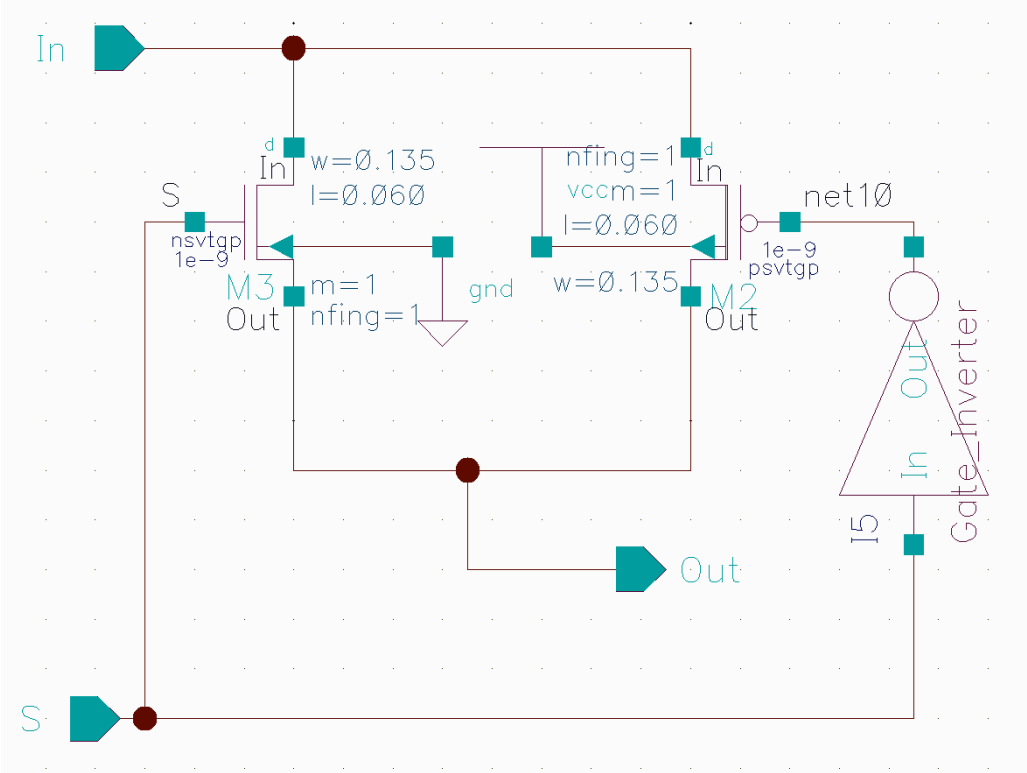

Figure 4.21: Schematic, CMOS Transmission Gate

It works almost like a clocked inverter, except it enables a signal through, when the switch signal S is high, as can be seen from table 4.8.

Table 4.8: Behavior of the CMOS Transmission Gate

S	In	Out
0	0	-
0	1	-
1	0	0
1	1	1

5. Soft-Supply Inverter

The Soft-Supply Inverter (SSI) is a new logical gate presented in this report. Based on the previous descriptions of the modules that makes up the SAR ADC, it will be attempted to find areas where this design can be used in order to compare the different results with regard to area and power consumption.

5.1 Idea

The design is based on regular inverter except that the supply voltage is used as a variable input signal, hence the name of the circuit. The only problem is that when the supply voltage is lower than the threshold voltage to the PMOS transistor, the output sees only a high resistance (see figure 5.1).

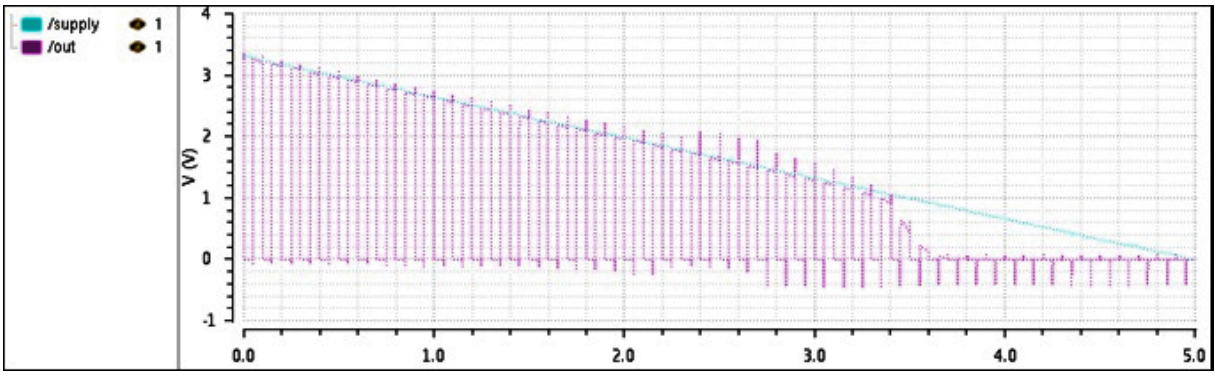

Figure 5.1: Inverter with a variable Supply Voltage

A way to solve the issue would be to change out the PMOS transistor with a CMOS transmission gate. The result is shown in figure 5.2 below.

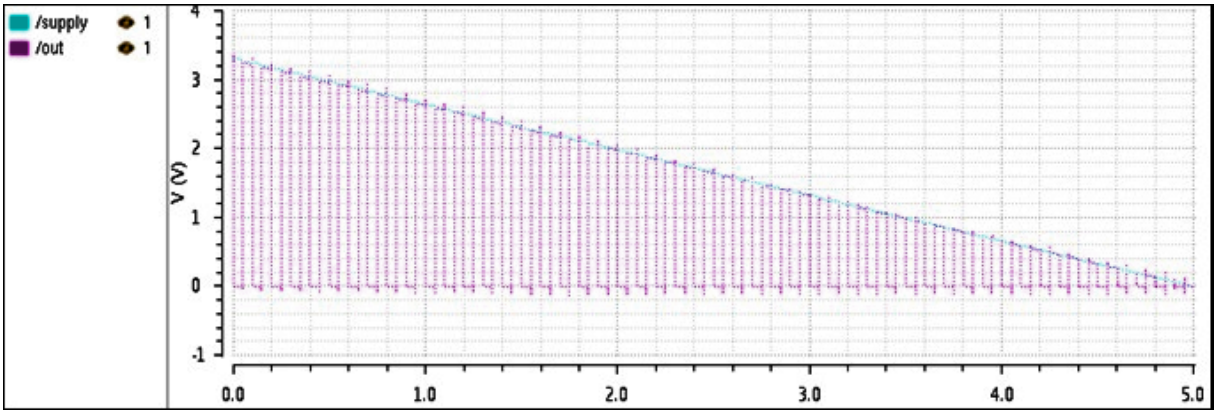

Figure 5.2: Soft-Supply Inverter

Thus, it should be possible to use the circuit as a logical gate.

5.2 Design

Figure 5.3 shows the schematic of the circuit on transistor level.

When the input A is high, it forces the output to ground at the same time as it closes the transmission gate. Whenever A is low, the output receives its signal from B through the open gate.

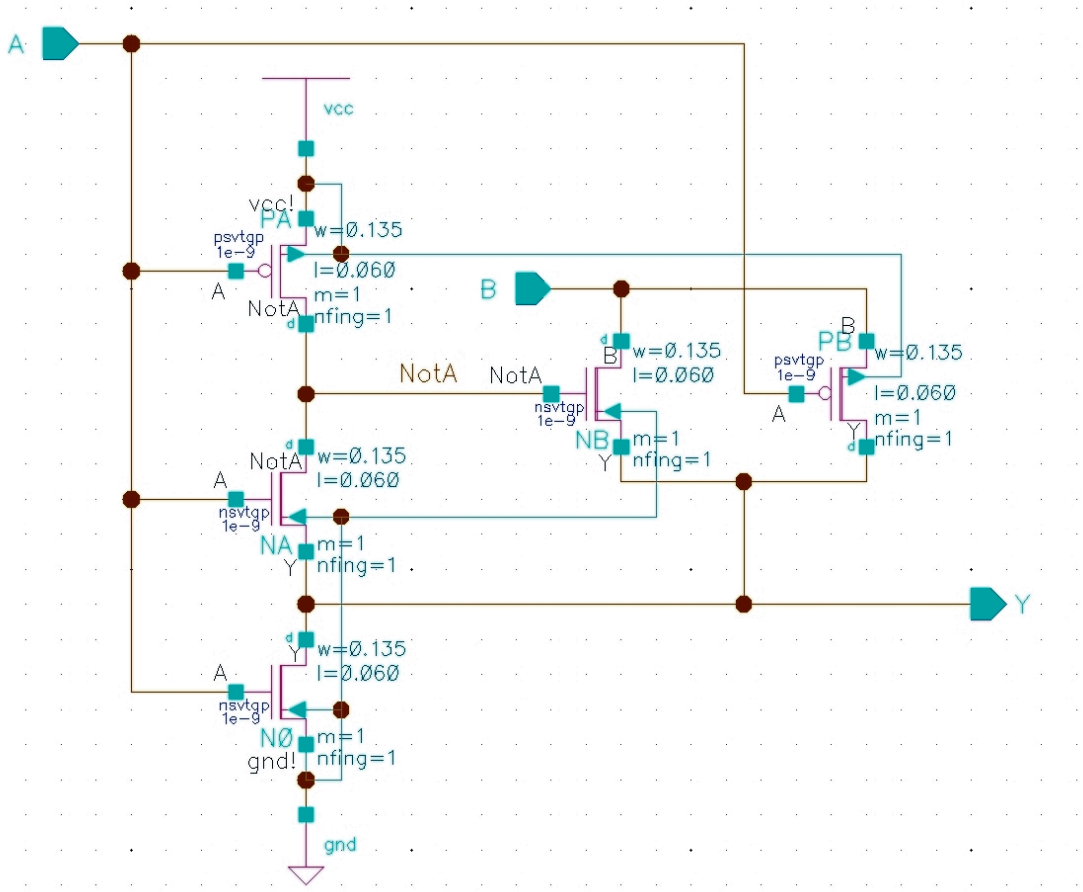

Figure 5.3: Schematic Soft-Supply Inverter

Here the inverter for signal A shares the same ground as the output's NMOS transistor. This should reduce the current leakage from supply, although it also makes the circuit slower.

The truth table is shown in table 5.1.

Table 5.1: Truth table, Soft-Supply Inverter

A	B	Y
0	0	0
0	1	1
1	0	0
1	1	0

As can be seen from the truth table, its behavior is the same as a NOR gate having one of its inputs inverted (see figure 5.4). From this simple logic behavior, it will be investigated where it can be applied and what possible improvements it may yield.

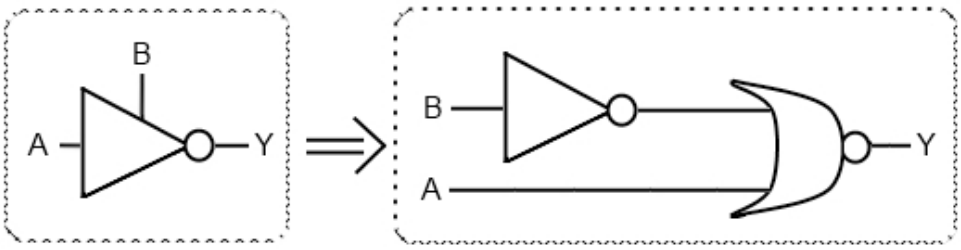

Figure 5.4: Soft-Supply Inverter, Model

The SSI uses one less transistor – five transistors instead of six transistors – which could result in smaller area and less power consumption. The design's schematic has been simulated as a part of this work, and both static and dynamic power consumption have been compared to the conventional design. The entire simulation procedure can be found in appendix A. The result was that the soft-supply inverter consumes 25% less static power, but that it also consumes more than double the dynamic power. This means that it might be well suited for a circuit with a slow clock, such as the SAR ADC implemented in this report.

Another discovery was that the soft-supply inverter consumes a rather high amount of power when both of its input signals are high, compared to other combinations. Thus, one should implement it in such a way that one avoids those situations as much as possible.

Another version of the soft-supply inverter (a "soft-ground" inverter) was also designed and simulated, but the revealed power consumption was too high, compared to the conventional design, that it was not of any interest to experiment more with it. The results of these simulations can be seen in appendix A.

5.3 SR-Latch

To see if it is possible to make an alternate version of the SR-Latch using the soft-supply inverter, the circuit in figure 5.5 is given blow.

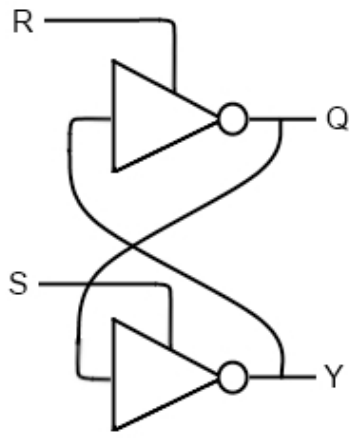

Figure 5.5: SR-Latch, SSI Design

The extended truth table for the new SR-Latch is shown in table 5.2.

Table 5.2: Extended Truth table for SR-Latch, New Design

R	S	Q	Y	Q_{NEXT}	Y_{NEXT}
0	0	0	0	**0**	**0**
0	0	0	1	**0**	**0**
0	0	1	0	**0**	**0**
0	0	1	1	**0**	**0**
0	1	0	0	**0**	**1**
0	1	0	1	**0**	**1**
0	1	1	0	**0**	**0**
0	1	1	1	**0**	**0**
1	0	0	0	**1**	**0**
1	0	0	1	**0**	**0**
1	0	1	0	**1**	**0**
1	0	1	1	**0**	**0**
1	1	0	0	**1**	**1**
1	1	0	1	**0**	**1**
1	1	1	0	**1**	**0**
1	1	1	1	**0**	**0**

To determine the behavior of this circuit, one considers what values Q and Y settle on when the input signals R and S change their value once. First of all, one sees that as long as both R and S are low, both Q and Y will also be forced low regardless of their initial value. Secondly, when R is low and S is high, Q will be forced low and Y will be forced high. Third, when R is high and S is low, Q will be forced high and Y will be forced high.

The last input combination, when both R and S are high, is not as trivial. One sees that in the case when Q and Y were equal, the circuit starts oscillating. However, if they were not, they will keep their previous values regardless of what they were.

The shortened down truth table derived from the extended truth table above is shown table 5.2 below.

Table 5.2: Truth table of SR-Latch, New Design

R	S	Q_{NEXT}	Y_{NEXT}
0	0	0 (Reset)	0 (Reset)
0	1	0	1
1	0	1	0
1	1	Q*	Y*

***When Q and Y are not equal.**

In conclusion, the circuit functions correctly as an SR-Latch as long as the outputs Q and Y remain unequal, that is if the inputs are never equal to zero at the same time. This is the same situation as in the conventional SR-latch shown previously.

5.4 D-Latch

An equivalent circuit of the one presented in figure 4.7 is shown in figure 5.6 below.

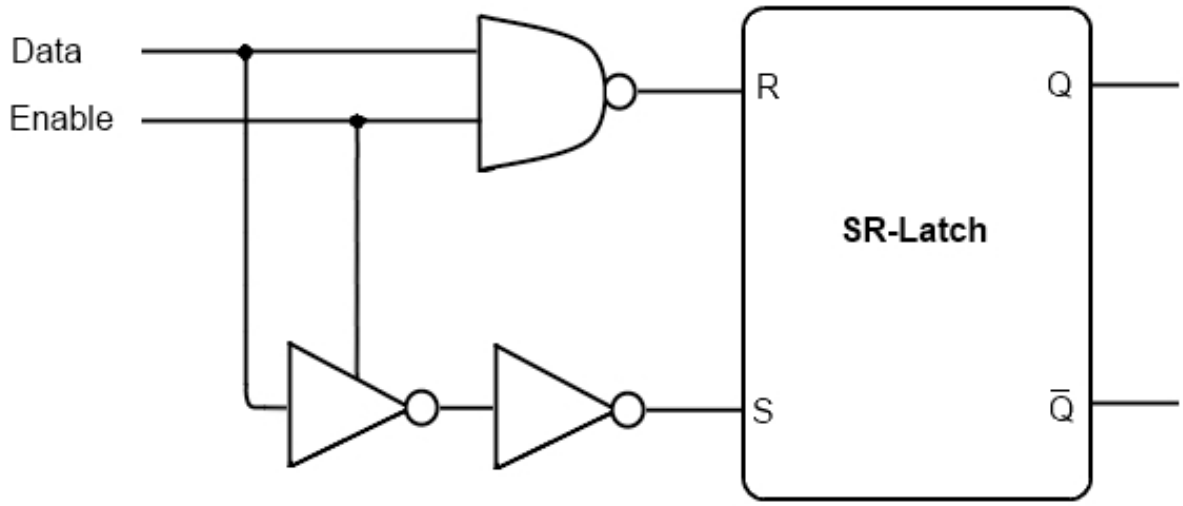

Figure 5.6: D-Latch, SSI Design I

The truth table is shown in table 5.3.

Table 5.3: Truth Table for D-Latch, SSI Design

E	D	R	S	Q
0	0	1	1	Q
0	1	1	1	Q
1	0	1	0	1
1	1	0	1	0

When comparing the two truth tables 4.3 and 5.3, one sees that they are almost identical. The only exception is that the new design stores the opposite value of Data. This can be easily fixed by letting Y, which is the same as Q(not), be connected to the output instead of Q.

So far, it seems that the transparent design of the D-Latch is better in terms of area. The new version of the SR-Latch uses two more transistors, and the extra logic for the D-Latch uses five more transistor. However, as we will see, it is possible to remove some components from redundance.

If one recalls the logical equality in figure 5.4 to change out the bottom SS-Inverter in figure 5.5 with the standard design, one can remove the inverter one receives from that exchange together with the inverter in figure 5.6. The result can be seen in figure 5.7.

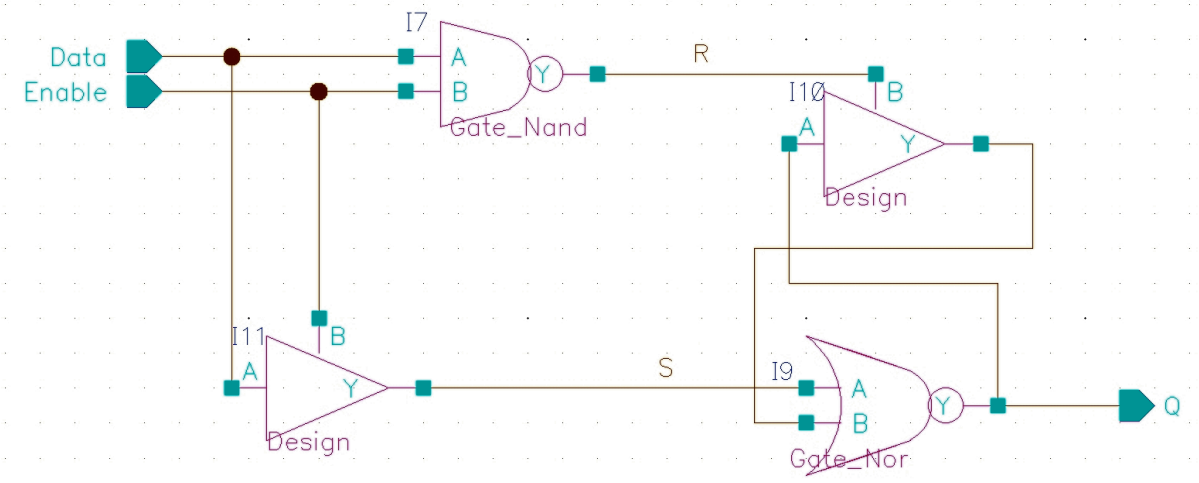

Figure 5.7: D-Latch, SSI Design II

By counting the transistors used by the gates in figure 4.8 and 5.7, one can see that the SSI design uses now only two more transistors than the transparent design.

5.5 D-Flip-Flop

Seen from the SS-inverter model in figure 5.4, the change in the standard DFF with reset circuitry in figure 4.13 is pretty straight forward. The result can be seen in figure 5.8 below.

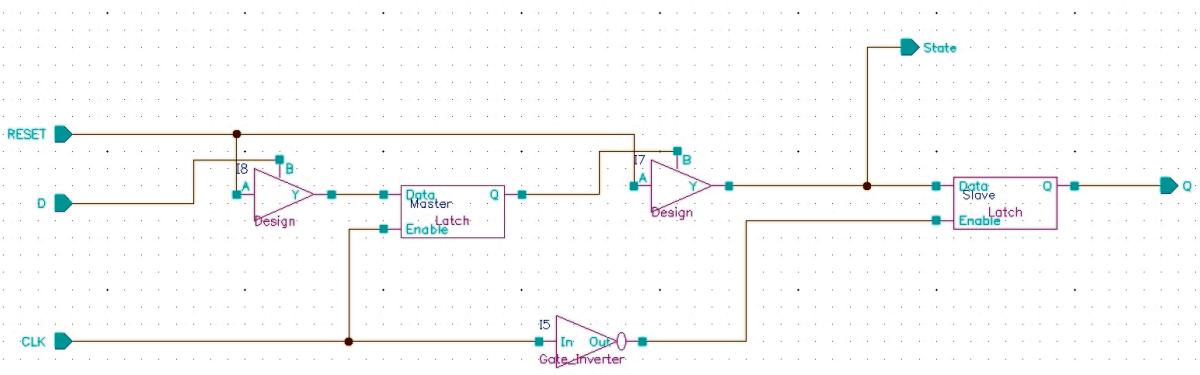

Figure 5.8: D-Flip-Flop, SSI Design

5.5.1 C^{2}MOS DFF v2

The reset circuitry in the slave latch has changed slightly from the one in figure 5.9.

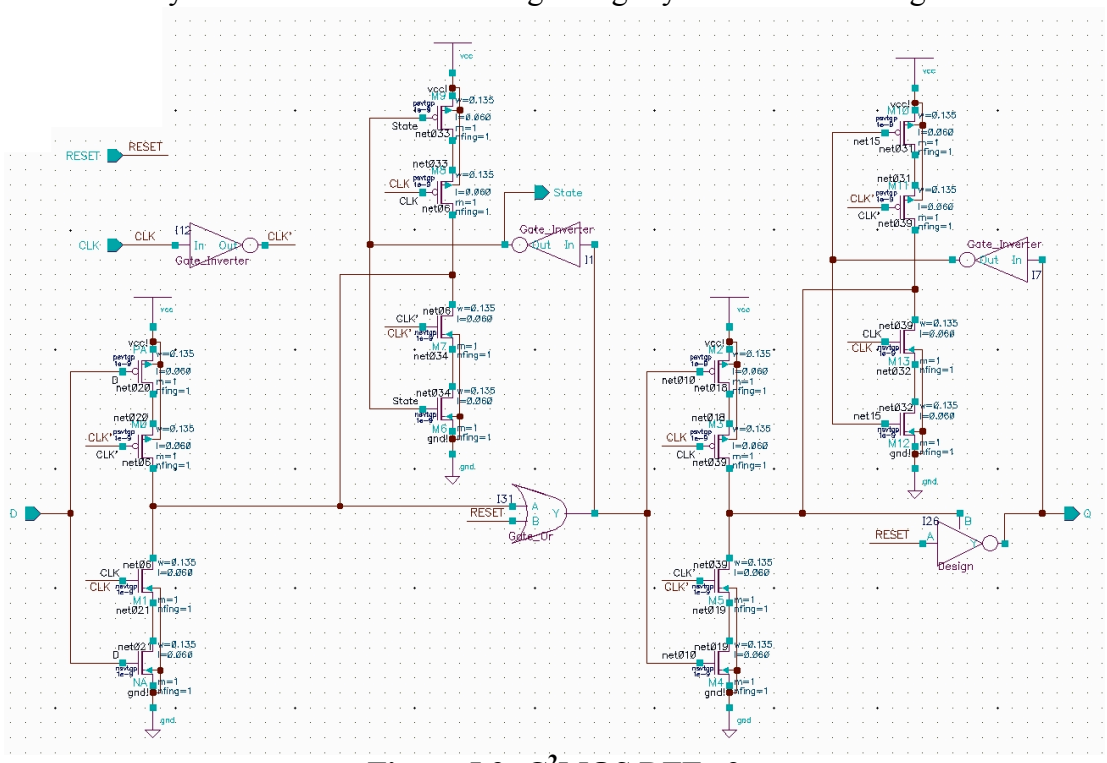

Figure 5.9: C^{2}MOS DFF v2

5.5.2 PowerPC 603 v2

The reset circuitry in the master latch has changed slightly from the one in figure 5.10.

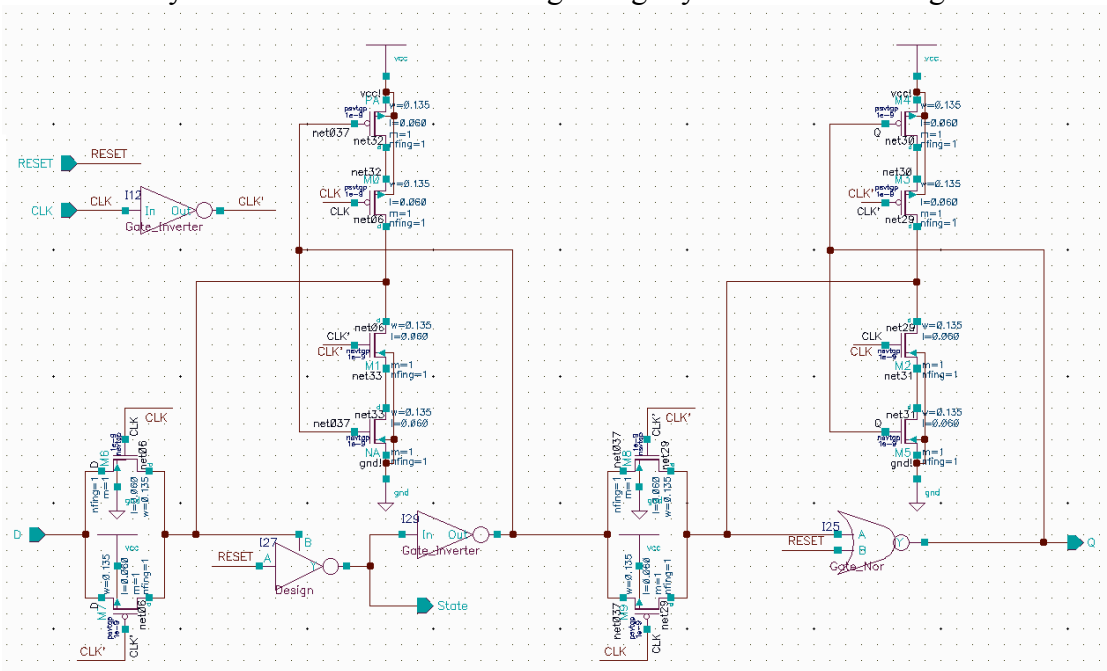

Figure 5.10: PowerPC 603 DFF v2

5.6 Mux, 4-1

The 4-1 mux shown in figure 4.5 works by enabling the transmission gates through each of its NOR-gates by routing the two control signals through two inverters. In a similar fashion, it is possible to enable the transmission gates using the NOR-gate, two SSIs and one AND-gate; this eliminates the need for the two inverters. The result can be seen in figure 5.11.

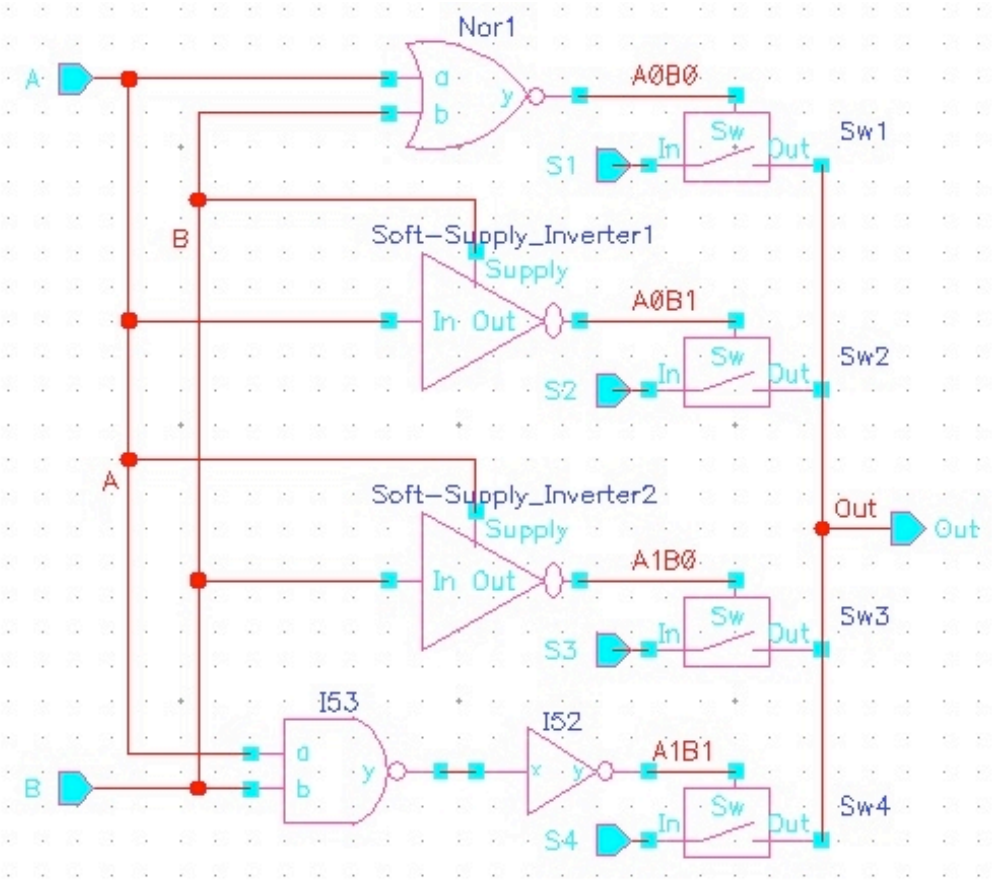

Figure 5.11: 4-1 Mux, SSI

6. Signal Mapping

In this chapter, the signal mapping of the 4-1 mux will be discussed based on the properties of the SSI and the logic module designed so far.

6.1 Mux, 4-1 – Signal Mapping

Now as the 4-1 mux has been designed, it is only a matter of selecting where the input signals to the DAC should be placed. An important to note is that the placement of the four signals to the DAC – those that will be enabled through the transmission gates in the mux – will decide what state the mux will stay in the longest. Recollecting that the SSI consumes a lot more power when both inputs are high, one should place the sampling voltage at S4 since sampling is done only one-tenth of the time during a conversion.

The other three signals should be placed in a way that simplifies the switch control logic; this is done with gray-coding. It would be easy to change the signals from S1 (A = 0, B = 0) to either S2 (A = 0, B = 1) or S3 (A = 1, B = 0), as that would require a change in only a single bit. This is done when a switch changes from vcm to either vref or gnd.

The resulting signal mapping is shown in table 6.1.

Table 6.1: 4-1 Mux Input Signal Mapping

S#	Signal	(A, B)
S1	VCM	(0, 0)
S2	GND	(0, 1)
S3	VREF	(1, 0)
S4	Vinn / Vinp	(1, 1)

Thus, the DAC will be implemented with the muxes as shown in figure 6.1.

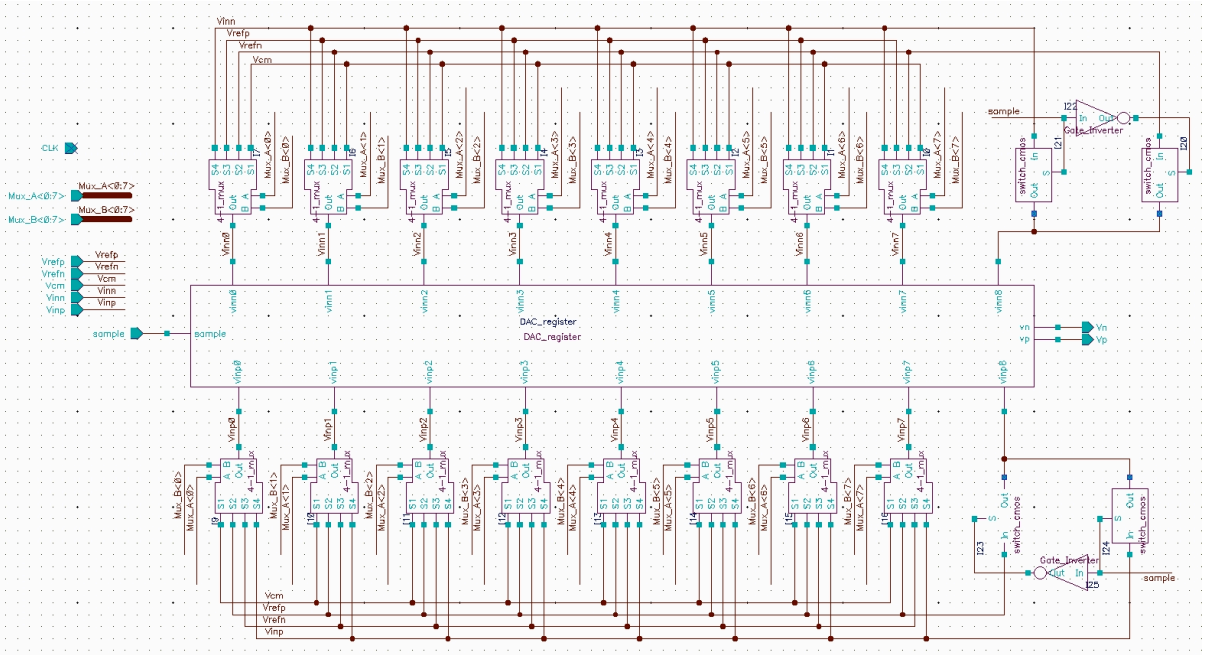

Figure 6.1: DAC and Muxes

6.2 Switch Control

Now, as the 4-1 mux input signal mapping is known, the logic to the switch controls can be designed. Table 6.2 shows the relationship between the input signals and the output signals.

Table 6.2: Switch Controls Input / Output Relationship

Start	State	Data	Mux state	Output
0	-	-	Vinn / Vinp	A = 1, B = 1
1	0	-	Vcm	A = 0, B = 0
1	1	0	VREF	A = 1, B = 0
1	1	1	GND	A = 0, B = 1

Whenever the converter is sampling, the DAC must be put in sampling mode by continuously applying the input signals onto the capacitors, regardless of previously stored bits in the register. Then, when the converter begins converting, the muxes must switch the voltage to the Vcm to store the sampled input voltage.

As the SAR ADC keeps calculating the different bits throughout a conversion, each and every mux must independently change the voltage applied to its own capacitor in order for the DAC to perform the binary search algorithm. The *state* variable in table 6.2 is considered '1' when the given mux has either reached its state or exceeded it during a conversion. The OR-gates in figure 6.2 ensures this functionality.

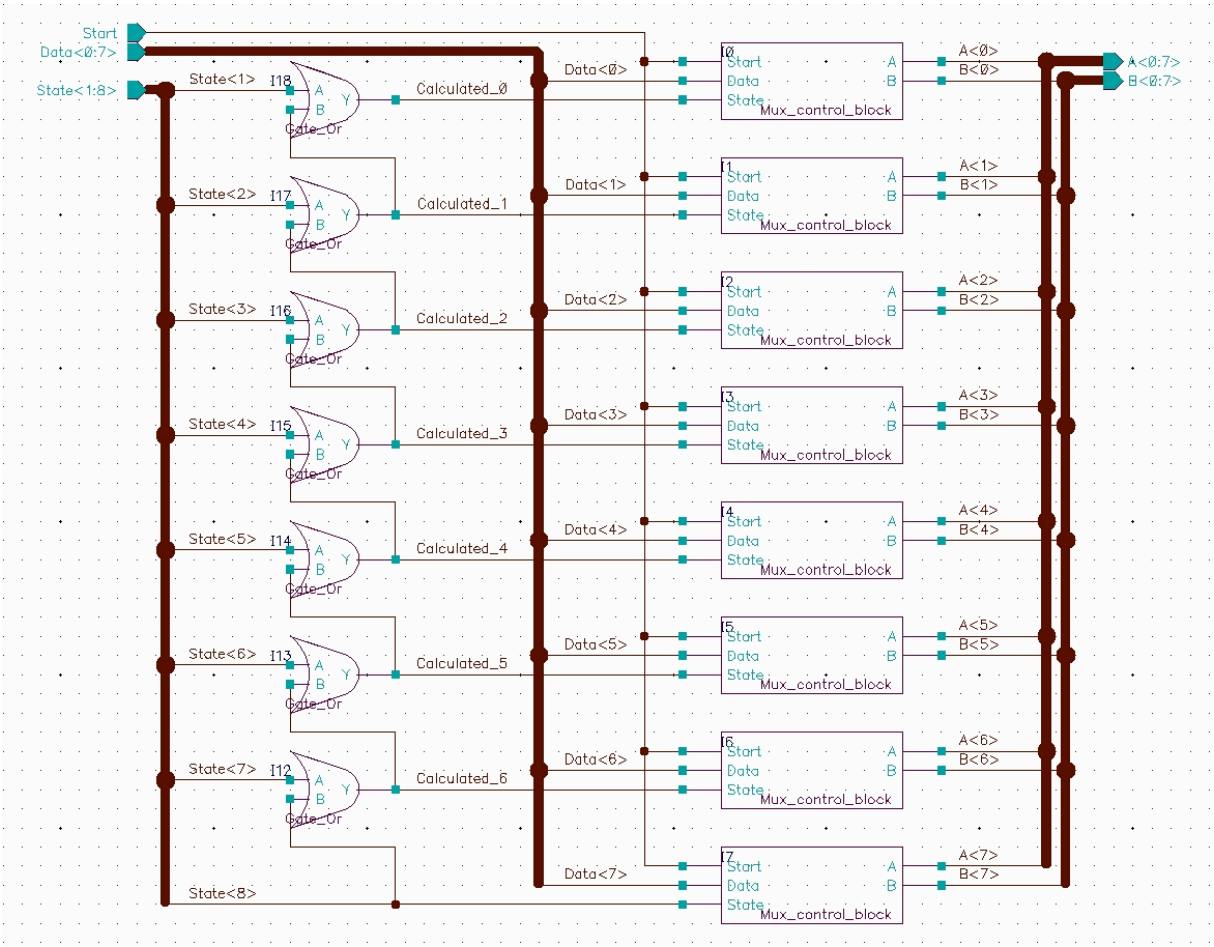

Figure 6.2: Mux Control Design

The design for the Mux Control Block is shown in figure 6.3.

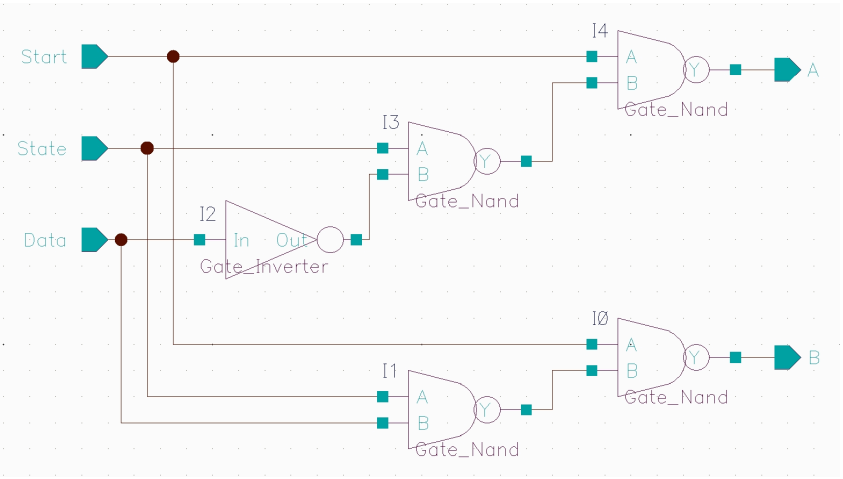

Figure 6.3: Design Mux Control Block

7. Simulations

All simulations with ADE-XL in Cadence version IC6.1.5.500.15. The modules were first simulated separately in order to ensure correct behavior. Afterwards, the entire SAR ADC was simulated to measure power consumption.

With the state machine and full SAR ADC, the simulations were also done using a 2-phase clock generator for the state machine to see much the power dissipation would change.

7.1 Mux, 4-1

The testbench for the mux is shown in figure 7.1. The four inputs w1 through w4 have the voltages 0V, 200mV, 400mV and 600mV respectively.

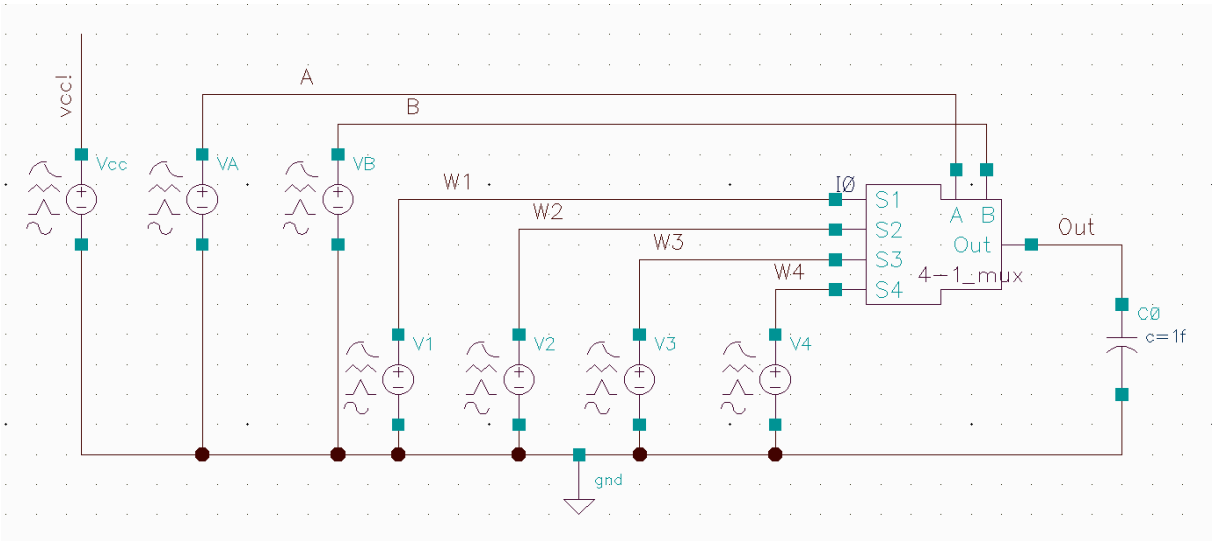

Figure 7.1: Testbench, 4-1 Mux

As the control signals A and B change, the output changes accordingly proving that the design works. The /Vcc/PLUS graphs in figure 7.2 and 7.3 shows the transient power consumption during the two different simulations.

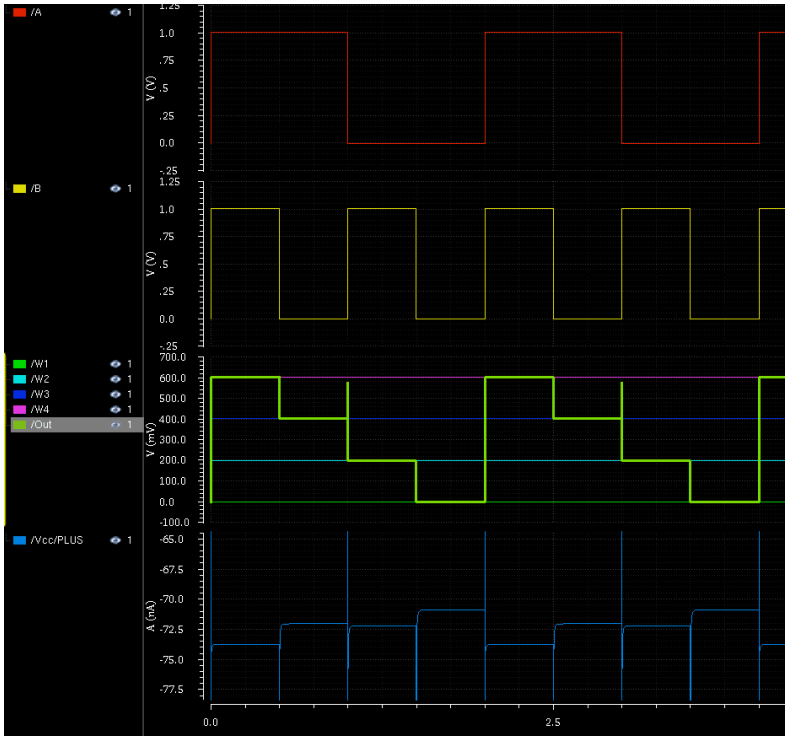

Figure 7.2: Simulation 4-1 Mux, Standard

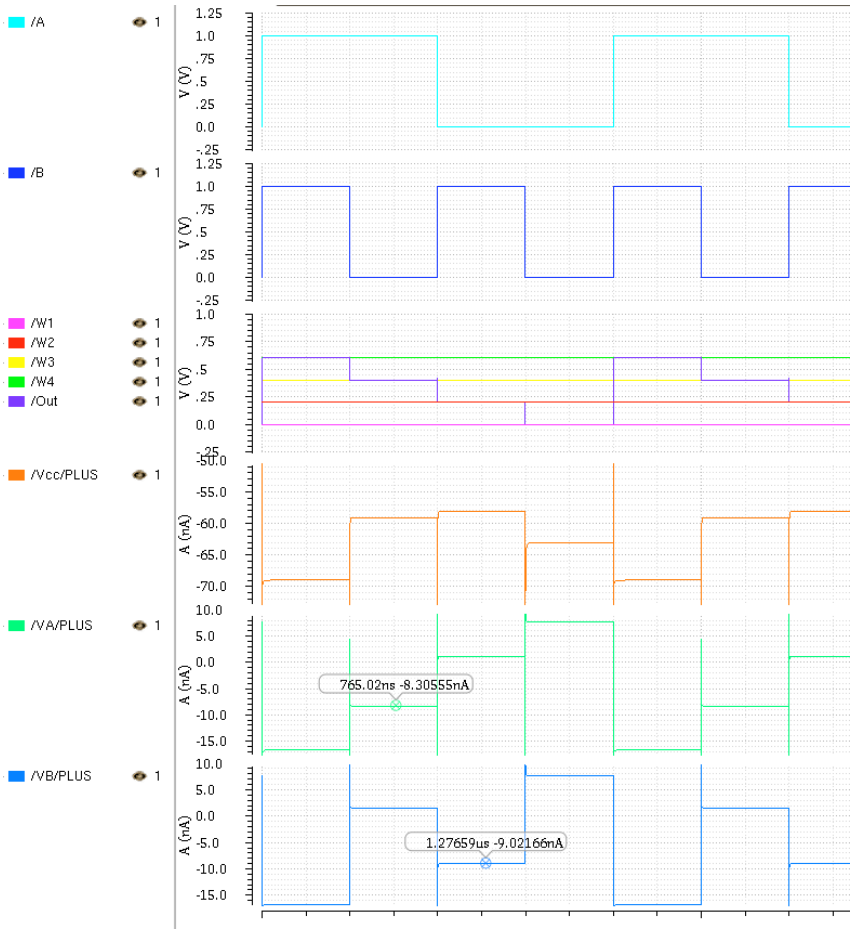

Figure 7.3: Simulation 4-1 Mux, SSI

7.2 State Machine

The testbench for the state machine is shown in figure 7.4 below.

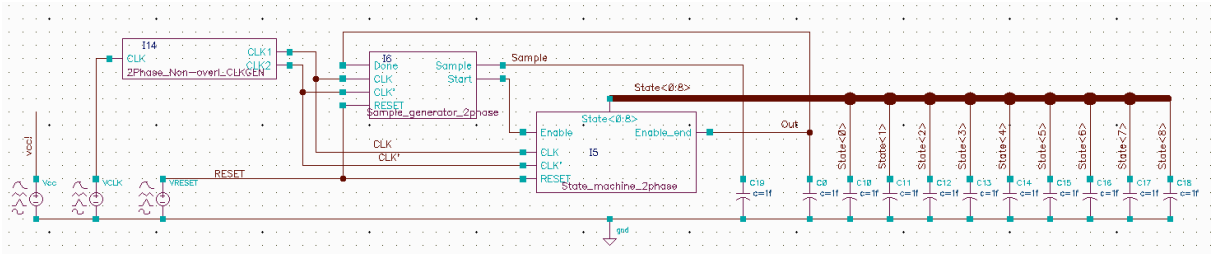

Figure 7.4: Testbench, State Machine

The signal graphs in figure 7.5 proves that the design works as intended. The signal that starts in the sample generator propagates through the DFFs until it reaches the final state and then returns back to the sample generator for another loop through the state machine.

It also shows that the system reacts correctly when the reset signal is applied during a conversion. Every DFF in the state machine is forced low while the sample generator is forced high. When the reset signal goes low again, the system starts from the beginning.

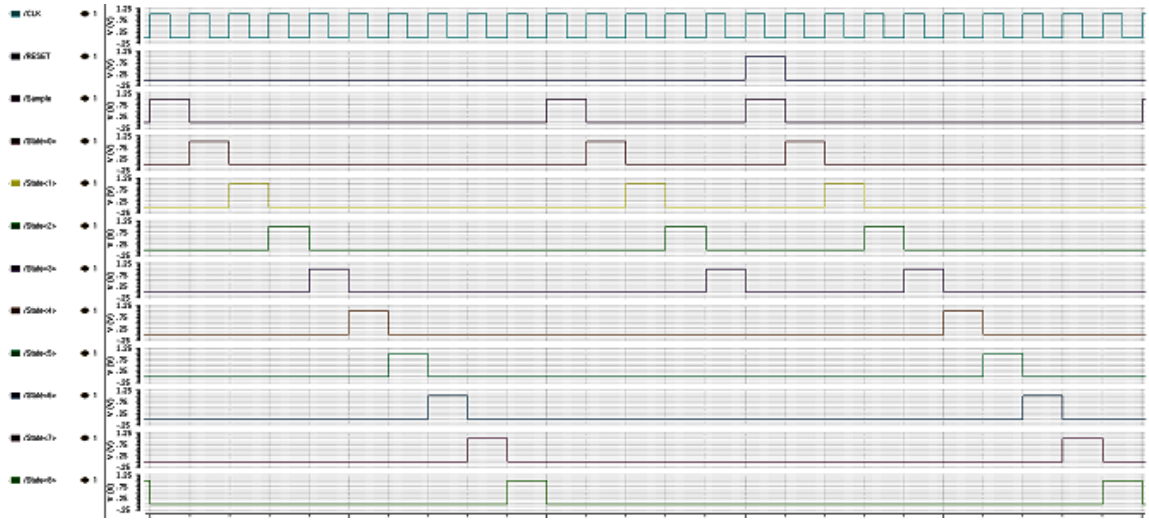

Figure 7.5: Simulation, State Machine

The same procedure was performed with all the different DFFs including both of the reset versions of C2MOS and PowerPC 603 and a 2-phase clock generator. The power consumption is listed in table 7.1.

Table 7.1: State Machine Power Consumption

	Power Consumption (nW)	
	Single clock	**2-Phase clock-gen**
Transparent	1158	1141
SS-Inverter	696.7	713.4
C^2MOS v1	733.8	714.3
C^2MOS v2	632.2	612.7
PowerPC 603 v1	519.3	499.8
PowerPC 603 v2	417	398.2

The different versions of C^2MOS and PowerPC 603 is that version 1 is with standard reset circuitry (figures 4.15 and 4.16), while version 2 is with SSI reset circuitry (figures 5.9 and 5.10). There is reduction in power dissipation with the use of SSI reset circuitry: 16% with C^2MOS and 24% with PowerPC 603.

7.3 SAR ADC

The testbench for the entire SAR ADC is shown in figure 7.6. The ideal DAC was used to recreate the original signal so that it could be used to calculate the ENOB with the spectrum tool in Virtuoso Visualization[18]. The ENOB was calculated to make sure that the modules were able to co-operate correctly with each other.

Also, an ideal ADC was used to compare with the performance of the SAR ADC. Since only the control logic was implemented on transistor level, both of these results should be equal.

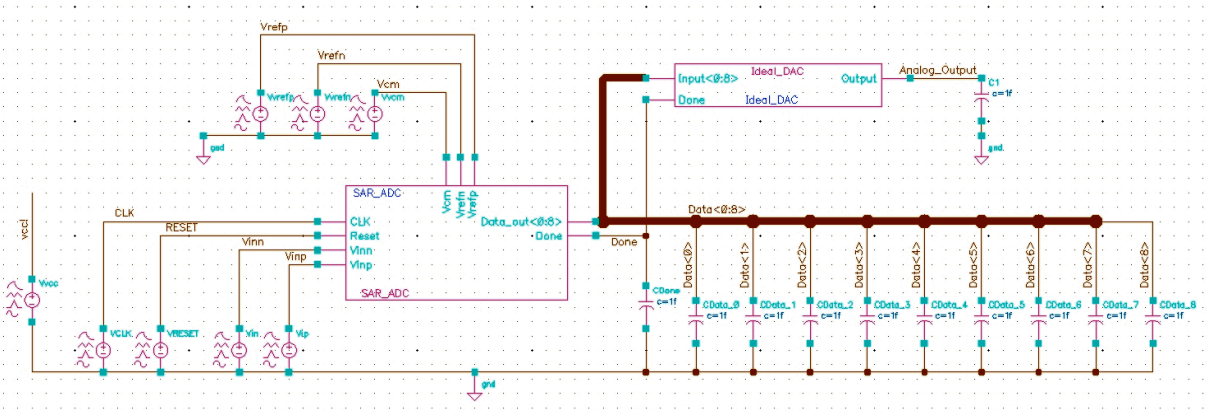

Figure 7.6: Testbench, SAR ADC

The simulation was done with coherent sampling using the variables presented in figure 7.7.

Figure 7.7: Variable Setup

Figure 7.8 is an exerpt from a full ENOB simulation showing a single conversion from 16ms to 17ms. In this particular situation, the sampled input signal is close enough to supply voltage that all the digital bits become high. As can be seen by the mux control signals A<0:7> and B<0:7>, only the B signals go high one by one while the A signals remain low, as expected.

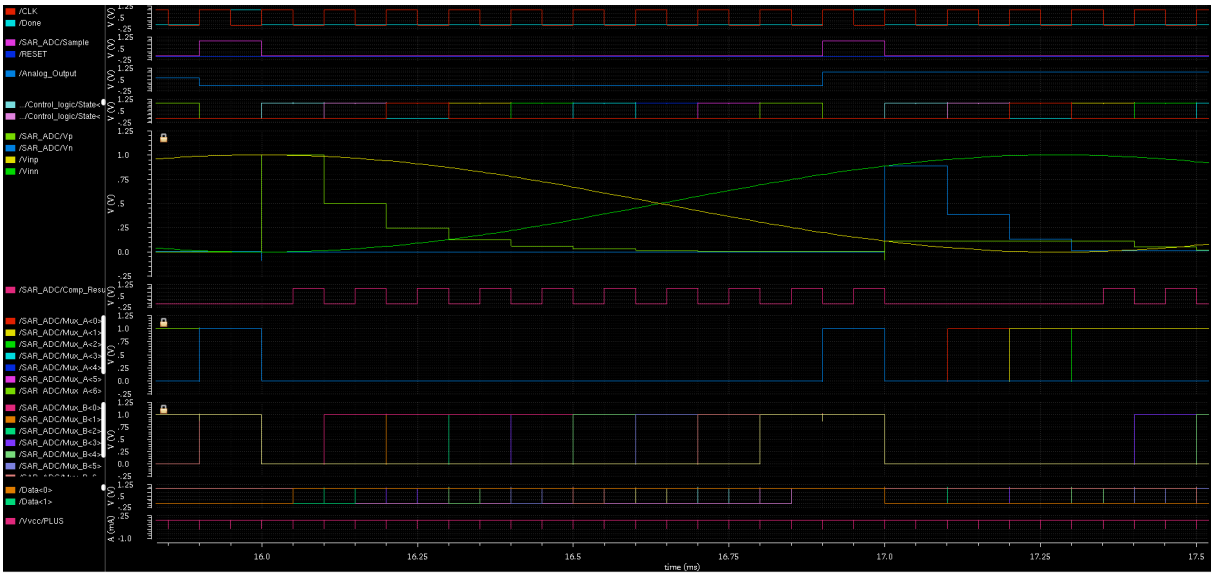

Figure 7.8: Simulation, SAR ADC

The SAR ADC was simulated with all the different D-Flip Flops and D-Latches both with and without the 2-phase clock generator. The results are shown in table 7.2.

Table 7.2: SAR ADC Power Consumption

	Power Consumption (nW)	
	Single clock	**2-Phase clock-gen**
Transparent	2,009	1.992
SS-Inverter	1.547	1.564
C^{2}MOS v1	1.501	1.482
C^{2}MOS v2	1.339	1.380
PowerPC 603 v1	1.286	1.266
PowerPC 603 v2	1.184	1.165

The output of the ideal ADC proved to be identical to the restored output of the ideal DAC. Also, the ENOB was calculated to be around 9 bit.

8. Discussion

This chapter discusses the chosen designs and the results presented in chapter 7 and from appendix A.

8.1 Soft-Supply Inverter

The simulations performed in appendix A revealed that the soft-supply inverter has a 25% decrease in static power consumption and a 100% increase in dynamic power consumption assuming an equal input signal combination. This presents an opportunity to design logical modules with reduced power consumption. However, since dynamic power consumption increases with frequency, it was decided use the design in a low-speed system.

Since the static power consumption when A and B are both '1' is much higher than the other combinations (almost three times larger than the second highest) it considered possible to reduce the power consumption further by avoiding this input combination as much as possible.

A great amount of work was put into figuring out where the gate might be utilized: from adders and counters to latches and switch controls. Because of the simple logical behavior, the gate could be used in virtually any logic design.

The similar design soft-ground inverter, however, was lacking in both areas and any further work was therefore considered void.

8.2 D-Latch

The latch that was designed with the soft-supply inverter has two more transistors than the transparent alternative. However, the SSI inside the SR-latch never has both of its input signals high. Also, the other SSI has almost never both inputs high, which only happens when the latch is writing a high bit value to the circuit. Seeing how each latch only writes once during an entire conversion, this reduces the power consumption a lot.

8.3 Reset Circuitry

Reset circuitry was applied to the DFFs of the sample generator and state machine. For some of the difference between the two reset circuitry designs was minimal, yet for C^2MOS and PowerPC 603 it had a about 14% and 20% further reduction in power consumption, respectively.

The slight increase in power consumption with the 2-phase clock generator for the SSI is a bit odd, especially when there was a slight decline for the other designs.

8.4 Mux, 4-1

The new 4-1 mux design uses the same number of transistors as the standard design, but the transient simulation still showed a slight reduction in power dissipation for the new design. Also, a strategical signal mapping was proposed to avoid the undesired high A and B to further decrease the overall power consumption, although this would have to be confirmed by simulating it as a part of the entire SAR ADC.

8.5 State Machine

The simulation results in table 7.1 shows that the SSI designs of the latches has a significant improvement to the standard design with a 40% reduction in power dissipation. However, other designs of the latch proves an even better reduction in power dissipation. Additionally, it can be argued that the difference between the SSI design and C^2MOS / PowerPC 603 designs could be even larger, because the inclusion of reset circuitry increases the transistor count and thus the power consumption.

8.6 Literature Study

The results from previous work [4], [11], [5], [12], [3] and [13] were discussed trying to find a current state-of-the-art SAR ADC. Based on Walden Field-of-Merit and total power consumption, both [4] and [11] are here considered state-of-the-art as Class-A SAR ADCs.

8.7 SAR ADC

The resulting ENOB of about 9 is caused by uncertainty because the calculations are based on probability. As already discussed in chapter 2, the ENOB can never be as high as the number of bits in the design (without oversampling). Nevertheless, it proved that the design works correctly which makes the measurement of power dissipation valid.

Because of time constraints, the 4-1 mux wasn't included in the SAR ADC simulations. However, with the promising results shown so far with the mux simulations, as well as the strategic signal mapping proposed in chapter 6, there are reasons to believe that it would reveal an even further reduction in power consumption compared to conventional design.

As with the state machine, the SSI design for the SAR ADC in table 7.2 shows a 23% reduction in power consumption, but an even further reduction with the other two alternatives. The reason it is not as high as with the state machine is that the SAR ADC also includes the switch controls which does not include SSI.

9. Conclusion

Simulations of the soft-supply inverter (SSI) revealed a 25% lower static power consumption and a 100% higher dynamic power consumption, compared to standard logic design. Since the particular input combination where both inputs on the SSI are high consume much more power than the other combinations, signals were mapped to avoid this as much as possible. Additionally, because of the high dynamic power, it was concluded that the gate would be best suited for low-speed applications.

A 1kS/s 9-bit SAR ADC was implemented based on the simplified control logic of [11]. New versions of the D-latch, D-Flip Flop (DFF), 4-1 mux and additional reset circuitry was designed using the SSI. In order to compare with alternative solutions, C^2MOS and PowerPC 603 latches were also used.

The digital modules were first simulated separately using Cadence version 6.1.5.500.15 and ADE-XL to confirm correct behavior. Then, the entire SAR ADC was simulated to ensure coherent co-operation between the modules by calculating the ENOB to near-ideal values using the spectrum tool in Virtual Visualization. Because of time constraints, the 4-1 mux was not included as a part of the SAR ADC, yet the separate simulation revealed a slightly reduced power dissipation.

The state machine simulation revealed a 40% reduction in power consumption with the use of the SSI latch and DFF. However, the C^2MOS and PowerPC 603 versions revealed an even further reduction by as much as 64%. Regardless, the reset circuitry for these alternate designs showed a 14% and 20% decrease in power consumption, respectively, when the SSI was applied. The total SAR ADC showed similar results, with the SSI providing a 23% reduction in power consumption, while the C^2MOS and PowerPC 603 reduced it by as far as 41%.

The results have shown an obvious advantage of the SSI over the standard logical design, although the latch design is somewhat lacking compared to C^2MOS and PowerPC 603. However, because of its simple logical behavior, the SSI can be used in virtually any logical circuit.

10. Suggestions to Future Work

Firstly, since the SSI's energy consumption is more limited by speed than its standard counterpart, more investigation should be made to define an area of efficiency for the new gate. Secondly, a layout must be designed to simulate a more realistic power consumption. Also, the 4-1 mux should be properly investigated to see exactly how much power can be saved by using the SSI version. Finally, more designs should be investigated for application of the SSI.

11. References

[1] S.-H. Cho, C.-K. Lee, J.-K. Kwon, S.-T. Ryu, “A 550-µW 10-b 40MS/s SAR ADC With Multistep Addition-Only Digital Error Correction”, in IEEE Journal of Solid-State Circuits. Vol 46, No. 8, August 2011.

[2] B.-E. Jonsson, “A/D-converter performance evolution (1974-2012)”, 2013 v1.1.

[3] M. van Elzakker, E. van Tuijl, P. Geraedts, D. Schinkel, E. Klumperink, B. Nauta, “A 1.9µW 4.4fJ/Conversion-step 10b 1MS/s Charge-Redistribution ADC” in IEEE ISSCC 2008.

[4] P. Harpe, E. Cantatore, A. Van Roermond, “A 10b/12b 40kS/s SAR ADC With Data-Driven Noise Reduction Achieving up to 10.1b ENOB at 2.2 fJ/Conversion-Step”, in IEEE Solid-State Circuits, December 2013.

[5] M. Ahmadi, W. Namgoong, “A 3.3fJ/conversion-step 250kS/s 10b SAR ADC Using Optimized Vote Allocation” in Department of Electrical Engineering, University of Texas at Dallas, IEEE, 2013.

[6] D.-A. Johns, K. Martin, “Analog Integrated Circuit Design”, by John Wiley & Sons, Inc., 1997.

[7] G. Wegmann, E.-A. Vittoz, F. Rahali, “Charge Injection in Analog MOS Switches”, in IEEE Journal of Solid-State Circuits, Vol sc-22, No. 6, December 1987.

[8] P.-M. Figueiredo, J.-C. Vital, “Kickback Noise Reduction Techniques for CMOS Latches Comparators”, in IEEE Transactions on Circuits and Systems, 2006.

[9] J.-A. Croon, M. Rosmeulen, S. Decoutere, W. Sansen, H.-E. Maes, “An Easy-to-Use Mismatch Model for the MOS Transistor”, in IEEE Journal of Solid-State Circuits. Vol 37, No. 8, August 2002.

[10] A. Wiltgen Jr., K.-A. Escobar, A.-I. Reis, R.-P. Ribas, “Power Consumption Analysis in Static CMOS Gates”, Institute of Informatics, Federal University of Rio Grande doSul – UFRGS, Porto Alegre, Brazil, IEEE, 2013.

[11] D. Zhang, A. Alvandpour, “A 3-nW 9.1-ENOB SAR ADC at 0.7 V and 1 kS/s”, in IEEE, 2012.

[12] A. Shikata, R. Sekimoto, T. Kuroda, H. Ishikuro, “A 0.5V 1.1MS/sec 6.3fJ/conversion-step SAR-ADC with Tri-Level Comparator in 40nm CMOS”, in Symposium on VLSI Circuits Digest of Technical Papers, 2011.

[13] H.-Y. Tai, H.-W. Chen, H.-S. Chen, “A 3.2fJ/c.-s. 0.35V 10b 100KS/s SAR ADC in 90nm CMOS”, in IEEE Symposium on VLSI Circuit Digest of Technical Papers”, 2012.

[14] H. Fan, X. Han, Q. Wei, “A 12-bit self-calibrating SAR ADC achieving a Nyquist 90.4-dB SFDR”, in Springer Science+Business Media, New York, 2012.

[15] S.-I. Kim, B.-G. Min, C.-W. Ju, “Comparative Study of Static and Dynamic D-type Flip-Flop Circuit suing InP HBT’s”, in IEEE Asia-Pacific Conference on Advanced System Integrated Circuits, Aug. 4-5, 2004.

[16] D. Markovic, B. Nikolic, R.-W. Brodersen, “Analysis and Design of Low-Energy Flip-Flops”, Berkely Wireless Research Center, University of California, Berkely, 2001.

[17] G. Gerosa, S. Gary, C. Dietz, D. Pham, K. Hoover, J. Alvarez, H. Sanchez, P. Ippolito, T. Ngo, S. Litch, J. Eno, J. Golab, N. Vanderschaaf, J. Kahle, “A 2.2 W, 80MHz Superscalar RISC Microprocessor”, in IEEE Journal of Solid-State Circuits. Vol 29 No. 12, December 1994.

[18] “Virtuoso Visualization and Analysis XL User Guide Product Version 6.1.5”, December 2012.

Appendix A – Soft-Supply Inverter Simulations

Both the static and dynamic power consumption of the soft-supply inverter will be simulated and compared to the conventional method seen in figure A.1 (repeated here for convenience).

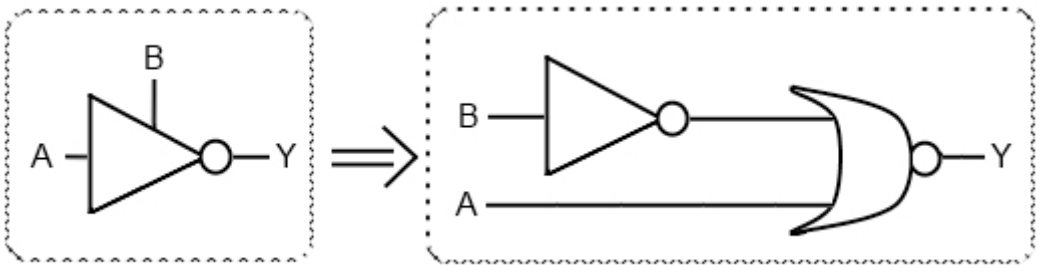

Figure A.1: Soft-Supply Inverter, Model

Another version of the soft-supply inverter will also be simulated. It is based on the same idea as the original idea, except the variable input is placed at the previous ground instead of where the supply voltage were, thus it has been given the name soft-ground inverter. The transistor level schematic of the circuit can be seen in figure A.2 below.

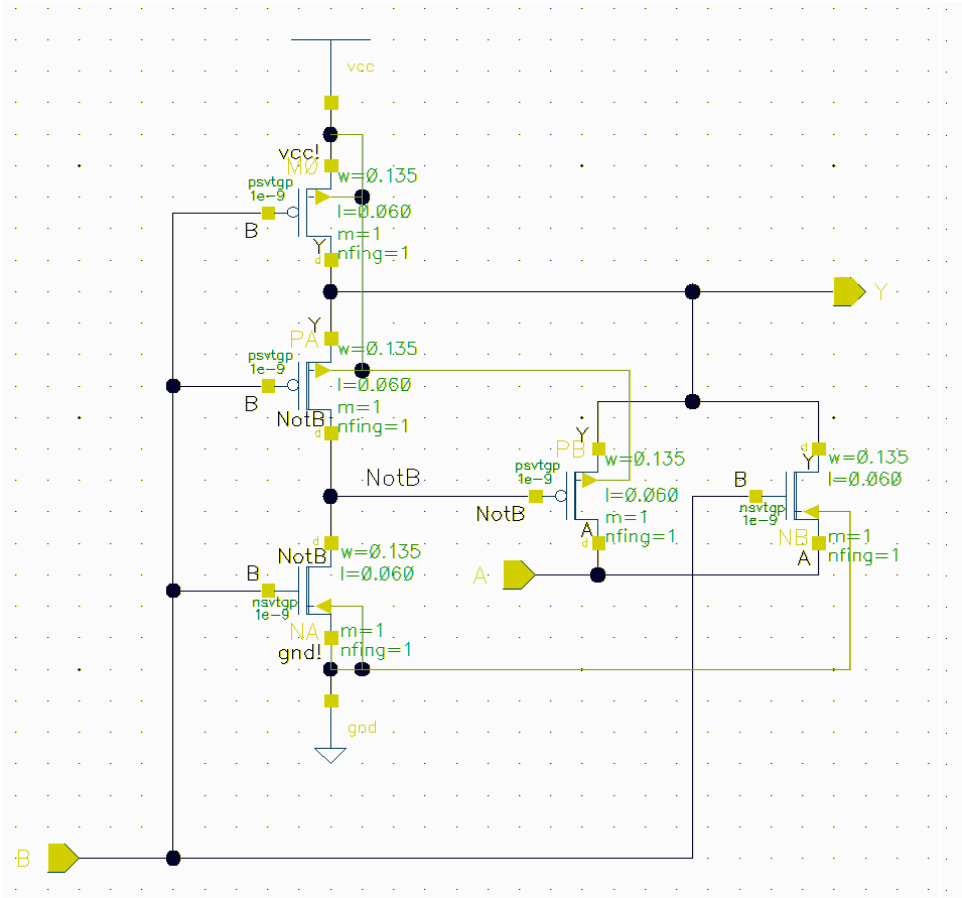

Figure A.2: Schematic Soft-Ground Inverter

The behavior is an inverted version of the soft-supply inverter, which can be seen by table A.1 below.

Table A.1: Truth table, Soft-Ground Inverter

A	B	Y
0	0	1
0	1	0
1	0	1
1	1	1

Static Power Consumption

The setup for simulating the static power consumption is shown in figure A.3. The element "Design" is the system under test. The other elements are regular inverters with their own voltage supply separated from "Design". Both supply voltages were set to 1V.

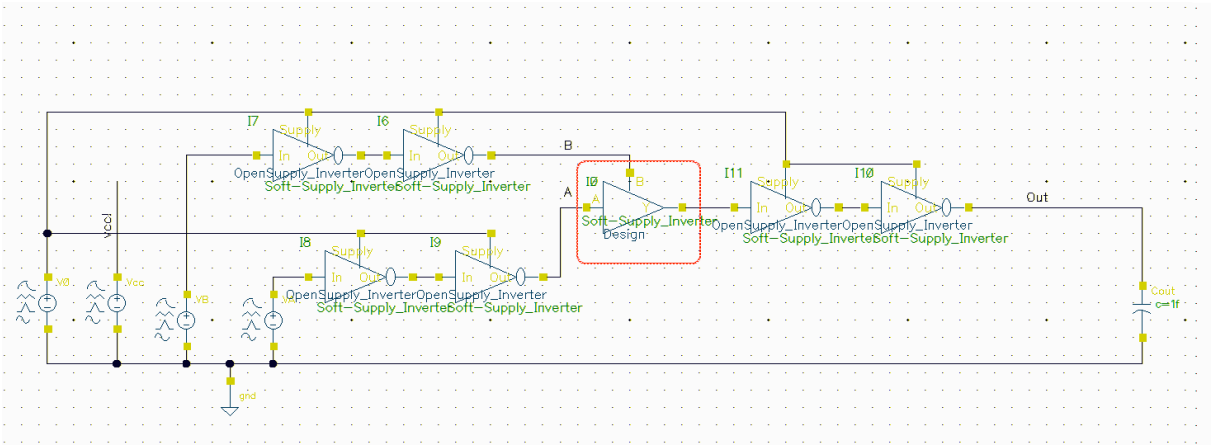

Figure A.3: Setup Simulation, Static Power Consumption

As mentioned earlier, both the SS-inverter and the SG-inverter will pull current from the output of a preceding gate for some of its input combinations. For the SS-inverter it is when A = 0. For the SG-inverter it is when B = 1.
To measure the correctly, the total power consumption equals the consumption from both Vcc and the mentioned input signal.

Soft-Supply Inverter:

NOR design:

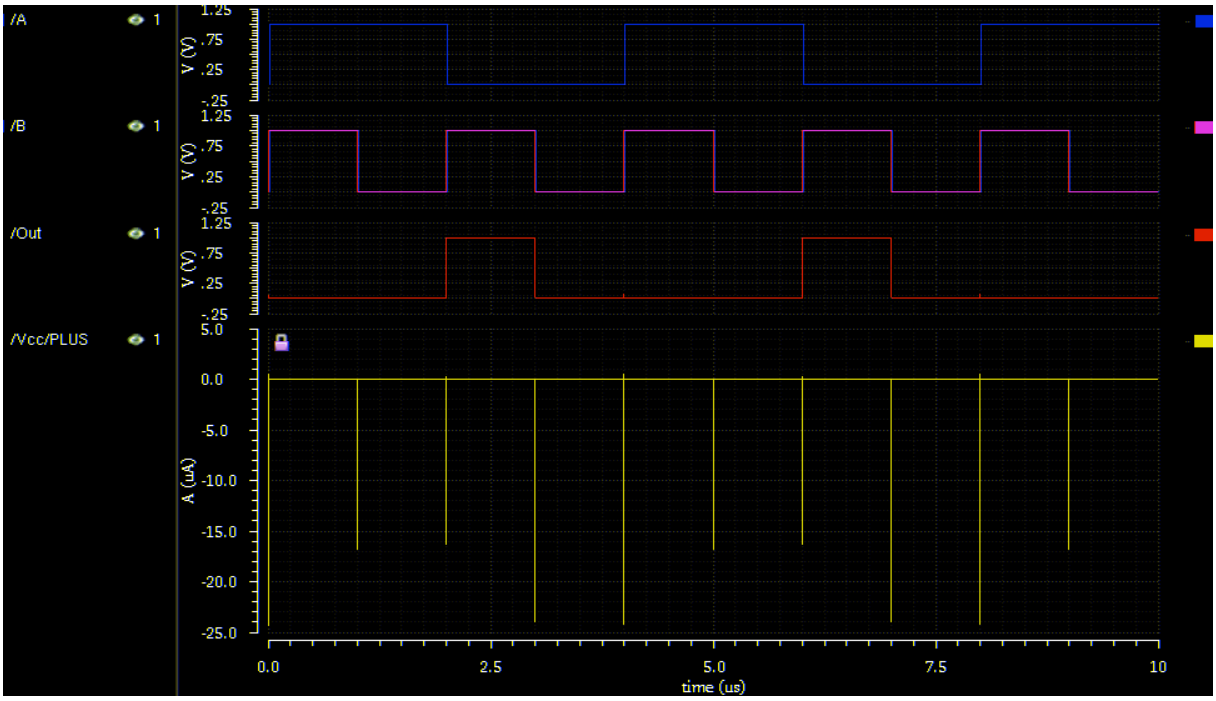

Figure A.4: Transient Simulation, NOR Design

DC Analysis:

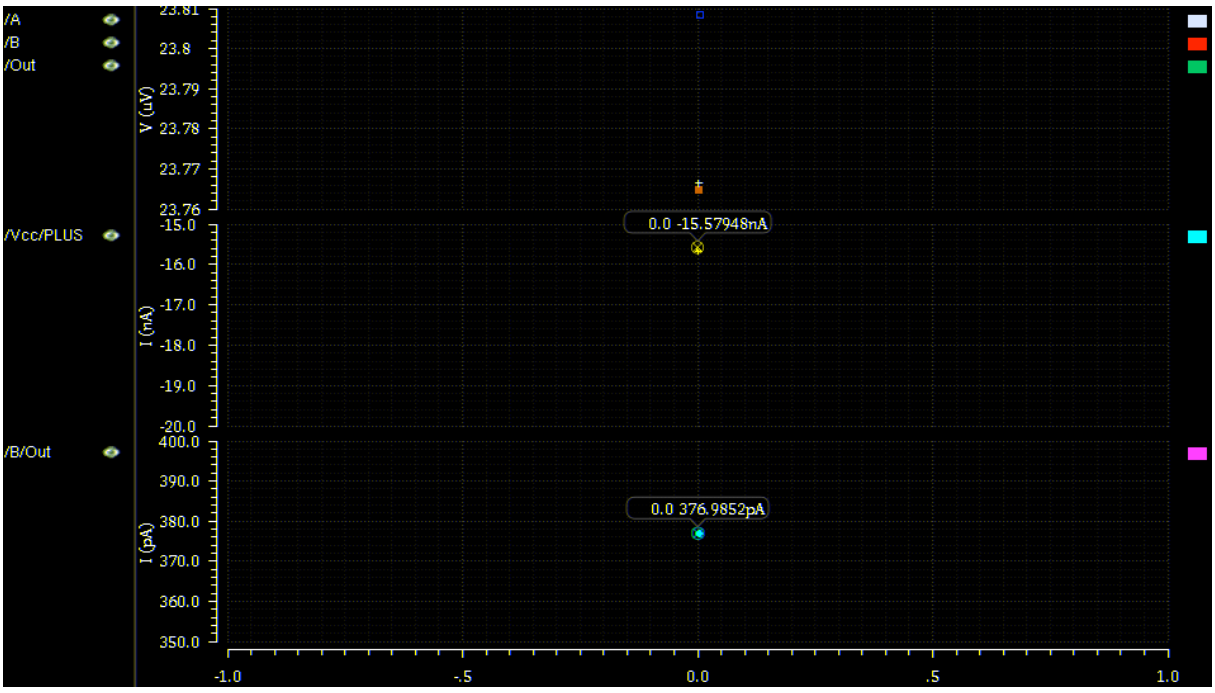

Figure A.5: Static Power Consumption, NOR Design (A=0, B=0, Y=0)

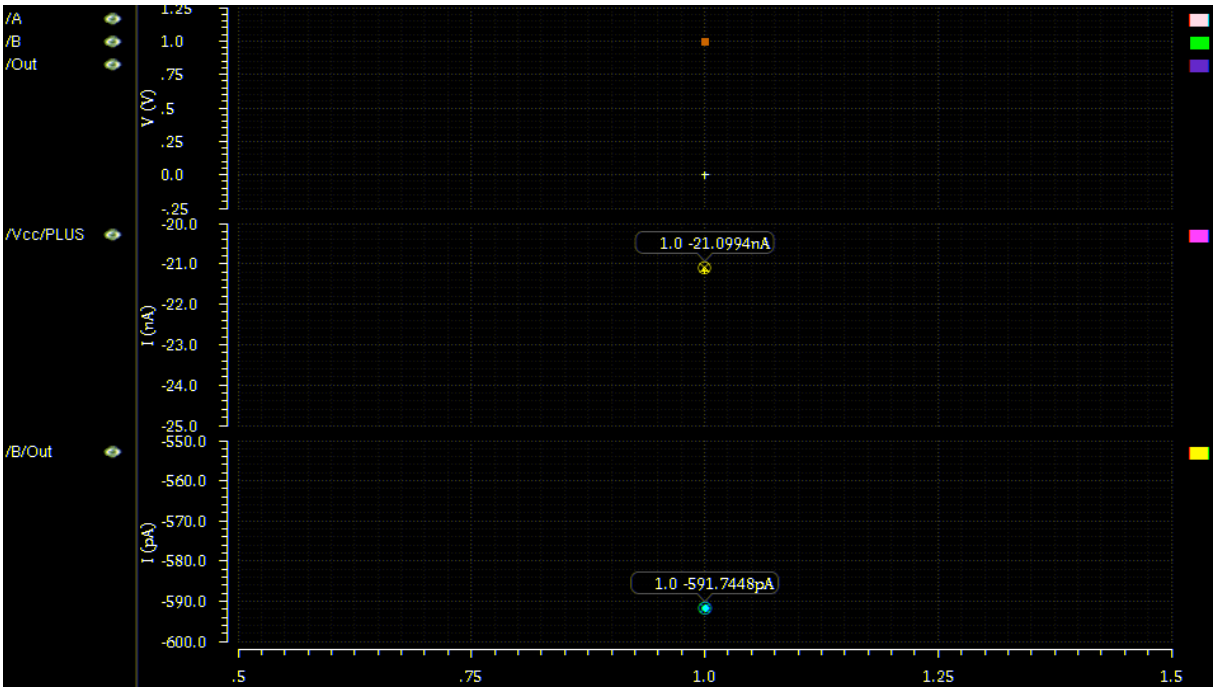

Figure A.6: Static Power consumption, NOR design (A=0, B=1, Y=1)

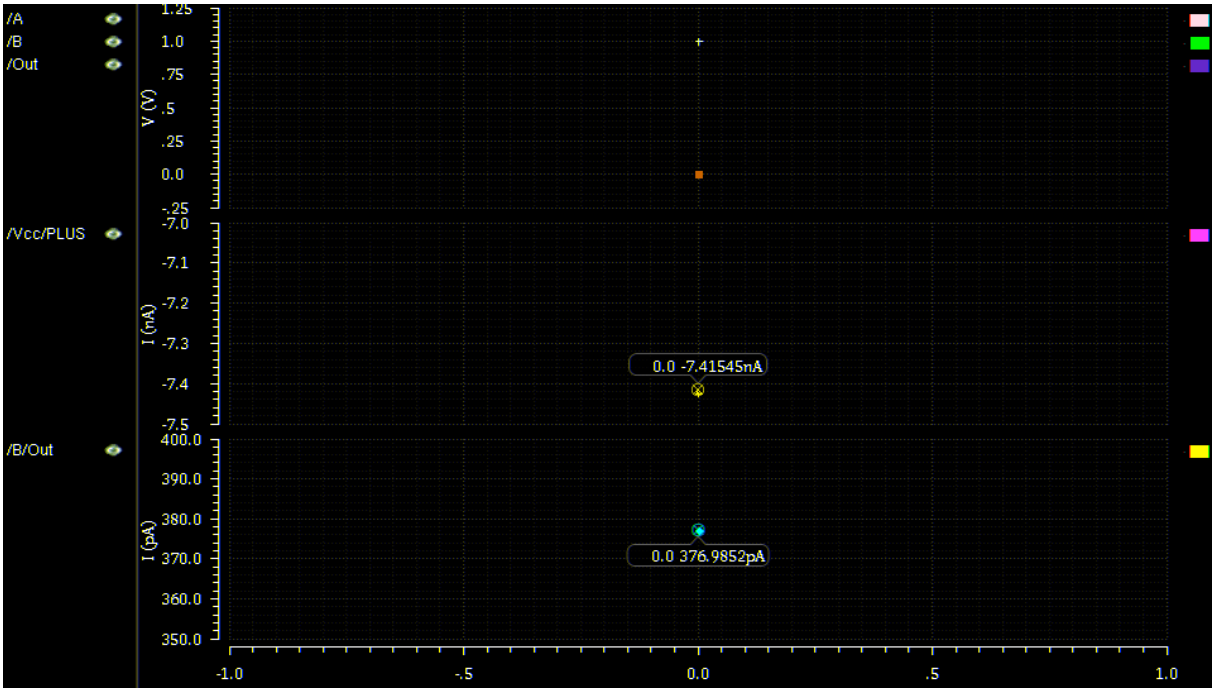

Figure A.7: Static Power consumption, NOR design (A=1, B=0, Y=0)

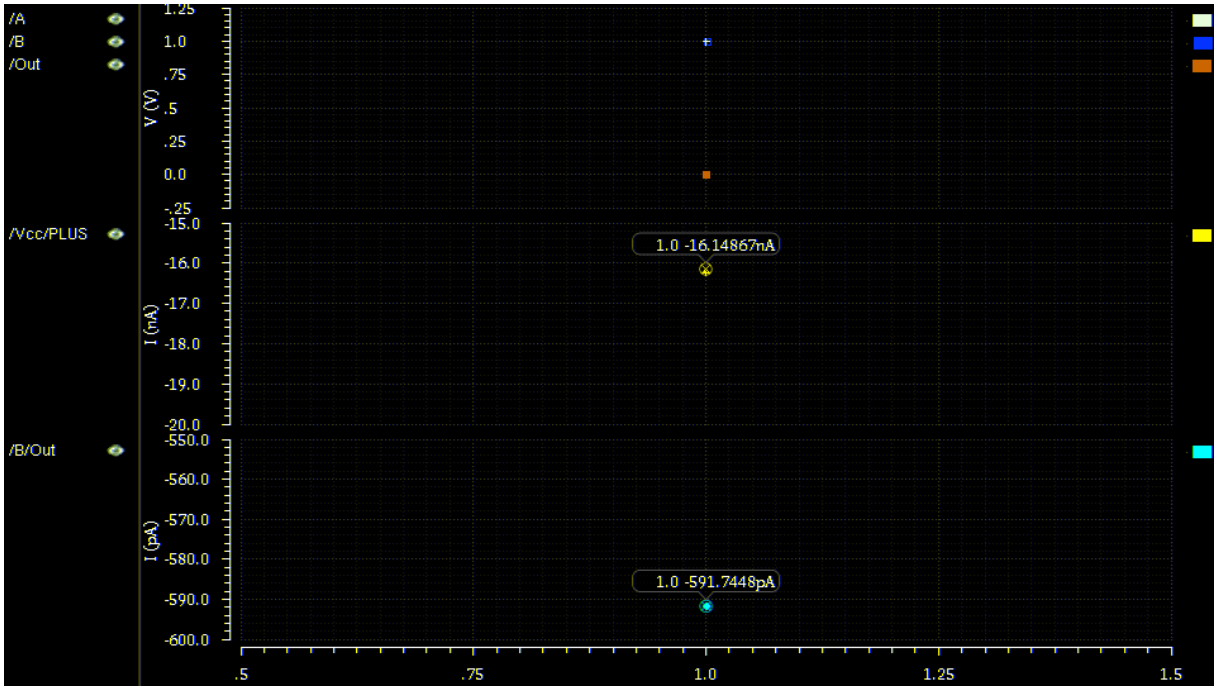

Figure A.8: Static Power consumption, NOR design (A=1, B=1, Y=0)

Soft-Supply Inverter:

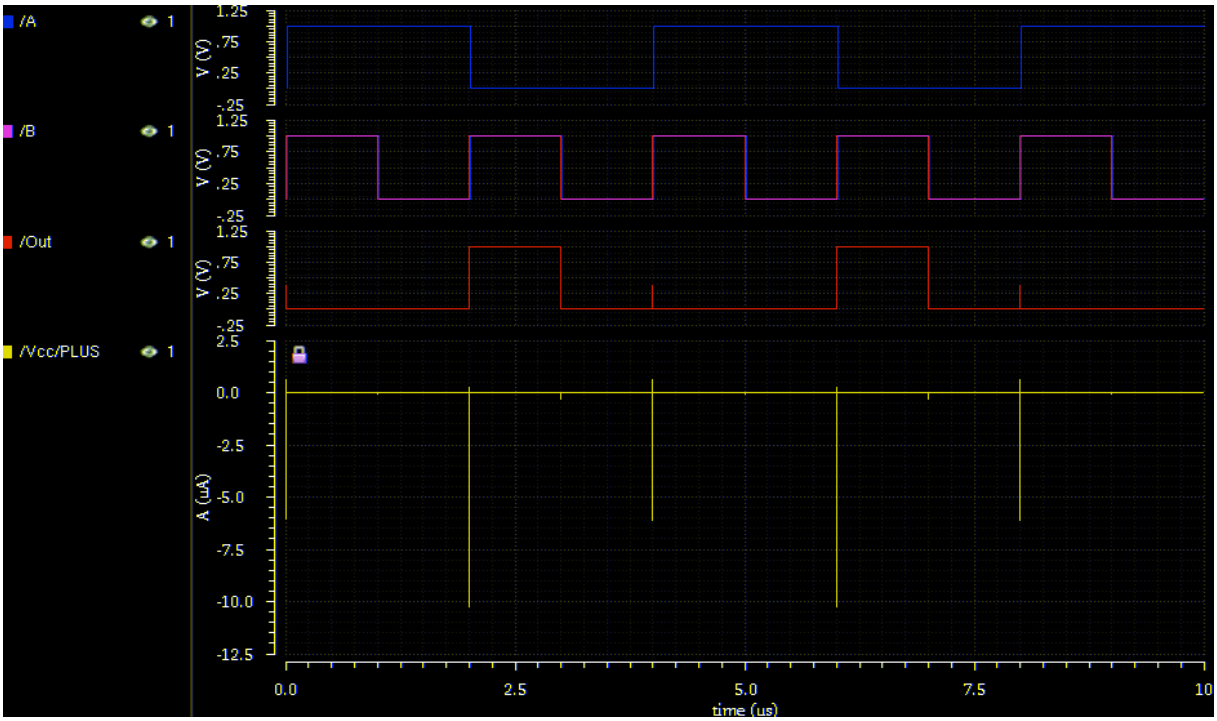

Figure A.9: Transient Simulation, SS-Inverter

DC Analysis:

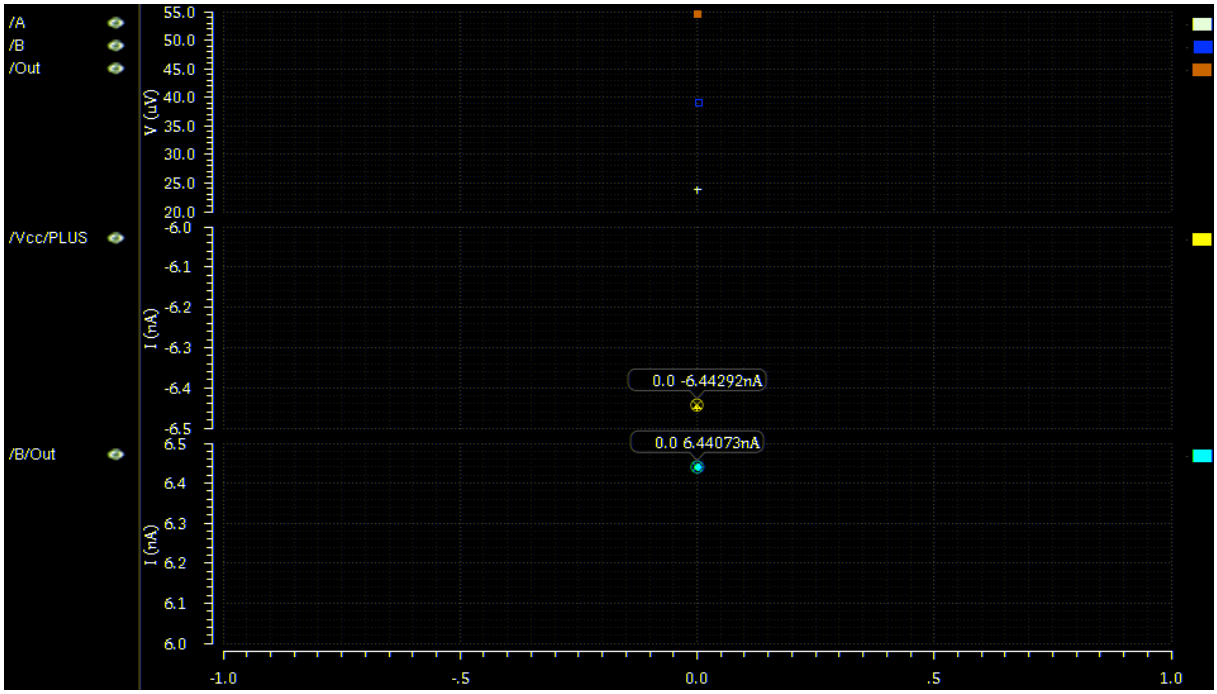

Figure A.10: Static Power Consumption, SS-Inverter (A=0, B=0, Y=0)

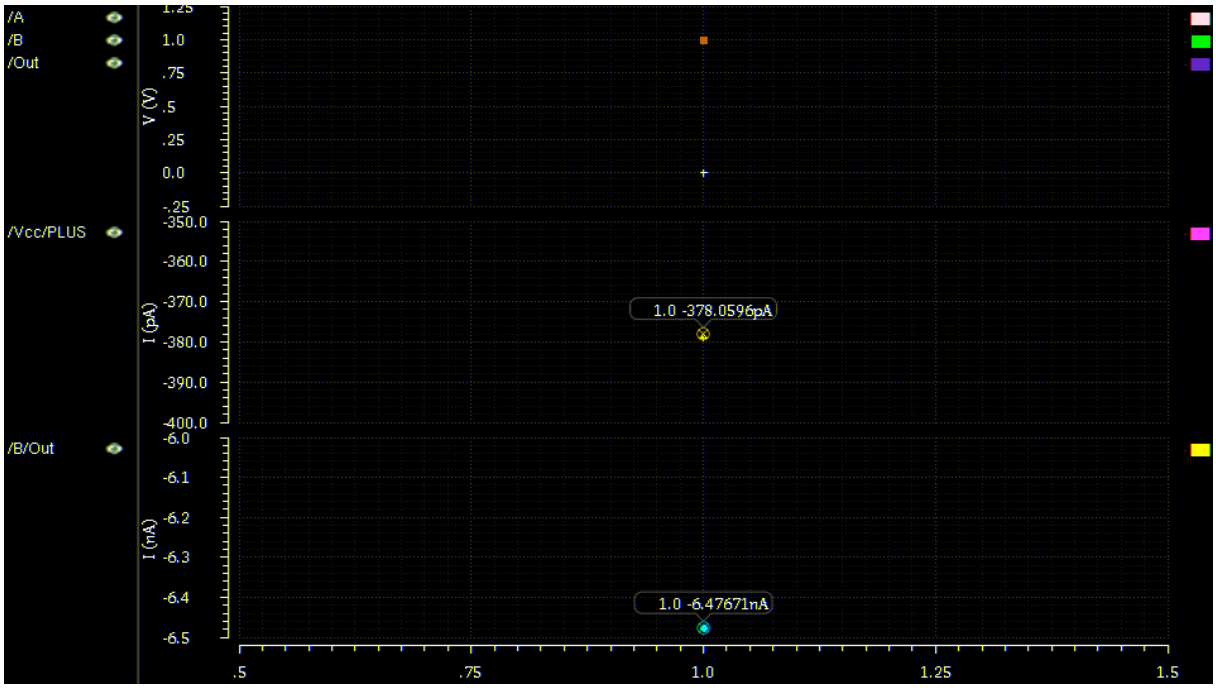

Figure A.11: Static Power Consumption, SS-Inverter (A=0, B=1, Y=1)

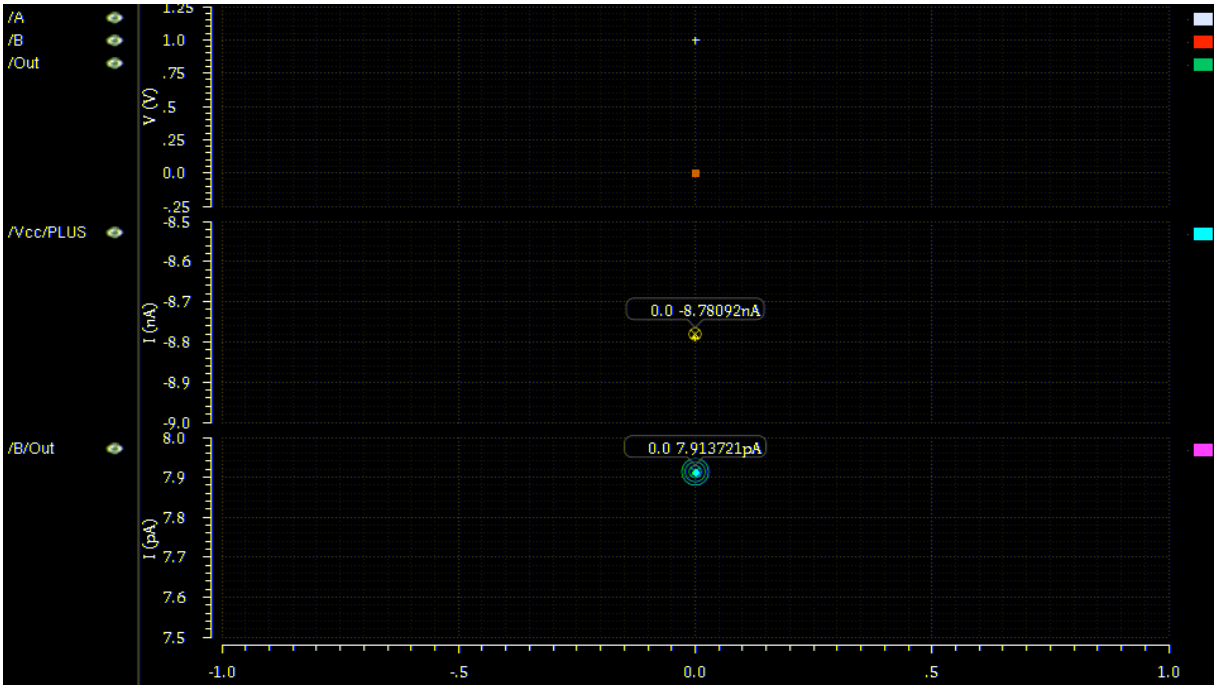

Figure A.12: Static Power Consumption, SS-Inverter (A=1, B=0, Y=0)

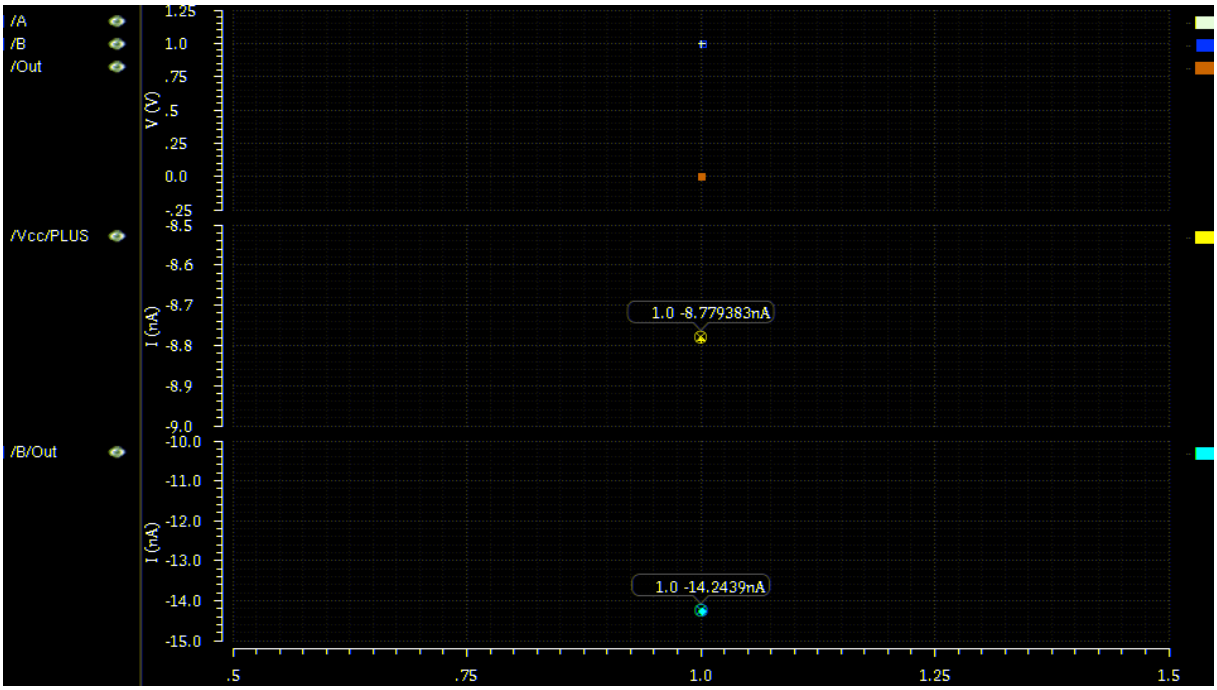

Figure A.13: Static Power Consumption, SS-Inverter (A=1, B=1, Y=0)

Soft-Ground Inverter:

NAND design:

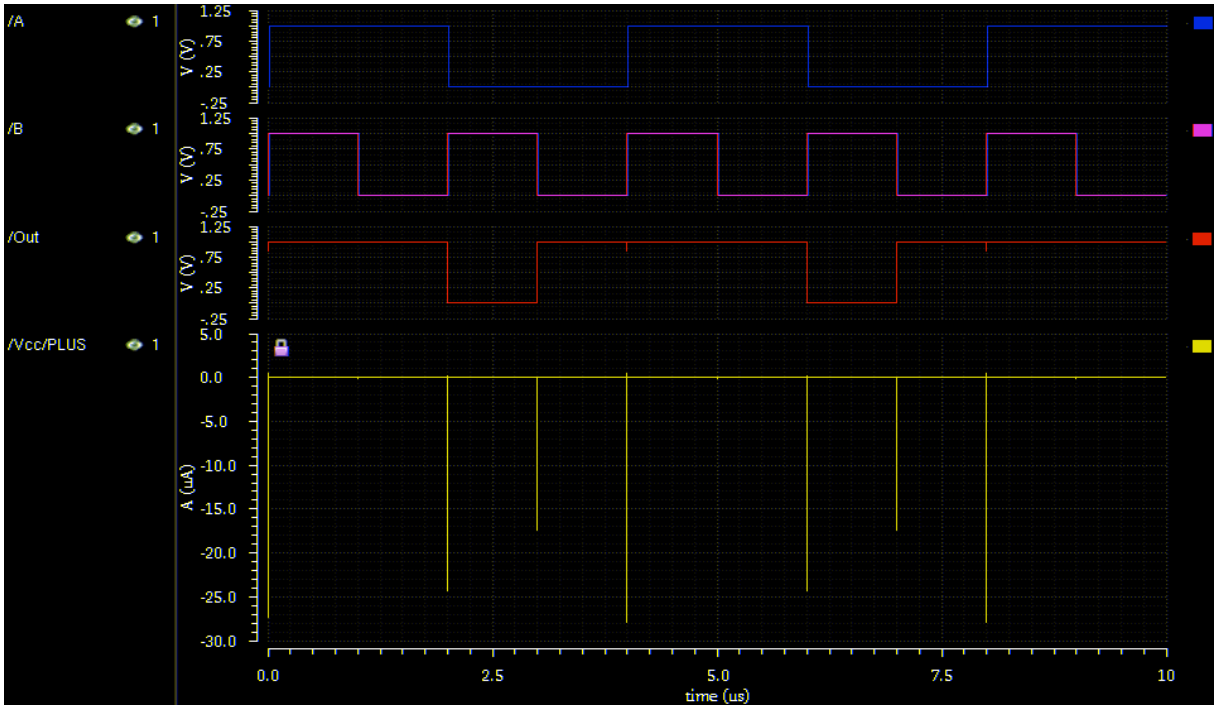

Figure A.14: Transient Simulation, NAND Design

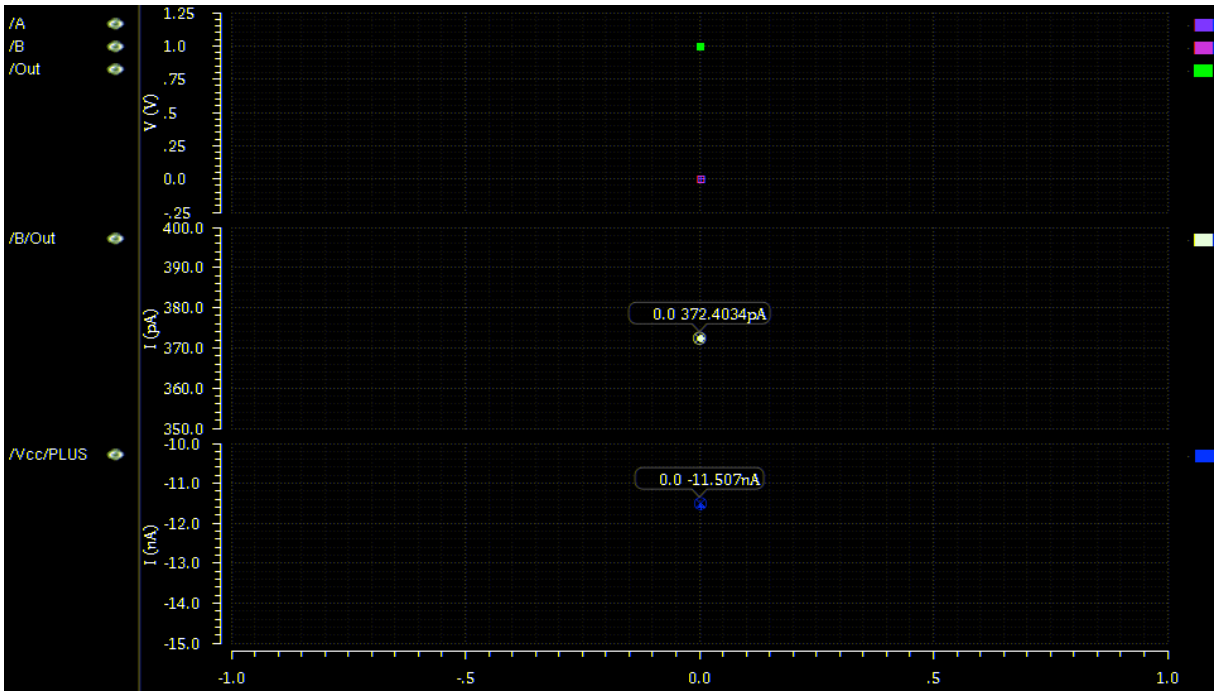

Figure A.15: Static Power Consumption, NAND Design (A=0, B=0, Y=1)

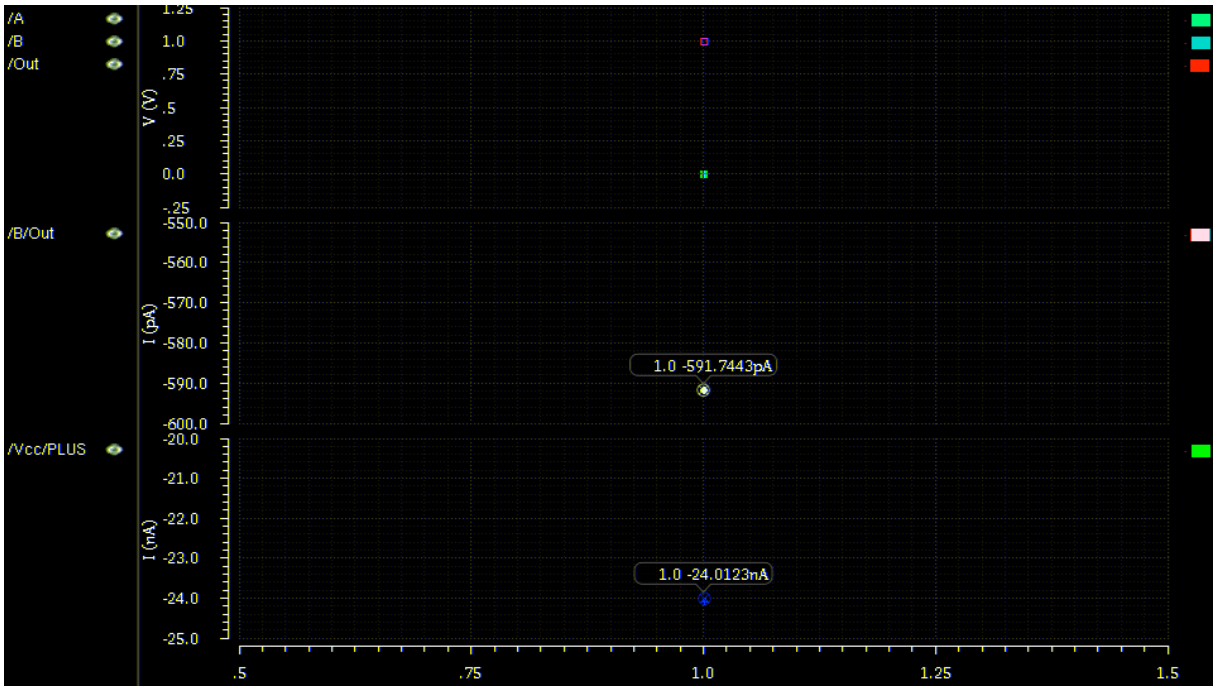

Figure A.16: Static Power Consumption, NAND Design (A=0, B=1, Y=0)

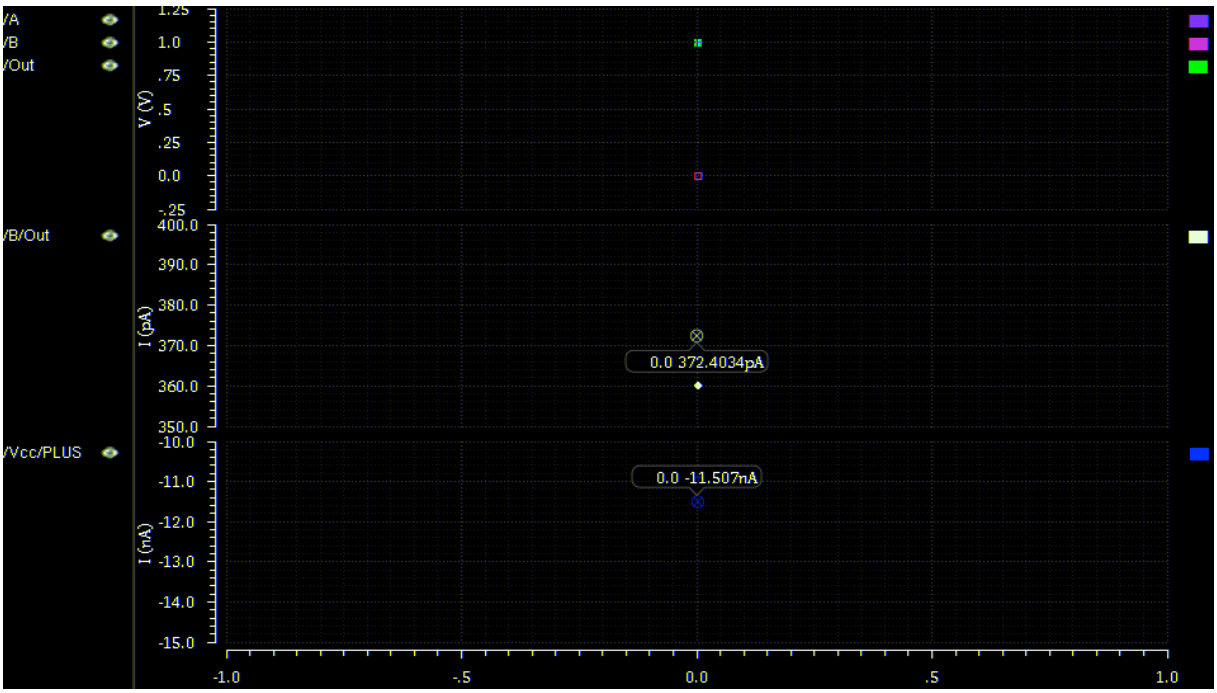

Figure A.17: Static Power Consumption, NAND Design (A=1, B=0, Y=1)

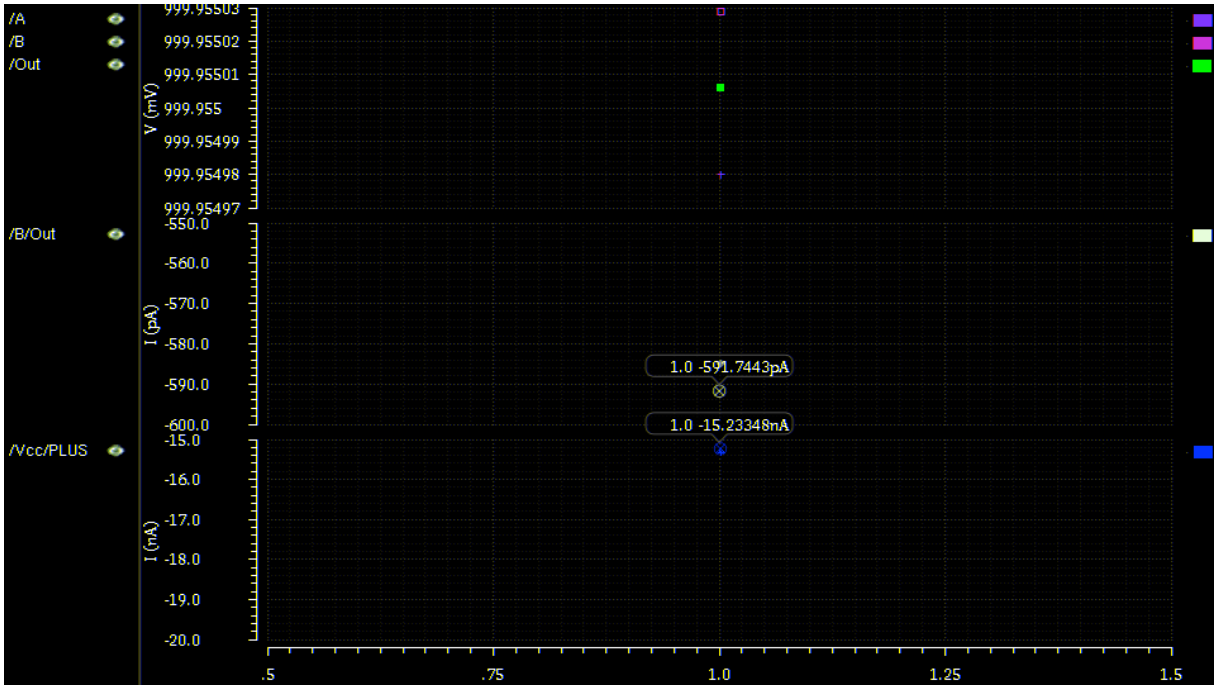

Figure A.18: Static Power Consumption, NAND Design (A=1, B=1, Y=1)

Soft-Ground Inverter:

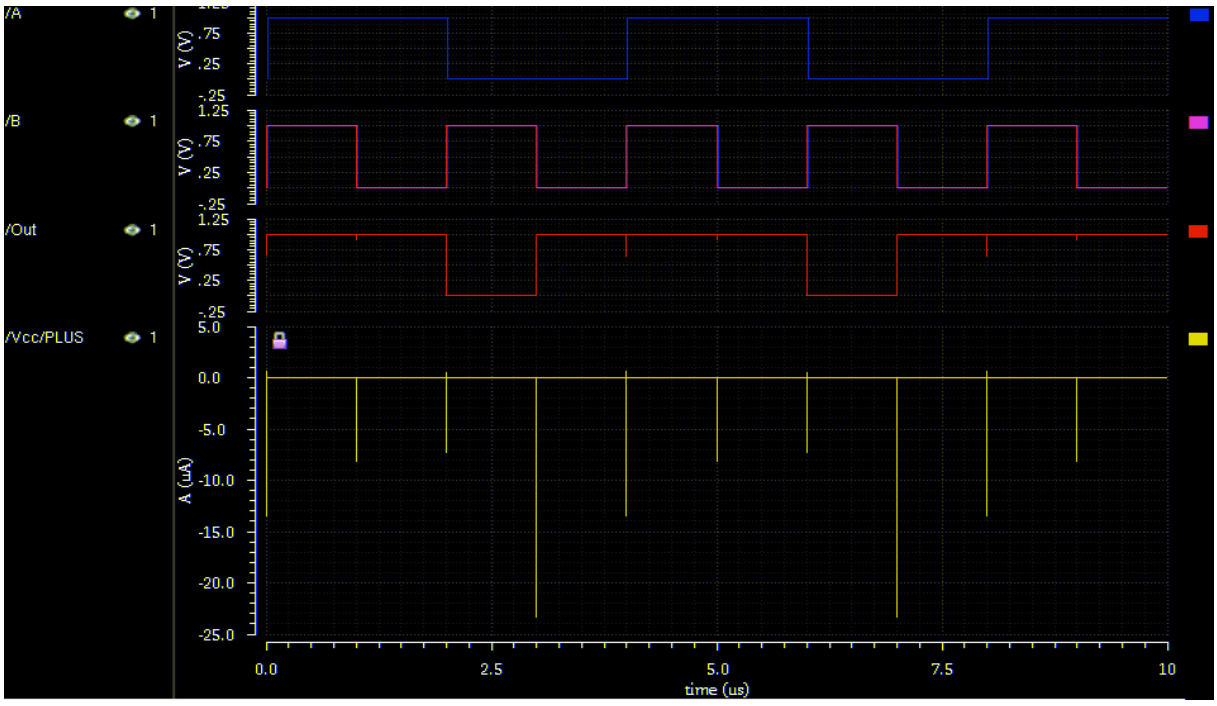

Figure A.19: Transient Simulation, SG-Inverter

DC Analysis:

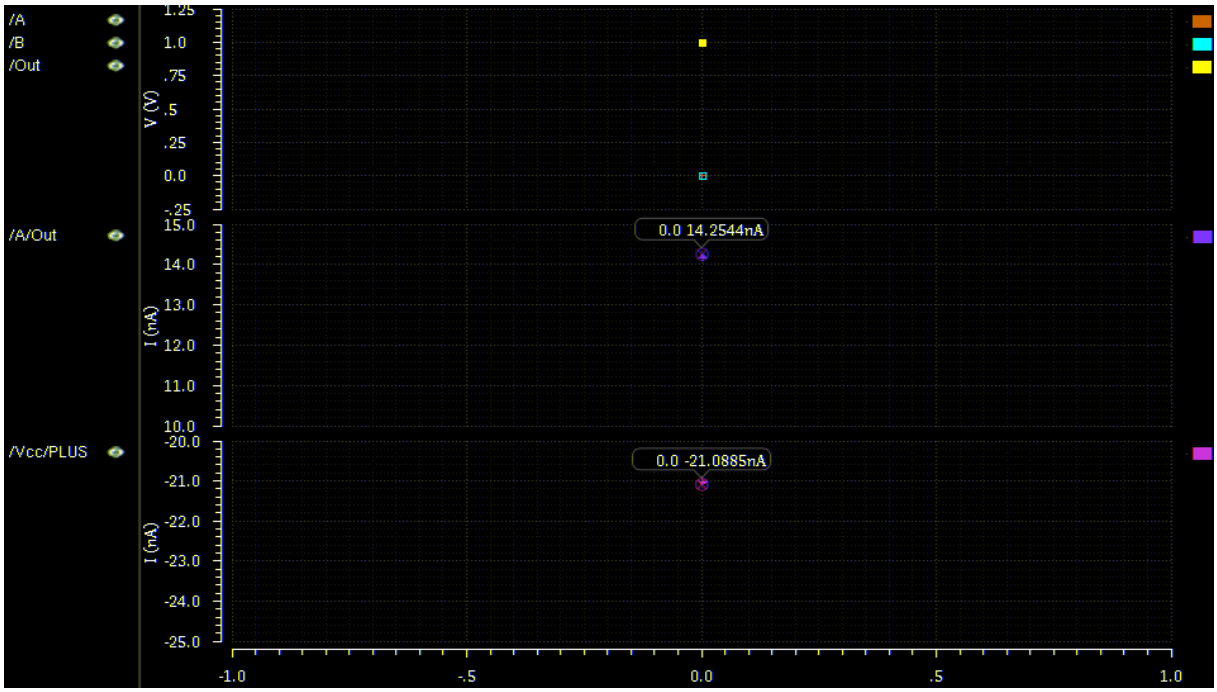

Figure A.20: Static Power Consumption, SG-Inverter (A=0, B=0, Y=1)

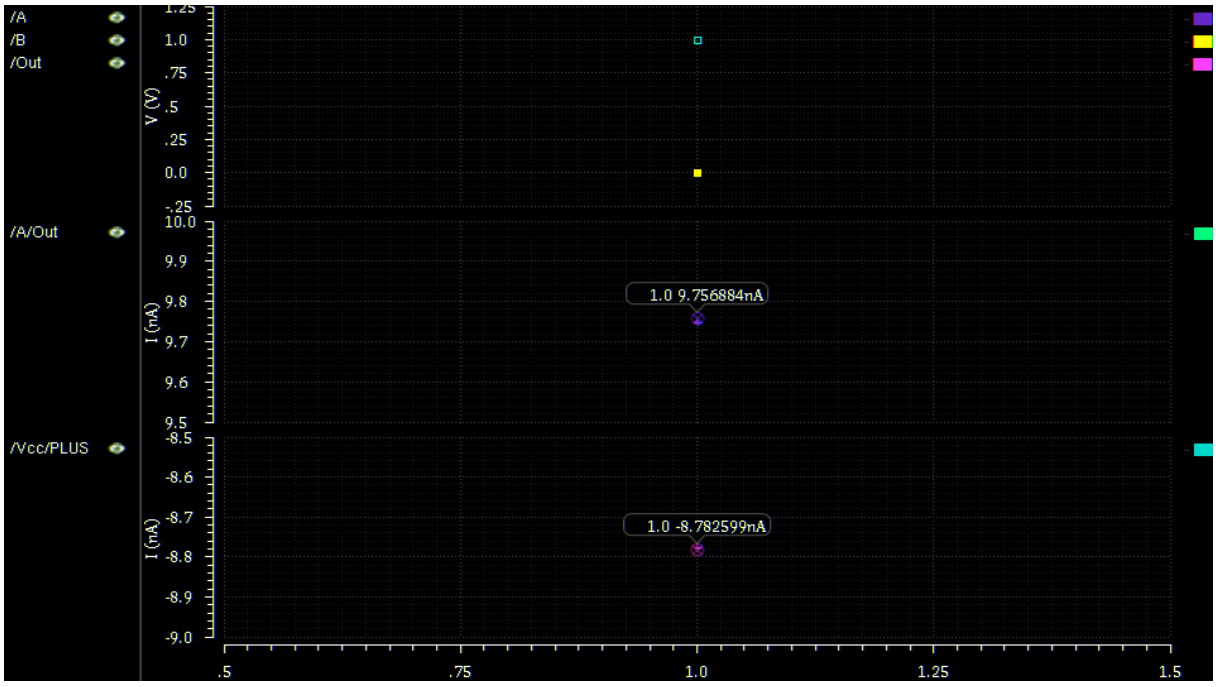

Figure A.21: Static Power Consumption, SG-Inverter (A=0, B=1, Y=0)

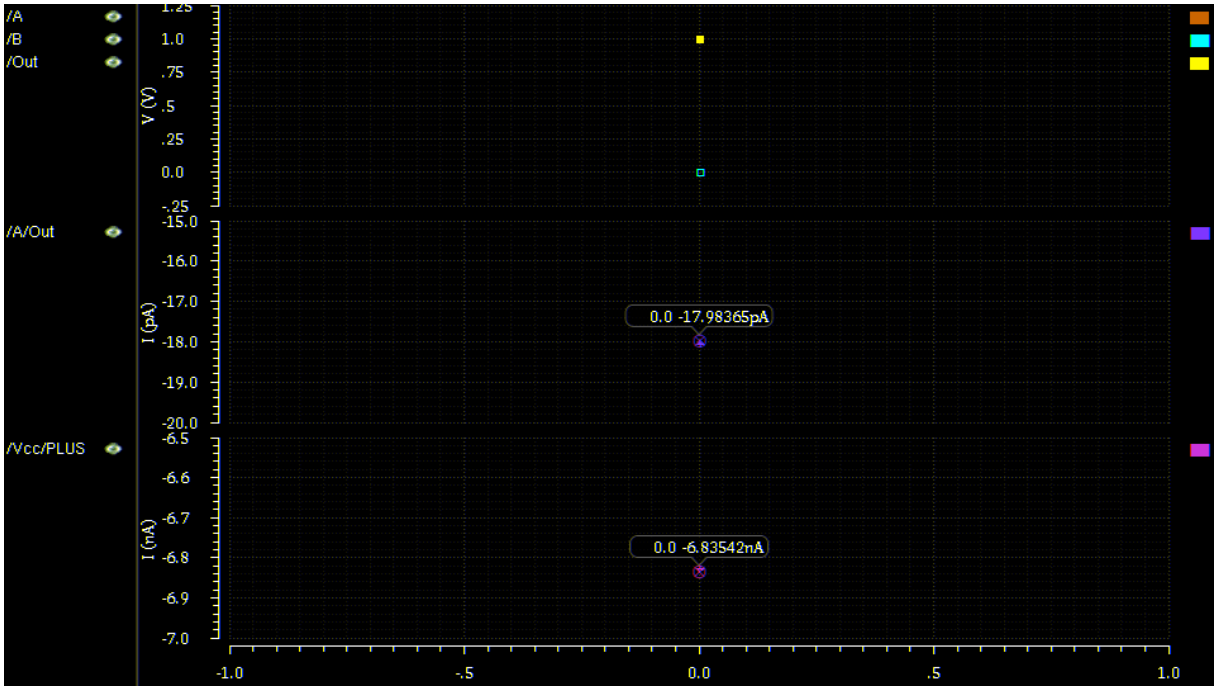

Figure A.22: Static Power Consumption, SG-Inverter (A=1, B=0, Y=1)

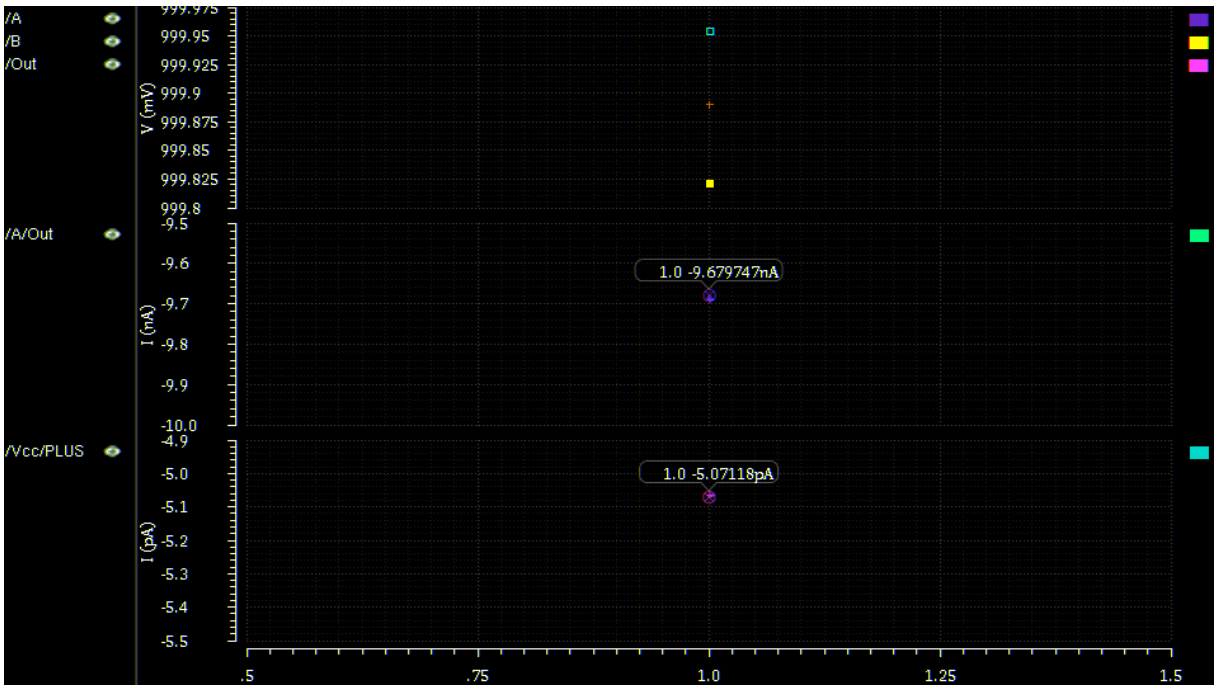

Figure A.23: Static Power Consumption, SG-Inverter (A=1, B=1, Y=1)

Summary

All of the simulations results from figures A.4-A20 are summarized in table A.2. It shows the comparisons between standard designs and the new designs. The bold values show the lowest power consumption for the given input combinations, as well as their sums.

Table A.2: Static Power Consumption Summary

Inputs	NOR Design	SS-Inverter	NAND Design	SG-Inverter
A = 0, B = 0	15.58nW	**6.442nW**	**11.51nW**	21.09nW
A = 0, B = 1	21.10nW	**6.855nW**	24.01nW	**8.783nW**
A = 1, B = 0	**7.415nW**	8.781nW	**11.51nW**	24.82nW
A = 1, B = 1	**16.15nW**	23.02nW	15.23nW	**14.75nW**
SUM	60.25nW	**45.10nW**	**62.26nW**	69.44nW

The soft-supply inverter shows a reduction of about 25% in average static power consumption. Note that when both A and B the power consumption is much higher than for the other input combinations.

The soft-ground inverter shows an increase of about 11.5% in average static power consumption.

Dynamic Power Consumption

The different designs were simulated by putting several circuits in a loop. Initial conditions were placed so that the circuit would begin oscillating once the simulation started. Here, all elements share the same supply voltage (1V).

Because of the different logical behaviors, the simulation setup had to be different for each design.

NOR Design

The setup for simulating the dynamic power consumption of the NOR design is shown in figure A.24 below.

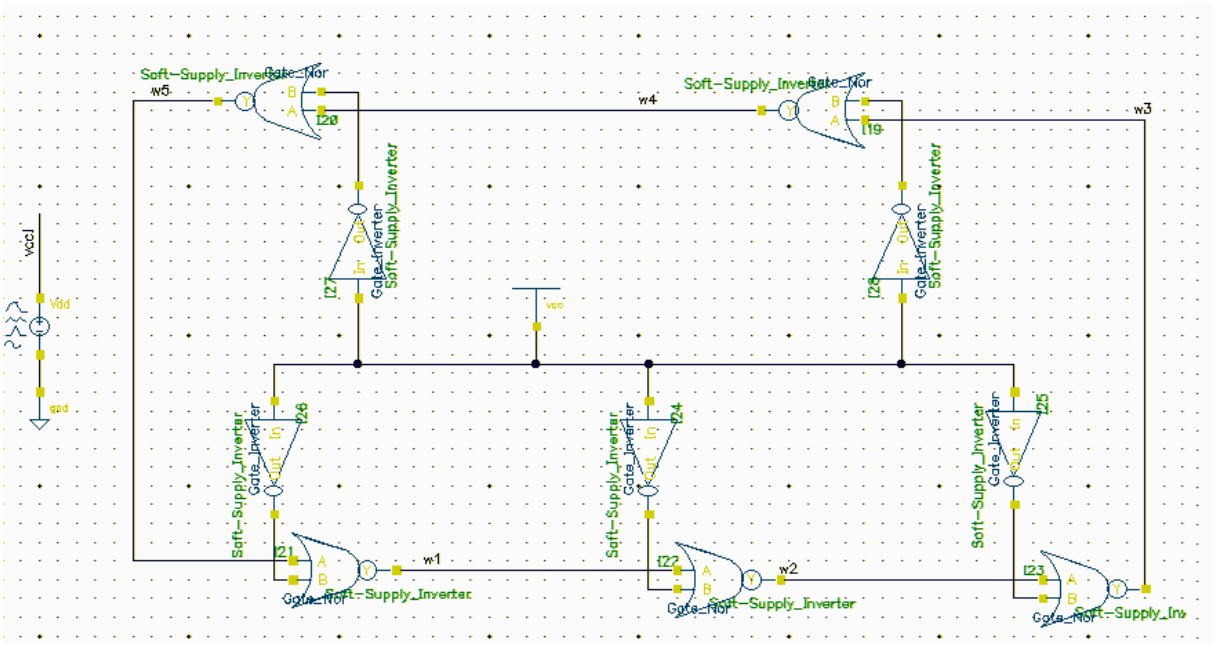

Figure A.24: Setup for Dynamic Power Consumption, NOR Design

The input B on the gate is set to a constant high input value while input A will continuously change as the circuit oscillates.

Figure A.25 shows the transient simulation.

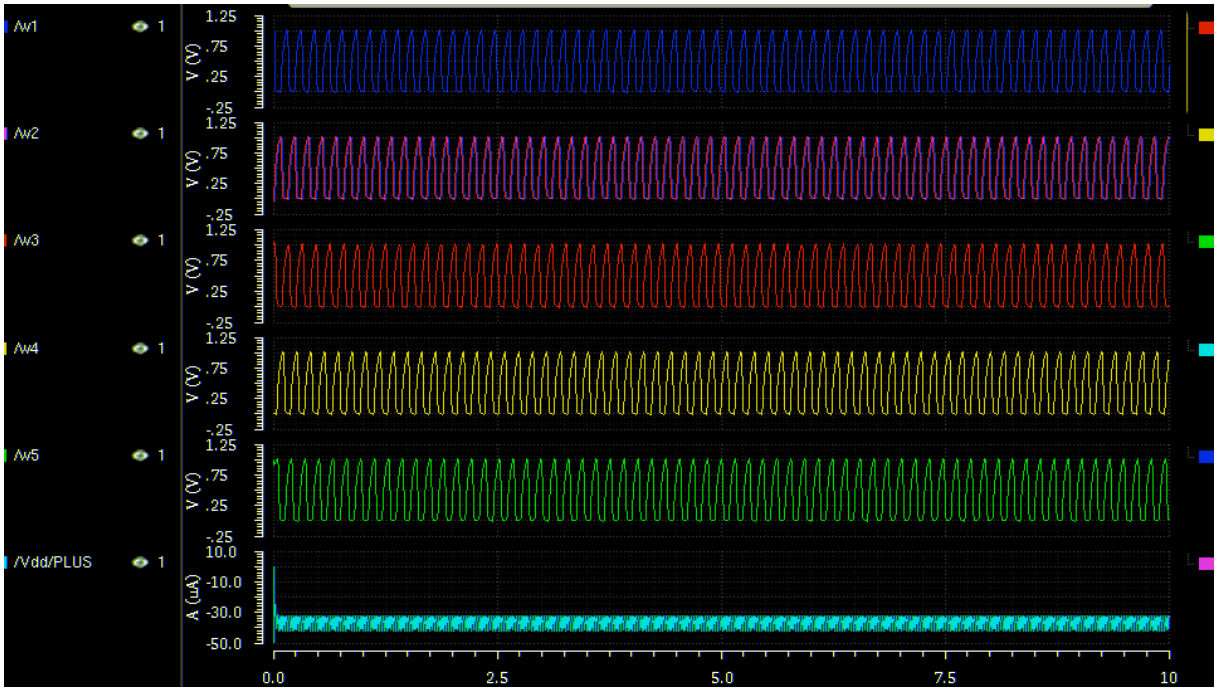

Figure A.25: Dynamic Power Consumption, NOR Design

5 elements in oscillation

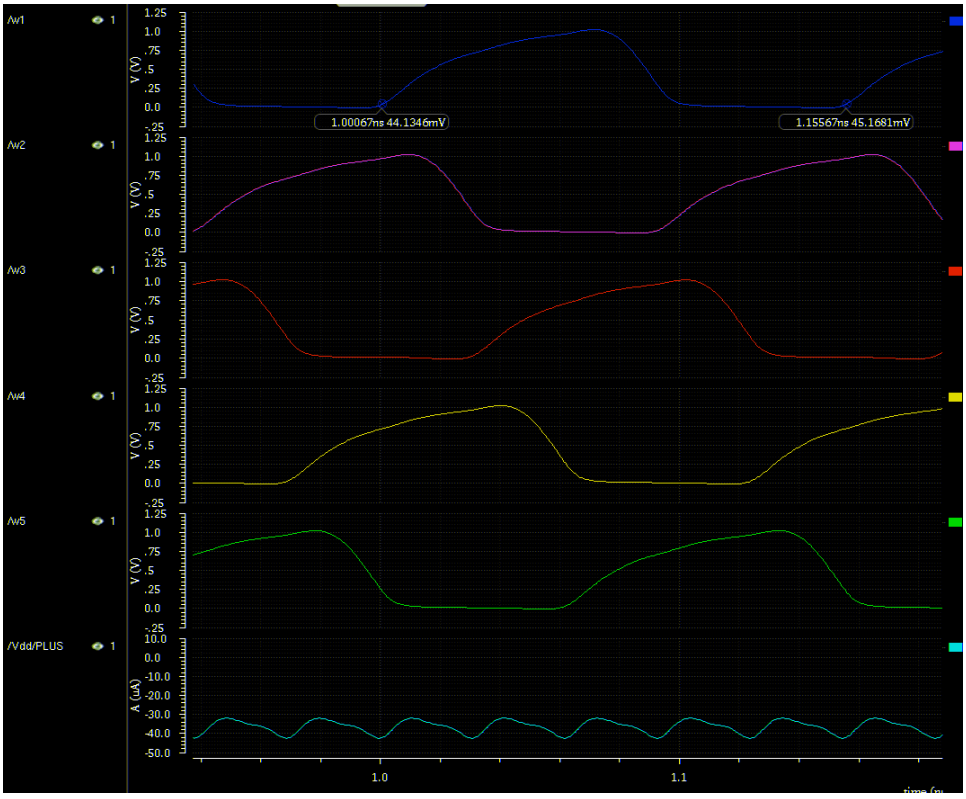

Figure A.26: 5 elements oscillating, NOR Design

The power consumption was **–36.51uA**.
A period took **–0.156ns**.

3 elements in oscillation

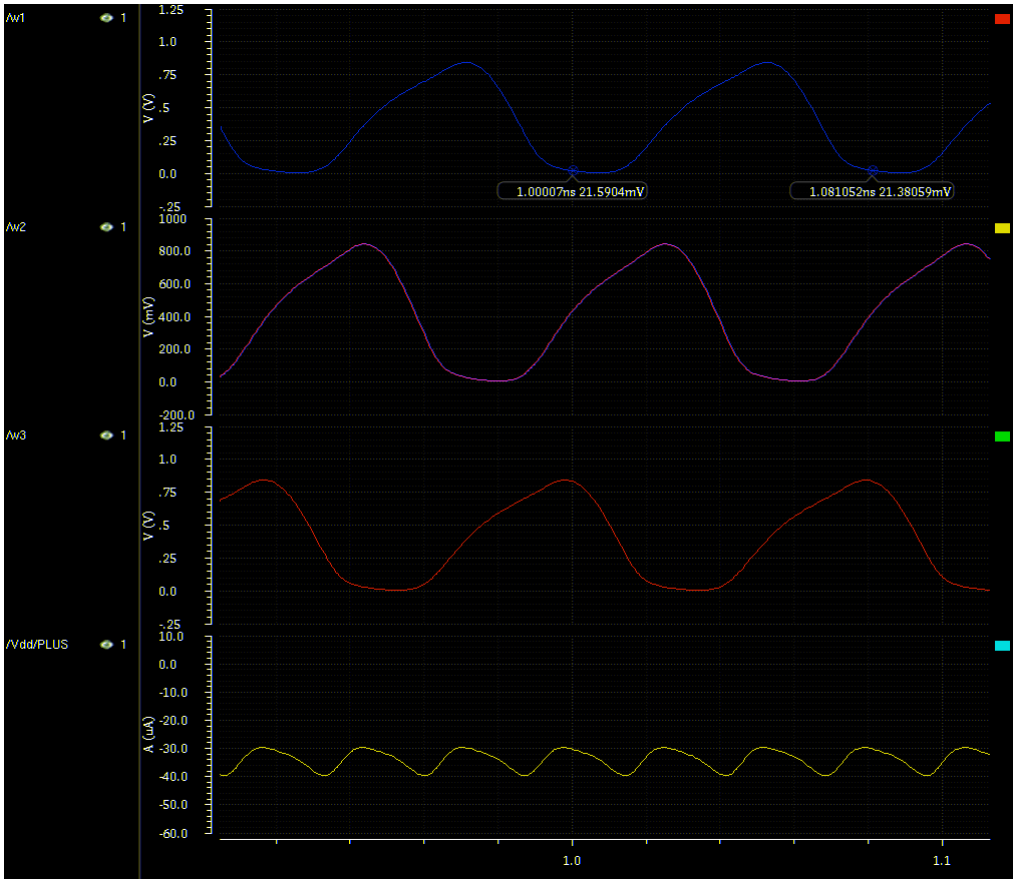

Figure A.27: 3 elements oscillating, NOR Design

The power consumption was **–34.01uA**.
A period took **–0.081ns**.

Soft-Supply Inverter

The setup for simulating the dynamic power consumption of the soft-supply inverter is shown in figure A.28 below.

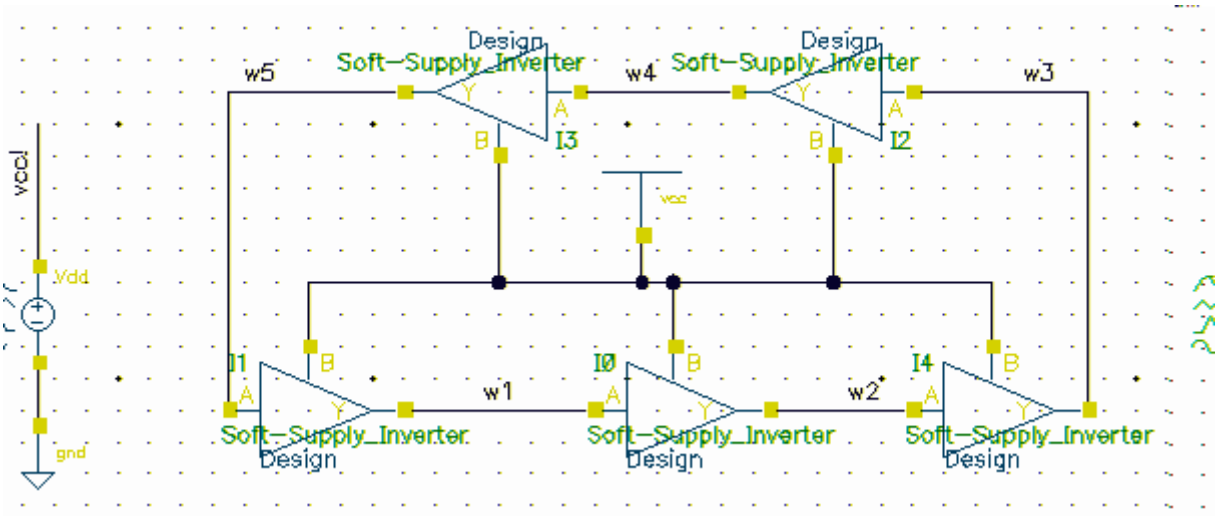

Figure A.28: Setup for Dynamic Power Consumption, SS-Inverter

The input B is set to a constant high input value while input A will continuously change as the circuit oscillates. That way, the inverter inside the design will also be used in every loop.

Figure A.29 shows the transient simulation.

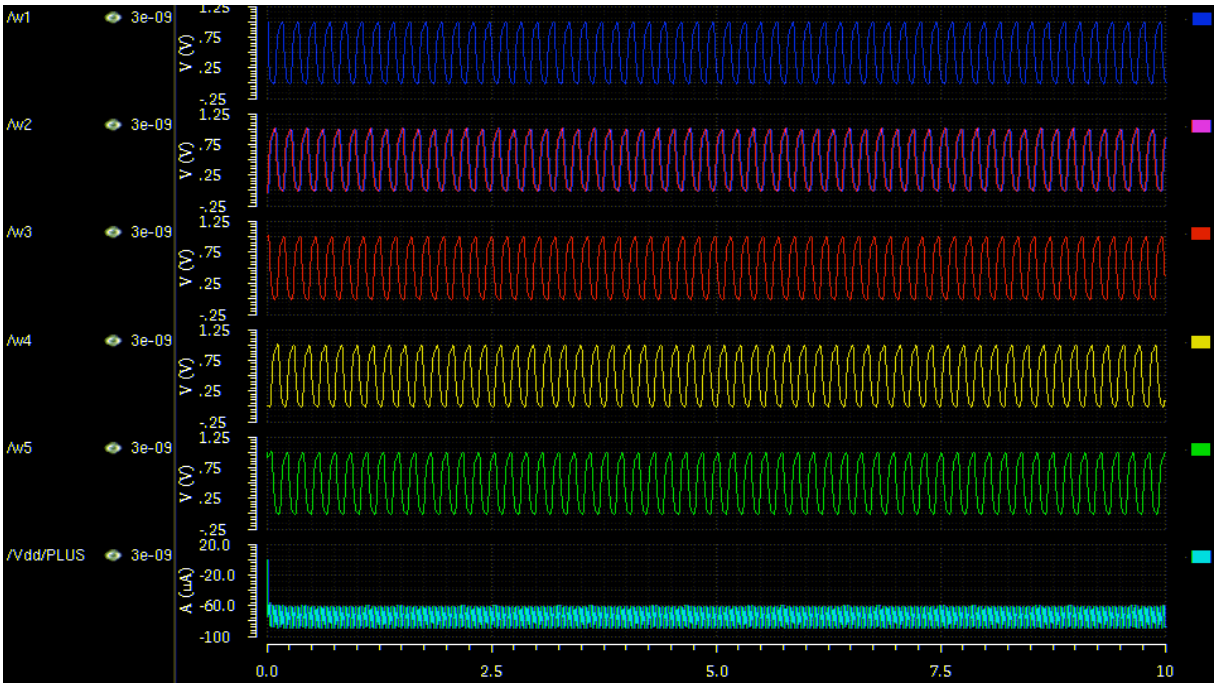

Figure A.29: Dynamic Power Consumption, SS-Inverter

5 elements in oscillation

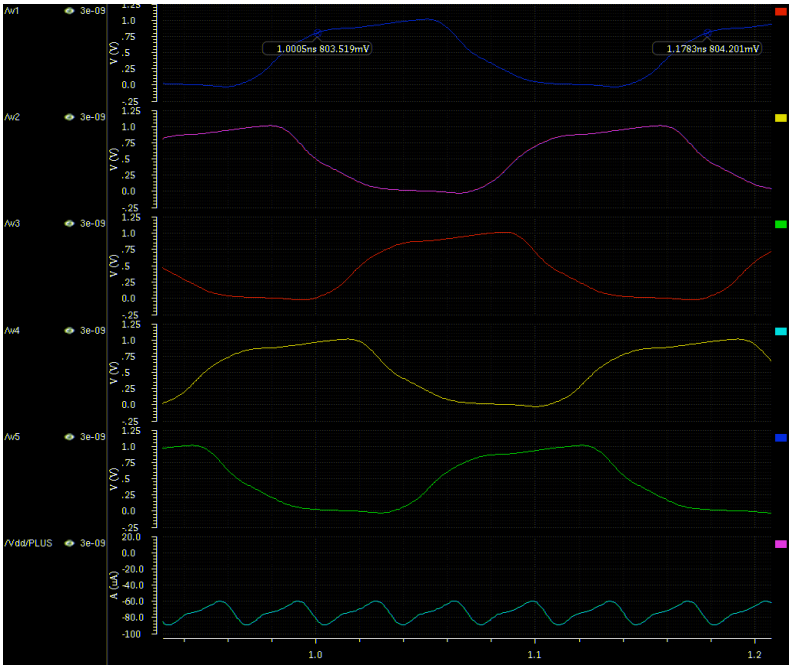

Figure A.30: 5 elements oscillating, SS-Inverter

The power consumption was **–79.09uA**.
A period took **–0.178ns**.

3 elements in oscillation

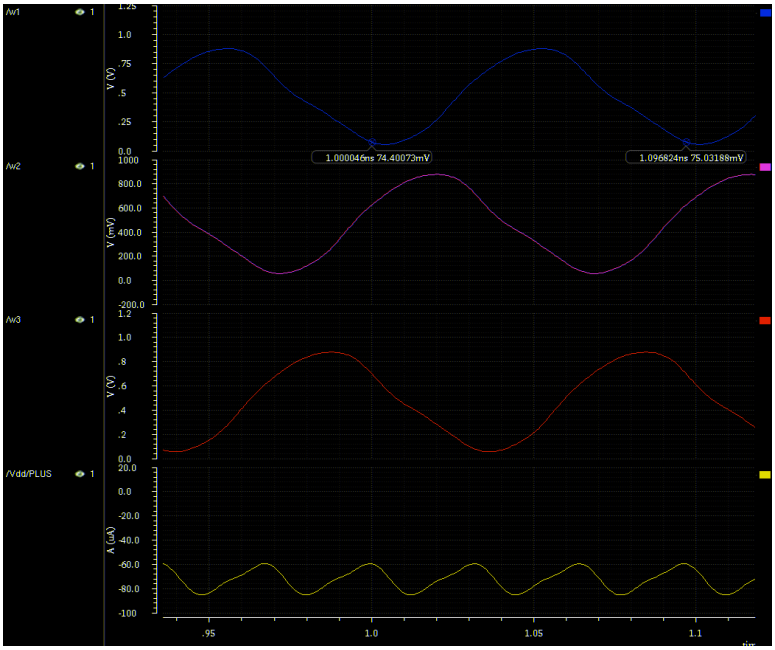

Figure A.31: 3 elements oscillating, NOR Design

The power consumption was **–72.33uA**.
A period took **–0.096ns**.

NAND Design

The setup for simulating the dynamic power consumption of the soft-supply inverter is shown in figure A.32 below.

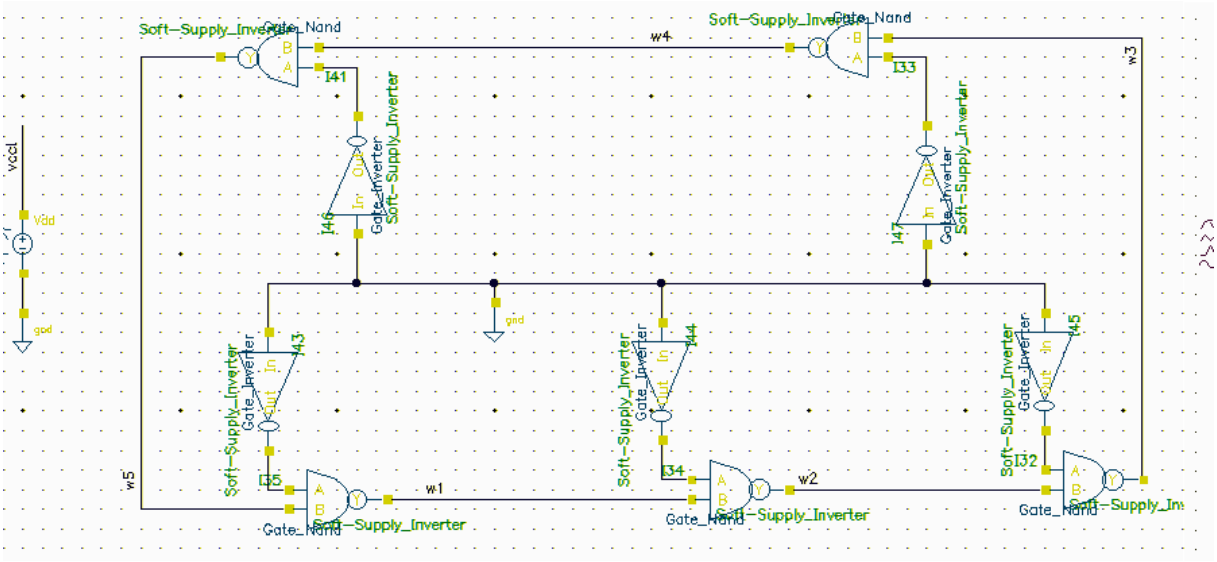

Figure A.32: Setup for Dynamic Power Consumption, NAND Design

The input B is set to a constant high input value while input A will continuously change as the circuit oscillates.

Figure A.33 shows the transient simulation.

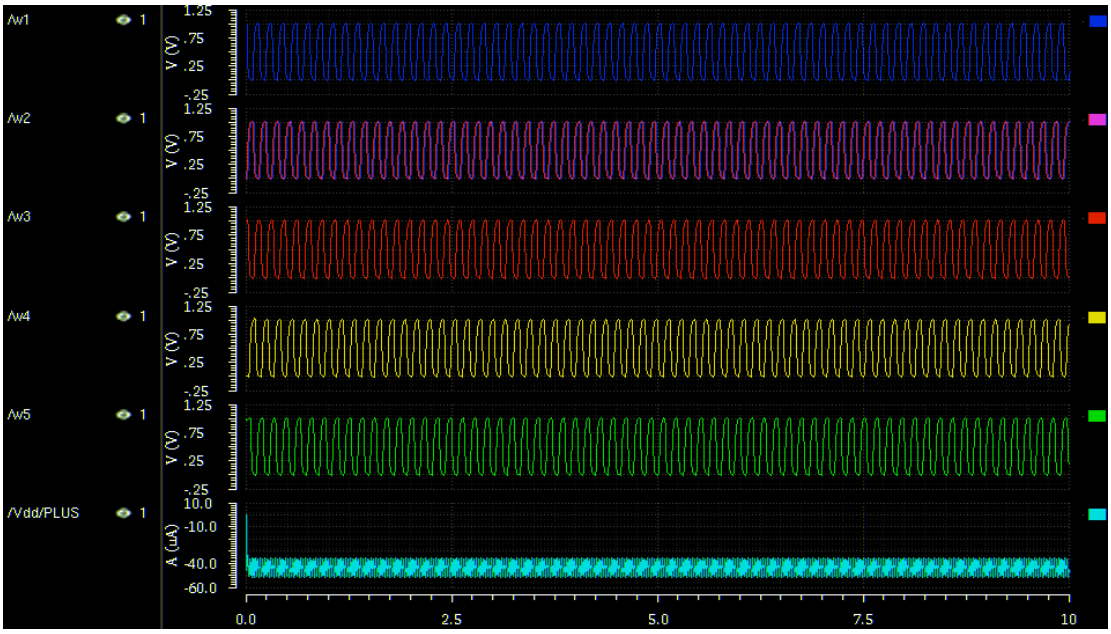

Figure A.33: Dynamic Power Consumption, NAND Design

The power consumption was **–44.22uA**.

Soft-Ground Inverter

The setup for simulating the dynamic power consumption of the soft-supply inverter is shown in figure A.34 below.

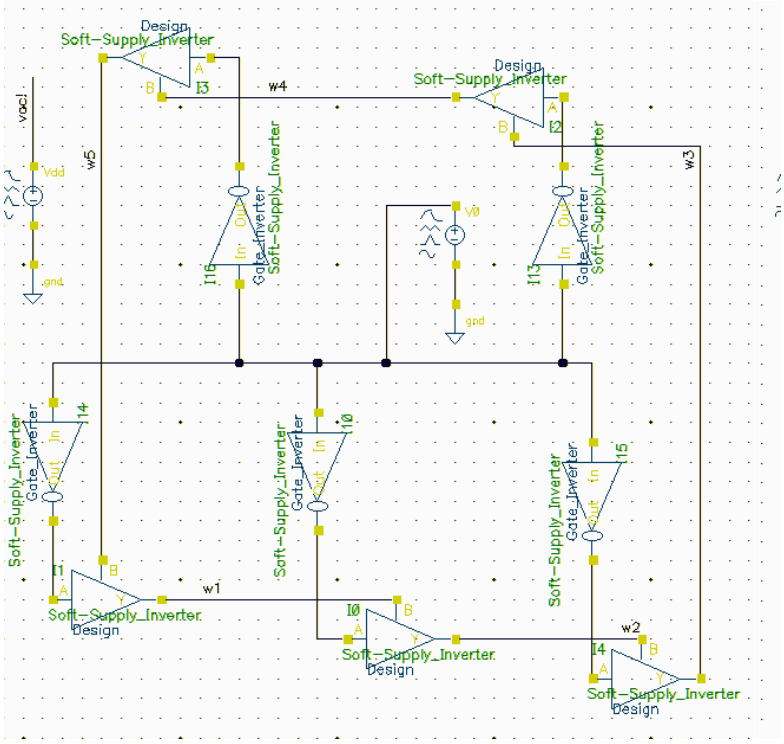

Figure A.34: Setup for Dynamic Power Consumption, SG-Inverter

The input B is set to a constant high input value while input A will continuously change as the circuit oscillates.

Figure A.35 shows the transient simulation.

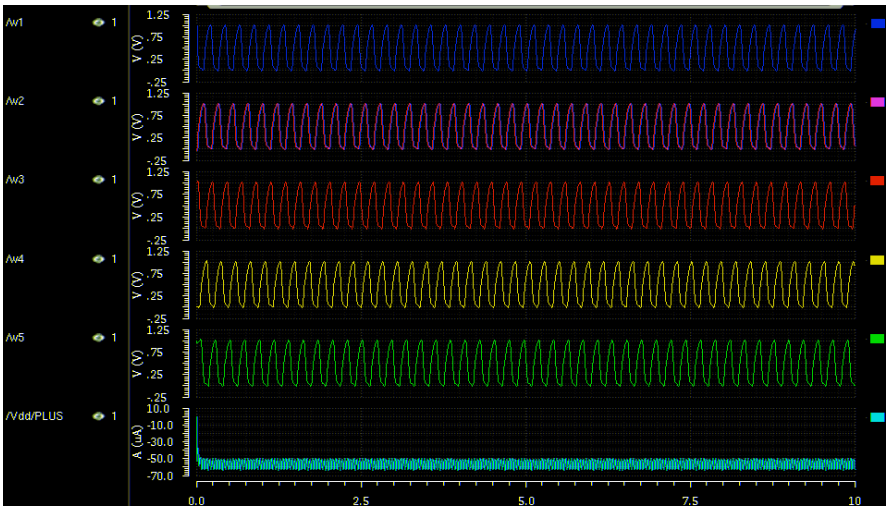

Figure A.35: Dynamic Power Consumption, SG-Inverter

The power consumption was **–56.28uA**.

Summmary

Because the results from simulating the soft-ground inverter revealed little to no improvement, it was not of any interest to experiment more with it.

The simulations are summarized in table A.3 and A.4.

Table A.3: Dynamic Power Consumption Summary

# Elements	NOR Design	SS-Inverter	NAND Design	SG-Inverter
5	**36.51uA**	74.09uA	**44.22uA**	56.28uA
3	**34.01nA**	72.33uA	-	-

Table A.4: Period of oscillation

# Elements	NOR Design	SS-Inverter	NAND Design	SG-Inverter
5	**0.156ns**	0.178ns	-	-
3	**0.081ns**	0.096ns	-	-

It would seem that the soft-supply inverter is slower than the conventional design. Additionally, the dynamic power consumption is about twice as much (table A.5).

Table A.5: Dynamic Power Consumption

# Elements	NOR Design	SS-Inverter
5	**5.70fJs**	13.2fJs
3	**2.75fJs**	6.94fJs

Appendix B – VerilogA Scripts

Here are the VerilogA codes used in the work.

B.1 DAC

```
// VerilogA for SAR_ADC, DAC, veriloga


`include "constants.vams"
`include "disciplines.vams"

module DAC(CLK, Vn, Vp, Mux_A, Mux_B, Vcm, Vinn, Vinp, Vrefn, Vrefp,
sample);
output Vn;
electrical Vn;

output Vp;
electrical Vp;

input CLK;
electrical CLK;

input [0:7] Mux_A;
electrical [0:7] Mux_A;

input [0:7] Mux_B;
electrical [0:7] Mux_B;

input Vcm;
electrical Vcm;

input Vinn;
electrical Vinn;
input Vinp;
electrical Vinp;

input Vrefn;
electrical Vrefn;
input Vrefp;
electrical Vrefp;

input sample;
electrical sample;

parameter real delay = 2n;
parameter real trise = 1n;
parameter real tfall = 1n;

//Output from DAC
real VN, VP;

//Temporary values used for bit-cycling
real VIN, VIP;

//Input pins
real vn0, vn1, vn2, vn3, vn4, vn5, vn6, vn7,
      vp0, vp1, vp2, vp3, vp4, vp5, vp6, vp7;

analog begin
```

```
// Sampling
@(cross( V(sample) - 0.5, -1)) begin
      VIN = V(Vinn);
      VIP = V(Vinp);
end

      // Bit0
      if ( V(Mux_A[0]) < 0.5 ) begin
            if ( V(Mux_B[0]) < 0.5 ) begin      // AB = 00;
                  vn0 = 0;
                  vp0 = 0;
            end else begin                      // AB = 01;
                  vn0 = 0;
                  vp0 = 1;
            end
      end else begin
            if ( V(Mux_B[0]) < 0.5 ) begin      // AB = 10;
                  vn0 = 1;
                  vp0 = 0;
            end else begin                      // AB = 11;
//                VN = V(Vinn);
//                VP = V(Vinp);
            end
      end

      // Bit1
      if ( V(Mux_A[1]) < 0.5 ) begin
            if ( V(Mux_B[1]) < 0.5 ) begin      // AB = 00;
                  vn1 = 0;
                  vp1 = 0;
            end else begin                      // AB = 01;
                  vn1 = 0;
                  vp1 = 1;
            end
      end else begin
            if ( V(Mux_B[1]) < 0.5 ) begin      // AB = 10;
                  vn1 = 1;
                  vp1 = 0;
            end else begin                      // AB = 11;
//                VN = V(Vinn);
//                VP = V(Vinp);
            end
      end

      // Bit2
      if ( V(Mux_A[2]) < 0.5 ) begin
            if ( V(Mux_B[2]) < 0.5 ) begin      // AB = 00;
                  vn2 = 0;
                  vp2 = 0;
            end else begin                      // AB = 01;
                  vn2 = 0;
                  vp2 = 1;
            end
      end else begin
            if ( V(Mux_B[2]) < 0.5 ) begin      // AB = 10;
                  vn2 = 1;
                  vp2 = 0;
            end else begin                      // AB = 11;
//                VN = V(Vinn);
//                VP = V(Vinp);
```

```
            end
      end


      // Bit3
      if ( V(Mux_A[3]) < 0.5 ) begin
            if ( V(Mux_B[3]) < 0.5 ) begin      // AB = 00;
                  vn3 = 0;
                  vp3 = 0;
            end else begin                      // AB = 01;
                  vn3 = 0;
                  vp3 = 1;
            end
      end else begin
            if ( V(Mux_B[3]) < 0.5 ) begin      // AB = 10;
                  vn3 = 1;
                  vp3 = 0;
            end else begin                      // AB = 11;
//                VN = V(Vinn);
//                VP = V(Vinp);
            end
      end

      // Bit4
      if ( V(Mux_A[4]) < 0.5 ) begin
            if ( V(Mux_B[4]) < 0.5 ) begin      // AB = 00;
                  vn4 = 0;
                  vp4 = 0;
            end else begin                      // AB = 01;
                  vn4 = 0;
                  vp4 = 1;
            end
      end else begin
            if ( V(Mux_B[4]) < 0.5 ) begin      // AB = 10;
                  vn4 = 1;
                  vp4 = 0;
            end else begin                      // AB = 11;
//                VN = V(Vinn);
//                VP = V(Vinp);
            end
      end

      // Bit5
      if ( V(Mux_A[5]) < 0.5 ) begin
            if ( V(Mux_B[5]) < 0.5 ) begin      // AB = 00;
                  vn5 = 0;
                  vp5 = 0;
            end else begin                      // AB = 01;
                  vn5 = 0;
                  vp5 = 1;
            end
      end else begin
            if ( V(Mux_B[5]) < 0.5 ) begin      // AB = 10;
                  vn5 = 1;
                  vp5 = 0;
            end else begin                      // AB = 11;
//                VN = V(Vinn);
//                VP = V(Vinp);
            end
      end
```

```
      // Bit6
      if ( V(Mux_A[6]) < 0.5 ) begin
            if ( V(Mux_B[6]) < 0.5 ) begin      // AB = 00;
                  vn6 = 0;
                  vp6 = 0;
            end else begin                      // AB = 01;
                  vn6 = 0;
                  vp6 = 1;
            end
      end else begin
            if ( V(Mux_B[6]) < 0.5 ) begin      // AB = 10;
                  vn6 = 1;
                  vp6 = 0;
            end else begin                      // AB = 11;
//                VN = V(Vinn);
//                VP = V(Vinp);
            end
      end

      // Bit7
      if ( V(Mux_A[7]) < 0.5 ) begin
            if ( V(Mux_B[7]) < 0.5 ) begin      // AB = 00;
                  vn7 = 0;
                  vp7 = 0;
            end else begin                      // AB = 01;
                  vn7 = 0;
                  vp7 = 1;
            end
      end else begin
            if ( V(Mux_B[7]) < 0.5 ) begin      // AB = 10;
                  vn7 = 1;
                  vp7 = 0;
            end else begin                      // AB = 11;
//                VN = V(Vinn);
//                VP = V(Vinp);
            end
      end

      VN = VIN - vn0*pow(0.5,1)*V(Vrefp)
                        - vn1*pow(0.5,2)*V(Vrefp)
                        - vn2*pow(0.5,3)*V(Vrefp)
                        - vn3*pow(0.5,4)*V(Vrefp)
                        - vn4*pow(0.5,5)*V(Vrefp)
                        - vn5*pow(0.5,6)*V(Vrefp)
                        - vn6*pow(0.5,7)*V(Vrefp)
                        - vn7*pow(0.5,8)*V(Vrefp);

      VP = VIP - vp0*pow(0.5,1)*V(Vrefp)
                        - vp1*pow(0.5,2)*V(Vrefp)
                        - vp2*pow(0.5,3)*V(Vrefp)
                        - vp3*pow(0.5,4)*V(Vrefp)
                        - vp4*pow(0.5,5)*V(Vrefp)
                        - vp5*pow(0.5,6)*V(Vrefp)
                        - vp6*pow(0.5,7)*V(Vrefp)
                        - vp7*pow(0.5,8)*V(Vrefp);

V(Vn) <+ transition( VN, delay, trise, tfall );
V(Vp) <+ transition( VP, delay, trise, tfall );

end //analog end
endmodule
```

B.2 Comparator

```
// VerilogA for SAR_ADC, Comparator, veriloga

`include "constants.vams"
`include "disciplines.vams"


module Comparator(vn, vp, clk, out);
input vn, vp, clk;
output out;

electrical vn, vp, clk, out;

parameter real supply = 1.0;
parameter real delay = 2n;
parameter real trise = 1n;
parameter real tfall = 1n;

integer VOUT;

analog begin

@(cross( V(clk) - 0.5, -1)) begin
      if ( V(clk) < 0.5) begin
            VOUT = supply*( V(vn) < V(vp) );
      end
end

if ( V(clk) > 0.5) VOUT = 0;

V(out) <+ transition( VOUT, delay, trise, tfall );

end //analog end

endmodule
```

B.3 Ideal ADC

```
// VerilogA for SAR_ADC, Ideal_ADC, veriloga

`include "constants.vams"
`include "disciplines.vams"

module Ideal_ADC(Input, Done, Output);

input Input;
electrical Input;

input Done;
electrical Done;

output [0:8] Output;
electrical [0:8] Output;

parameter real trise = 1n;
parameter real tfall = 1n;
parameter real delay = 2n;

real sampl;

integer Vout[0:8];
integer i;

analog begin

@(cross( V(Done) - 0.5, -1)) begin
      sampl = V(Input);
      for ( i=0;i<=8;i=i+1) begin
            if( (sampl-pow(0.5,i+1)) > 0 ) begin
                  Vout[i] = 1;
                  sampl = sampl - pow(0.5,i+1);
            end else begin
                  Vout[i] = 0;
            end
      end
end

V(Output[0]) <+ transition(Vout[0], trise, tfall, delay);
V(Output[1]) <+ transition(Vout[1], trise, tfall, delay);
V(Output[2]) <+ transition(Vout[2], trise, tfall, delay);
V(Output[3]) <+ transition(Vout[3], trise, tfall, delay);
V(Output[4]) <+ transition(Vout[4], trise, tfall, delay);
V(Output[5]) <+ transition(Vout[5], trise, tfall, delay);
V(Output[6]) <+ transition(Vout[6], trise, tfall, delay);
V(Output[7]) <+ transition(Vout[7], trise, tfall, delay);
V(Output[8]) <+ transition(Vout[8], trise, tfall, delay);

end //analog

endmodule
```

B.4 Ideal ADC

```
// VerilogA for SAR_ADC, Ideal_DAC, veriloga

`include "constants.vams"
`include "disciplines.vams"

module Ideal_DAC(Input, Done, Output);

input [0:8] Input;
electrical [0:8] Input;

input Done;
electrical Done;

output Output;
electrical Output;

real OUT;
real v0;
real v1;
real v2;
real v3;
real v4;
real v5;
real v6;
real v7;
real v8;

parameter real tfall = 1n;
parameter real trise = 1n;
parameter real delay = 2n;

analog begin

v0 = V(Input[0]);
v1 = V(Input[1]);
v2 = V(Input[2]);
v3 = V(Input[3]);
v4 = V(Input[4]);
v5 = V(Input[5]);
v6 = V(Input[6]);
v7 = V(Input[7]);
v8 = V(Input[8]);

@(cross(V(Done) - 0.5, +1)) begin
      if (V(Done) > 0.5) begin
            OUT = 0.5*v0 + pow(0.5,2)*v1 + pow(0.5,3)*v2 + pow(0.5,4)*v3 +
pow(0.5,5)*v4 + pow(0.5,6)*v5 + pow(0.5,7)*v6 + pow(0.5,8)*v7 +
pow(0.5,9)*v8;
            //$display("", v0, v1, v2, v3, v4, v5, v6, v7, v8);
      end
end

V(Output) <+ transition(OUT,trise, tfall, delay);

end // analog

endmodule
```