Interval Bisection Quantization Circuit for an 8-bit Analog to Digital Converter

A Thesis

Presented to

the Faulty of the Department of Electrical and Computer Engineering

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in Electrical Engineering

by

Colin Taylor

December 2012

Interval Bisection Quantization Circuit for an 8-bit Analog to Digital Converter

_____
Colin Taylor

Approved:

_____
Chair of the Committee
E. Joe Charlson, Professor,
Electrical and Computer Engineering

Committee Members:

_____
Wanda Wosik, Associate Professor,
Electrical and Computer Engineering

_____
Rajeev Rajan Pillai, Post Doctoral Fellow,
Department of Physics

_____
Suresh K. Khator, Associate Dean,
Cullen College of Engineering

_____
Badrinath Roysam, Professor and Chair,
Electrical and Computer Engineering

Acknowledgements

I would first like to thank my advisor, Dr. Charlson, for inspiring me to pursue VLSI design both as a field of study and a career path. I would also like to thank all my professors for providing me with the necessary background knowledge for this project. Finally, I would like to thank my family without whose support I could not have made it through my studies.

Interval Bisection Quantization Circuit for an 8-bit Analog to Digital Converter

An Abstract

of a

Thesis

Presented to

the Faulty of the Department of Electrical and Computer Engineering

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in Electrical Engineering

by

Colin Taylor

December 2012

Abstract

A common type of analog to digital converter (ADC) is called the successive approximation ADC. It works by setting each bit in the digital output code individually and thus its conversion time is determined by the clock frequency and the resolution of the converter. A speed improvement may be made by using a quantization circuit that sets the entire output code at once. The goal of this project is to design such a quantization circuit based on the interval bisection algorithm. The circuit was designed using Cadence Schematic and Virtuoso and was simulated using spice. Spice simulations show that, on average, the circuit converts faster than the conventional successive approximation converter.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

One type of analog to digital converter (ADC) is called the successive approximation ADC. This ADC works by first assuming a value for the most significant bit (MSB) in a binary number and then sending that word to a digital to analog converter (DAC). The resulting analog signal is then compared to the input analog signal using a comparator. The value of the MSB will then remain the same or be changed based on the comparator output. This procedure will be repeated for each bit until all the bits in the binary number have been set and the conversion process is completed [1].

This means that the number of steps it takes to be sure that the binary number is correct is equal to the number of bits in the digital number. This can be improved upon by designing a converter that assumes an initial value for, and then manipulating based on comparator output, the entire binary number. Successfully designing, having fabricated, and testing such a converter is the goal of this project.

It should be noted that a proper ADC consists of four parts. The first is an antialiasing filter that prevents the higher frequency aliases of the input signal, produced by sampling, from being input into the ADC. The second is a sample and hold circuit that takes the value of the input signal when the conversion

process begins and holds it constant for the duration of the process. The third is the quantizer which divides the reference voltage into 2^N segments, where N is the resolution of the converter (the number of bits in the output digital number), and find the one that most closely represents the value of the input. The fourth is the encoder which outputs a binary number based on the segment selected by the quantizer [1]. This project will only be concerned with the quantizer and encoder parts, but as will be shown, the quantization and encoding will occur as part of the same process.

1.2 The Interval Bisection Algorithm

The interval bisection algorithm, as it applies to this project, may be stated as follows. First, the interval is defined as being [0,255] with 127 being the midpoint, only integer values are allowed. Next, the value is compared with the input and the interval either becomes [127,255] or [0,127] depending on whether the analog value corresponding to 127 is higher or lower than the input value. This process will continue until the analog value corresponding to the interval midpoint is within one least significant bit (LSB) of the input.

This algorithm is a good choice for implementing an ADC because the electronic components needed are relatively straightforward, and the algorithm gives good performance. The number of steps required in the worst case scenario is equal to the resolution of the converter. The best case scenario is when the initial midpoint, equal to $2^N - 1$, is the best fit for the analog input signal. In this case, only one step is required for conversion. Figure 1-1 shows

the average conversion times of the proposed converter vs. the conversion times of a conventional successive approximation converter at different resolutions.

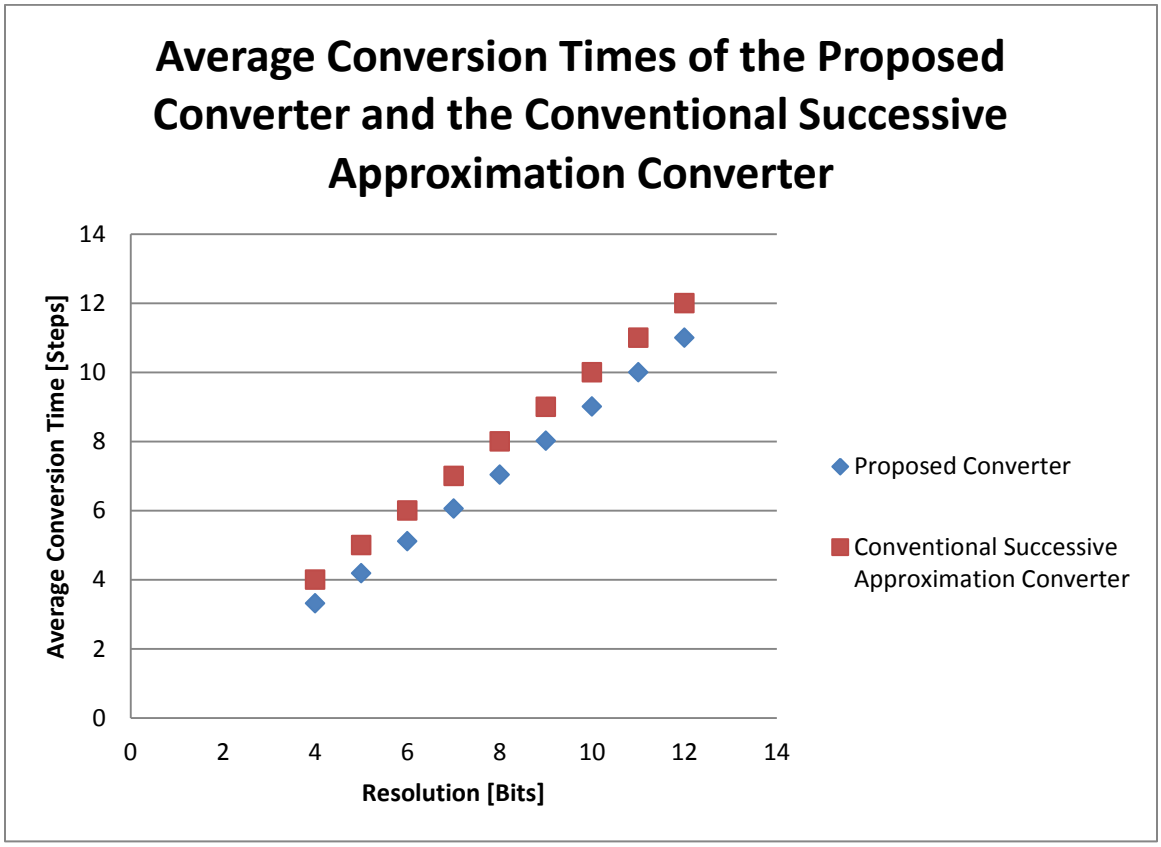

Figure 1-1: Average conversion times of proposed converter and conventional successive approximation converter.

Figure 1-1 shows that the proposed converter consistently converts faster than the conventional successive approximation converter. The Perl script used to obtain this data can be found in appendix A.

1.3 System Overview

The components required for the circuit are as follows: multiplexors to select an input to the registers based on comparator output, registers to hold the left and right endpoints of the interval, an 8-bit binary adder whose output will be

shifted to the right by one bit to accomplish the division by two, a DAC, two comparators, digital logic to test for convergence, and a register to hold the output number. Figure 1-2 shows a block diagram of the system.

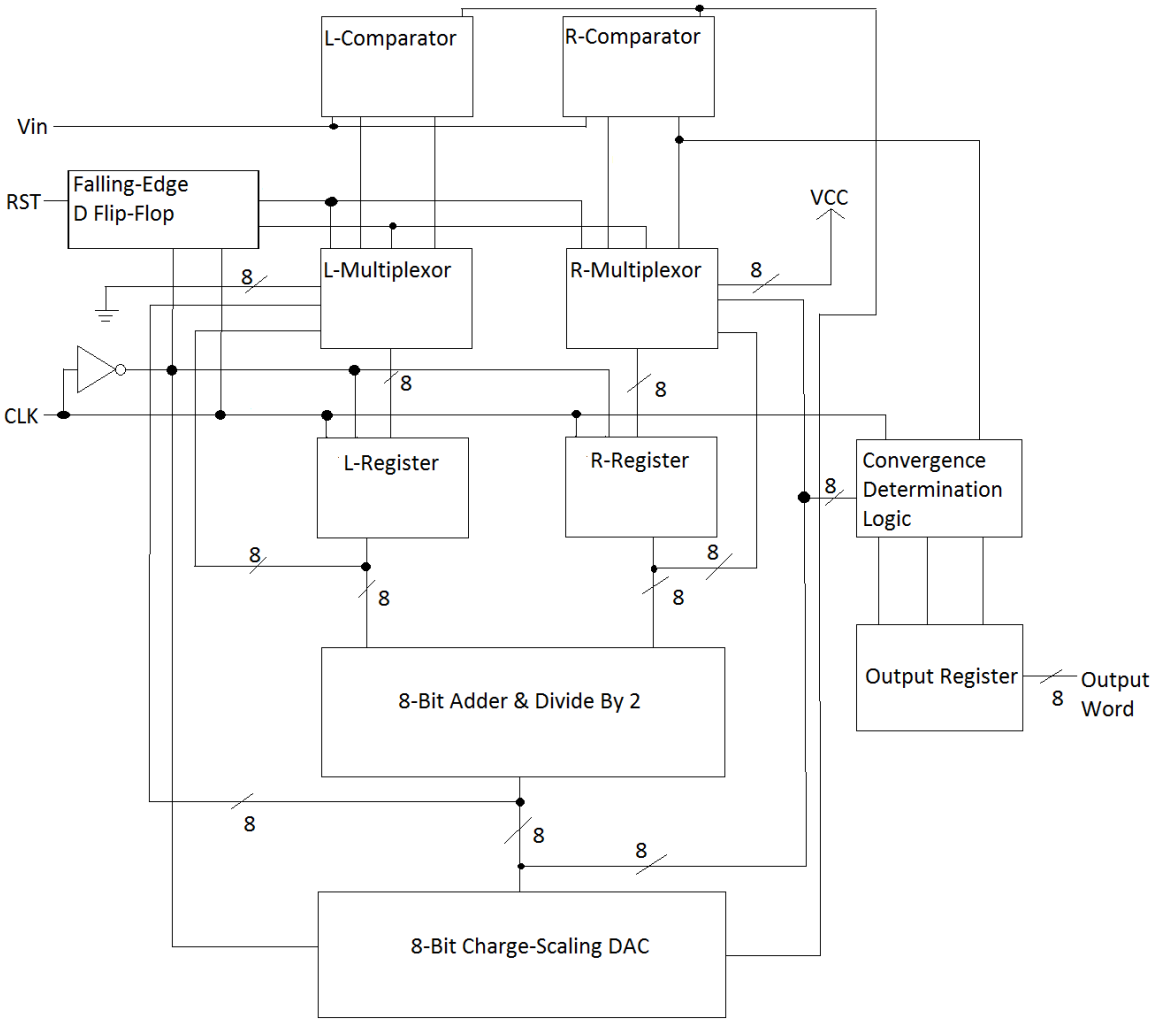

Figure 1-2: The block diagram of the proposed converter.

In this circuit the reference voltage will be the supply voltage, V_{CC}, for simplicity. Each of the parts, or in the case of the registers their constituent flip-flops, will have a chapter dedicated to it later in this thesis.

For now however, the best way to understand the circuit is to go through one complete conversion cycle. First, the circuit is reset and the number stored in the L-register is 0 and the number in the R-register is 255. The adder then adds these two numbers and the result is divided by two to obtain the midpoint. Since only unsigned integers are used, the division can be accomplished by making the final carry out the MSB and discarding the LSB, and thus the result is divided by two. Next, the result is sent to the DAC where it is converted into an analog signal. This analog signal is sent to the L and R comparators.

Let the analog signal from the DAC and the input signal be called v_D and v_I respectively. Also, let the output of the L-comparator be called HL and the output of the R-comparator be called HR. With these definitions the comparators will behave like this:

$$\text{HL} = 0,\ v_I < v_D - \text{LSB}/2, \tag{1 – 1}$$

$$\text{HL} = 1,\ v_I > v_D - \text{LSB}/2, \tag{1 – 2}$$

$$\text{HR} = 0,\ v_I < v_D + \text{LSB}/2, \tag{1 – 3}$$

$$\text{HR} = 1,\ v_I < v_D + \text{LSB}/2. \tag{1 – 4}$$

If both comparator outputs are 1 the L-register will be set to the midpoint of the interval and the R-register will remain unchanged. If both comparator outputs are 0 the R-register will be set to the midpoint of the interval and the L-register will remain unchanged. Then the process will begin again with the new interval and continue until HL is 1 and HR is 0 indicating that the conversion is complete.

Notice that when this condition occurs, both registers will be set to the midpoint and thus it will be preserved at the output of the addition and divide by two circuits.

HL = 1 and HR = 0 is one condition that the convergence determination logic will test for. The other is a special case that arises when the correct output code is 255. Since any fractional part of the result of division by two will be lost it is impossible for the addition and division by two circuits to output 255. Therefore, the convergence determination logic will also need to test for the condition

$$Y = HR^{*}(\text{output} = 254), \qquad (1 - 5)$$

where the * indicates the logical AND operation. When this condition is true, the convergence determination logic will output a special signal to change the midpoint from 254 to 255.

The convergence determination logic will also need to take into account the fact that the comparator output is not valid for the entire clock cycle because the DAC output is only valid for half of the clock cycle. Once the convergence determination logic outputs a 1 the output register is set to the midpoint and the conversion cycle ends.

Chapter 2

Multiplexors and Flip-Flops

2.1 Introduction

This chapter will cover the multiplexors and D flip-flop (DFF) circuits used to implement the registers. The purpose of the multiplexors is to select the appropriate input to the registers containing the left and right endpoints of the interval. The DFFs will also be used to implement the output register. Both circuits will use standard CMOS logic.

2.2 The Multiplexor Circuit

The purpose of a multiplexor is to select one signal from many to pass to the output. The multiplexors used in this project will have three possible inputs to pass to the output. For the multiplexors that control the input to the left endpoint register, the possible inputs are: 0 upon reset, the previous value contained in the left endpoint register if the output of the left comparator is low, and the midpoint if the output of the left comparator is high. For the multiplexors that control the input to the right endpoint register, the possible inputs are: 1 upon reset, the midpoint if the output of the right comparator is low, and the previous value contained in the right endpoint register if the output of the right comparator is high.

The multiplexor was designed by picturing it as a series of digitally controlled switches. There are multiple ways to implement switches in CMOS

technology. The simplest is to use a single N-channel transistor with the appropriate digital control signal connected to its gate. This would be a good option for saving area, but overall it is a poor choice for two reasons.

The first issue is simply speed. During part of the time when the output is switching the transistor will be in the triode region of operation and thus less current will be available to charge or discharge the load capacitance at the output.

The second issue is the so called “poor 1” problem. This arises when the output is changing from a 0 to a 1. In this situation the gate and input are connected to the positive supply voltage (a logical “1”), thus the input acts as the drain, and the output will initially be at 0V acting as the source. When the voltage difference between the gate and the output drops below the threshold voltage of the transistor, the transistor turns off and the output is held at a value below the positive supply voltage. This value can be found by solving the following equation:

$$V_O = V_{CC} - [V_{thN} + \gamma_N(\sqrt{V_O + 2\theta_F} - \sqrt{2\theta_F})], \qquad (2-1)$$

where V_O is the output voltage, V_{CC} is the positive supply voltage, V_{thN} is the threshold voltage of the transistor, γ_N is the body effect parameter of the transistor, and θ_F is the Fermi potential of the transistor. Equation 2-1 shows that the output value can be significantly less than a normal logical “1” which can cause problems with noise margins and driving subsequent logic gates [2].

The CMOS transmission gate is made of an N-channel transistor and a P-channel transistor connected in parallel. The gate of the N-channel device is connected to the digital control signal and the gate of the P-channel device is connected to its complement. The CMOS transmission gate addresses the speed issue by providing more current for charging or discharging the output capacitance (another way to think of it is reducing the resistance of the switch and thus the time constant of the charging or discharging process). It addresses the “poor 1” issue because when the N-channel device turns off the P-channel device remains on and continues to charge the output capacitance until it reaches a logical “1”.

Figure 2-1 below shows the schematic of the multiplexor circuit.

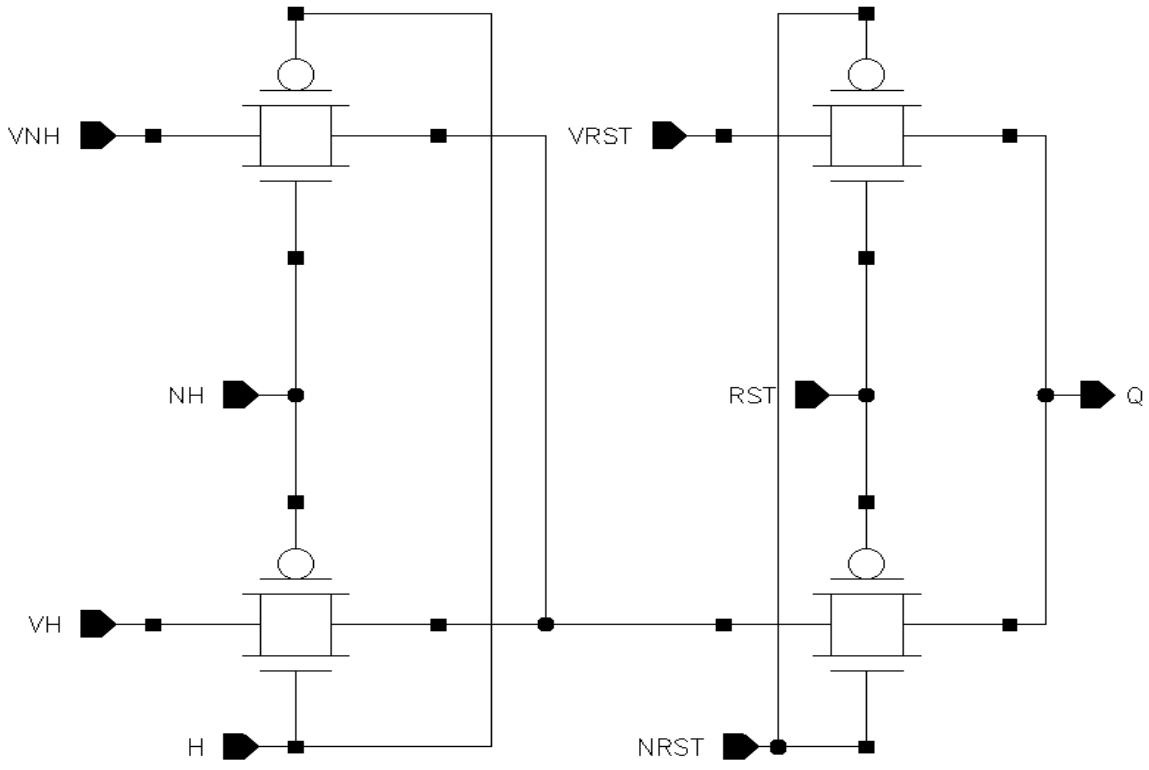

Figure 2-1: The schematic of the multiplexor circuit. Transistors with a circle on the gate are P-channel.

The signals represent the following: H is the output of the left or right comparator, NH is the complement of H, VH is the signal that will be passed if H is high, VNH is the signal that will be passed if H is low, RST is the reset signal, NRST is the complement of RST, VRST is the signal that will be passed if RST is high, and Q is the output. The first pair of transmission gates passes a signal based on whether H is high or not. This signal is then sent to another transmission gate that passes it if RST is low, otherwise the signal VRST will be passed.

2.3 Multiplexor Physical Layout and Simulation Results

Figure 2-2 shows the physical layout of the multiplexor circuit.

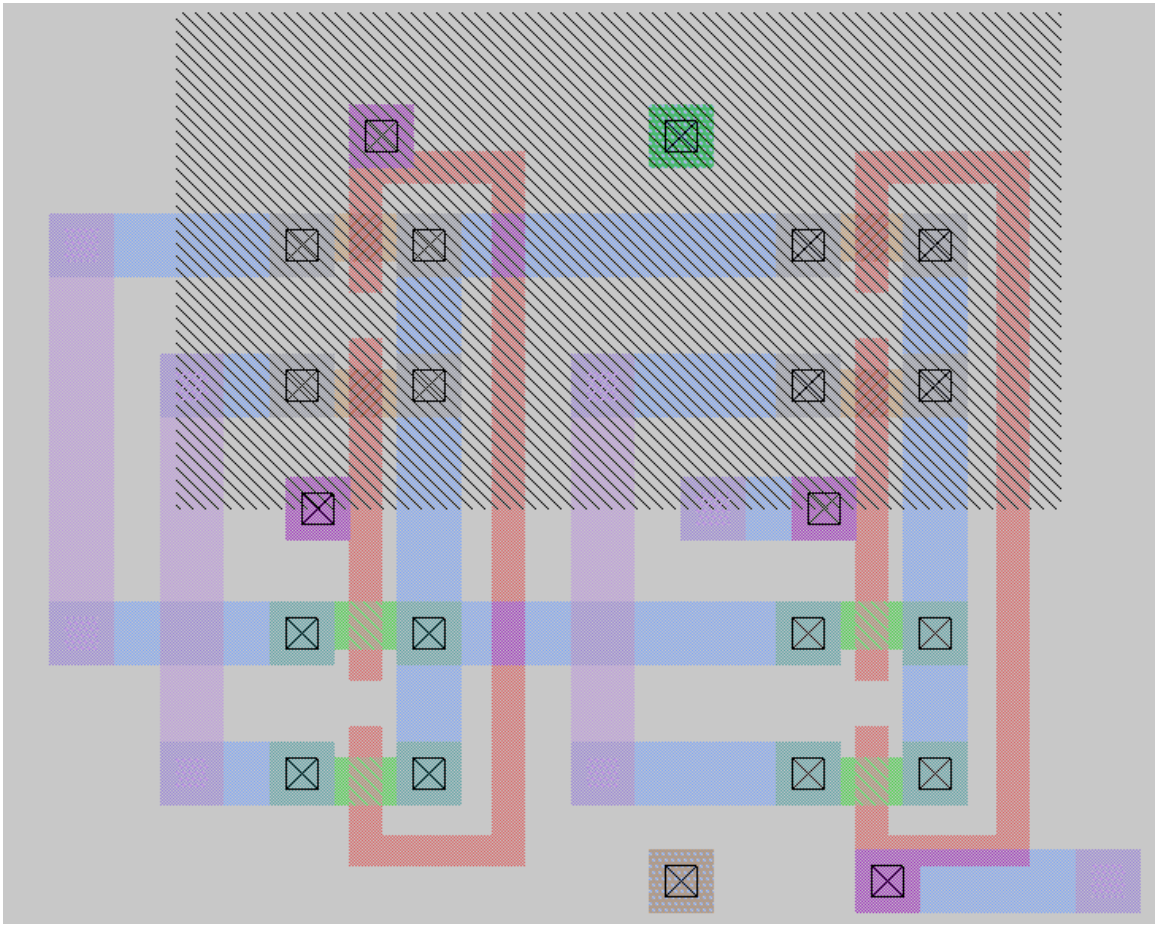

Figure 2-2: The multiplexor cell. All transistors are of minimum dimensions.

A layout of a certain circuit intended to be part of a larger system is often referred to as a cell by VLSI designers, therefore it is more appropriate to say that figure 2-2 is the multiplexor cell. This terminology will be used for the remainder of this chapter.

To minimize the area taken up by the circuit all transistors have the minimum dimensions allowed by the MOSIS scalable CMOS design rules:

$$W = 0.9\ [\mu m], \tag{2 – 2}$$

$$L = 0.6\ [\mu m]. \tag{2 – 3}$$

Additionally, the transmission gates are not laid out individually. Instead, the two transmission gates in each pair are interwoven with each other so that all the P-channel devices can be put in a single well.

The cell is also laid out to make connections with other cells easier. For example, the inputs of the first pair of transmission gates are connected with the second metal so that the control signals can be brought in from the comparators using the first metal. A similar technique is used for the control signals of the second pair, only the signals will be entering and leaving the cell in the vertical direction.

The two isolated objects are called wafer plugs and their purpose is to make sure that all the transistors in the cell are only connected in the ways defined by the metal and polycrystalline silicon layers. The top wafer plug is connected to the most positive potential in the circuit and the bottom plug is

connected to the most negative potential in the circuit. One of these will also be connected to the input of the second pair of transmission gates depending on whether that multiplexor controls a bit of the left endpoint register or the right endpoint register.

The simulation results are shown below in figure 2-3.

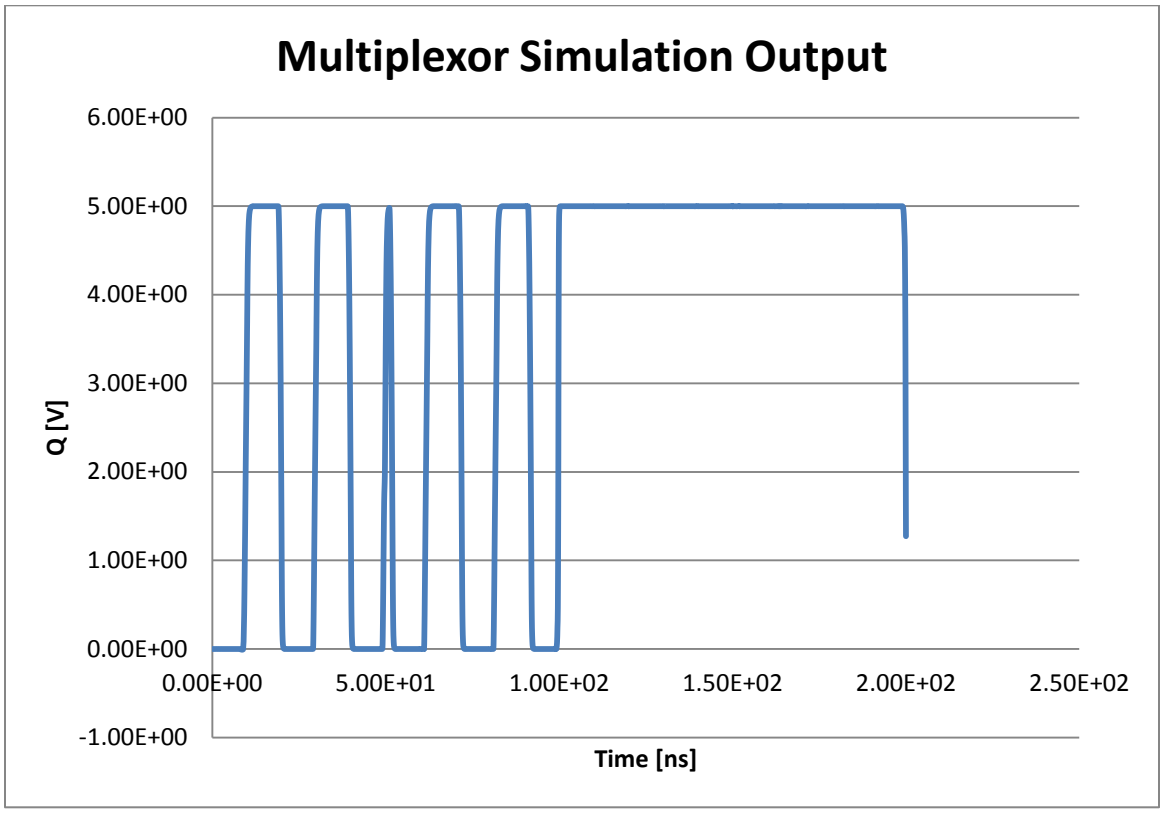

Figure 2-3: Simulation results for the multiplexor cell.

The SPICE code used to obtain this data was extracted from the cell layout and may be found in appendix B. In this simulation VH and VNH are two square waves that are out of phase by 180 degrees and VRST is a constant 5V. The control signal H is initially low, but switches every 50ns. The control signal RST is also initially low, but goes high at 100ns. The output reflects these control

signals. For the first 50ns the signal VNH is passed, then for the next 50ns VH is passed. For the next 100ns (the remainder of the simulation) the VRST signal is passed. Therefore, the multiplexor cell was designed and laid out correctly.

A word about the transistor models and simulation methodology is in order before continuing on to flip-flops. Cadence provides transistor models for simulations, but these models are outdated. Therefore, it is more appropriate to use more recent transistor models. In order to make these models compatible with the version of SPICE being used the LEVEL parameter must be changed to 8 and the VERSION parameter to 3.2. Other than that they are unchanged. These models were obtained from the MOSIS datasheet of the July 2012 run of the AMI C5 process, the same process that will be used to fabricate this project. These models will be used in all simulations for this writing.

The results were produced by running the simulation in SPICE and then sending the data to a Microsoft Excel spreadsheet for plotting. This approach will be used in all simulations in this writing.

2.4 The Flip-Flop Circuit

The purpose of a DFF is to store a single bit of data over the course of a clock cycle. A group of flip-flops comprise a register. The DFF is designed to be edge triggered, that is the bit at the input will only be stored in the flip flop when the clock signal goes from low to high or high to low. This ensures that the output does not track the input during a significant part of the clock cycle as this may cause issues with the subsequent digital circuitry [2].

The most basic form of the DFF is the master/slave configuration shown in figure 2-4.

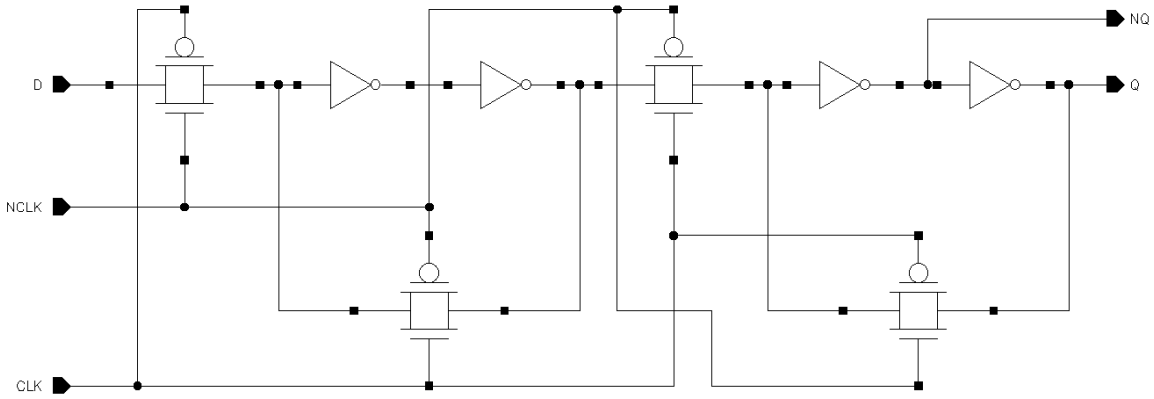

Figure 2-4: A master/slave DFF implemented with CMOS inverters and transmission gates.

The signals are as follows: D is the data to be put into the flip-flop, CLK is the clock signal, NCLK is the complement of the clock signal, Q is the output, and NQ is the complement of the output. This DFF is made up of two sub-circuits called latches. The latch on the left is the master and the latch on the right is the slave.

A latch operates in the following manner. During one half of the clock cycle the feedback loop is open and the output tracks the input. During the second half of the clock cycle the input is disconnected from the first inverter and the feedback loop is closed thus holding the output at the previous value of the input.

To understand the operation of the circuit first assume that the CLK is low. In this state the master is tracking the input with its feedback loop closed, and the input to the slave is disconnected from the output of the master. The slave's

feedback loop is closed, holding the value of the previous input. Then CLK goes high. When this happens the master is disconnected from the input and its feedback loop is closed, preserving the value of the input when CLK went high at its output. At the same time the input of the slave is connected to the output of the master and its feedback loop is opened, thus tracking the held output of the master [2]. In order to make a falling edge DFF, all one need do is exchange the CLK and NCLK inputs.

2.5 The Flip-Flop Cell and Simulation Results

The cell for the flip-flop is shown in figure 2-5.

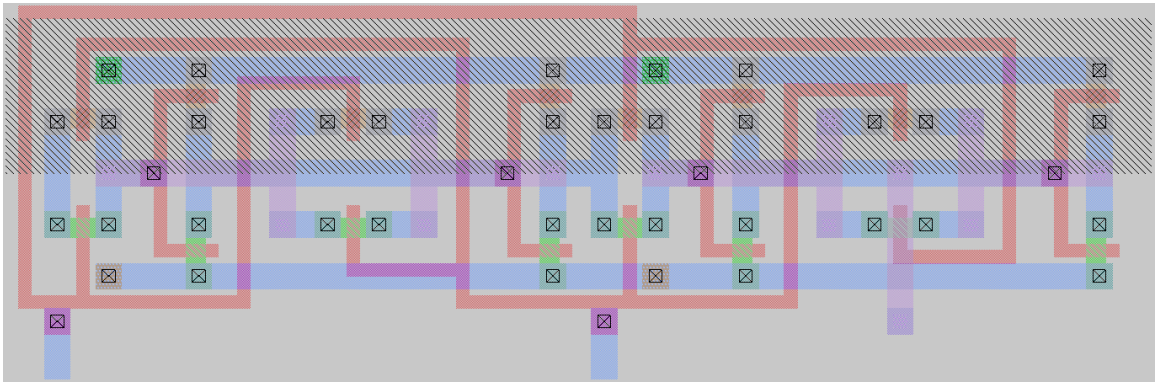

Figure 2-5: The cell for the rising edge DFF.

This cell has also been designed to minimize area and make connections easier. Therefore, all transistors once again have the minimum dimensions allowed by the process design rules. Also, the CLK and NCLK inputs (NCLK is the input on the bottom left and CLK is on the bottom at the center) have been designed so that a single second metal wire can be used to connect all CLK and NCLK inputs. The inputs and outputs of the transmission gates are in second metal so that the inverters in the latches can be connected using only the first metal. To make

metal based connections easier the CLK and NCLK signals are connected to all transmission gates using only polycrystalline silicon (shown in red). Normally connection with polycrystalline silicon is not a good idea because of its high resistivity relative to metal, but in this case it can be done because the only things being connected are gates and gates draw virtually no current.

The clock signal is shown in figure 2-6.

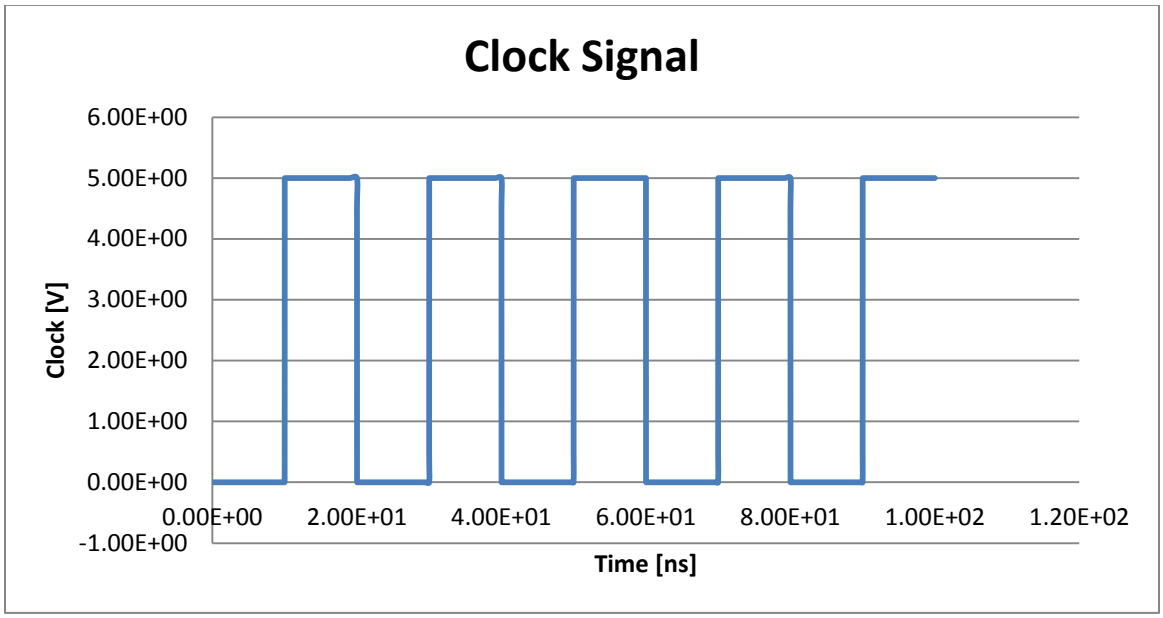

Figure 2-6: The clock signal for the rising edge DFF simulation.

Note that the rising clock edges are located at:

$$t = 10(1 + 2i)\ [ns], i = 0,1, \ldots, \qquad (2-4)$$

and the falling clock edges are located at:

$$t = 20i\ [ns], i = 1,2, \ldots. \qquad (2-5)$$

The simulation results for the rising edge DFF are located in figure 2-7, the input curve has diamond shaped figure on it and the output curve has square shaped figures on it.

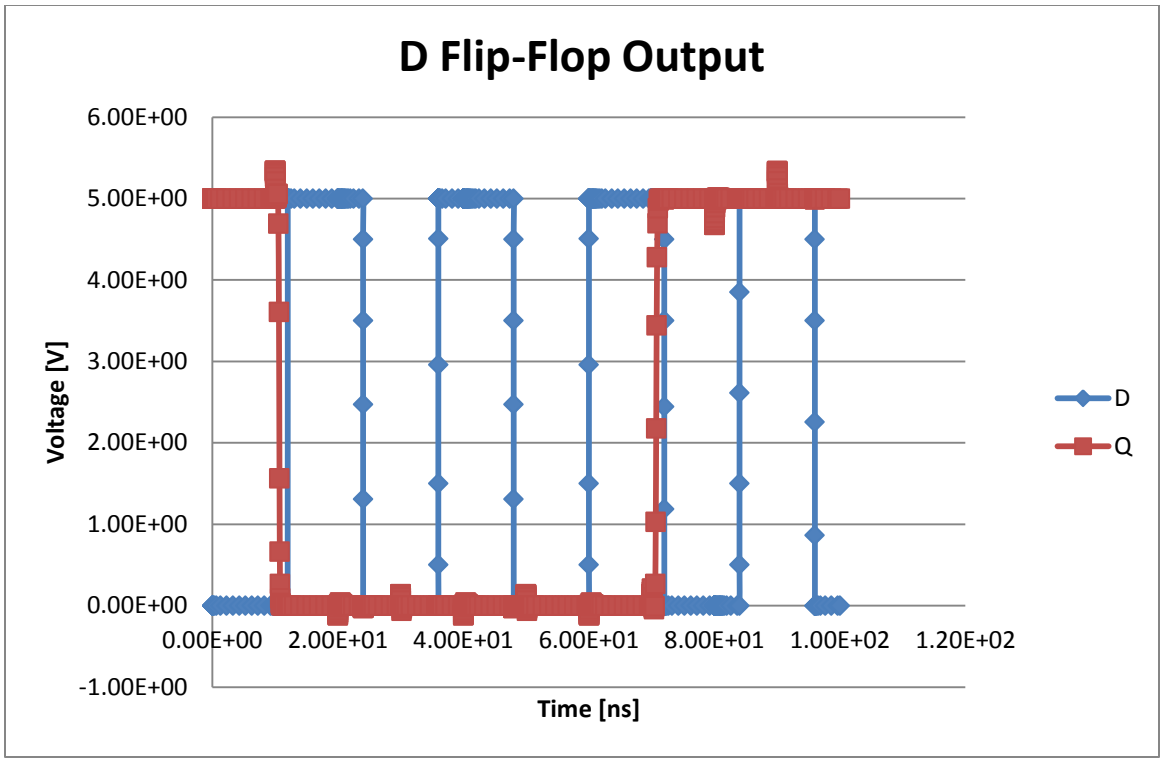

Figure 2-7: The output of the rising edge DFF simulation.

The SPICE code used to produce figures 2-6 and 2-7 was extracted from the cell layout and is shown in appendix B. This simulation uses the clock signal of figure 2-6 and D is a square wave that has a half period of 12ns. When the first rising edge occurs at 10ns D is low and so the DFF output is made low. For the next two rising edges D is low and so the output remains low. Then, at 70ns, D is high so the output goes high in response and remains high for the remainder of the simulation. These results show that the rising edge DFF cell was designed and laid out correctly.

Chapter 3

Addition and Division by Two Circuit

3.1 Introduction

This chapter focuses on the addition and division by two circuit. The purpose of this circuit is to compute the midpoint of the interval from the left and right endpoints of the interval stored in the left and right registers. The addition process is achieved by a CMOS adder with a fast carry scheme to allow for operation at higher speeds. The division by two is accomplished by discarding the LSB from the adder output and making the adder's final carry out the MSB, thus shifting the output to the right by one bit.

3.2 CMOS Logic Gate Design Process

A conventional CMOS logic gate is made of two networks: a pull up network (PUN) and a pull down network (PDN). The PUN is made of P-channel devices and is designed to connect the gate output to VCC (the power supply voltage) when the output of the logic function being implemented is a logical 1. Likewise, the PDN is made of N-channel devices and is designed to connect the gate output to ground when the output of the logic function being implemented is a logical 0 [2].

Now the question becomes how to obtain the PUN and PDN from the logic function the gate is supposed to carry out. The first step in this process is to invert the logic function. The resulting logic function will be 1 when the original

function is 0, and therefore gives the PDN. The PUN can be obtained either from the original logic function or using the duality property. The duality property refers to the fact that a series connection of a set of transistor with given control variables in one network implies that the transistors with the same control variables in the other network will be connected in parallel. If the PUN is obtained from the original logic function all variables must be complemented to account for the fact that the P-channel devices in the PUN will turn on when the voltage on their gates is low [2].

The goal when determining the size of the transistors in a PUN or a PDN is to make the time it takes to pull the output up or down, the circuit's drive capability, identical to a simple CMOS inverter in the worst case scenario. First, a given PDN or PUN may have multiple paths between the output and the voltage it will connect the output to so these paths must first be identified. Second, imagine any transistor that is conducting as a resistor. The value of this resistance is inversely proportional to the transistor's aspect ratio (its width divided by its length). Therefore, to make the drive capability of a given path equal to the CMOS inverter all of the transistors in that path must be made N times as wide as a transistor in the CMOS inverter, where N is the number of transistors in the path. It is also possible make the transistor length 1/N times that of a transistor in the CMOS inverter, but since all lengths are typically at the minimum length allowed by the process making the transistors N times wider is what is generally done [2].

One more thing must be considered in matching drive capability. Typically the carrier mobility of N-channel is much larger than that of the P-channel device. To compensate for this the P-channel devices would have to be made much larger than the N-channel devices and this size mismatch would result in the gate taking up additional space. A good compromise is to make both transistors in the basic inverter the minimum dimensions allowed by the process and then scale the transistors in the PDN and PUN according to this minimum size. This is the approach that will be followed for the adder circuits.

3.3 Adder Cell

Now that a procedure for designing CMOS logic gates has been presented, the adder cell can be designed. The truth table of the logic function for the adder cell is given in table 3-1 and it can be expressed as:

$$Y = C'[LR' + L'R] + C[L'R' + LR]. \qquad (3-1)$$

Table 3-1: Truth Table of Adder Cell

C	R	L	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

In Equation 3-1, Y is the output of the adder cell, L is the input bit from the L register, R is the input bit from the R register, and C is the carry out from the previous cell (more on carries later). In logic expressions the apostrophe

represents the complement of a variable, so L' represents the complement of L for example.

Since Equation (3-1) and Table (3-1) are given, the simplest way to obtain the PDN is by looking at Table 3-1 and obtaining a logic function from it. This gives:

$$\mathrm{PDN} = \mathrm{L'R'C'} + \mathrm{LRC'} + \mathrm{LR'C} + \mathrm{L'RC}. \qquad (3-2)$$

Equation (3-2) may be simplified to yield

$$\mathrm{PDN} = \mathrm{C'[L'R'} + \mathrm{LR]} + \mathrm{C[LR'} + \mathrm{L'R]}. \qquad (3-3)$$

Equation (3-3) is the logic function for the PDN. It is clear from Equation (3-3) that the worst case scenario path between the output node and ground passes through three transistors. Therefore, all transistors in the PDN should have a width of three times the minimum width or 2.7[μm].

Equation (3-1) provides the logic function for the PUN. However, since the PUN will be made of P-channel transistors, every variable in Equation (3-1) must be inverted for the PUN to function correctly. This process gives the function for the PUN as:

$$\mathrm{PUN} = \mathrm{C'[L'R'} + \mathrm{LR]} + \mathrm{C[LR'} + \mathrm{L'R]}. \qquad (3-4)$$

Equation (3-4) shows that the worst case scenario path from the positive supply voltage to the output node again passes through three transistors. Therefore, like the PDN, all transistors should have a width of three times the minimum or 2.7[μm].

This gate has the unusual property of having an identical PUN and PDN, making design and layout very straightforward. The schematic for the gate is shown in Figure 3-1.

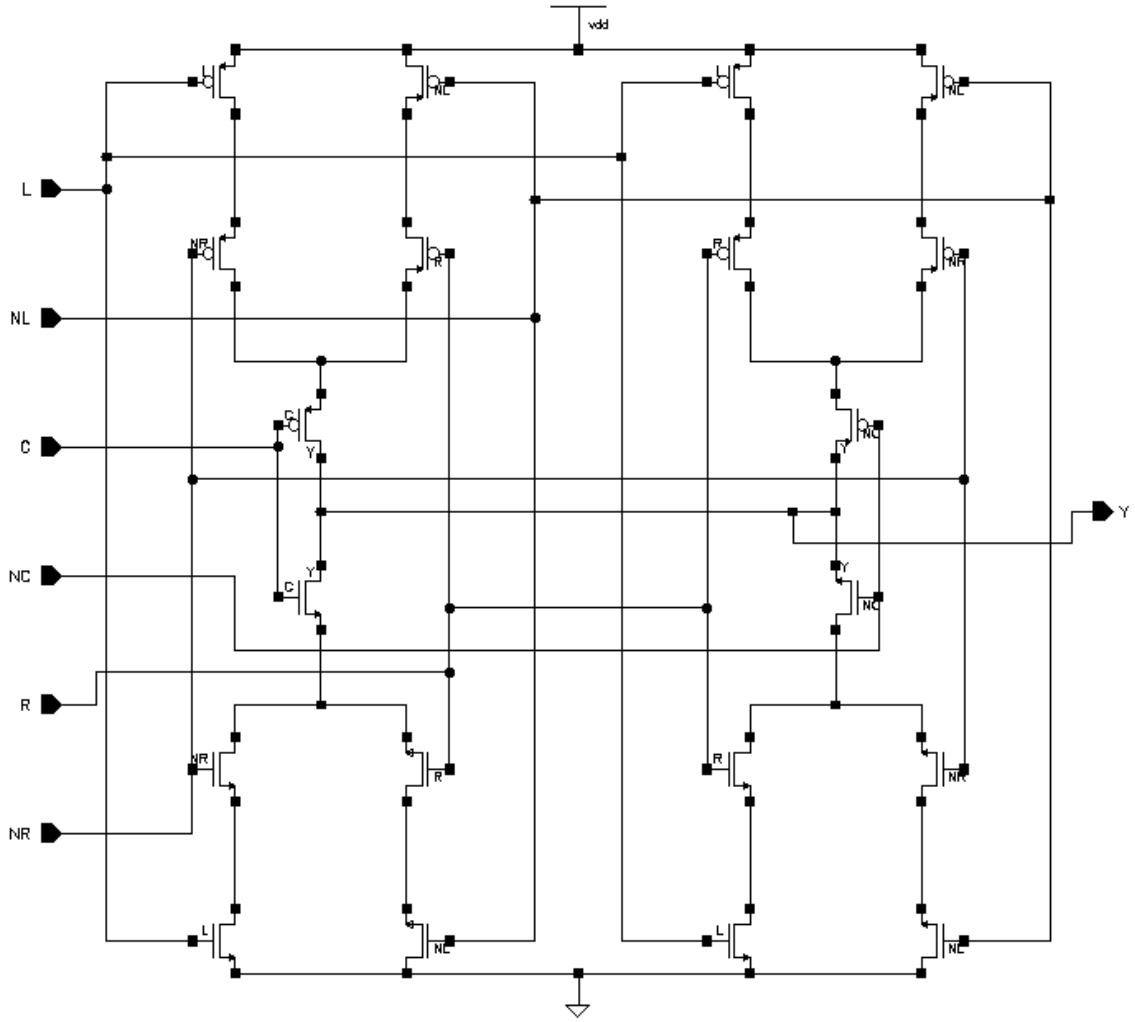

Figure 3-1: The schematic of the basic adder cell, signals beginning with N are complemented signals, inverters not shown.

In Figure 3-1 signals beginning with N are complemented signals, for example NL would correspond to L'. Figure 3-1 shows how the PUN and PDN are joined at the output node Y to form the complete adder gate. When the logic function representing the PUN is a 1, the PUN connects the output to the positive supply voltage V_{DD} making the output Y a 1. Likewise, when the logic function

representing the PDN is a 1 the PDN connects the output to ground making Y a 0.

Simulation confirms the operation described above. Figure 3-2 shows the simulation results obtained from the circuit of Figure 3-1.

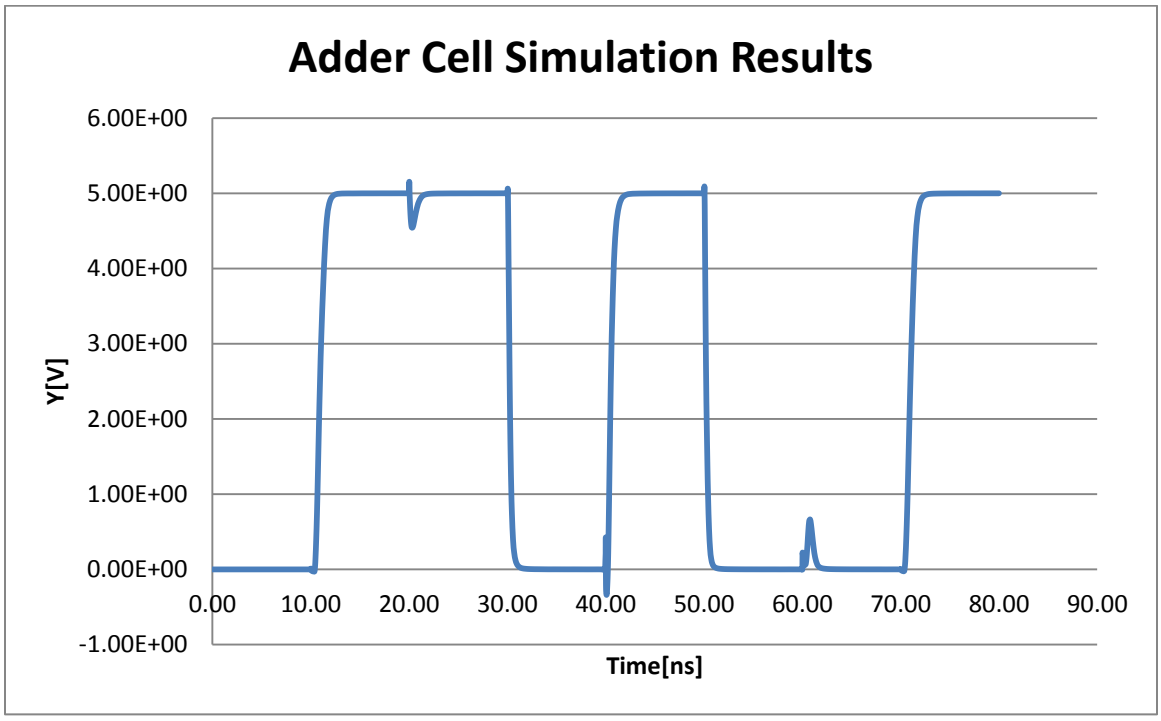

Figure 3-2: Simulation results for the adder circuit of Figure 3-1 for inputs described below.

At the beginning of the simulation all signals: L, R, and C are set to 0 and remain there for the first 10[ns]. According to Table 3-1 the output should be 0 and the simulation shows this is the case. Then from 10[ns] to 20[ns] L is set to 1 and all other signals remain 0, Table 3-1 predicts and the simulation shows that the output is a 1 in this case. From 20[ns] to 30[ns] L is returned to 0, R is set to 1, and C remains at 0. Table 3-1 indicates that the output should remain at 1 in this instance and this occurs in the simulation. From 30[ns] to 40[ns] R remains at 1, L is set to 1, and C remains at 0. In this situation Table 3-1 shows that the output

should be a 0 and the simulation confirms this. From 40[ns] onwards C is set to 1 and L, R follow the same pattern as before. In these situations Table 3-1 shows that the output should follow the pattern: 1,0,0,1. The simulation output follows the same pattern for 40[ns] to 80[ns].

3.4 Fast Carry

The most logical way to handle carries is to make each cell generate its own carry and then just connect the carry out of one bit to the carry in of the next bit. This means that the bits can only be evaluated one after another, one at a time. A speed improvement can be made by determining what a carry will be before the preceding bits are computed. The fastest way to do this is to predict carries based on the inputs to the adder; however this is impractical due to hardware complexity [3]. To illustrate this consider the logic functions for the first three carry outs:

$$c_1 = l_1 r_1, \tag{3 – 5}$$

$$c_2 = l_2 r_2 + (l_2 + r_2) l_1 r_1, \text{ and} \tag{3 – 6}$$

$$c_3 = l_3 r_3 + (l_3 + r_3) l_2 r_2 + (l_3 + r_3)(l_2 + r_2) l_1 r_1. \tag{3 – 7}$$

In the preceding equations the c's represent the carry outs and the l's and r's represent the inputs to the adder from the L and R registers. Clearly, implementing this system for an 8-bit adder would be impractical.

A fast carry scheme can still be implemented without resorting to the impractical complexity described above. This scheme can be implemented by

looking at Equations (3-5) to (3-7) and noticing that each bit past the first one needs to determine whether it will pass along the carry from the previous bit or produce a carry of its own. To that end, each bit past the first one needs to produce two additional signals called propagate and generate. Propagate determines whether or not a bit will pass along a carry from the previous bit and generate determines whether or not a bit will produce its own carry. The logic equations for these signals are [3]:

$$p_i = l_i + r_i, \text{ and} \tag{3 – 8}$$

$$g_i = l_i r_i \ . \tag{3 – 9}$$

The first bit in the adder will only need to produce a generate signal because it has no carry in to propagate and any result of addition will be lost in the division by two.

Every two bits of these signals are used to compute a “super-carry” that determines whether those two bits will produce a carry out to the next bit [3]. The logic equation for the super-carries is

$$C_i = g_{i-1} + p_{i-1}g_{i-2} + p_{i-1}p_{i-2}C_{i-1}. \tag{3 – 10}$$

Here, the capital c’s represent super-carries with C_i being the super-carry being produced and C_{i-1} being the previous super-carry. Since there is no carry-in to the first group of bits, the first super-carry is given as:

$$C_1 = g_2 + p_2 g_1. \tag{3 – 11}$$

The speed of this scheme comes from the fact that all p_i and g_i are evaluated in parallel, thus when a given super-carry is evaluated the next can be immediately determined while the intervening bits are still being evaluated. Therefore, many bits can be computed in parallel resulting in increased speed.

The next task is to design the circuitry for the scheme just described. First, consider the equations for the super-carries. Since these equations involve only un-complemented variables, the CMOS gates that implement them will only need the complements of those variables, therefore each bit should produce the complements of p_i and g_i. This means using a NOR gate to produce p_i and a NAND gate to produce g_i. This is advantageous for speed and area because NAND and NOR are the "natural" gates, meaning they do not require inverters at their inputs. This also means that the previous super-carry will also need to be complemented, but this will need to be done anyway for the addition and this value can also be used for the next super-carry.

In addition, every odd numbered bit with the exception of the first one will need to generate its own carry out. The logic function for this is given as:

$$c_i = l_i r_i + (l_i + r_i) c_{i-1}. \qquad (3-12)$$

Like the super-carries this gate will need the complemented versions of all variables. The complements of l_i and r_i can be easily obtained from the D flip-flops, and c_{i-1} comes from the preceding super-carry and the complement will be produced for the adder cell anyway and therefore is also easily accessible. Both

the super-carry gates and the carry out gates for the odd numbered bits have been designed using the procedure described in section 3.2.

3.5 Physical Layout and Simulation Results

Now that the required logic circuits have been discussed the full addition and division by two circuit can be shown. Figure 3-3 below shows the layout of the circuit.

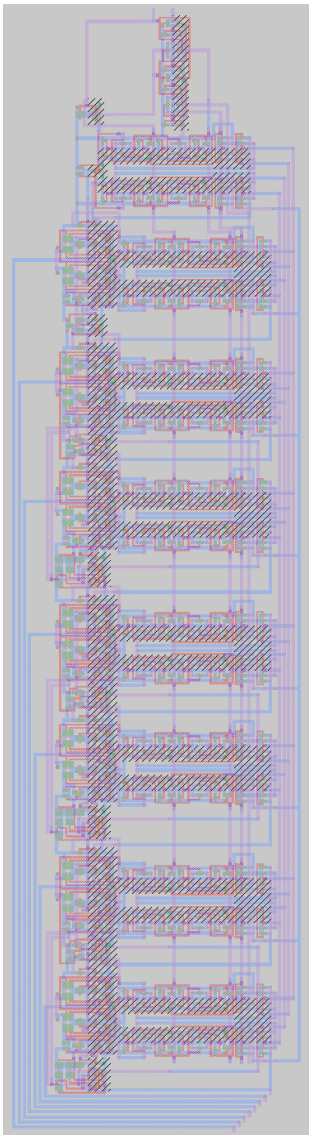

Figure 3-3: The physical layout of the addition and division by two circuit.

Starting from the top, the first component is a falling edge D flip-flop for synchronous reset. The component directly to its left before the main adder is a CMOS inverter to produce the complement of the clock signal; it has slightly larger dimensions on both transistors because it must drive a substantial load. Everything below this inverter is the main circuit. The components above the addition and division by two circuit components are the multiplexors and flip-flops of the L and R registers and L and R multiplexors.

The main circuit is constructed as discussed in the previous two sections. Each bit with the exception of the first has an adder cell at its heart, as well as a NAND and NOR gate to produce propagate and generate. The odd numbered bits with the exception of the first have their own carry out gates. The first bit is only a NAND gate to produce the generate signal. Finally, there is a super-carry every two bits as well as after the last bit to produce the final carry out. The MSB is set to the final carry out to accomplish the division by two.

The simulation begins by resetting the circuit and setting the control signals for the adder to count down. From 0 to 160[µs], the first eight clock cycles, the adder counts down until it reaches 0. The outputs at each clock cycle are the midpoints of the interval, with the first interval being [0,255] and the right endpoint being set to the result of the previous clock cycle.

At 160[µs] the circuit is reset again and the control signals are set for the adder to count up. The adder spends another eight clock cycles, 160[µs] to 320[µs] counting up. This proceeds in the same fashion as before except that

the left endpoint is set to the result of the previous clock cycle. The final result is 254 as expected, because fractional data is simply lost (no rounding takes place). Figure 3-4 shows these results graphically.

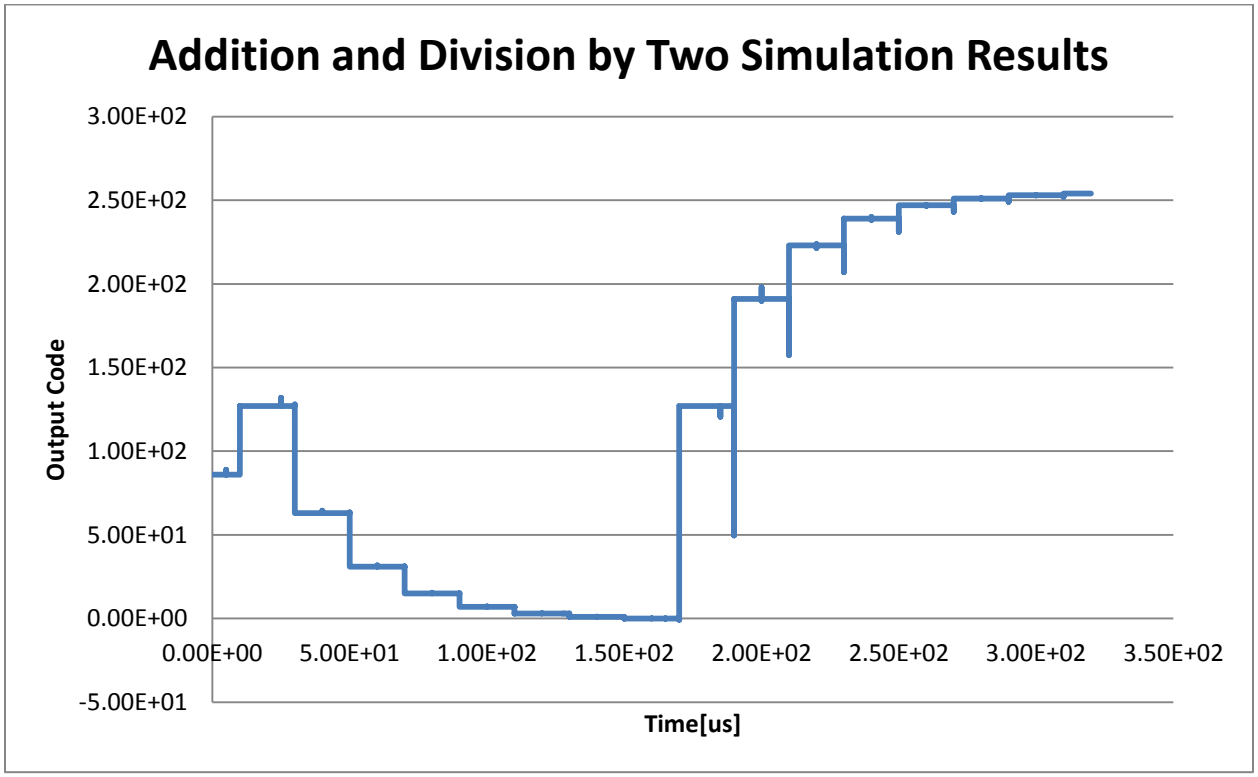

Figure 3-4: Simulation results for the circuit shown in Figure 3-3.

The simulation was carried out using a clock speed of 50[kHz] (a period of 20[µs]). This was done to make the circuit behavior clear even though the circuit has been successfully simulated with clock speeds of up to 50[MHz] (a period of 20[ns]). The simulation results with the higher clock speed can be found in appendix C. The simulation results show that the various components discussed in this chapter function together as intended. Furthermore, a general procedure for designing CMOS logic gates has been outlined and demonstrated. This procedure will be used again later in the project.

Chapter 4

Digital to Analog Converter

4.1 Introduction

The purpose of the digital to analog converter (DAC) is to convert the digital output code candidate, produced by the addition and division by two circuits, into an analog signal that the comparators can test against the analog input signal. The results of the comparators will be used to determine whether the current digital output code is the correct one, or whether the correct output code is higher or lower.

The DAC will have a resolution of eight bits. The primary requirements of the DAC are a rail-to-rail (in this case 0 – 5[V]) output capability and high-accuracy. The secondary requirement is speed. The DAC should be capable of high speed operation, but the accuracy and rail-to-rail requirements take priority. The DAC input will only be valid in the second half of the clock cycle, when the addition and division by two circuits have finished their operation.

There are three possible candidates for the DAC architecture to use: current scaling, voltage scaling, or charge scaling. Current scaling DACs take the reference voltage and create a series of binary weighted currents from it. The digital input code then selects some of these currents and sends them to an output amplifier to be turned into an output voltage. The currents that are not used are returned to ground or the negative power supply [1].

The current scaling DAC is not suitable for this application primarily because the requirement of an output amplifier makes obtaining rail-to-rail operation extremely difficult. Static power dissipation, due to the fact that the currents are always flowing, is another disadvantage of this DAC.

The voltage scaling DAC works by using a network to divide the reference voltage into a set of smaller voltages and then the digital input code is sent to a series of switches to select one of these voltages as the output [1]. This DAC has the advantage of not needing an output amplifier if the load is very highly resistive with a small capacitance, although it technically can drive a capacitive load without an output amplifier in which case speed will suffer. In this scenario, the output of the DAC will only be connected to the gates of MOS transistors, therefore the conditions for not needing an output amplifier are met.

However, the voltage scaling DAC has some disadvantages that make it unsuitable for this application. The primary disadvantage is size and complexity. Since the converter needs to have eight bit resolution, the converter will need to divide the reference voltage into 256 different voltages and this requires a long string of resistors. In addition, the circuit would require an 8-to-256 decoder and 256 switches to select the appropriate output voltage. Needless to say, the required circuitry would be large and complex which is undesirable. Also, like the current scaling DAC, this DAC will exhibit static power dissipation due to the resistor string.

The only remaining architecture is the charge scaling DAC. The charge scaling DAC works by "binarily dividing the total charge applied to a capacitor array" [1]. The charge scaling DAC has the advantages of being simple and accurate. It can be made to not need an output amplifier by making sure that the output capacitance is at all times much greater than the load capacitance (since the load will be the gates of MOS transistors, this can be easily accomplished). High speed can also be easily achieved. Additionally, the charge scaling DAC has no static power dissipation because when the capacitors have been charged/discharged to their final value no current flows. Due to these advantages, the charge scaling DAC is the best choice for this application.

4.2 Basic 4-bit Charge Scaling DAC

Before presenting the full 8-bit charge scaling DAC, it is useful to examine the simpler 4-bit charge scaling DAC. This makes it simpler to understand the basic principle of the charge scaling DAC and, as will be seen later, the best way to make the 8-bit version is by using two 4-bit charge scaling DACs. The schematic of the basic 4-bit charge scaling DAC is shown in Figure 4-1 below [1].

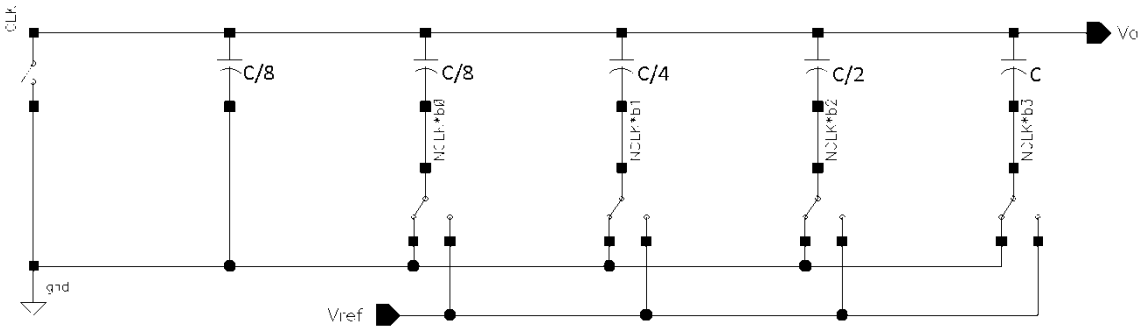

Figure 4-1: Schematic of the 4-bit charge scaling DAC.

The circuit consists of capacitors, switches, and logic gates that control the switches. The logic gates are not shown, but their logic functions are shown near the switches that they control. The capacitors are binary weighted. The capacitor with capacitance C/8 is the smallest capacitor in the circuit and its capacitance will be referred to as the fundamental capacitance. The switches will be implemented with CMOS transmission gates and will be controlled by CMOS logic gates. The last capacitor with capacitance C/8 is there to make the total capacitance 2C. It will be shown later why this is necessary [1].

The circuit of Figure 4-1 works by first grounding both terminals of the capacitors when the CLK signal is high to prepare for the next conversion [1]. This is accomplished while the addition and division by two circuit is performing its task, computing the next digital code to be converted. When the CLK signal is low (NCLK is high) the equivalent circuit of Figure 4-2 results.

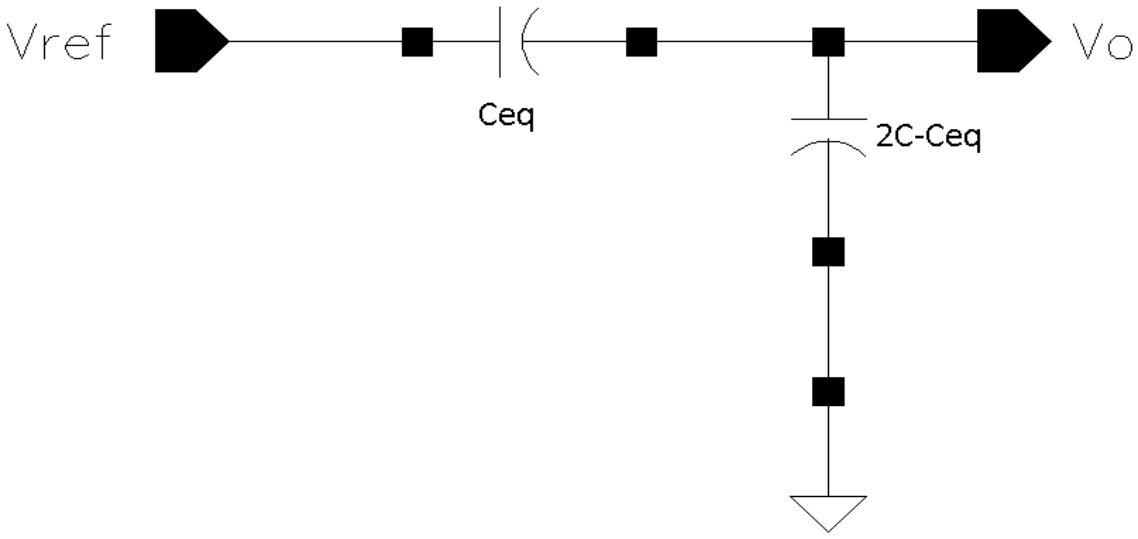

Figure 4-2: Equivalent circuit when NCLK is high and the output is valid.

The time when NCLK is high is the time when the output of the DAC is valid [1].

It will now be shown that Figure 4-2 results in a 4-bit DAC. Note the following derivation follows material found in reference [1]. The capacitance C_{eq} represents all capacitors connected to the reference voltage V_{ref} (the positive power supply in this application), or

$$C_{eq} = C\left(\frac{b_0}{8} + \frac{b_1}{4} + \frac{b_2}{2} + b_3\right). \quad (4-1)$$

The total charge on both capacitors must be equal, therefore:

$$C_{eq}(V_{ref} - V_o) = (2C - C_{eq})V_o, \quad (4-2)$$

$$C_{eq}V_{ref} - C_{eq}V_o = 2CV_o - C_{eq}V_o, \quad (4-3)$$

$$C_{eq}V_{ref} = 2CV_o, \text{ and} \quad (4-4)$$

$$V_o = V_{ref}\left(\frac{b_0}{16} + \frac{b_1}{8} + \frac{b_2}{4} + \frac{b_3}{2}\right). \quad (4-5)$$

Clearly, this is the output of a 4-bit DAC. The total capacitance of 2C was necessary to ensure that each bit was scaled properly in the output.

4.3 Extension to 8-bit Resolution

Now it is time to consider the problem of making an 8-bit charge scaling DAC. The most obvious solution is to extend the concept of Figure 4-1 all the way to 8-bits. There are two significant problems with this approach. The first is that the largest capacitor in the circuit would be 256 times larger than the smallest and this is impractical due to the large area consumed. The second is

that matching accuracy between the capacitors decreases as the difference between their sizes increases [1].

A better option is to combine two 4-bit DACs to form an 8-bit DAC. The way to do this for the charge scaling DAC is to connect the two 4-bit DACs with a connecting capacitor of appropriate value [1]. Figure 4-3 illustrates this approach.

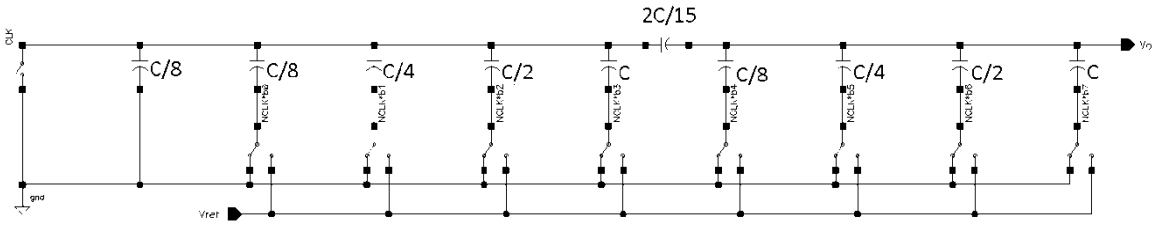

Figure 4-3: Method of using two 4-bit charge scaling DACs to make an 8-bit charge scaling DAC.

The left hand side of Figure 4-3 is identical to the circuit in Figure 4-1. This will be referred to as the LSB array because it handles the four least significant bits. The right hand side of Figure 4-3 will be referred to as the MSB array because it handles the four most significant bits. The MSB array is similar to the LSB array, but it lacks a terminating capacitor. This is because the connecting capacitor and the equivalent capacitance of the LSB array will form the terminating capacitor for the MSB array [1].

The derivation of the value of the connecting capacitor has been taken from [1] and proceeds according to the fact that the connecting capacitor in series with the Thevenin equivalent capacitance of the LSB array must terminate the MSB array. This can be described mathematically as:

$$\frac{C}{8} = \frac{1}{\frac{1}{C_S} + \frac{1}{2C}}, \quad (4-6)$$

$$\frac{1}{C_s} = \frac{8}{C} - \frac{1}{2C}, \text{ and} \quad (4-7)$$

$$C_s = \frac{2C}{15}. \quad (4-8)$$

In the above equations C_s is the capacitance of the connecting capacitor.

Now that the purpose and requirements for the connecting capacitor have been explained, it can be shown that the circuit of Figure 4-3 functions as an 8-bit DAC. Again, the derivation follows material from [1]. Figure 4-4 shows the Thevenin equivalent circuit of Figure 4-3.

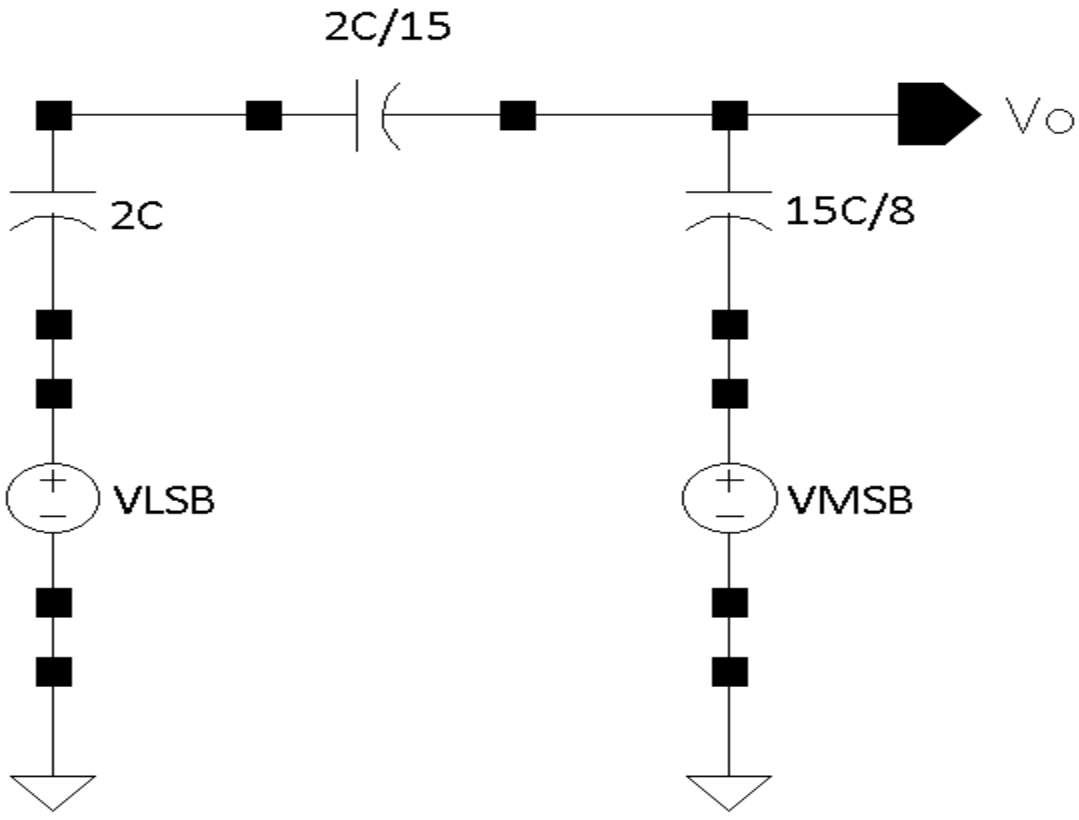

Figure 4-4: Thevenin equivalent circuit of figure 4-3.

Thevenin equivalents are found by finding two parameters: the Thevenin equivalent voltage and the Thevenin equivalent impedance (in this case capacitance). The Thevenin equivalent voltage is found by making the node of interest an open circuit, and finding the voltage between the node of interest and ground. The Thevenin equivalent capacitance is found by setting all voltage sources to zero and finding the equivalent capacitance between the node of interest and ground.

Using this method, it is clear that the Thevenin equivalent capacitances of the LSB array and MSB array are 2C and 15C/8 respectively. The Thevenin equivalent voltages are given by:

$$V_{LSB} = V_{ref}(\frac{b_0}{16} + \frac{b_1}{8} + \frac{b_2}{4} + \frac{b_3}{2}), \text{ and} \quad (4-9)$$

$$V_{MSB} = V_{ref}(\frac{b_4}{15} + \frac{2b_5}{15} + \frac{4b_6}{15} + \frac{8b_7}{15}). \quad (4-10)$$

Here, V_{LSB} is the Thevenin equivalent voltage of the LSB array and V_{MSB} is the Thevenin equivalent voltage of the MSB array.

The final output can be found by using superposition, which is finding the output with only one Thevenin equivalent voltage active and then summing the results. The output voltage from the LSB array can be found as follows:

$$\frac{15C}{8}V_{o1} = \frac{C}{8}(V_{LSB} - V_{o1}), \quad (4-11)$$

$$15V_{o1} = V_{LSB} - V_{o1}, \text{ and} \quad (4-12)$$

$$V_{o1} = \frac{V_{LSB}}{16}. \quad (4-13)$$

In the previous equations, V_{o1} is the component of the output resulting from the LSB array and $\frac{C}{8}$ is the capacitance that results from the series combination of 2C and 2C/15. The output voltage from the MSB array can be found in a similar fashion:

$$\frac{C}{8}V_{o2} = \frac{15C}{8}(V_{MSB} - V_{o2}), \quad (4-14)$$

$$V_{o2} = 15(V_{MSB} - V_{o2}), \text{ and} \quad (4-15)$$

$$V_{o2} = \frac{15}{16}V_{MSB}. \quad (4-16)$$

Combining these two results gives:

$$V_o = V_{o1} + V_{o2} = \frac{V_{LSB}}{16} + \frac{15V_{MSB}}{16}, \text{ and} \quad (4-17)$$

$$V_o = V_{ref}(\frac{b_0}{256} + \frac{b_1}{128} + \frac{b_2}{64} + \frac{b_3}{32} + \frac{b_4}{16} + \frac{b_5}{8} + \frac{b_6}{4} + \frac{b_7}{2}). \quad (4-18)$$

This is clearly the correct output for an 8-bit DAC.

4.4 Physical Layout and Simulation Results

This circuit will use capacitors made from two different polycrystalline silicon (poly) layers. This structure has a capacitance per unit area of roughly 0.92 [fF/μm^2] [4]. Figure 4-5 on the next page shows the physical layout of a capacitor. The central square is the capacitor itself and it has dimensions of 21[μm] x 21[μm] giving a capacitance of roughly 400[fF]. The two poly layers have as many contacts as possible in order to make the contact resistance as small as possible.

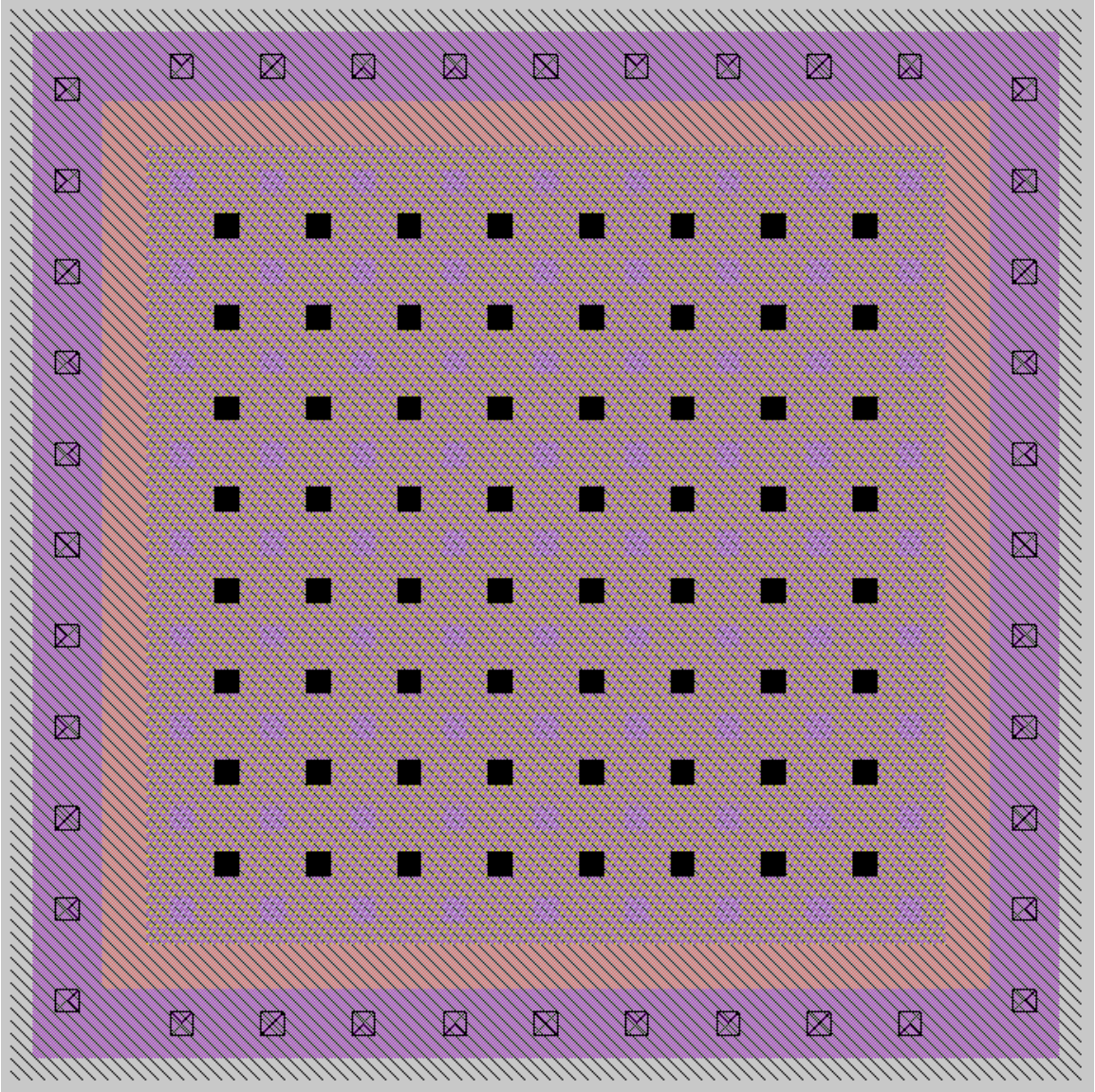

Figure 4-5: Physical layout of a poly-poly2 capacitor.

The purpose of the n-well surrounding the capacitor is to “minimize field leakage” [5].

The capacitor in Figure 4-5 will serve as the fundamental capacitor for DAC. All other capacitors, with the exception of the connecting capacitor, will be made by using multiple copies of the fundamental capacitor connected in parallel. The reason for this is to improve the matching accuracy between the

capacitors [1]. Figure 4-6 shows the complete layout of the 8-bit charge scaling DAC.

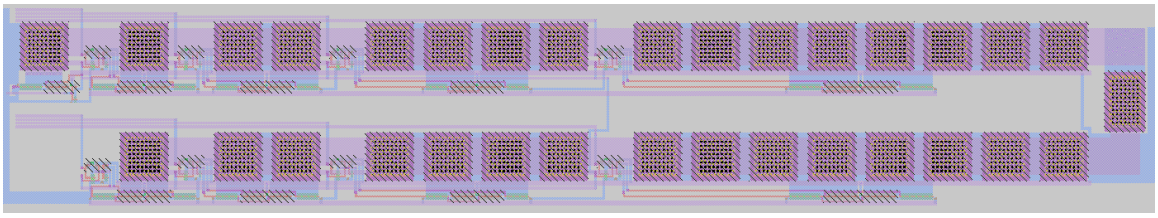

Figure 4-6: Complete layout of the DAC.

The switches below the different capacitors are conventional CMOS transmission gates. The widths of the transistors in these switches can be determined by noting that the minimum width of a transistor in an analog application for this process should be 3[µm] and associating this width with a minimum capacitance, say 100[fF]. Due to the fact that the minimum capacitance in this circuit is around 400[fF], the minimum transistor width in the switches should be scaled up to four times the minimum, or 12[µm]. The final capacitor in each capacitor is significantly larger than the others in the array so it is prudent to make their switches slightly larger than the others, in this case 18[µm].

The logic gates that control these transmission gates are conventional CMOS NAND gates. The outputs of these gates are then sent to inverters to realize the necessary logic functions. This may seem a bit crude, but no performance is lost since both the output of the logic functions and there complements are needed to control the transmission gates.

The circuit was simulated using a clock speed of 100[kHz]. Since showing all 256 input cases on the same plot is not practical, only every 17^{th} input code

will be shown. In other words the digital input codes for this simulation have the following values:

$$Input\ Code_i = 17i, i = 0,1,\dots,15. \qquad (4-19)$$

The simulation results are shown in Figure 4-7.

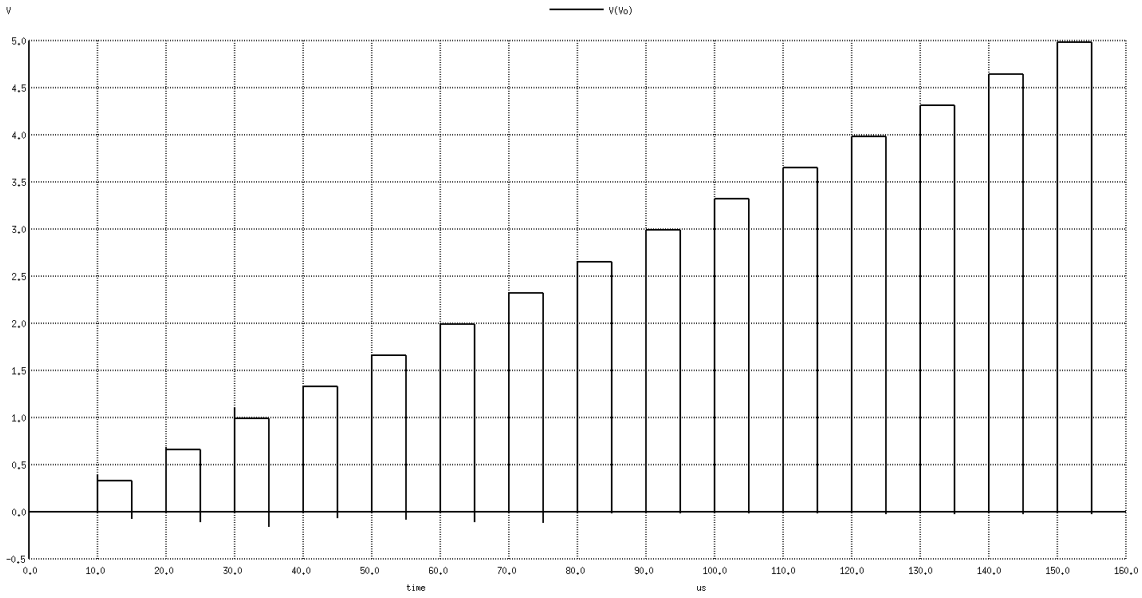

Figure 4-7: Simulation results for the DAC of Figure 4-6.

From Figure 4-7, the converter appears to be monotonic and have good accuracy. A closer examination of each output value reveals that it is within ±0.5[LSB] of the correct value. Any further improvement beyond this will be made useless by the quantization error [1]. The specific output values for each input code can be found in Appendix D.

Chapter 5

Comparators

5.1 Introduction

The purpose of the comparators is to determine whether the current digital output code from the addition and division by two circuit is higher than the correct output code, lower than the correct output code, or is the correct output code. There are two comparators: the L-comparator and the R-comparator. The L-comparator compares the output of the DAC with the analog input minus one half of a least significant bit (LSB). Likewise, the R-comparator compares the output of the DAC with the analog input plus one half of a LSB. The functionality of the comparators can be summarized in the following equations:

$$\text{HL} = 0,\ v_I < v_D - \text{LSB}/2, \tag{5 – 1}$$

$$\text{HL} = 1,\ v_I > v_D - \text{LSB}/2, \tag{5 – 2}$$

$$\text{HR} = 0,\ v_I < v_D + \text{LSB}/2, \text{ and} \tag{5 – 3}$$

$$\text{HR} = 1,\ v_I < v_D + \text{LSB}/2. \tag{5 – 4}$$

Here HL is the output of the L-comparator and HR is the output of the R-comparator.

So, if HL and HR are 0 then the output of the addition and division by two circuit is below the correct output code and this value will become the next value

in the L-register. Likewise, if HL and HR are 1 then the output of the addition and division by two circuit is above the correct output code and this value will become the next value in the R-register.

The requirements for the comparators are rail-to-rail operation and a resolution of less than 10[mV]. The comparators will have three major stages. The first stage consists of CMOS differential amplifiers that convert the difference between the two voltage inputs into a difference between two currents. Differential amplifiers will also be used to create the ±LSB/2 offsets. The second stage is a latch to determine the output of the comparator. The latch uses positive feedback and converts the current difference back to a voltage difference. The final stage is another differential amplifier to give drive capability and convert the voltage difference from the latch to logic levels.

5.2 Models

Before describing each of these stages, the models that will be used must be discussed. Since the channel lengths are all submicron, the simple current voltage equations found in introductory electronics texts are not adequate. Instead, models that account for various short channel effects should be used, however these models can be too complicated for the hand calculations used for initial design work. The following equation accounts for the velocity saturation effect and is simple enough to use for hand calculation [6]:

$$I_{ds} = W v_{sat} C_{ox} \frac{{V_{on}}^2}{V_{on} + E_c L}. \qquad (5-5)$$

In equation 5 – 5: W is the width of the transistor, v_{sat} is the carrier saturation velocity, C_{ox} is the oxide capacitance, v_{on} is the amount by which the voltage between the gate and the source exceeds the threshold voltage and the minimum voltage between the drain and source to keep the transistor in the saturation region of operation, E_c is the critical field where the carrier velocity saturates, and L is the transistor length.

The body effect is crucial to achieving rail-to-rail input common mode range; however model parameters for it are not readily available on process data sheets. Also, a more accurate model to the basic one is desirable. An excellent model to use is provided by [7]:

$$V_{th} = V_{th0} + K_1\left(\sqrt{2\phi_F + V_{sb}} - \sqrt{2\phi_F}\right) + K_2 V_{sb}. \qquad (5-6)$$

This model takes into account the fact that the doping in the body of the transistor is non-uniform in the vertical direction, and it is simple enough for hand calculation [7]. Furthermore, the parameters K_1 and K_2 can readily be found in the SPICE model cards on the process run data sheets.

The physical parameters for the transistors are also found on these process run datasheets. Since there is variability in these parameters, it is prudent to examine them over several runs and calculate the average, maximum, and minimum values. The circuits will then be designed so that the requirements can be met even with the worst case scenario parameters. The values were examined over 10 runs of the C5 process and the values are summarized in the tables on the next page.

Table 5-1: Parameters for C5 N-Channel Transistors [4]

Value	V_{th0} $[V]$	K_1 $[\sqrt{V}]$	K_2	k' $[\frac{\mu A}{V^2}]$	v_{sat} $[\frac{cm}{s}]$	E_cL [V]
Minimum	0.586	0.845	-0.112	57.1	19700000	3.9044912
Average	0.6198	0.8927	-0.1017	59.13	19840000	4.908404
Maximum	0.669	0.9155	-0.085	60.3	20000000	5.0950522

Table 5-2: Parameters for C5 P-Channel Transistors [4]

Value	$\lvert V_{th0}\rvert$ $[V]$	K_1 $[\sqrt{V}]$	K_2	k' $[\frac{\mu A}{V^2}]$	v_{sat} $[\frac{cm}{s}]$	E_cL [V]
Minimum	0.915	0.5535	0.007787	19.1	7000000	4.5508748
Average	0.915	0.5535	0.007862	19.37	10093000	7.669336
Maximum	0.915	0.5535	0.00787	19.8	13200000	10.2154

Table 5-3: Other C5 Parameters [4]

Value	C_{ox} $[\frac{F}{cm^2}]$	High Resistance $[\frac{k\Omega}{Square}]$
Minimum	2.43173E-07	0.992
Average	2.49527E-07	1.051
Maximum	2.53901E-07	1.103

5.3 Differential Amplifier Design

Now that the necessary equations and parameters have been presented, it is time to proceed with the design of the differential amplifiers for the first stages of the comparators. The N-channel and P-channel differential amplifiers should produce a current difference of at least one microampere for an input voltage difference of 10[mV] and provide rail-to-rail input common mode input range. These requirements are what will drive the design process for the differential amplifiers. The main task of the design process is to find the widths of all transistors that make up the differential amplifiers. In particular, there are three transistor widths that need to be found: the widths of the input transistors, the widths of the diode connected load transistors, and the widths of the

transistors in the bias current sources. The requirements for the differential amplifiers are summarized in tables 5-4 and 5-5.

Table 5-4: Design Parameters for N-Channel Differential Amplifiers

Transconductance $g_m[\frac{\mu A}{V}]$	100
Maximum Input Common Mode Voltage $V_{CM,Max}\ [V]$	5
Minimum Input Common Mode Voltage $V_{CM,Min}\ [V]$	2

Table 5-5: Design Parameters for P-Channel Differential Amplifiers

Transconductance $g_m[\frac{\mu A}{V}]$	100
Maximum Input Common Mode Voltage $V_{CM,Max}\ [V]$	2.2
Minimum Input Common Mode Voltage $V_{CM,Min}\ [V]$	0

In addition, the desired bias current for both differential amplifiers will be 10[μA] to allow for rail-to-rail input range with reasonable transistor sizes.

The P-channel differential amplifier will be designed first. To find the widths of the input transistors, consider the expression for transcondtuctance:

$$g_m = 2\sqrt{k'_p \frac{W_I}{L}_p \frac{I_B}{2}}. \qquad (5-7)$$

Using the worst case scenario parameters from table 5-2 and solving equation 5-7 for $\frac{W_I}{L}_p$ with $g_m = 100[\frac{\mu A}{V}]$ gives $W_I = 15.9[\mu m]$.

Next, the widths of the load transistors can be found from the minimum input common mode voltage requirement, or:

$$V_{CM,Min} = V_{THN} + V_{on,L} - |V_{THP}|. \quad (5-8)$$

Solving equation 5-8 for $v_{on,L}$ using worst case scenario parameters from tables 5-1 and 5-2 and using equation 5-5 and tables 5-1 and 5-3 to get the transistor width reveals that the transistor width is below the minimum width for analog applications using the C5 process. Therefore, the P-channel rail-to-rail requirement can be met using minimum width transistors ($W_L = 3[\mu m]$). In addition, the body-effect in the input transistors will improve the minimum input common mode voltage further.

The final step is to find the widths of the transistors in the bias current source. This can be done by using the maximum input common mode range requirement, or:

$$V_{CM,Max} = V_{CC} - 2V_{cs} - (|V_{thp}| + V_{on,I}). \quad (5-9)$$

Again using worst case scenario parameters from tables 5-1 to 5-3 as well as equation 5-6 to solve equation 5-9 for V_{CS}, and using worst case scenario parameters as well as equation 5-5 gives the widths of the current source transistors as $W_{CS} = 4.2[\mu m]$.

A similar procedure can be used to design the N-channel differential amplifier. The expression for transconductance:

$$g_m = 2\sqrt{k'_n \frac{W_I}{L}_n \frac{I_B}{2}}, \quad (5-10)$$

can again be used to find the widths of the input transistors. Doing so, using worst case scenario parameters, gives an input transistor width of $W_I = 9[\mu m]$.

The next step is to find the widths of the load transistors using the maximum input common mode voltage requirement. The maximum input common mode range requirement can be expressed as:

$$V_{CM,Max} = V_{CC} + V_{thn} - (|V_{thp}| + V_{on,L}). \quad (5-11)$$

Equation 5-11 shows that as long as the threshold voltage of the input transistors is greater than the voltage drop across the diode connected load transistors, the maximum input common mode range requirement will be satisfied. The threshold voltage of the input transistors can be found using equation 5-6 with:

$$V_{sb} = V_{CC} - V_{thp} - V_{on,L} - V_{on,I}. \quad (5-12)$$

Using a load transistor width of $W_L = 10.2[\mu m]$ meets the requirements of equations 5-11 and 5-12 with the worst case scenario parameters.

The final step is to find the widths of the current source transistors using the minimum input common mode voltage requirement, which can be expressed as:

$$V_{CM,Min} = V_{thn} + V_{on,I} + 2V_{cs}. \quad (5-13)$$

Using worst case scenario parameters and equation 5-6 to find V_{thn}, the current source transistors should have widths of $W_{CS} = 6.6[\mu m]$.

The final step is to design the resistor that will bias both current sources. The resistor will be connected in between N-channel and P-channel diode connected transistors with widths found previously for the bias current sources. Using this information and equation 5-5 with worst case scenario parameters gives a voltage drop of 3.071[V] across the resistor. Using Ohm's law and the desired bias current gives a desired resistance of 307.1[kΩ]. The differential amplifiers are shown in figure 5-1.

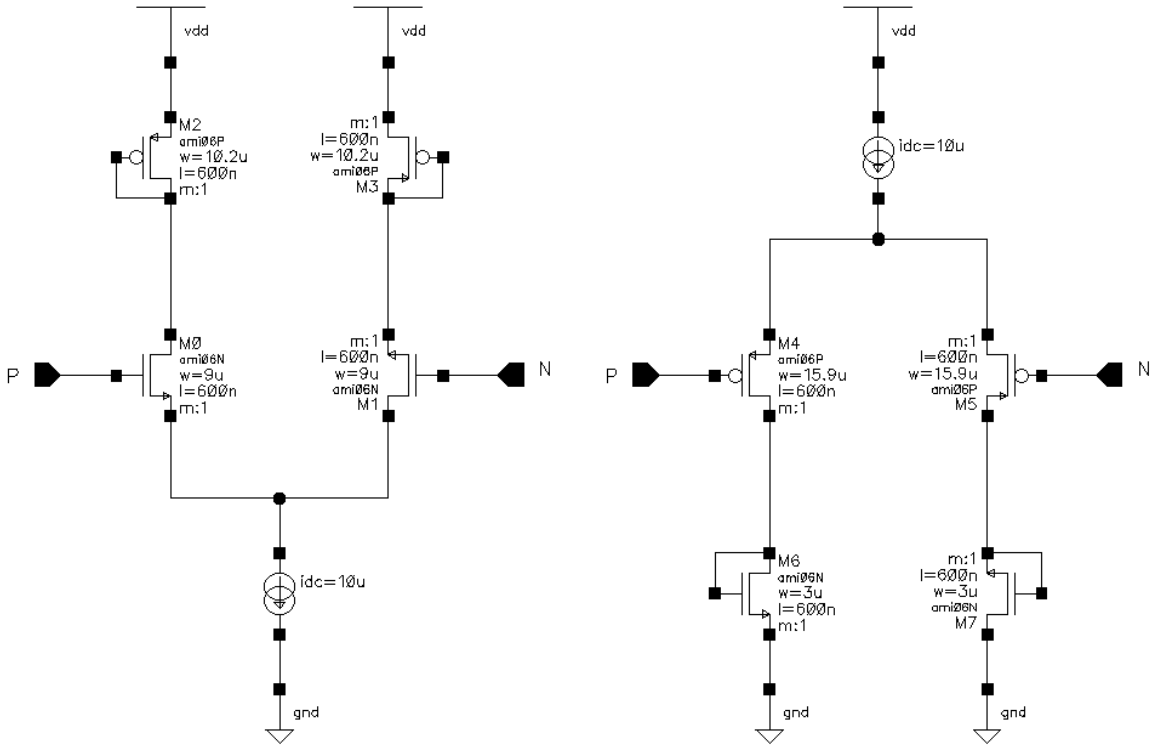

Figure 5-1: Schematics of the N and P channel differential amplifiers.

These differential amplifiers will also be used to generate currents that will be used to produce the necessary LSB/2 offsets. The following equations describe all the currents that will be used in the comparators:

$$I_{+1n} = \frac{I_{1n}}{2} + \frac{g_{m1n}}{2}(V_D - V_i), \qquad (5-14)$$

$$I_{-1n} = \frac{I_{1n}}{2} - \frac{g_{m1n}}{2}(V_D - V_i), \quad (5-15)$$

$$I_{+2n} = \frac{I_{2n}}{2} + \frac{g_{m2n}}{2}\frac{LSB}{2}, \quad (5-16)$$

$$I_{-2n} = \frac{I_{2n}}{2} - \frac{g_{m2n}}{2}\frac{LSB}{2}, \quad (5-17)$$

$$I_{+1p} = \frac{I_{1p}}{2} + \frac{g_{m1p}}{2}(V_D - V_i), \quad (5-18)$$

$$I_{-1p} = \frac{I_{1p}}{2} - \frac{g_{m1p}}{2}(V_D - V_i), \quad (5-19)$$

$$I_{+2p} = \frac{I_{2p}}{2} + \frac{g_{m2p}}{2}\frac{LSB}{2}, \text{ and} \quad (5-20)$$

$$I_{-2p} = \frac{I_{2p}}{2} - \frac{g_{m2p}}{2}\frac{LSB}{2}. \quad (5-21)$$

The differential amplifiers 1n and 1p are the differential amplifiers that produce a current difference based on the difference between the output of the DAC and the input voltage. The differential amplifier 2n produces the necessary offset currents by connecting its positive input to the positive power supply and the negative input terminal to a voltage divider circuit that produces the input voltage of VCC – LSB/2. The differential amplifier 2p produces the necessary offset currents by connecting its positive input to a voltage divider that produces the input voltage of LSB/2 and the negative input terminal to ground. These currents will be combined in various ways to produce the necessary currents for the comparators.

The L-comparator can be made by combining equations 5-14 through 5-21 in the following manner:

$$I_+ = I_{+1n} + I_{+2n} + I_{+1p} + I_{+2p}$$

$$= I_B + \frac{1}{2}[(g_{m1n} + g_{m1p})(V_D - V_i) + (g_{m2n} + g_{m2p})\frac{LSB}{2}], \text{ and} \quad (5-22)$$

$$I_- = I_{-1n} + I_{-2n} + I_{-1p} + I_{-2p}$$

$$= I_B - \frac{1}{2}[(g_{m1n} + g_{m1p})(V_D - V_i) + (g_{m2n} + g_{m2p})\frac{LSB}{2}]. \quad (5-23)$$

Here, I_B is the bias component from the differential amplifiers. The switching point will be when I_+ and I_- are equal. This occurs when:

$$V_D = V_i - \frac{g_{m2n} + g_{m2n}}{g_{m1n} + g_{m1p}}\frac{LSB}{2}. \quad (5-24)$$

In order to preserve the offset across the entire input range g_{m1n} must at all times equal g_{m2n} and g_{m1p} must at all times equal g_{m2p}. To accomplish this, 2n and 2p must have the same bias currents as 1n and 1p. How this is done will be shown later.

The R-comparator can be made the same way. To make the R-comparator, equations 5-14 through 5-21 are combined in the following manner:

$$I_+ = I_{+1n} + I_{-2n} + I_{+1p} + I_{-2p}$$

$$= I_B + \frac{1}{2}[(g_{m1n} + g_{m1p})(V_D - V_i) - (g_{m2n} + g_{m2p})\frac{LSB}{2}], \text{ and} \quad (5-25)$$

$$I_- = I_{-1n} + I_{-2n} + I_{-1p} + I_{-2p}$$

$$= I_B - \frac{1}{2}[(g_{m1n} + g_{m1p})(V_D - V_i) - (g_{m2n} + g_{m2p})\frac{LSB}{2}]. \quad (5-26)$$

The point at which the currents become equal, and the R-comparator switches, is given by:

$$V_D = V_i + \frac{g_{m2n} + g_{m2p}}{g_{m1n} + g_{m1p}} \frac{LSB}{2}. \qquad (5-27)$$

The need to make the bias currents in the N and P differential amplifiers equal is again seen. Since all currents will come from the same differential amplifiers, ensuring the offset of the L-comparator will also ensure the offset of the R-comparator.

5.4 High Accuracy Current Mirrors

The ability to produce highly accurate copies of the currents from the differential amplifiers is crucial because the decision making components of the comparators, the latches, rely on current inputs. High accuracy can be achieved by using conventional cascode current mirrors, however the required voltage drop across the input transistors is too large to allow the differential amplifiers to achieve rail-to-rail input range. The ideal current mirror will have a single diode connected input transistor and the high output resistance of a cascode current mirror.

High output resistance can be achieved by using negative feedback. Figure 5-2 on the next page shows the schematics of the high accuracy current mirror [8]. The advantages of this current mirror are that it has a single transistor input, which allows for a low voltage drop across the input, and negative

feedback to create high output resistance [8]. In addition, its design is reasonably straightforward and easily allows for physical layout.

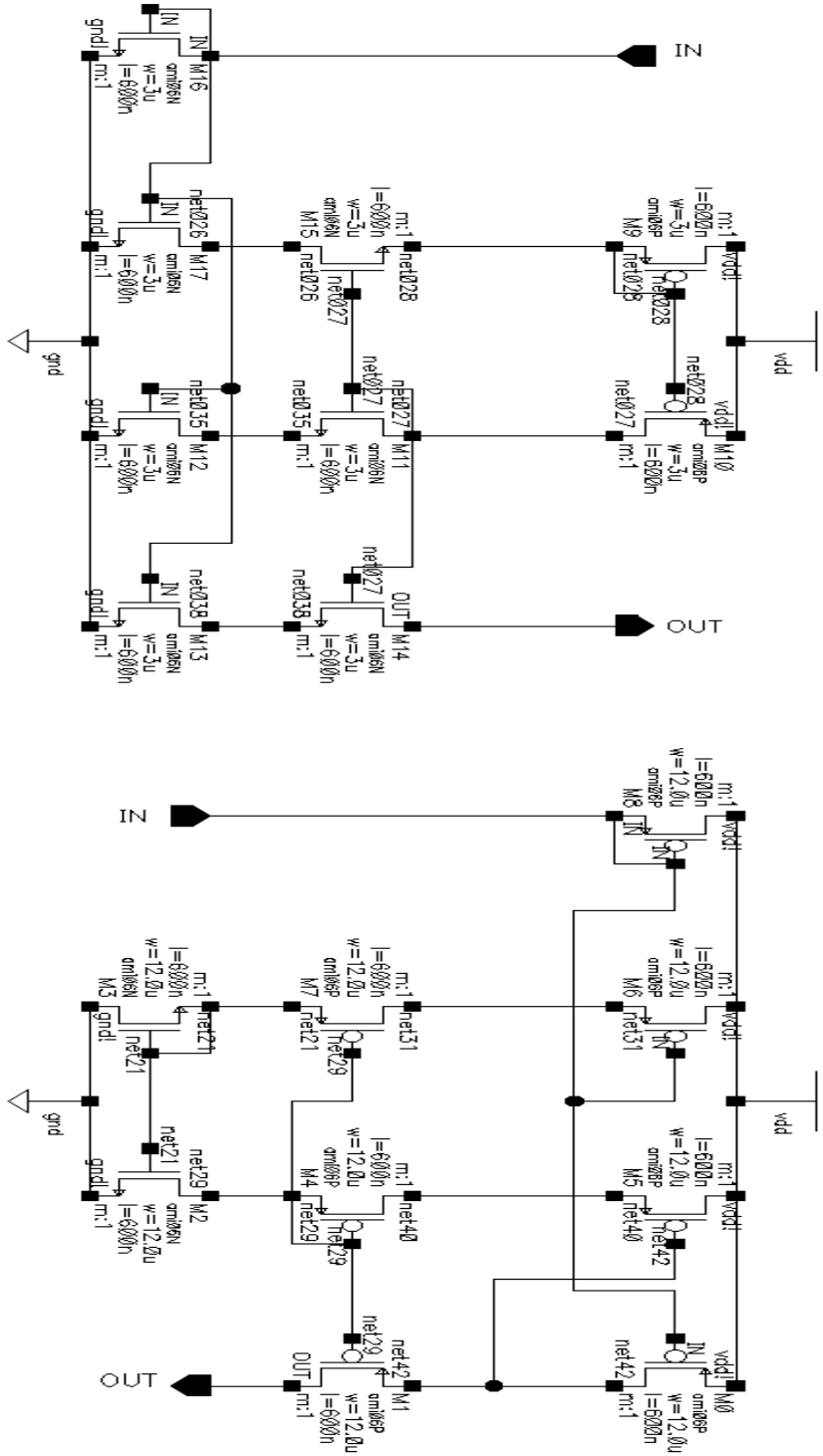

Figure 5-2: Schematics for the N and P channel high accuracy current mirrors.

Figure 5-3 shows the accuracy of this type of current mirror versus various input currents, the current mirror of interest is the more flat curve [8].

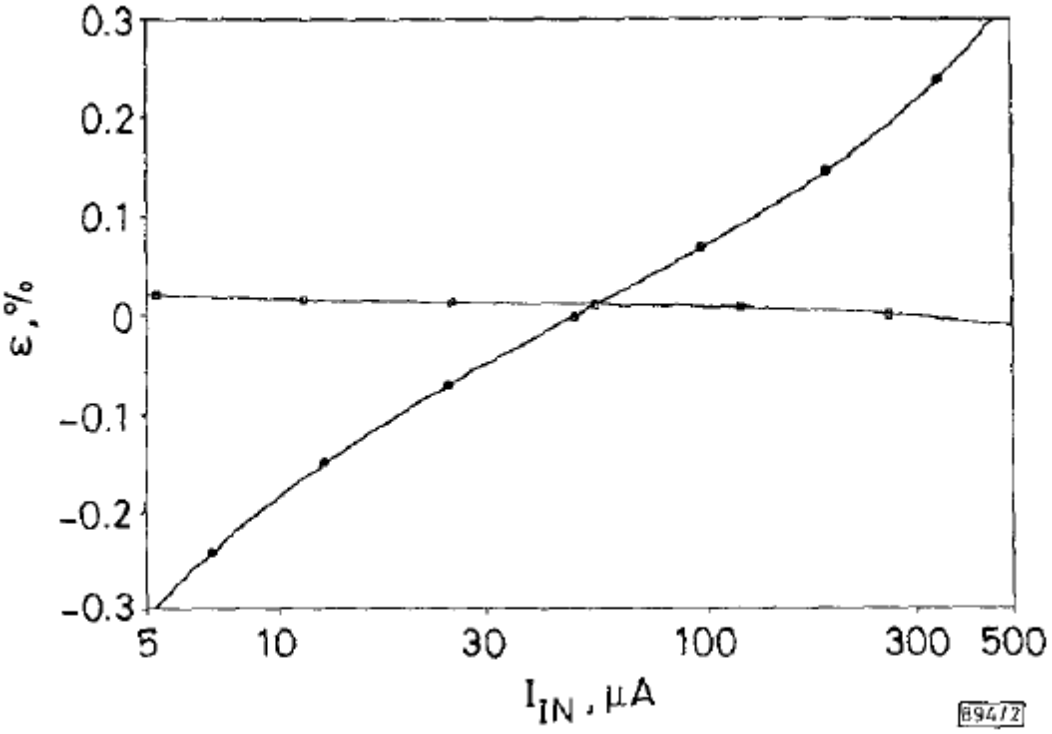

Figure 5-3: The accuracy of the current mirror versus input currents.

The current mirror that produced these results was an N-channel current mirror, like that in figure 5-2 made using a different process, but the quality of this current mirror is clear [8]. The high accuracy combined with the single input transistors to allow the differential amplifiers to reach rail-to-rail input range make these current mirrors ideal for this application.

5.5 Latches

The latch is the component of the circuit that actually determines which signal is larger [9]. The latch accepts the currents described in section 5.3 and

uses regenerative feedback to determine the larger of the two currents. Figure 5-4 below shows the schematic of the latch [9].

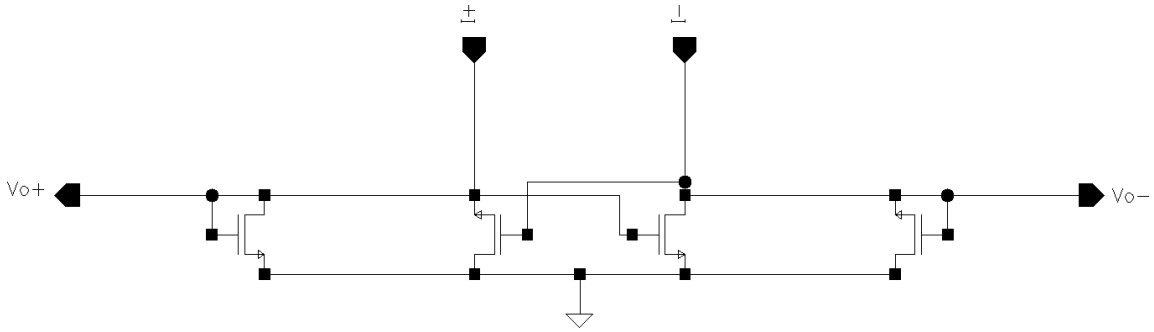

Figure 5-4: Schematic of the latch, responsible for determining the larger signal in each comparator.

The transistors are all N-channel and minimum dimension for analog applications (W=3[µm], L=0.6[µm]). The maximum output voltage is approximately twice the threshold voltage of the transistors [9].

To understand how the circuit of figure 5-4 works, let the transistors be numbered 1 to 4, going from left to right. Following the description given in [9], first assume I_+ is much larger than I_-. This means that transistors 1 and 3 are on and transistors 2 and 4 are off. Furthermore,

$$V_{o+} = \sqrt{\frac{I_+}{k'_n(\frac{W}{L})}}, \text{ and} \qquad (5-28)$$

$$V_{o-} \approx 0. \qquad (5-29)$$

Next, suppose that I_+ starts decreasing and I_- starts increasing thus lowering the gate-source voltages of transistors 1 and 3 and causing V_{o-} to increase. The switching point is when the drain source voltage of transistor 3 (V_{o-}) becomes

equal to the threshold voltage of transistor 2. Since all transistors in the circuit are identical, this point is when both input currents are equal. Also at this point transistors 2 and 4 turn on and transistors 1 and 3 turn off. If this analysis is repeated with I_- much larger than I_+ the same switching point is obtained with the roles of the transistors being reversed.

Since this circuit uses positive feedback it is important to isolate its output from the input differential amplifiers due to the fact that rapid changes in output can short through gate-source/drain capacitances and appear at the input [9]. Fortunately, this design does so automatically since the currents are all provided by current mirrors. In addition, because N-channel transistor latches are being used the output currents from the P-channel differential amplifiers will need additional P-channel current mirrors to make sure that they are flowing in the right direction.

5.6 Output Differential Amplifiers

The final stage of the comparator is the differential amplifier that converts the voltage difference from the latch into logic level output voltages (0 and 5[V]). The requirements for the output amplifiers are a large input range because the output voltages of the latches can be quite low, and a high output drive capability to minimize delay time [9]. A self-biased CMOS differential amplifier successfully accomplishes both of these goals [10]. The schematic for this circuit is shown in figure 5-5 on the next page [10]. All bias voltages in the circuit are taken from a single point (the negative output), thus allowing negative feedback to stabilize the

bias voltage [10]. This self-biasing allows the circuit to provide very high output currents for high drive capability [10].

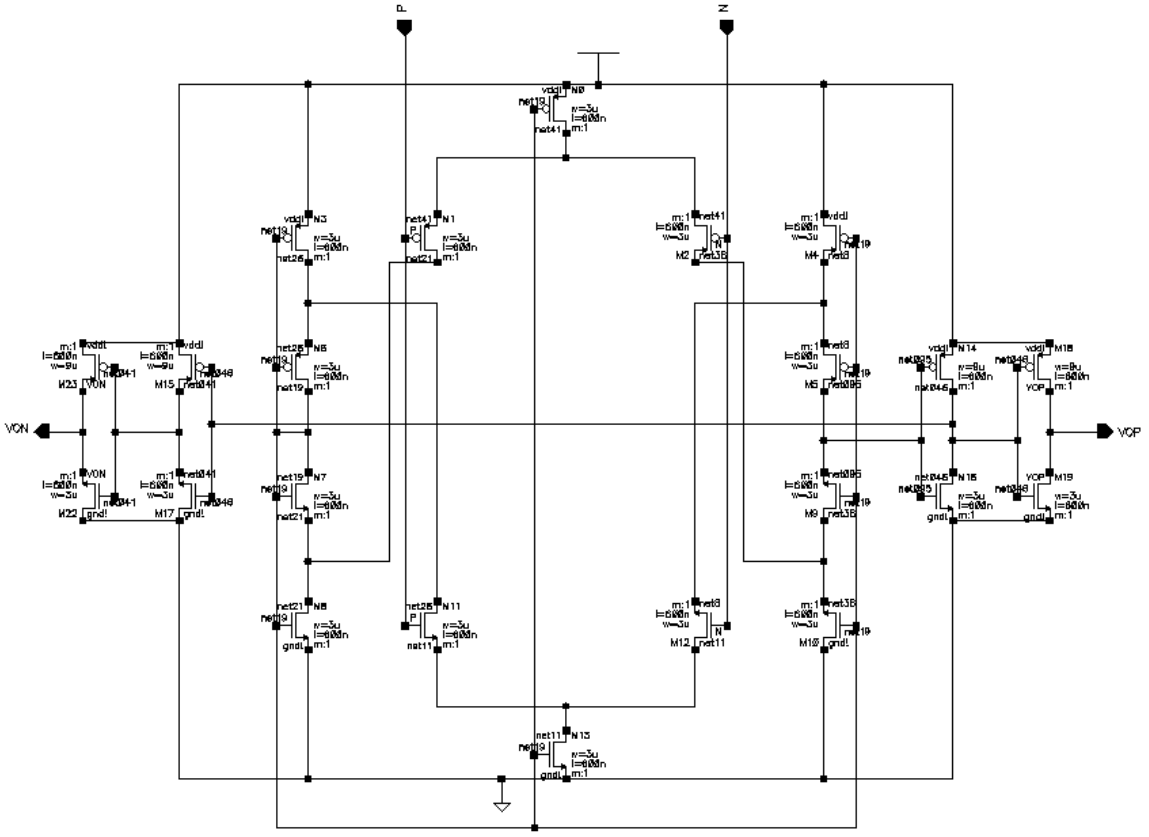

Figure 5-5: The self-biased CMOS differential amplifier that will be the output amplifier of the comparator.

In addition, two CMOS inverters have been connected to the outputs in order to ensure that the output voltages are at logic levels. These inverters have been designed so that their switching points are close to VCC/2.

5.7 Physical Layout and Simulation Results

The primary concern when laying out the comparators is to make sure that the properties of all transistors except for the physical dimensions are as identical as possible, particularly the input transistors because a mismatch can lead to an

offset voltage. Therefore, the common centroid approach should be used in the layout [1]. This means that each transistor's surroundings should be as identical as possible. In order to accomplish this each component and the overall layout will be made as mirror symmetric as possible.

Referring to the schematics presented in this chapter, making them mirror symmetric should be straightforward. For example, the current mirror output stages can be made mirror symmetric by moving the two output transistors to the center of the structure (there will be no input transistors because they are part of the differential amplifiers).

The whole layout should also be made as mirror symmetric as possible. Figure 5-6 on the next page shows the layout of the comparators. The differential amplifiers and the resistors are placed along the same axis. The resistor in the center is the biasing resistor and the resistors outside the main structure are the resistors that create the input voltages for the differential amplifiers that create the offset currents.

Adjoining the differential amplifiers are the current mirror output stages that produce the copies of the currents necessary for combination into the latch input currents. Notice that there are two extra components on each side of the first P-channel differential amplifier. These are the P-channel high accuracy current mirrors that reverse the direction of the currents from the P-channel differential amplifiers for the latches. Also notice that the offset differential

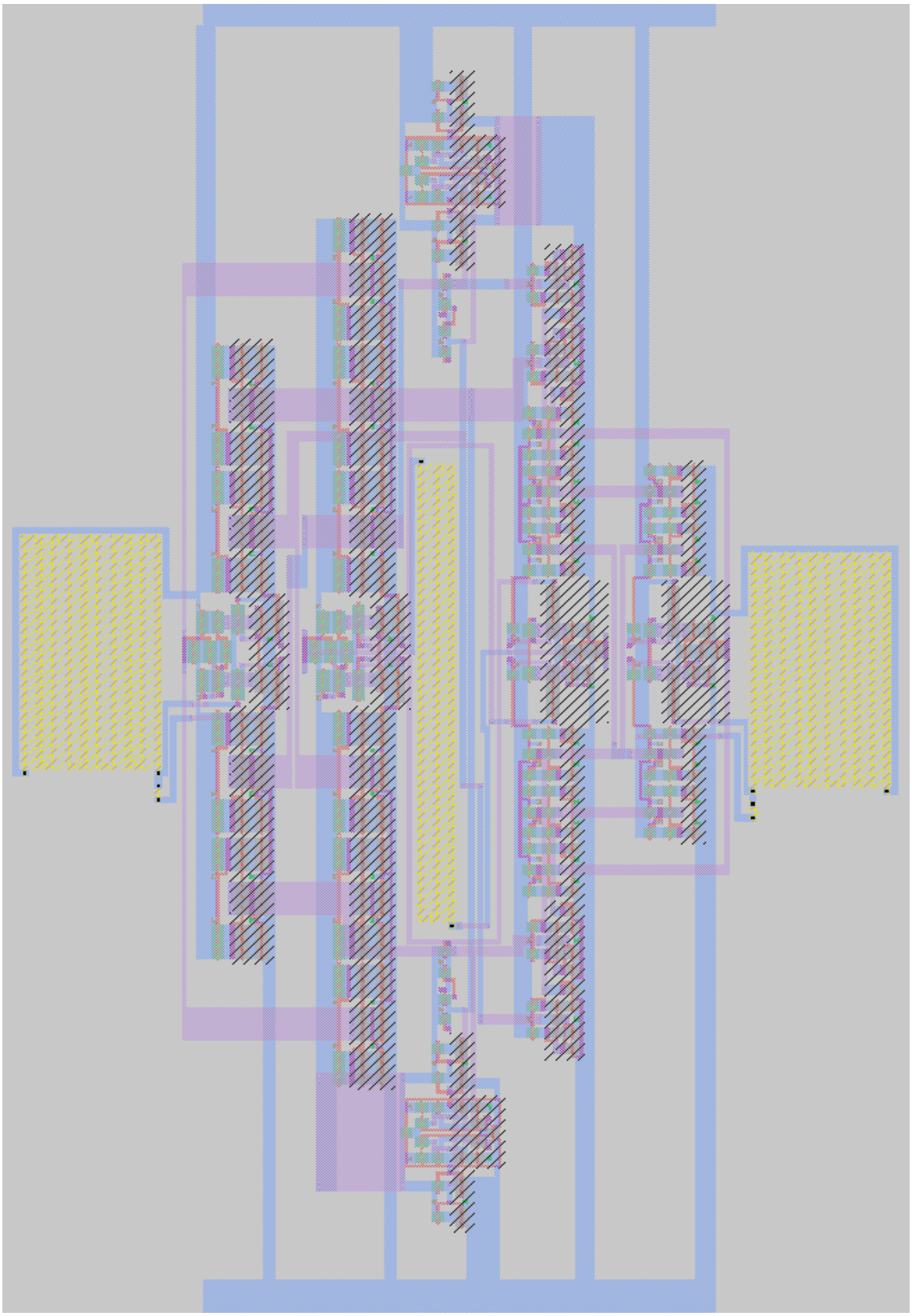

Figure 5-6: The physical layout of the L and R comparators.

amplifiers are biased by the output of the input differential amplifiers. This is how the bias point is preserved as described in section 5.3.

The goal of the first simulation is to verify the DC characteristics of the comparators. The simulation was conducted by holding the negative input voltage constant at 2.5[V] and sweeping the positive input voltage from 2.4[V] to 2.6[V], and the results are shown in figure 5-7.

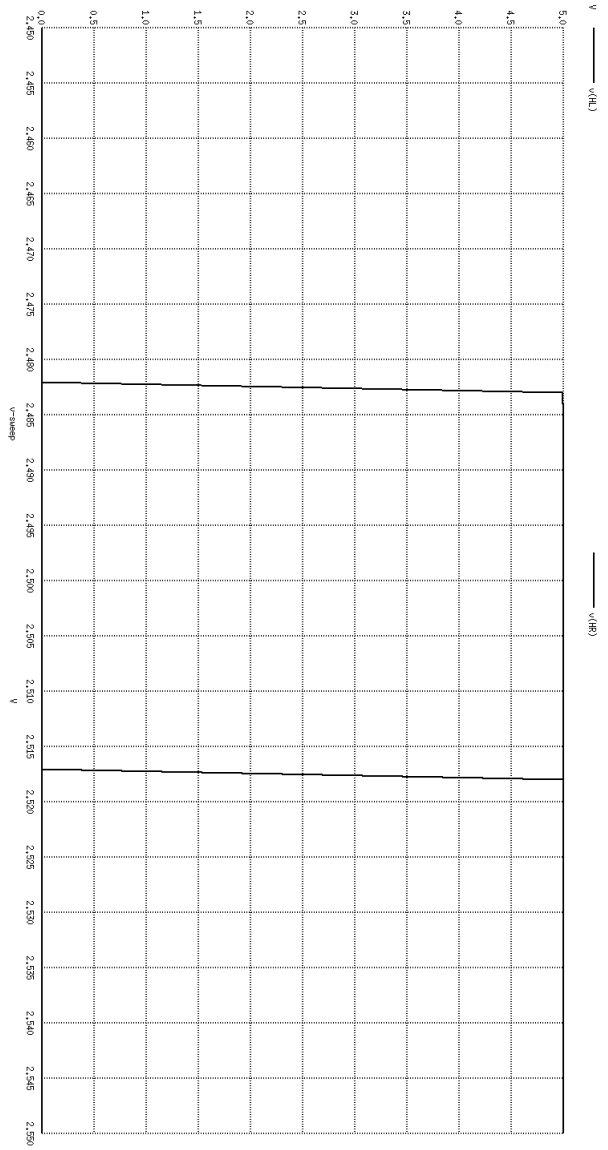

Figure 5-7: DC simulation results for the comparators.

The results of this simulation show that the offsets are slightly larger than LSB/2, however the circuit extractor did not give the resistors the value expected from the layout and this is most likely the cause. Whether this error is just the extractor or an actual layout issue remains to be seen. Other than the offset issue the comparators have the expected DC characteristics.

The final simulation will test the comparators' input range. This simulation was conducted by connecting the negative input to a piece wise linear source which starts at 2.5[V], goes to 5[V] at 100[µs], and finishes at 0[V] at 300[µs]. The positive input is connected to a square wave with an amplitude of 100[mV] and a period of 20[µs], it begins at -100[mV]. The simulation results are shown in figure 5-8.

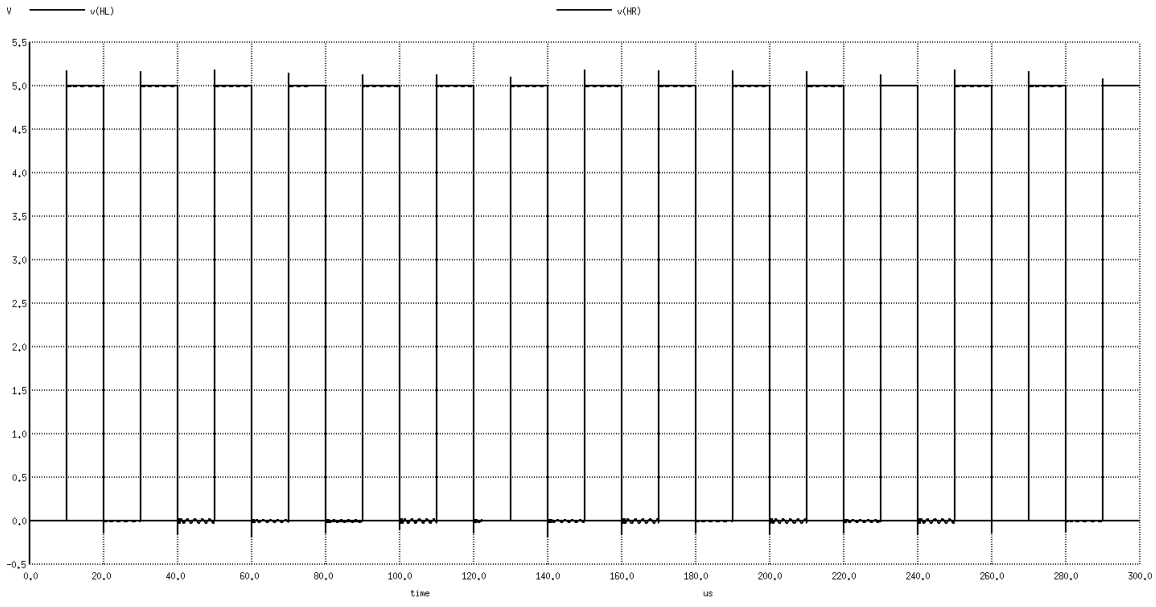

Figure 5-8: The input range simulation results.

The curves for HL and HR are virtually identical, as expected with this large of a voltage difference between the inputs. Furthermore, the desired rail-to-rail behavior is observed.

Chapter 6

Convergence Logic, Output and Final Circuit

6.1 Introduction

The purpose of the convergence determination logic is to determine when the correct digital output code has been reached and the conversion cycle is complete. The convergence criterion can be expressed in the following manner:

$$V_i - \frac{LSB}{2} < DAC(DigitalOutputCode) < V_i + \frac{LSB}{2}. \qquad (6-1)$$

The conditions expressed in 6-1 occur when HL=1 and HR=0, therefore the logic equation for the convergence determination circuit is:

$$CONV = HL * NHR. \qquad (6-2)$$

This is adequate for all cases except when the correct output code is 255. Since the addition and division by two circuit cannot output 255, and additional layer of logic is needed to test for this condition. The complete logic function can be expressed as:

$$CONV = HL * NHR + HR * b8 * b7 * b6 * b5 * b4 * b3 * b2. \qquad (6-3)$$

The second term of equation 6-3 addresses the 255 output issue, it will have its own gate because it will need to control the input to the flip-flop of the output register that holds the first bit.

The output register will be nearly identical in structure to the L and R registers shown in chapter 2. However, there are two major differences between the output register and the L and R registers. The first is that the flip-flops will not be driven by the clock signal, but by the output of the convergence determination logic. The second is that there will only be one bit with input control, the first bit to allow 255 to be output as discussed earlier.

The final circuit combines all of the circuits discussed in this chapter and the previous four chapters to implement the block diagram in chapter 1. The physical layout of the final circuit will be presented along with the results of simulations of the final circuit. This simulation will be the ultimate test of all the circuits discussed in this writing, determining whether they work together as predicted.

6.2 Convergence Determination Logic Design

The two terms of equation 6-3 will be combined using a NAND gate by the property:

$$NAND(NA, NB) = OR(A, B). \qquad (6-4)$$

This means that the complements of each term are required. This is simple to accomplish by using a conventional NAND gate to obtain the first term. Producing the second term is slightly more complex. At first glance it is just as simple as the first term, but with eight inputs. Using a single gate with eight inputs is not a good design because of large input capacitances. A good rule of thumb for the maximum number of inputs to a single gate is four [2]. Therefore,

the second term will be produced by two four input NAND gates whose outputs are combined by a NOR gate by the following property:

$$NOR(NA, NB) = AND(A, B). \qquad (6-5)$$

There will also be an inverter to produce the complement of this output for controlling the transmission gates that select the input to the first bit of the output register and for combining the two terms of equation 6-3.

There is an additional issue with the convergence logic; the comparator output is not valid at all times. The reason for this is because the DAC output is only valid when the clock signal is low. To address this, the output of the gates that implement equation 6-3 will be sent to a gated D-latch.

The function of the gated D-latch is to have its output (RDY) follow its input (CONV) while the clock signal (CLK) is low and hold the previous value of the input while the clock signal is high. Figure 6-1 below shows the schematic of the gated D-latch.

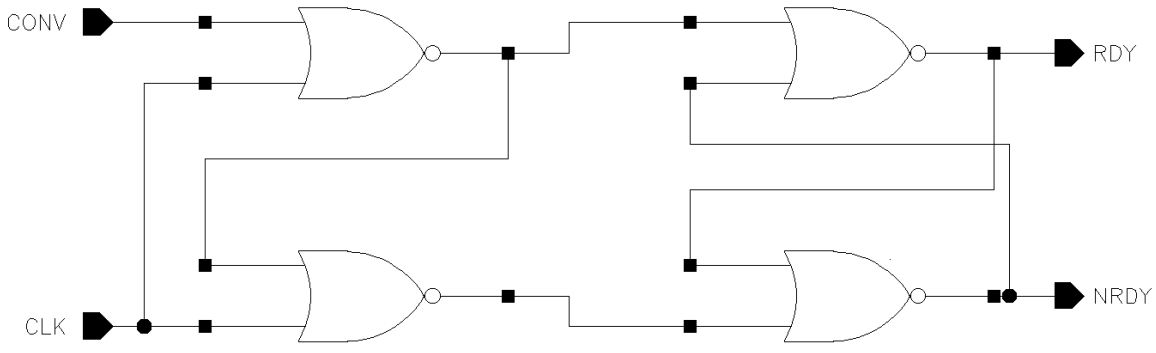

Figure 6-1: Schematic of the gated D-latch circuit.

The two NOR gates on the right form a conventional S-R latch. The R input is on the top and the S input is on the bottom. When S is 0 and R is 1 the output of the latch (RDY) is 0 (NRDY=1). When S is 1 and R is 0 RDY becomes a 1 (NRDY=0). When S and R are both 0 RDY will hold its previous value. The state where both S and R are 1 is not allowed and the logic gates at the input do not produce this state.

The two NOR gates on the left produce the inputs for the S-R latch. The resulting truth table is shown below.

Table 6-1: Truth Table for Gated D-Latch Input Logic Gates

CLK	CONV	S	R
0	0	0	1
0	1	1	0
1	0	0	0
1	1	0	0

The output of the logic gates clearly produces the desired inputs for the S-R latch, thus allowing the RDY signal to follow the output of equation 6-3 when the comparator output is valid and hold the previous value when the comparator output is invalid. The RDY and NRDY signals will act as the CLK and NCLK signals for the output register and be used to tell when a conversion is complete.

The logic controlling the input to the first bit of the output register is fairly straightforward. This logic consists of two CMOS transmission gates. The first transmission gate has as its input the least significant bit of the addition and division by two circuit output and is turned on when the second term of equation 6-3 produces a 0. The second transmission gate has a constant 1 as its input

and is turned on when the second term of equation 6-3 produces a 1, thus allowing the output to reach 255.

6.3 Physical Layouts and Simulation Results

The first components to be shown are the logic gates that implement equation 6-3. Figure 6-2 shows the physical layout of these circuits.

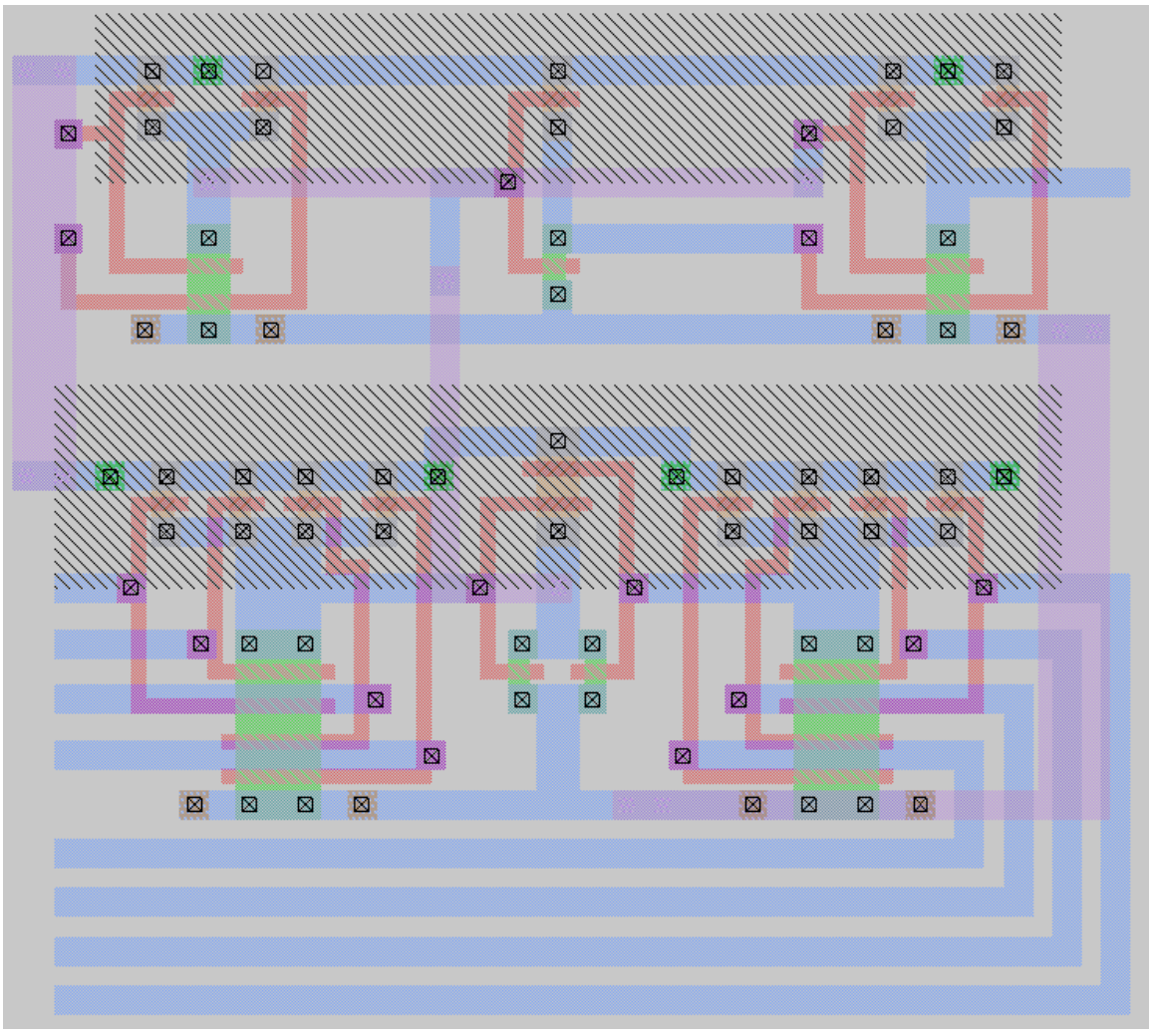

Figure 6-2: Physical layout that implements equation 6-3.

The gates were all designed using the procedure outlined in chapter 3, hence the wider N-channel transistors in the NAND gates. The NOR gate in the bottom

center combines the outputs of the two four-input NAND gates as discussed earlier. The NAND gate in the top right combines the complement of the output of the gates on the bottom, produced by the inverter in the top center, with the output of the NAND gate in the top left. This implements the logic function of equation 6-3. The simulation results for the circuit in figure 6-2 are shown below in figure 6-3.

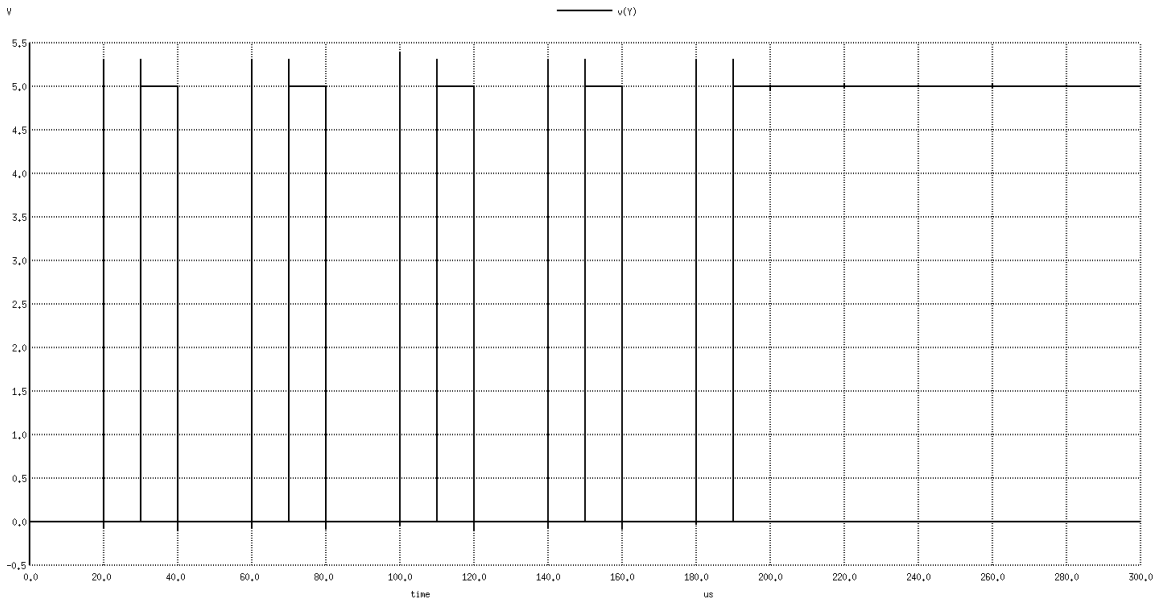

Figure 6-3: Simulation results for the logic gates of figure 6-2.

In this simulation HL is a pulse source starting at 0[V] and rising to 5[V] with a period of 20[µs]. NHR is also a pulse source with the same limits and a period of 40[µs]. Therefore, from 30[µs] to 40[µs], 70[µs] to 80[µs], etc. the output Y will be high, and that is what is observed. This pattern continues until 200[µs] when HR as well as the seven most significant bits go high and thus the output goes high and remains high for the remainder of the simulation.

The physical layout of the gated D-latch shown in figure 6-2 is shown in figure 6-4 below. Each component in this circuit is a 2-input CMOS NOR gate designed using the process discussed in chapter 3.

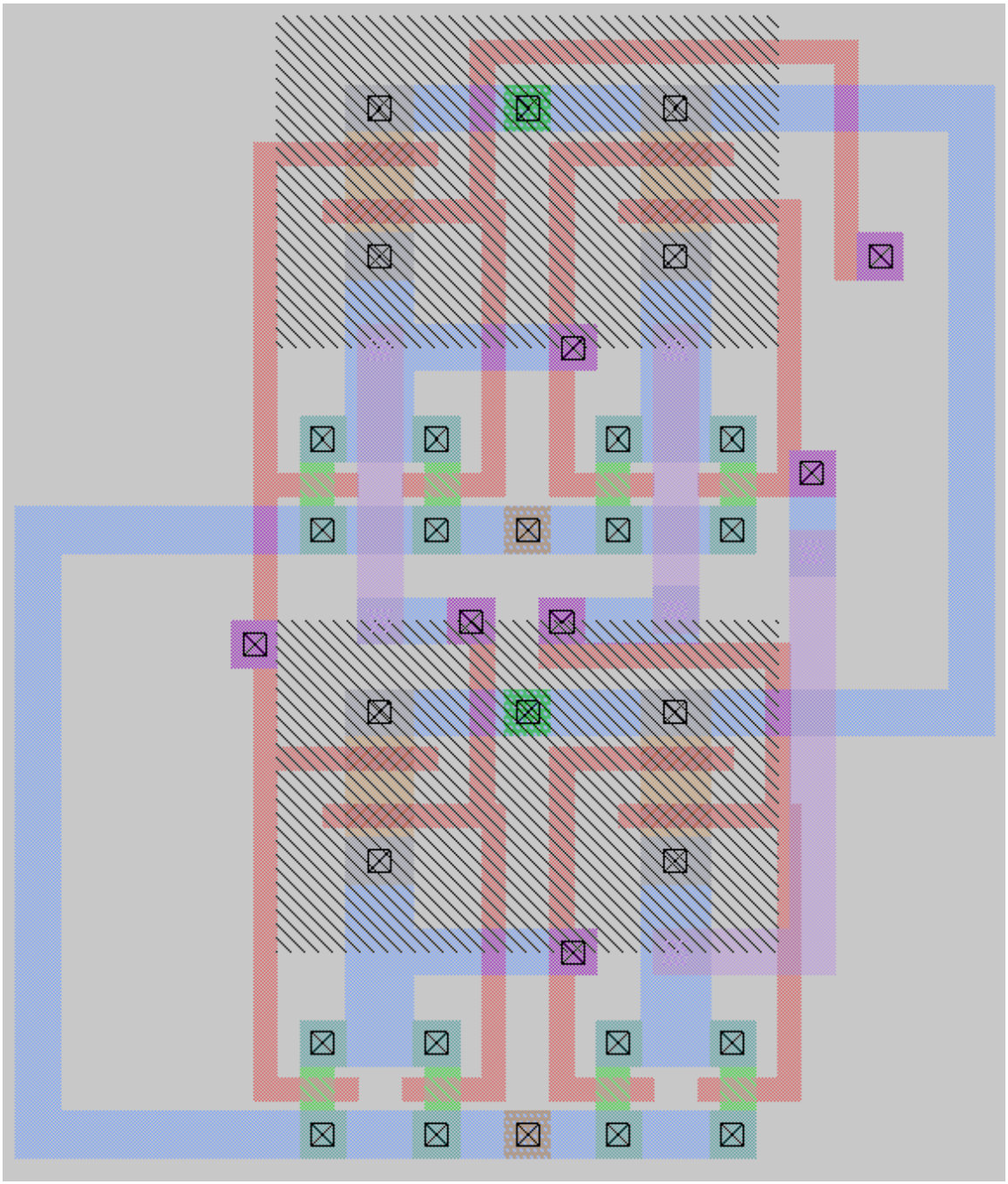

Figure 6-4: Physical layout of the gated D-latch.

The gates are placed very much like they appear in figure 6-1. The two gates on the left are form the logic that control the S-R latch, which is formed by the two gates on the right. The simulation results for this circuit are shown in figure 6-5.

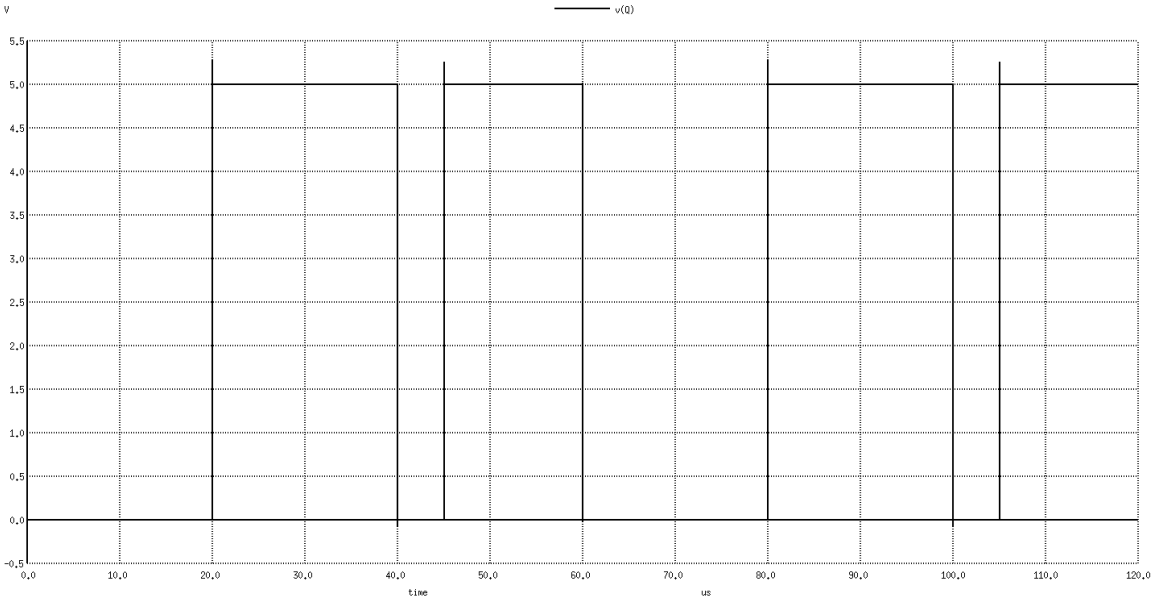

Figure 6-5: Simulation results for the gated D-latch.

This simulation uses a clock signal that starts low and is low every other 10[µs] interval, and is of course high in the other intervals. The input signal is a pulse that starts low, then goes to 5[V] at 15[µs] and has a period of 30[µs]. It is clear that the circuit's output follows the input at the even numbered 10[µs] intervals and retains its previous output at the odd numbered 10[µs] intervals. Thus, the gated D-latch of figure 6-4 functions correctly.

6.4 Final Circuit

The full circuit is laid out in the same manner as the block diagram in chapter 1. The comparators are on the top, the addition and division by two circuit is in the middle, the DAC is on the bottom, and the convergence

determination logic as well as the output register are to the right of everything else. Figure 6-6 shows the full layout.

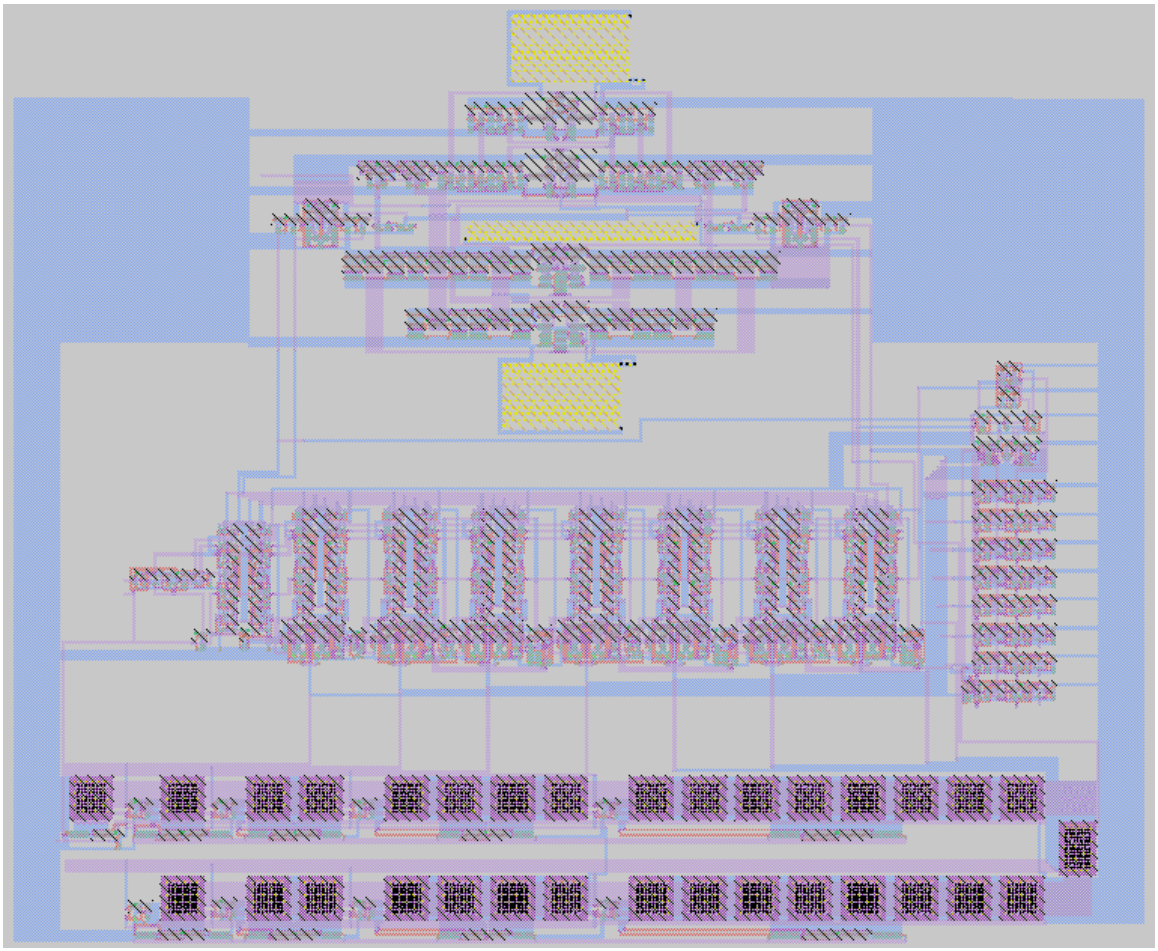

Figure 6-6: Final layout of the circuit.

The connections for the analog input, clock, and reset signals all extend towards the left where they will eventually be connected to bonding pads. A similar situation exists for the output signals. The conversion complete signal as well as all the bits of the digital output code extend towards the right where they will also be connected to bonding pads. The circuit was submitted to the MOSIS service for fabrication.

The final task is to simulate the final circuit. The circuit will be simulated with a clock speed of 1[MHz]. Analog inputs will run the entire range from 0 to 5[V] at 0.5[V] intervals. The simulation results are summarized in table 6-2.

Table 6-2: Simulation Results for the Final Circuit

Analog Input [V]	Output Code	Conversion Time [Clock Cycles]	Error
0	0	8	0
0.5	25	7	-1
1	51	6	0
1.5	77	7	0
2	103	5	1
2.5	128	8	0
3	153	7	-1
3.5	179	6	0
4	205	7	0
4.5	231	5	1
5	255	8	0

Figure 6-7 below shows a sample conversion cycle.

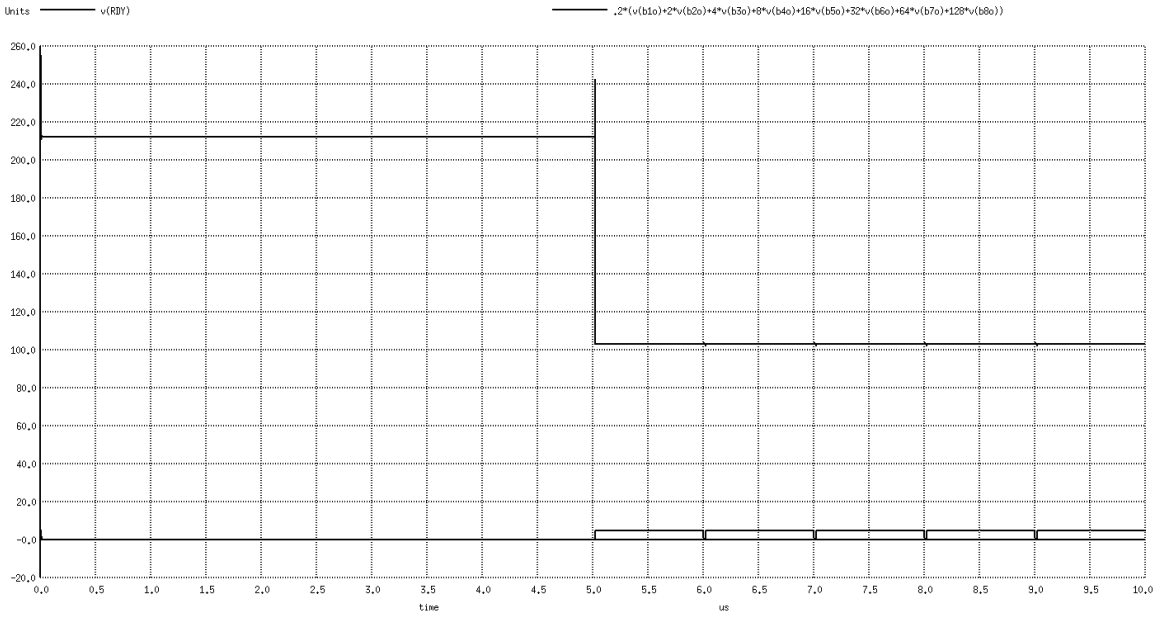

Figure 6-7: Conversion cycle for an input of 2[V].

In figure 6-7 the top curve is the digital output code and the bottom curve is the RDY signal. Since the clock frequency is 1[MHz] a clock cycle has a time period of 1[μs]. During the first clock cycle the circuit is reset and the conversion begins. Including the reset cycle, it takes five clock cycles to make the conversion.

The conversion times and error for several other analog inputs are shown in table 6-2. There is some error in some values, including 2[V]. This could be due to the possibility that the circuit extractor produced the wrong values for the offset resistors in the comparators. However, the circuit seems to generally work as predicted over the whole input range.

Chapter 7

Summary and Conclusion

7.1 Summary

The purpose of this project was to design and test an interval bisection quantization circuit for and 8-bit analog to digital converter. The reason for doing this is that the conversion times of most successive approximation ADCs in clock cycles are equal to the resolution of the converter, and it has been shown that this time can be improved upon by designing a quantization circuit that examines the entire digital output code at once.

The circuit has six major components: comparators with offsets of ±LSB/2, two registers whose next values are determined by multiplexors (controlled by the output of the comparators), a circuit that adds the values in the registers together and then divides the result by two, a charge scaling DAC that converts the output of the addition and division by two circuit into an analog voltage for the comparators, digital logic to determine when the conversion cycle has ended, and another register to hold the output digital word. Each of these components has been successfully designed, laid out, and simulated.

The final circuit, consisting of the six major pieces discussed previously, has also been successfully laid out and simulated. The simulation results show that the circuit does indeed convert faster in several cases than a conventional successive approximation ADC. The worst case scenario conversion time is

equal to that of the conventional successive approximation ADC. The circuit has been submitted to the MOSIS service for fabrication.

7.2 Conclusion

It has been demonstrated that this circuit successfully converts analog input signals to digital ones just as fast or faster than a conventional successive approximation ADC, however it still needs additional components before it can be considered a true analog to digital converter. The first component is a sample and hold circuit. This circuit is responsible for providing a constant value of the analog input signal for the entire conversion cycle. The second component is an anti-aliasing filter that prevents copies of the input signal aliased to higher frequencies by the sampling process from interfering with the input signal [1].

There are two main improvements that could be made to this circuit. The first is that it could be made using a process with a smaller feature size, such as 0.18[µm] or 0.15[µm]. This would dramatically reduce the area of the circuit. The difficulty with doing this is that transistor analytical models become far less accurate as feature sizes decrease due to short channel effects, such as carrier velocity saturation and drain induced barrier lowering. In particular, the reduced supply voltages of these small feature size processes would mean that the comparators would need to be much more sensitive and this would add an extra design challenge.

The second improvement is to make the comparators faster so that the circuit can better take advantage of the speed of the other components. The

best way to do this is to add additional amplifier stages, called preamplifiers, to the inputs of the comparators. The optimum configuration is six preamplifiers with a gain of 2.72 in cascade, but three preamplifiers with a gain of 6 in cascade gives almost identical results [1]. The challenge is designing these preamplifiers to have the necessary gain as well as meeting the rail-to-rail input requirements. This also requires additional area.

This circuit, along with the necessary additional components, is suited for any application that a successive approximation ADC is suited for. In particular, any application where the converter performs a conversion and then is turned off in between conversions is a good application for this converter due to the fact that this circuit converts faster than a conventional successive approximation ADC, thus allowing it to be turned off sooner saving power.

References

[1] Phillip E. Allen and Douglas R. Holdberg, “CMOS Analog Circuit Design,” Second Edition, Oxford University Press, 2003.

[2] Adel S. Sedra and Kenneth C. Smith, “Microelectronic Circuits,” Fifth Edition, Oxford University Press, 2004.

[3] John L. Hennessy and David A. Patterson, “Computer Organization and Design, The Hardware/Software Interface,” Fourth Edition, Morgan Kaufmann Publishers, 2009.

[4] MOSIS Scalable CMOS Design Rules Revision 8.00, https://www.mosis.com/files/scmos/scmos.pdf, Retrieved: April 2012.

[5] http://webpages.eng.wayne.edu/cadence/ECE6570/cap/Layout_of_Capacitor.html.

[6] David A. Hodges, Horace G. Jackson, and Resve A. Saleh, “Analysis and Design of Digital Integrated Circuits,” Third Edition, McGraw Hill, 2004.

[7] Yuhua Cheng, Mansun Chan, Kelvin Hui, Min-chie Jeng, Zhihong Liu, Jianhui Huang, Kai Chen, James Chen, Robert Tu, Ping K. Ko, Chenming Hu, “BSIM3v3 Manual (Final Version),” http://www.nikola.com/old/pdf/bsimset.pdf, Retrieved: September 2012.

[8] Zeki, A.; Kuntman, H.; , "Accurate and high output impedance current mirror suitable for CMOS current output stages," Electronics Letters , vol.33, no.12, pp.1042-1043, 5 Jun 1997.

[9] R. Jacob Baker, Harry W. Li, and David E. Boyce, “CMOS Circuit Design Layout, and Simulation,” John Wiley & Sons, Inc., 1998.

[10] Bazes, M.; , "Two novel fully complementary self-biased CMOS differential amplifiers," Solid-State Circuits, IEEE Journal of , vol.26, no.2, pp.165-168, Feb 1991.

Appendix A

Perl Script to Simulate Circuit Performance

```
#open or create output file in overwrite mode
open(OUTPUT,">output.txt") or die "Cannot open ouput file.\n";

#define a hash containing the resolutions under study
%exponents = (16 => 4, 32 => 5, 64 => 6, 128 => 7, 256 => 8,
            512 => 9, 1024 => 10, 2048 => 11, 4096 => 12);

#compute the average conversion time for different resolutions
for $i (16, 32, 64, 128, 256, 512, 1024, 2048, 4096) {
        $j = 0;
        $numBits = $exponents{$i};
        $avgTime = 0;
        while($j < ($i-1)) {
                $left = 0;
                $right = $i-1;
                $middle = int(($left+$right)/2);
                $numCycles = 1;
                until($middle == $j) {
                        if($middle < $j) {
                                $left = $middle;
                        }
                        else {
                                $right = $middle;
                        }
                        $middle = int(($left+$right)/2);
                        $numCycles++;
                }
                $avgTime+=$numCycles;
                $j++;
        }
        $avgTime+=$numBits;
        $avgTime/=$i;
        print OUTPUT "$numBits $avgTime $numBits\n";
}

#close output file
close(OUTPUT);
```

Appendix B

SPICE Code for the Multiplexor and D Flip-Flop Circuits

B.1 Multiplexor

```
* # FILE NAME:
/NET/HYDRA/HOME/C/CMTAYLO5/CADENCE/SIMULATION/MUX/HSPICES/
* EXTRACTED/NETLIST/MUX.C.RAW
* NETLIST OUTPUT FOR HSPICES.
* GENERATED ON OCT 13 23:25:56 2012

* FILE NAME: THESIS_MUX_EXTRACTED.S.
* SUBCIRCUIT FOR CELL: MUX.
* GENERATED FOR: HSPICES.
* GENERATED ON OCT 13 23:25:56 2012.

M2 Q RST 1 VCC  AMI06P  L=600.000021222513E-9 W=899.999974990351E-9
+AD=1.70999998015675E-12 AS=1.70999998015675E-12
PD=4.50000015916885E-6
+PS=4.50000015916885E-6 M=1
M3 Q NRST VRST VCC  AMI06P  L=600.000021222513E-9
W=899.999974990351E-9
+AD=1.70999998015675E-12 AS=1.70999998015675E-12
PD=4.50000015916885E-6
+PS=4.50000015916885E-6 M=1
M4 1 NH VH VCC  AMI06P  L=600.000021222513E-9 W=899.999974990351E-9
+AD=1.70999998015675E-12 AS=1.70999998015675E-12
PD=4.50000015916885E-6
+PS=4.50000015916885E-6 M=1
M5 1 H VNH VCC  AMI06P  L=600.000021222513E-9 W=899.999974990351E-9
+AD=1.70999998015675E-12 AS=1.70999998015675E-12
PD=4.50000015916885E-6
+PS=4.50000015916885E-6 M=1
M6 Q NRST 1 GND  AMI06N  L=600.000021222513E-9
W=899.999974990351E-9
+AD=1.70999998015675E-12 AS=1.70999998015675E-12
PD=4.50000015916885E-6
+PS=4.50000015916885E-6 M=1
M7 Q RST VRST GND  AMI06N  L=600.000021222513E-9
W=899.999974990351E-9
```

```
+AD=1.70999998015675E-12 AS=1.70999998015675E-12
PD=4.50000015916885E-6
+PS=4.50000015916885E-6 M=1
M8 1 NH VNH GND  AMI06N  L=600.000021222513E-9
W=899.999974990351E-9
+AD=1.70999998015675E-12 AS=1.70999998015675E-12
PD=4.50000015916885E-6
+PS=4.50000015916885E-6 M=1
M9 1 H VH GND  AMI06N  L=600.000021222513E-9 W=899.999974990351E-9
+AD=1.70999998015675E-12 AS=1.70999998015675E-12
PD=4.50000015916885E-6
+PS=4.50000015916885E-6 M=1
VDD VCC 0 DC 5V
RGND GND 0 1e-6
VHLR H 0 PULSE(0 5V 49ns 1ns 1ns 49ns 100ns)
mA VCC H NH VCC AMI06P W=0.9u L=0.6u
mB NH H GND GND AMI06N W=0.9u L=0.6u
VRST RST 0 PULSE(0 5V 99ns 1ns 1ns 99ns 200ns)
mC VCC RST NRST VCC AMI06P W=0.9u L=0.6u
mD NRST RST GND GND AMI06N W=0.9u L=0.6u
V1 VNH 0 PULSE(0 5V 9ns 1ns 1ns 9ns 20ns)
V2 VH 0 PULSE(0 5V 1ns 1ns 1ns 9ns 20ns)
V3 VRST 0 DC 5V
.tran 1ns 200ns
.control
run Mux.out
.endc
* DATE: Jul 20/12
* LOT: v25u                WAF: 5101
* Temperature_parameters=Default
.MODEL AMI06N NMOS (                              LEVEL   = 8
+VERSION = 3.2           TNOM    = 27             TOX     = 1.38E-8
+XJ      = 1.5E-7        NCH     = 1.7E17         VTH0    = 0.6206755
+K1      = 0.9154941     K2      = -0.1070065     K3      = 23.1322127
+K3B     = -10.118157    W0      = 2.578041E-8    NLX     = 1.004874E-9
+DVT0W   = 0             DVT1W   = 0              DVT2W   = 0
+DVT0    = 0.7455079     DVT1    = 0.3803402      DVT2    = -0.5
+U0      = 454.9970557   UA      = 1.178822E-13   UB      = 1.468939E-18
+UC      = 8.099128E-12  VSAT    = 1.980587E5     A0      = 0.6470937
+AGS     = 0.1501036     B0      = 1.922533E-6    B1      = 5E-6
+KETA    = -4.48048E-3   A1      = 3.937357E-5    A2      = 0.3
+RDSW    = 950.3826959   PRWG    = 0.1146132      PRWB    = 8.477104E-3
+WR      = 1             WINT    = 2.512136E-7    LINT    = 4.599276E-8
+XL      = 1E-7          XW      = 0              DWG     = -4.777866E-9
+DWB     = 2.160953E-8   VOFF    = -9.792248E-5   NFACTOR = 0.9637953
+CIT     = 0             CDSC    = 2.4E-4         CDSCD   = 0
```

```
+CDSCB   = 0           ETA0    = 2.059644E-3    ETAB    = -2.60819E-4
+DSUB    = 0.0679985     PCLM    = 2.189859      PDIBLC1 = 9.878026E-5
+PDIBLC2 = 2.114418E-3    PDIBLCB = 0.0668401     DROUT   = 2.351591E-3
+PSCBE1  = 3.480502E8     PSCBE2  = 1.845207E-6    PVAG    = 0
+DELTA   = 0.01          RSH     = 80.9          MOBMOD  = 1
+PRT     = 0           UTE     = -1.5          KT1     = -0.11
+KT1L    = 0           KT2     = 0.022         UA1     = 4.31E-9
+UB1     = -7.61E-18     UC1     = -5.6E-11      AT      = 3.3E4
+WL      = 0           WLN     = 1           WW      = 0
+WWN     = 1           WWL     = 0           LL      = 0
+LLN     = 1           LW      = 0           LWN     = 1
+LWL     = 0           CAPMOD  = 2           XPART   = 0.5
+CGDO    = 1.93E-10      CGSO    = 1.93E-10      CGBO    = 1E-9
+CJ      = 4.161289E-4    PB      = 0.8283848     MJ      = 0.4275867
+CJSW    = 3.259973E-10   PBSW    = 0.8           MJSW    = 0.1846732
+CJSWG   = 1.64E-10      PBSWG   = 0.8           MJSWG   = 0.2019414
+CF      = 0           PVTH0   = 0.0277539     PRDSW   = 180.4529442
+PK2     = -0.0767092     WKETA   = -8.914329E-3   LKETA   = -2.154684E-3    )
*
*
.MODEL AMI06P PMOS (                          LEVEL   = 8
+VERSION = 3.2           TNOM    = 27            TOX     = 1.38E-8
+XJ      = 1.5E-7        NCH     = 1.7E17        VTH0    = -0.9152268
+K1      = 0.553472      K2      = 7.871921E-3    K3      = 8.9476091
+K3B     = 0.5321457     W0      = 1E-8          NLX     = 1.004257E-9
+DVT0W   = 0           DVT1W   = 0           DVT2W   = 0
+DVT0    = 0.4713552     DVT1    = 0.18506       DVT2    = -0.3
+U0      = 201.3603195    UA      = 2.48572E-9     UB      = 1.005454E-21
+UC      = -1E-10        VSAT    = 1.139712E5     A0      = 0.7835701
+AGS     = 0.120596      B0      = 8.005994E-7    B1      = 2.880961E-8
+KETA    = -4.865785E-3   A1      = 5.760921E-4    A2      = 0.5104845
+RDSW    = 3E3           PRWG    = -0.0219752     PRWB    = -0.0907724
+WR      = 1.01          WINT    = 2.298706E-7    LINT    = 9.979707E-8
+XL      = 1E-7          XW      = 0           DWG     = 2.212012E-9
+DWB     = -1.864478E-8   VOFF    = -0.0484713     NFACTOR = 0.4480928
+CIT     = 0           CDSC    = 2.4E-4        CDSCD   = 0
+CDSCB   = 0           ETA0    = 9.407653E-3    ETAB    = -0.2
+DSUB    = 1           PCLM    = 2.3394089     PDIBLC1 = 0.0769106
+PDIBLC2 = 4.01984E-3     PDIBLCB = -0.0292094     DROUT   = 0.2662061
+PSCBE1  = 8E10          PSCBE2  = 8.957714E-8    PVAG    = 0
+DELTA   = 0.01          RSH     = 106           MOBMOD  = 1
+PRT     = 0           UTE     = -1.5          KT1     = -0.11
+KT1L    = 0           KT2     = 0.022         UA1     = 4.31E-9
+UB1     = -7.61E-18     UC1     = -5.6E-11      AT      = 3.3E4
+WL      = 0           WLN     = 1           WW      = 0
+WWN     = 1           WWL     = 0           LL      = 0
```

```
+LLN     = 1            LW      = 0            LWN     = 1
+LWL     = 0            CAPMOD  = 2            XPART   = 0.5
+CGDO    = 2.28E-10     CGSO    = 2.28E-10     CGBO    = 1E-9
+CJ      = 7.095753E-4  PB      = 0.8613744    MJ      = 0.4858698
+CJSW    = 2.182984E-10 PBSW    = 0.8921837    MJSW    = 0.1908523
+CJSWG   = 6.4E-11      PBSWG   = 0.8          MJSWG   = 0.2261452
+CF      = 0            PVTH0   = 5.98016E-3   PRDSW   = 14.8598424
+PK2     = 3.73981E-3   WKETA   = 9.043859E-3  LKETA   = -0.0103186     )
*
*
* END OF NETLIST
.OP
.save
.OPTION  INGOLD=2 ARTIST=2 PSF=2
+        PROBE=0 TEMP=25
.END
```

B.2 D Flip-Flop Circuit

```
* # FILE NAME:
/NET/HYDRA/HOME/C/CMTAYLO5/CADENCE/SIMULATION/RISINGEDGEDF
F/
* HSPICES/EXTRACTED/NETLIST/RISINGEDGEDFF.C.RAW
* NETLIST OUTPUT FOR HSPICES.
* GENERATED ON OCT 13 23:55:56 2012

* FILE NAME: THESIS_RISINGEDGEDFF_EXTRACTED.S.
* SUBCIRCUIT FOR CELL: RISINGEDGEDFF.
* GENERATED FOR: HSPICES.
* GENERATED ON OCT 13 23:55:57 2012.

M5 Q CLK 4 VCC  AMI06P  L=600.000021222513E-9 W=899.999974990351E-9
+AD=1.70999998015675E-12 AS=1.70999998015675E-12
PD=4.50000015916885E-6
+PS=4.50000015916885E-6 M=1
M6 4 NCLK 3 VCC  AMI06P  L=600.000021222513E-9 W=899.999974990351E-
9
+AD=1.70999998015675E-12 AS=1.70999998015675E-12
PD=4.50000015916885E-6
+PS=4.50000015916885E-6 M=1
M7 3 NCLK 1 VCC  AMI06P  L=600.000021222513E-9 W=899.999974990351E-
9
+AD=1.70999998015675E-12 AS=1.70999998015675E-12
PD=4.50000015916885E-6
+PS=4.50000015916885E-6 M=1
M8 1 CLK D VCC  AMI06P  L=600.000021222513E-9 W=899.999974990351E-9
```

+AD=1.70999998015675E-12 AS=1.70999998015675E-12
PD=4.50000015916885E-6
+PS=4.50000015916885E-6 M=1
M9 Q NQ VCC VCC AMI06P L=600.000021222513E-9
W=899.999974990351E-9
+AD=1.70999998015675E-12 AS=1.70999998015675E-12
PD=4.50000015916885E-6
+PS=4.50000015916885E-6 M=1
M10 NQ 4 VCC VCC AMI06P L=600.000021222513E-9
W=899.999974990351E-9
+AD=1.70999998015675E-12 AS=1.70999998015675E-12
PD=4.50000015916885E-6
+PS=4.50000015916885E-6 M=1
M11 3 2 VCC VCC AMI06P L=600.000021222513E-9 W=899.999974990351E-
9
+AD=1.70999998015675E-12 AS=1.70999998015675E-12
PD=4.50000015916885E-6
+PS=4.50000015916885E-6 M=1
M12 2 1 VCC VCC AMI06P L=600.000021222513E-9 W=899.999974990351E-
9
+AD=1.70999998015675E-12 AS=1.70999998015675E-12
PD=4.50000015916885E-6
+PS=4.50000015916885E-6 M=1
M13 Q NCLK 4 GND AMI06N L=600.000021222513E-9
W=899.999974990351E-9
+AD=1.70999998015675E-12 AS=1.70999998015675E-12
PD=4.50000015916885E-6
+PS=4.50000015916885E-6 M=1
M14 4 CLK 3 GND AMI06N L=600.000021222513E-9 W=899.999974990351E-
9
+AD=1.70999998015675E-12 AS=1.70999998015675E-12
PD=4.50000015916885E-6
+PS=4.50000015916885E-6 M=1
M15 3 CLK 1 GND AMI06N L=600.000021222513E-9 W=899.999974990351E-
9
+AD=1.70999998015675E-12 AS=1.70999998015675E-12
PD=4.50000015916885E-6
+PS=4.50000015916885E-6 M=1
M16 1 NCLK D GND AMI06N L=600.000021222513E-9
W=899.999974990351E-9
+AD=1.70999998015675E-12 AS=1.70999998015675E-12
PD=4.50000015916885E-6
+PS=4.50000015916885E-6 M=1
M17 GND NQ Q GND AMI06N L=600.000021222513E-9
W=899.999974990351E-9

```
+AD=1.70999998015675E-12 AS=1.70999998015675E-12
PD=4.50000015916885E-6
+PS=4.50000015916885E-6 M=1
M18 GND 4 NQ GND  AMI06N  L=600.000021222513E-9
W=899.999974990351E-9
+AD=1.70999998015675E-12 AS=1.70999998015675E-12
PD=4.50000015916885E-6
+PS=4.50000015916885E-6 M=1
M19 GND 2 3 GND  AMI06N  L=600.000021222513E-9
W=899.999974990351E-9
+AD=1.70999998015675E-12 AS=1.70999998015675E-12
PD=4.50000015916885E-6
+PS=4.50000015916885E-6 M=1
M20 GND 1 2 GND  AMI06N  L=600.000021222513E-9
W=899.999974990351E-9
+AD=1.70999998015675E-12 AS=1.70999998015675E-12
PD=4.50000015916885E-6
+PS=4.50000015916885E-6 M=1
VDD VCC 0 DC 5V
RGND GND 0 1e-6
VD D 0 PULSE(0 5V 12ns 1e-2ns 1e-2ns 12ns 24ns)
VCLK CLK 0 PULSE(0 5V 10ns 1e-2ns 1e-2ns 10ns 20ns)
ma VCC CLK NCLK VCC AMI06P W=0.9u L=0.6u
mb NCLK CLK GND GND AMI06N W=0.9u L=0.6u
.tran 1ns 100ns
.control
run DFF.out
.endc
* DATE: Jul 20/12
* LOT: v25u                WAF: 5101
* Temperature_parameters=Default
.MODEL AMI06N NMOS (                          LEVEL   = 8
+VERSION = 3.2           TNOM    = 27           TOX     = 1.38E-8
+XJ      = 1.5E-7        NCH     = 1.7E17       VTH0    = 0.6206755
+K1      = 0.9154941     K2      = -0.1070065   K3      = 23.1322127
+K3B     = -10.118157    W0      = 2.578041E-8  NLX     = 1.004874E-9
+DVT0W   = 0             DVT1W   = 0            DVT2W   = 0
+DVT0    = 0.7455079     DVT1    = 0.3803402    DVT2    = -0.5
+U0      = 454.9970557   UA      = 1.178822E-13 UB      = 1.468939E-18
+UC      = 8.099128E-12  VSAT    = 1.980587E5   A0      = 0.6470937
+AGS     = 0.1501036     B0      = 1.922533E-6  B1      = 5E-6
+KETA    = -4.48048E-3   A1      = 3.937357E-5  A2      = 0.3
+RDSW    = 950.3826959   PRWG    = 0.1146132    PRWB    = 8.477104E-3
+WR      = 1             WINT    = 2.512136E-7  LINT    = 4.599276E-8
+XL      = 1E-7          XW      = 0            DWG     = -4.777866E-9
+DWB     = 2.160953E-8   VOFF    = -9.792248E-5 NFACTOR = 0.9637953
```

```
+CIT     = 0            CDSC    = 2.4E-4        CDSCD   = 0
+CDSCB   = 0            ETA0    = 2.059644E-3   ETAB    = -2.60819E-4
+DSUB    = 0.0679985    PCLM    = 2.189859      PDIBLC1 = 9.878026E-5
+PDIBLC2 = 2.114418E-3   PDIBLCB = 0.0668401     DROUT   = 2.351591E-3
+PSCBE1  = 3.480502E8    PSCBE2  = 1.845207E-6   PVAG    = 0
+DELTA   = 0.01         RSH     = 80.9          MOBMOD  = 1
+PRT     = 0            UTE     = -1.5          KT1     = -0.11
+KT1L    = 0            KT2     = 0.022         UA1     = 4.31E-9
+UB1     = -7.61E-18    UC1     = -5.6E-11      AT      = 3.3E4
+WL      = 0            WLN     = 1             WW      = 0
+WWN     = 1            WWL     = 0             LL      = 0
+LLN     = 1            LW      = 0             LWN     = 1
+LWL     = 0            CAPMOD  = 2             XPART   = 0.5
+CGDO    = 1.93E-10     CGSO    = 1.93E-10      CGBO    = 1E-9
+CJ      = 4.161289E-4  PB      = 0.8283848     MJ      = 0.4275867
+CJSW    = 3.259973E-10 PBSW    = 0.8           MJSW    = 0.1846732
+CJSWG   = 1.64E-10     PBSWG   = 0.8           MJSWG   = 0.2019414
+CF      = 0            PVTH0   = 0.0277539     PRDSW   = 180.4529442
+PK2     = -0.0767092   WKETA   = -8.914329E-3  LKETA   = -2.154684E-3  )
*
*
.MODEL AMI06P PMOS (                           LEVEL   = 8
+VERSION = 3.2          TNOM    = 27            TOX     = 1.38E-8
+XJ      = 1.5E-7       NCH     = 1.7E17        VTH0    = -0.9152268
+K1      = 0.553472     K2      = 7.871921E-3   K3      = 8.9476091
+K3B     = 0.5321457    W0      = 1E-8          NLX     = 1.004257E-9
+DVT0W   = 0            DVT1W   = 0             DVT2W   = 0
+DVT0    = 0.4713552    DVT1    = 0.18506       DVT2    = -0.3
+U0      = 201.3603195  UA      = 2.48572E-9    UB      = 1.005454E-21
+UC      = -1E-10       VSAT    = 1.139712E5    A0      = 0.7835701
+AGS     = 0.120596     B0      = 8.005994E-7   B1      = 2.880961E-8
+KETA    = -4.865785E-3 A1      = 5.760921E-4   A2      = 0.5104845
+RDSW    = 3E3          PRWG    = -0.0219752    PRWB    = -0.0907724
+WR      = 1.01         WINT    = 2.298706E-7   LINT    = 9.979707E-8
+XL      = 1E-7         XW      = 0             DWG     = 2.212012E-9
+DWB     = -1.864478E-8 VOFF    = -0.0484713    NFACTOR = 0.4480928
+CIT     = 0            CDSC    = 2.4E-4        CDSCD   = 0
+CDSCB   = 0            ETA0    = 9.407653E-3   ETAB    = -0.2
+DSUB    = 1            PCLM    = 2.3394089     PDIBLC1 = 0.0769106
+PDIBLC2 = 4.01984E-3    PDIBLCB = -0.0292094    DROUT   = 0.2662061
+PSCBE1  = 8E10         PSCBE2  = 8.957714E-8   PVAG    = 0
+DELTA   = 0.01         RSH     = 106           MOBMOD  = 1
+PRT     = 0            UTE     = -1.5          KT1     = -0.11
+KT1L    = 0            KT2     = 0.022         UA1     = 4.31E-9
+UB1     = -7.61E-18    UC1     = -5.6E-11      AT      = 3.3E4
+WL      = 0            WLN     = 1             WW      = 0
```

```
+WWN     = 1            WWL     = 0            LL      = 0
+LLN     = 1            LW      = 0            LWN     = 1
+LWL     = 0            CAPMOD  = 2            XPART   = 0.5
+CGDO    = 2.28E-10     CGSO    = 2.28E-10     CGBO    = 1E-9
+CJ      = 7.095753E-4  PB      = 0.8613744    MJ      = 0.4858698
+CJSW    = 2.182984E-10 PBSW    = 0.8921837    MJSW    = 0.1908523
+CJSWG   = 6.4E-11      PBSWG   = 0.8          MJSWG   = 0.2261452
+CF      = 0            PVTH0   = 5.98016E-3   PRDSW   = 14.8598424
+PK2     = 3.73981E-3   WKETA   = 9.043859E-3  LKETA   = -0.0103186    )
*
*
* END OF NETLIST
.OP
.save
.OPTION  INGOLD=2 ARTIST=2 PSF=2
+        PROBE=0 TEMP=25
.END
```

Appendix C

Addition and Division by Two High Speed Simulation Results

The results of the 50[MHz] simulation of the addition and division by two circuit were not shown in the main text because they are not as clear as the lower speed results. This is due to the fact that the output code was obtained by the following equation:

$$Output\ Code = 0.2[v(b0) + 2v(b1) + 4v(b2) + 8v(b3) + 16v(b4) + 32v(b5) + 64v(b6) + 128v(b7)]. \quad \text{(C – 1)}$$

The results are shown in figure C-1.

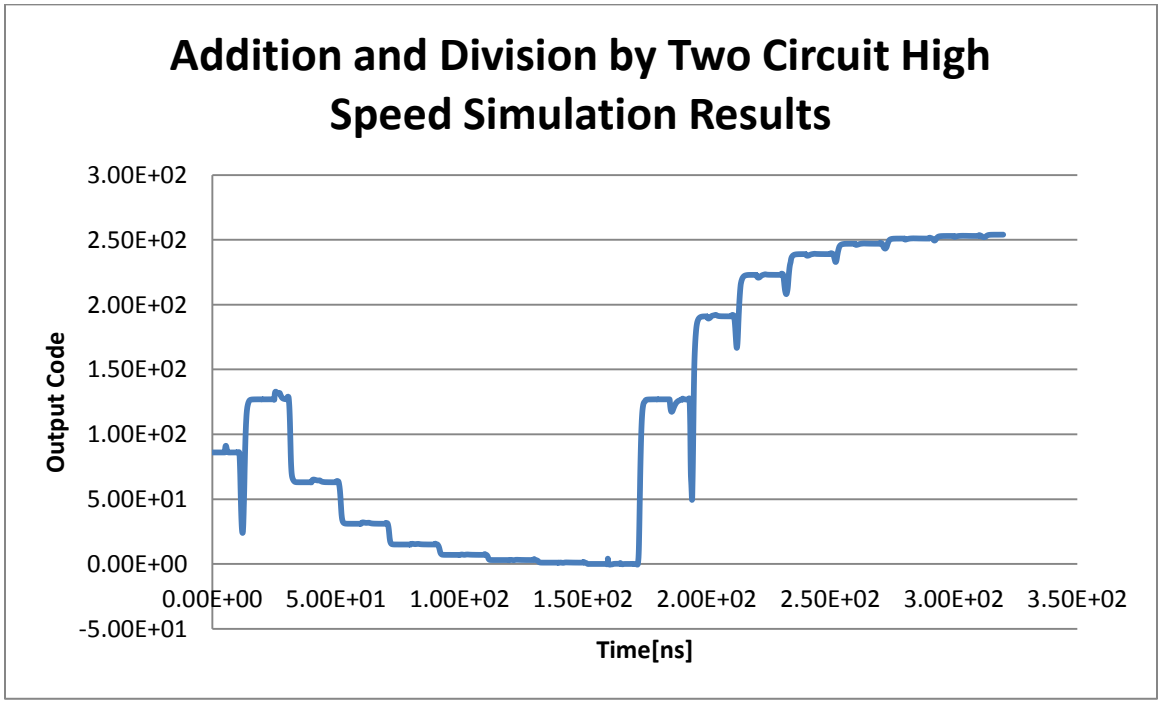

Figure C-1: 50[MHz] simulation results for the addition and division by two circuit.

Appendix D

Output Values of DAC Simulation

Table D-1: Output Voltages of DAC Simulation Rounded to Two Places After the Decimal Point

Input Code	Output Value[V]	|Error|[LSB]
0	0.00E+00	0
17	3.30E-01	0.104
34	0.66	0.208
51	1	0.2
68	1.33	0.096
85	1.66	0.008
102	1.99	0.112
119	2.32	0.216
136	2.66	0.192
153	2.99	0.088
170	3.32	0.016
187	3.65	0.12
204	3.98	0.224
221	4.32	0.184
238	4.65	0.08
255	4.98	0.024