Design and Implementation of an Ultra-Low Power Analog-to-Digital Converter

MADVLSI Final Report

Sam Becht, Jared Kirschner, Amy Whitcombe

December 17, 2012

1 Introduction

Mobile, embedded, and remote systems require the ability to run critical, high-performance operations for long durations of time. The available energy for such applications is often limited by size constraints, cost, or inconvenience. Thus, the ability to design circuitry which can operate at low supply voltages and with little power consumption is critical for such applications. An important component in many such applications is an analog-to-digital converter (ADC), which allows for a digital processor to interpret real-world signals. Heemin Yang proposed a novel ultra-low power ADC in his Ph.D. thesis titled "A Time-Based Energy-Efficient Analog-to-Digital Converter" for the Electrical Engineering and Computer Science Department at the Massachusetts Institute of Technology.[1] In this project, we implemented an ultra-low power ADC integrated circuit based on the design proposed in Yang's thesis.

2 Theory of Operation

The inspiration for this project comes from a unique time-based ADC presented in a thesis by Heemin Yang.[1] This particular ADC is designed to meet the following performance specifications with the MOSIS TSMC 0.35 μm process: operate on a 3 V supply with 75 μW of power consumption, thermal-noise-limited precision of 12-bits, and a $\approx$30 kHz sampling rate.[1][3] A later iteration of the design with a 0.18μm process achieved 8 bits of precision and a 45 kHz sample rate with only 960 nW of power consumption—potentially the most energy efficient ADC per quantization level reported to date.[2]

This time-based ADC has two separate phases used to digitize the analog signal. The first phase of the conversion process, shown in Figure 1, is a dual-slope stage which calculates the first two most-significant bits (MSBs). Capacitor C_1 is first charged with the input current I_{in} for one clock cycle. A reference current I_{ref} then charges capacitor C_2 until the voltages of the two capacitors are equal. The number of clock cycles which occur before V_{C_2} matches V_{C_1} represent the 2 MSBs. Thus, $I_{in} \leq 4 \cdot I_{ref}$ is a restriction on the input current.

The second phase of the ADC operation, shown in Figure 2, is an algorithmic phase started with a temporal residual from the dual-slope phase. This temporal residual is the difference between the end of the dual-slope phase and the start of the present clock cycle. If this residual is less than half of a clock cycle, the next bit is a 0; if this residual is more than half of a clock cycle, the next bit is a 1. Re-stated, the residual is equal to $0.5 + \epsilon$ where $-0.5 \leq \epsilon \leq 0.5$; the sign of epsilon encodes the next bit. The process of determining the sign of epsilon is started by converting the residual time $0.5 + \epsilon$ to a voltage: capacitor C_1 is charged with I_{ref} until the next clock cycle. This time-to-voltage stage charges C_1 for $1 - (0.5 - \epsilon) = 0.5 + \epsilon$ time. In the proceeding voltage-to-time stage, the capacitor C_2 is charged up to the voltage of C_1—a process which takes $0.5 + \epsilon$ time referenced against the start of the clock cycle. Next, one of the capacitors is grounded and then charged back up to the voltage of the other capacitor, again taking $0.5 + \epsilon$ time. In total, the amount of time elapsed from the end of the time-to-voltage stage is $2(0.5 + \epsilon) = 1 + \epsilon$ time. Thus, if $\epsilon < 0$, 0 clock cycles will have completed; if $\epsilon \geq 0$, 1 clock cycle will have completed. The time elapsed since the start of clock cycle is the temporal residual for the next iteration of the algorithm. A small temporal residual after the dual-slope stage results in 1 clock cycle elapsing before the end of the first iteration of the algorithm, the number of clock cycles (0 or 1) represents

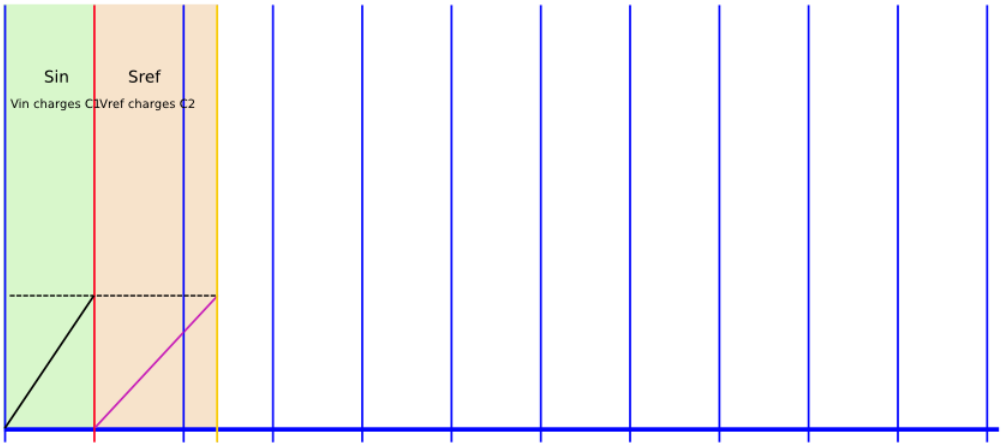

Figure 1: Dual-slope phase of the time-based ADC used to calculate the first two most-significant bits. In this case, the digitized value is 01 because one clock cycle occurred before the reference capacitor C_2 reached the voltage of the input capacitor C_1. Note: black lines denote charging by I_{in}, purple lines denote charging by I_{ref}, red vertical lines denote state machine transitions on clock edges, and yellow vertical lines denote state machine transitions on compator pulses.

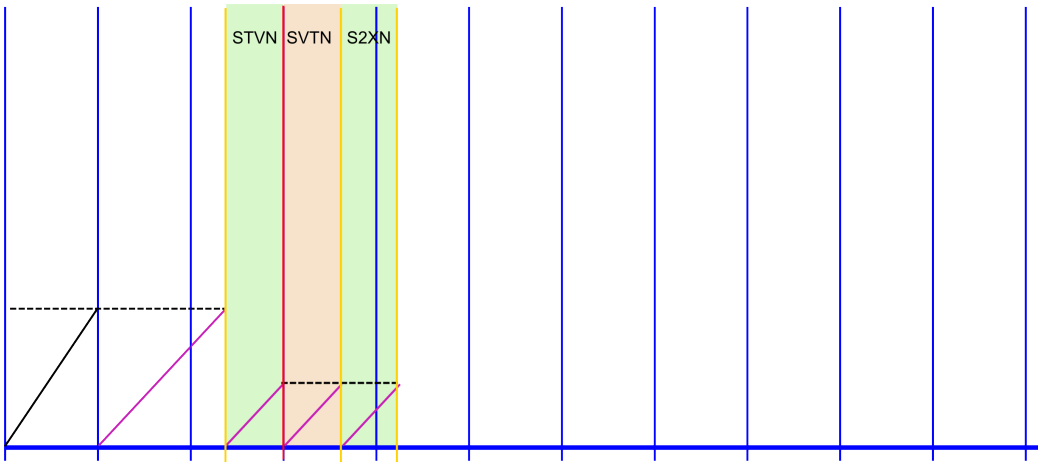

Figure 2: Algorithmic phase of the time-based ADC used to calculate the remaining bits of precision (bits 2 through n). For this bit, a temporal residual of less than half a clock cycle in the S_{TVN} state requires 1 clock edge to complete the S_{VTN} and S_{2XN} states. For the first bit, the inversion of the number of clock edges represents the bit value ($\overline{1} = 0$ in thie case). Note: black lines denote charging by I_{in}, purple lines denote charging by I_{ref}, red vertical lines denote state machine transitions on clock edges, and yellow vertical lines denote state machine transitions on compator pulses.

the inverted value of the bit for the first iteration. As each iteration inverts its residual, the mapping between 0 and 1 clock cycles and a binary value of 0 or 1 switches after each iteration. This process is continued until the desired precision of the ADC has been reached. Figure 3 demonsrates the dual-slope phase followed by several iterations of the algorithmic phase of operation.

A high-level block diagram presented in Yang's thesis is shown in Figure 4.[1] The analog switch network is responsible for handling the charging and discharging of the two capacitors used in the dual-slope and algorithmic phases of operation. The comparator outputs a pulse in response to one capacitor voltage exceeding the voltage of a reference capacitor voltage. A state machine controls the behavior of the state machine by sending control signals to the other ADC sub-circuits. Lastly, the output registers provide an interface for external circuitry

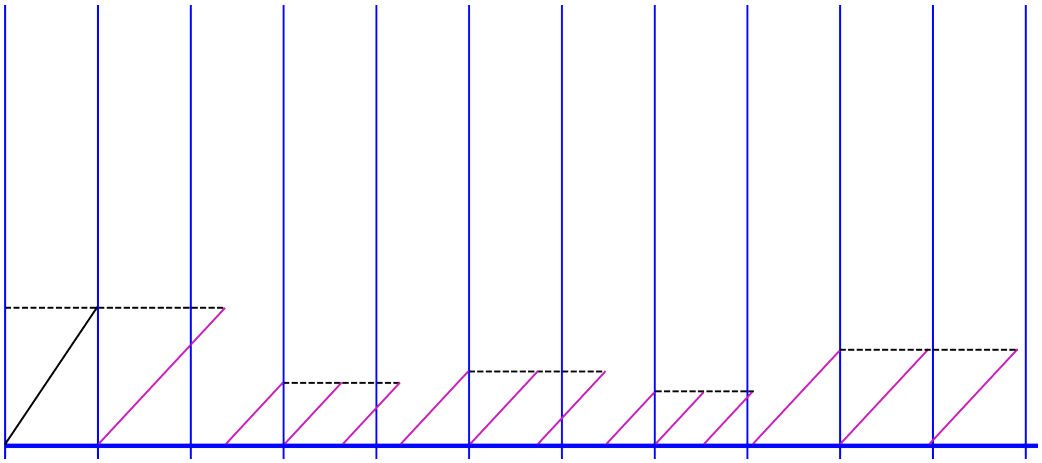

Figure 3: Algorithmic phase of the ADC after several iterations of the algorithm. The MSB stage encodes a binary value of 01 (see Figure 1). The LSB stage encodes a binary value of $\bar{1}1\bar{1}1 = 0101$, yielding a binary value of 010101 after 6 bits have been calculated.

to access the digitized value. The high-level block diagram presented in Figure 4 does not exactly match our implementation of this ADC, but provides a solid high-level description of its behavior.

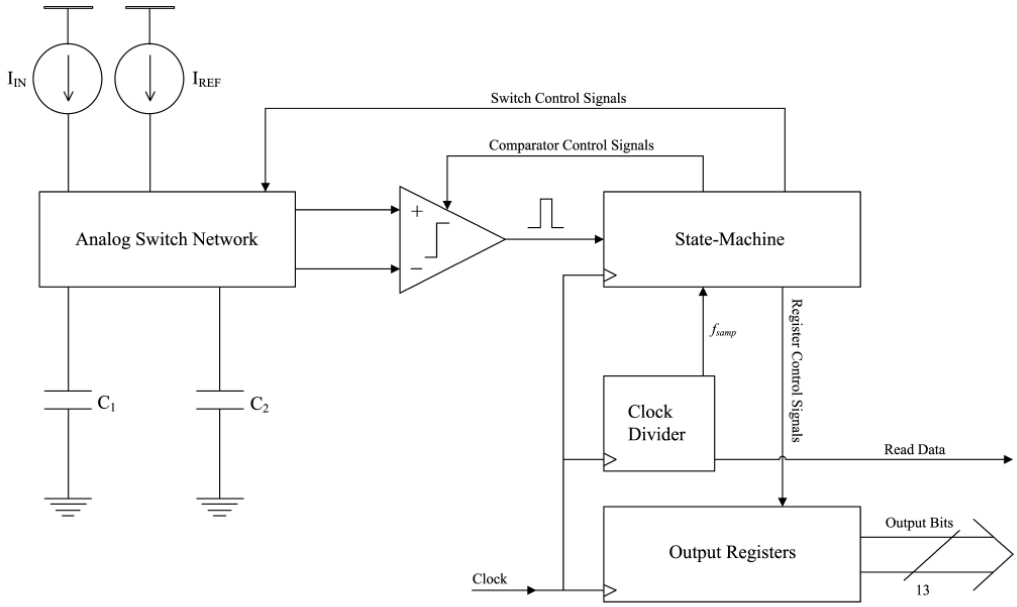

Figure 4: High-level block diagram of the ADC presented in [1].

3 Implementation

Though the thesis upon which our project is based has a wealth of information regarding the operation of the ADC, there are many implementation details that are left as an exercise to the reader to determine. In the following subsections, we present our implementation of the various components of the ADC and the related design decisions. All schematic capture (LTSpice), simulation (LTSpice), and layout (Magic 8.0) files for the design are available at `https://github.com/jkirschner/ultra-efficient-adc`. Through the process of implementing

the circuit, we gained a greater understanding of the many design decisions that are left unexplained in the thesis. We also gained a greater appreciation for the difficulty of implementing an elegant high-level design; in this case, the devil really is in the details (see Section 5).

3.1 Analog Switch Network

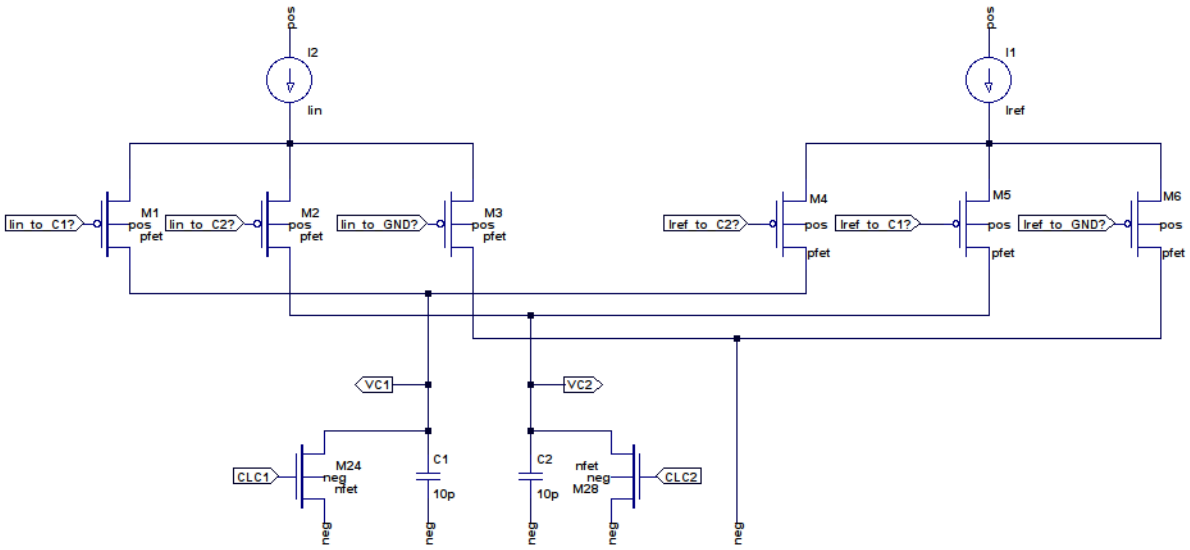

Figure 5: Analog switch network with three-way pMOS differential pair for current steering.

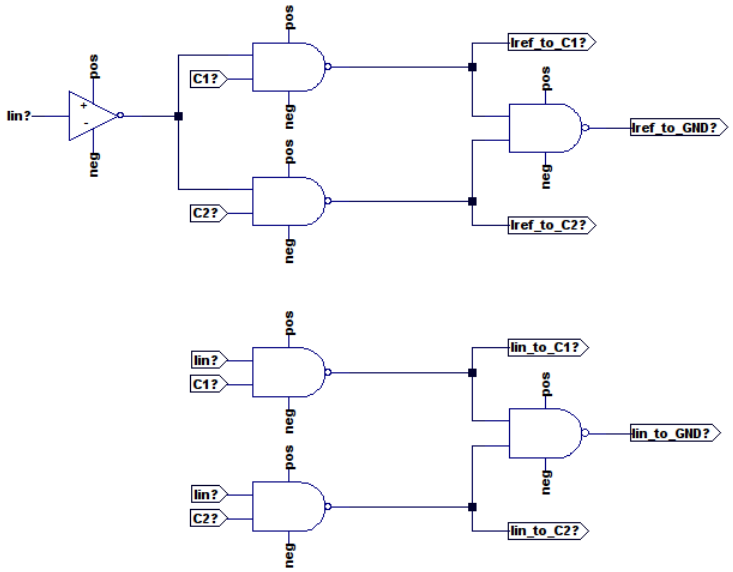

Figure 6: Logic used to generate the control signals for the analog switch network.

The analog switch network is responsible for managing the charging and discharging of the two matched capacitors, C_1 and C_2, which are the basis of the operation of the ADC. The final precision of the ADC will be limited by capacitor mismatch and capacitor voltage errors if not designed carefully. At any given time, the analog switch network steers the input current I_{in} and the reference current I_{ref} to one of the two capacitors or

shunts this current to ground. We chose to implement the current steering of each of the two current sources with a three-branch pMOS differential pair, as shown in Figure 5. The inputs to the gates of the differential pair are digital, and the state machine will guarantee that, at any given time, exactly one of the pMOS differential pair gate signals will be logic low. Thus, all of the bias current (I_{in} or I_{ref}) going into the differential pair is steered into exactly one of the branches. For each of the two current sources, the three branches of the differential pair are respectively connected to C_1, C_2, and ground. If a sourced current is not being steered either to C_1 or C_2, the current will be steered to ground as a result of the digital circuit shown in Figure 6. This solution was chosen instead of turning off the bias current sources when not actively charging a capacitor to avoid charge injection from switching the bias current source on and off. The source of a strong nMOS transistor is connected to the positive plate of each capacitor and can be used to reset the capacitor voltage. The capacitors C_1 and C_2 are shorted to ground when state machine control signals CLC1 and CLC2 respectively are high and the comparator output is high.

3.2 Comparator

The purpose of the comparator is to output a pulse in response to the voltage of a capacitor exceeding the voltage of a reference capacitor. The comparator consists of several components: a pre-amplifier, a gain/latch stage, and a pulse-width control circuit. A bias generator also generated a cascode bias voltage needed by the pre-amplifier.

3.2.1 Bias Generator

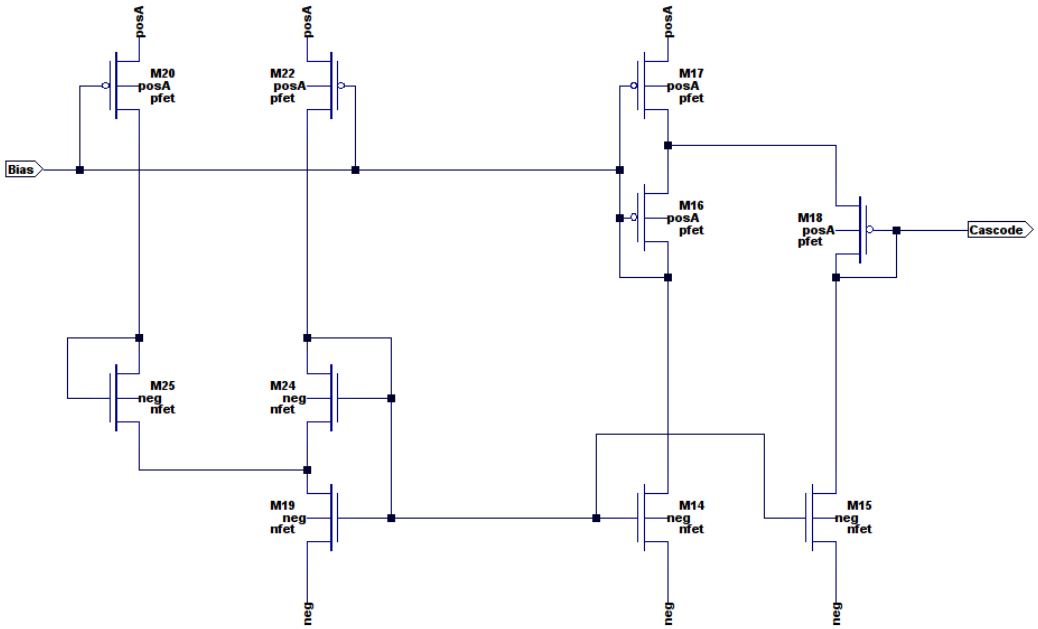

Figure 7: Cascode bias voltage generator for the comparator.

The bias generator presented in Figure 7 was adapted from a bias generator provided in Machine Problem 3 for Bradley Minch's Mixed Analog-Digital Very Large Scale Integration (MADVLSI) class at Olin College of Engineering. It produces a pMOS cascode bias voltage needed by the pre-amplifier stage.

3.2.2 Pre-Amplifier

The comparator pre-amplifier presented in Figure 8 is a simple differential pair with a cascoded pMOS bias current source. The V+ and V- outputs will be sent to the gain/latch stage in order to latch a logic high value when the input voltage exceeds the reference voltage.

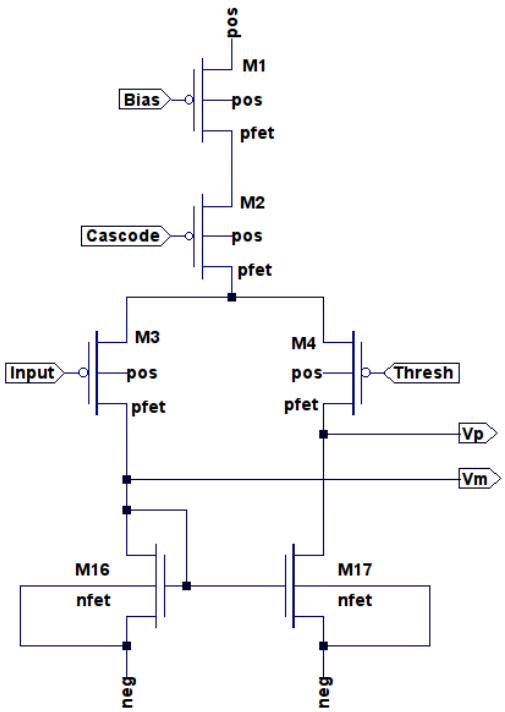

Figure 8: Comparator preamplifier.

3.2.3 Gain/Latch Stage

The gain/latch stage of the amplifier presented in Figure 9 is taken directly from pg. 59 of [1]. As V+ becomes greater than V-, the pMOS differential pair will cause the voltage of Node D_{out} to increase from ground. Once it reaches the threshold voltage of transistor M8, the pMOS current mirror formed by M1 and M5 causes D_{out} to increase with positive feedback, "latching" the output value high. D_{out} is passed to a pulse-width control circuit which will generate a pulse in response to a latched logic high value. The Disable input allows for the disabling of the current mirror formed by M1 and M5 and pulls the output to logic low via nMOS transistor M13, resetting the circuit for the next comparison. The Disable input will be used by the state machine to reset the gain/latch stage after a pulse has been emitted.

3.2.4 Pulse-Width Control Circuit

The pulse-width control circuit is used to output a pulse of fixed duration in response to a logic high input. The intended usage of the pulse-width control circuit within the context of the comparator is to output a pulse when the voltage of one capacitor surpasses the voltage of the second capacitor, triggering a state transition within the algorithmic stage of the ADC operation. The comparator pulse is also used in conjunction with signals from the state machine to ground one or both capacitors, a necessity for the ADC operation. The schematic for the pulse-width control circuit shown in Figure 10 is taken directly from Yang's thesis.[1]

The pulse-width circuit has a double-inverter buffer at its input stage. An intermediate node, labeled X, is connected to the buffered input through a transmission gate which conducts when the circuit's output is logic low. Node X is also connected to a current sink transistor gated by voltage V_{pw1} through a transmission gate which conducts when the circuit's output is logic high. The value of Node X goes through a double-inverter buffer to produce the pulse-width control circuit's output. Once the output goes logic high, the input is disconnected from Node X; instead, the current sink transistor gated by voltage V_{pw1} is now connected to Node X. Node X is kept logic high for some time by a positive feedback transistor. However, this positive feedback transistor is in series with another transistor gated by Node H which slowly charges up from ground to the power rail via a current source transistor gated by V_{pw2} when the output is logic high. As Node H approaches the rail, the positive feedback transistor on Node X becomes current starved, allowing the current sink transistor gated by

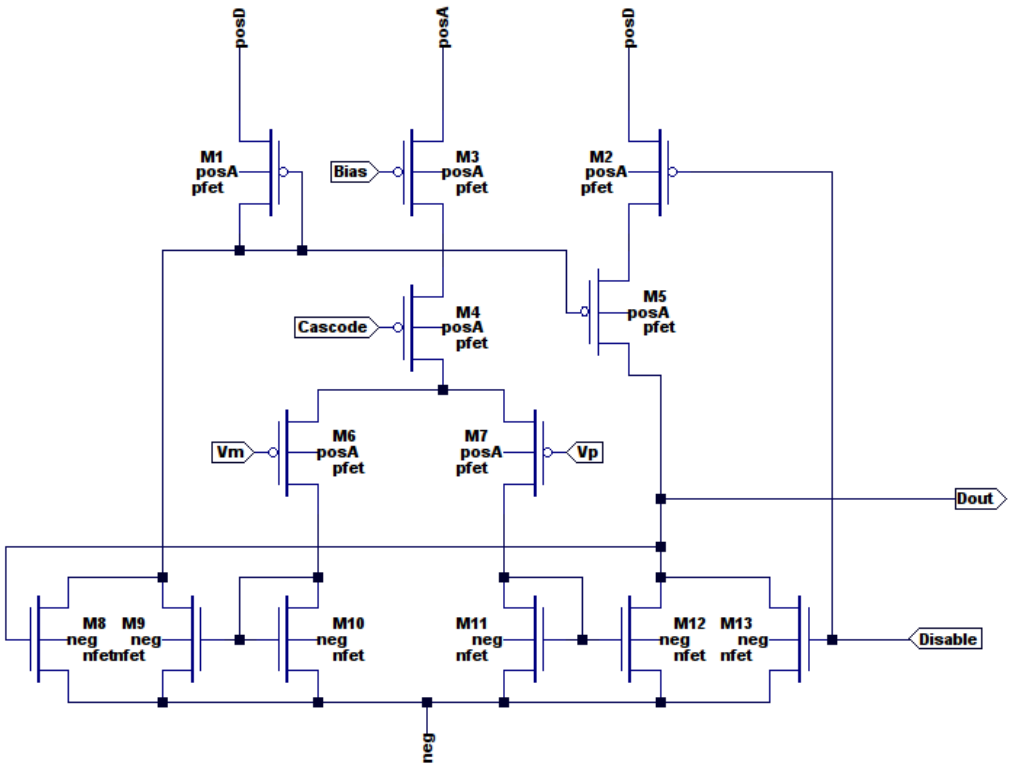

Figure 9: Gain-latch stage of the comparator.

V_{pw1} to pull Node X to logic low. The output of the pulse-width control circuit then becomes low, and the input signal is once again connected to Node X through a transmission gate. The duration of the pulse is set by the values of the bias voltages V_{pw1} and V_{pw2}; a sronger current sink or a stronger current source will lead to a shorter pulse, and vice-versa.

3.2.5 Comparator Integration

The pulse-width control circuit is normally thought to produce a pulse of fixed duration in response to a rising edge on its input. This occurs because the pulse-width control circuit is disconnected from its input while its output is high. However, once the pulse has been emitted, the pulse-width control circuit is no longer disconnected from its input. If the input value is still high, another pulse will be emitted. This behavior is undesirable; if more than one pulse is emitted from the pulse-width control circuit in response to the gain/latch stage output, the state machine will transition states prematurely in response to the extra pulse output. To prevent this behavior, a logic high value of the pulse output is used to disable the gain/latch stage, resetting its output to ground. Both the gain/latch stage and the pulse-width control circuit are disabled by the state machine when the present state is not waiting on a comparator output pulse before transitioning. As pulses from the comparator are used to reset capacitors C_1 and C_2 to ground, an unintended pulse from the comparator would interfere with the operation of the ADC; thus, the ability to disable the comparator makes the operation of the ADC more robust.

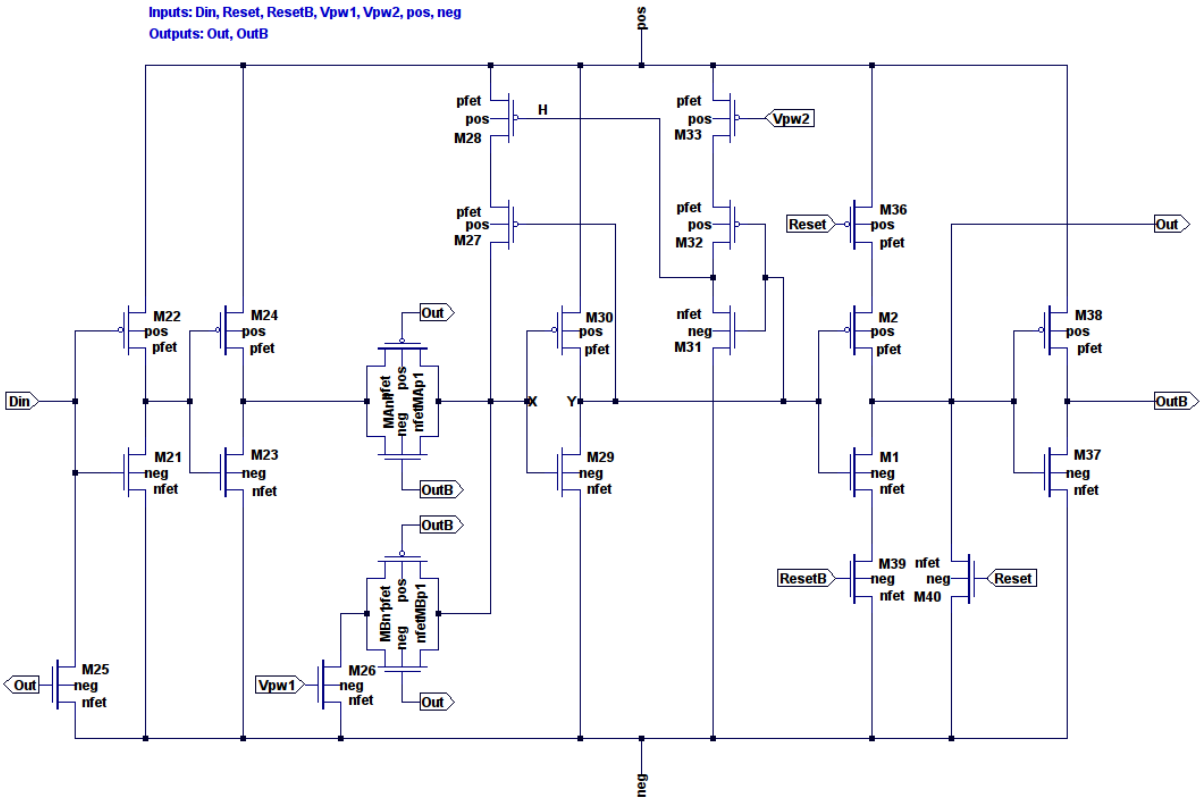

Figure 10: Comparator pulse-width control circuit.[1]

3.3 Intermediate Registers

The intermediate registers are separated into two parts: the most-significant bit (MSB) registers and the least-significant bit (LSB) registers, corresponding to the input, dual-slope and residual, algorithmic stages of the ADC operation respectively.

3.3.1 MSB Intermediate Registers

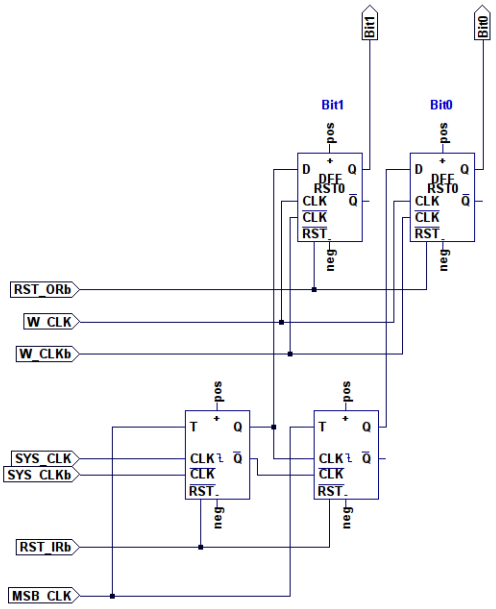

Figure 11: Most significant bit registers.

The dual-slope stage computes the first two most-significant bits (bits 0 and 1) of the digitized input signal by counting the number of falling clock edges which occur before the comparator signals that C_2 (charged by the reference current) has reached a higher voltage than C_1 (charged by the input current). As the input, dual-slope stage of the ADC is restricted to less than four clock edges, the number of falling clock edges can be tracked with a two-bit counter that is reset after the dual-slope stage has finished. This is implemented using a cascade of two Toggle Flip Flops (TFFs), as shown in Figure 11. The MSB_CLK control signal enables toggling on falling clock edges when logic high; the RST_IRb control signal resets the TFFs when logic low.

3.3.2 LSB Intermediate Registers

The least-significant bits (bits 2 through 11) are computed by a time-based algorithmic stage which operates on a temporal residual from the previous computed bit. Each algorithm iteration determines the value of its bit by determining whether the preceeding temporal residual was greater than or less than half the duration of a clock cycle and then produces a residual for the next algorithm iteration. The temporal residual is converted to a voltage by charging up capacitor C_1 with I_{ref} until the next clock edge (state S_{TV}). I_{ref} is then used to charge up C_2 to the voltage of C_1 (state S_{VT}); at this point, C_1 is discharged and then charged back up to the voltage of C_2 using I_{ref} (state S_{2X}). The value of the bit is determined by counting the number of clock cycles (0 or 1) which occur during the S_{VT} and S_{2X} stages. The next temporal residual is the time remaining between the end of the S_{2X} state and the next clock cycle. Because the S_{VT} state essentially inverts the temporal residual, the mapping between 0 or 1 clock cycles and a binary value inverts after each successive iteration. Even iterations (0-indexed) of the algorithm map 0 or 1 clock cycles to the binary values 1 and 0 respectively; odd iterations (0-indexed) of the algorithm map 0 or 1 clock cycles to the binary values 0 and 1 respectively.

The state diagram presented in Yang's thesis and shown in Figure 14 suggests an implementation in which different states (negative-index and positive-index) alternate iterations of the algorithm. Instead, one can apply

an odd number of inversions to the number of clock cycles counted on even algorithm iterations, resulting in a net inversion; conversely, one can apply an even number of inversions to the number of clock cycles counted on odd algorithm iterations, resulting in a net non-inversion. This is achieved in our implementation, shown in Figure 12, by passing the number of clock cycles counted at each algorithm iteration through a specially connected shift register; instead of passing the Q output of a DFF to the D input of the next DFF, the $\overline{Q}$ output is passed forward. After bit 11 has been computed, bits 2 through 11 will have been inverted the requisite number of times (even or odd) to produce the correct output for each bit.

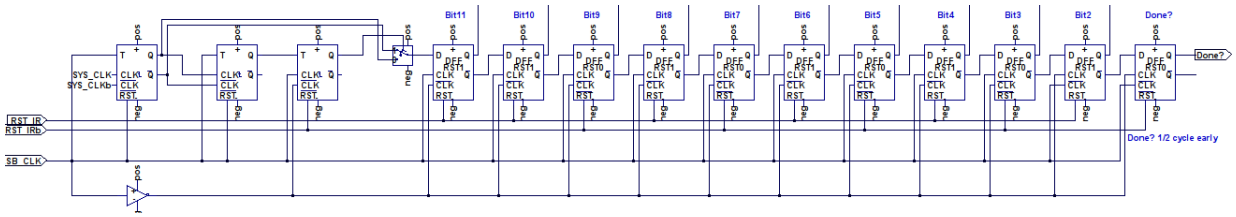

Figure 12: Least significant bit registers.

The implementation presented in Figure 12 has some several peculiarities which need further explanation.

Rather than a simple 1-bit counter to count 0 or 1 clock edges, this implementation uses a 2-bit counter of cascaded TFFs. The output of the MSB of this 2-bit counter is passed to a third TFF. The purpose of this segment of the LSB register circuit is to detect when an iteration occurs over more than 1 clock edge (i.e. 2 clock edges) due to non-idealities in the circuit behavior (e.g. comparator and state machine delays). In the event of 2 clock edges, we attempt to correct the error by giving the present clock transition count a value of 0 and inverting all successive clocks transition counts ($0 \rightarrow 1$, $1 \rightarrow 0$) (see pg. 49 of [1] for more detail). The occurrence of 2 clock edges during an iteration causes the value of the third TFF to toggle after the bit has been computed; as the third TFF output is a selector between Q and $\overline{Q}$ for the output of the LSB of the 2-bit clock edge counter, the inversion of all successive bits is achieved.

For the state machine to operate correctly, some element of the circuit needs to signal when the desired level of precision (12 bits) has been reached; as the dual-slope stage calculates the first 2 MSB bits, the algorithmic stage calculates the remaining 10 LSB bits. This could be implemented by a 4-bit counter with some output logic that determines whether or not the binary output corresponds to the decimal value of 10. However, the number of transistors required to produce the desired signal can be reduced by taking advantage of the existing shift register structure; an n bit shift register with a logic high value loaded in the first DFF will produce a logic high output at the n^{th} DFF. The 11^{th} DFF in the shift register is intended to produce a logic high value once 10 algorithm iterations have been completed and logic low in all other cases. To produce logic low in all other cases, the initial values of DFFs 2 through 11 must produce a logic low at the 11^{th} DFF. The initial values of DFFs 2 through 11 must then take into account the number of inversions which will occur in the binary value before reaching DFF 11. The same consideration applies to DFF 1, which must produce a binary value of 1 at the output of DFF 11 after the 10^{th} algorithm iteration. This output, DONE?, signals the state machine to proceed to the exit state of the state machine after finishing the present iteration. For this transition to occur reliably, the DONE? output must be logic high before the algorithm completes. To accomplish this, the last two DFFs in the shift register latch their values on the rising edge of the clock rather than the falling edge of the clock (as in all preceeding shift registers in the shift register).

3.4 Output Registers

The output registers are implemented as D Flip Flops with inputs coming from the outputs of the MSB and LSB intermediate registers. Data is latched into the output registers on the falling edge of the W_CLK control signal from the state machine (occurring at the transition from the exit to the reset stage). This allows for the computed digital conversion to be stored and made available for output while the intermediate registers are being filled with the next conversion.

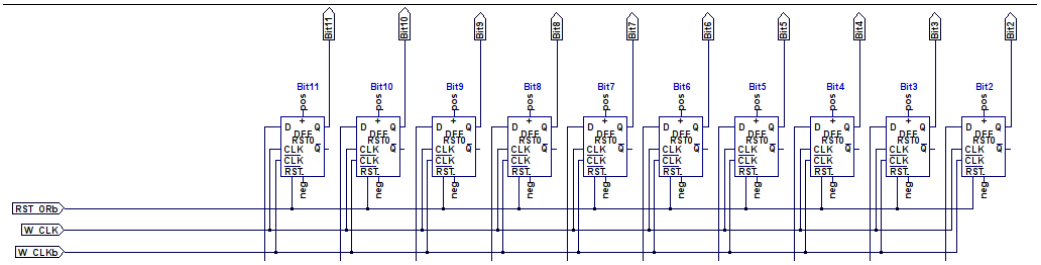

Figure 13: Output register file. The values stored in each of the LSB registers are stored in a flip flop corresponding to each output bit.

3.5 State Machine

The majority of the system control signals required for proper functioning of the Analog Switch Network, Comparator, Intermediate Registers, and Output Registers are generated by the State Machine. In general, a state machine has an internal state and a set of rules for each possible state that dictates which sets of inputs will result in a transition to a new state. Thus, a state machine allows for a implementation of a process which can be outlined in a state diagram, such as the one presented by Heemin Yang in Figure 14.

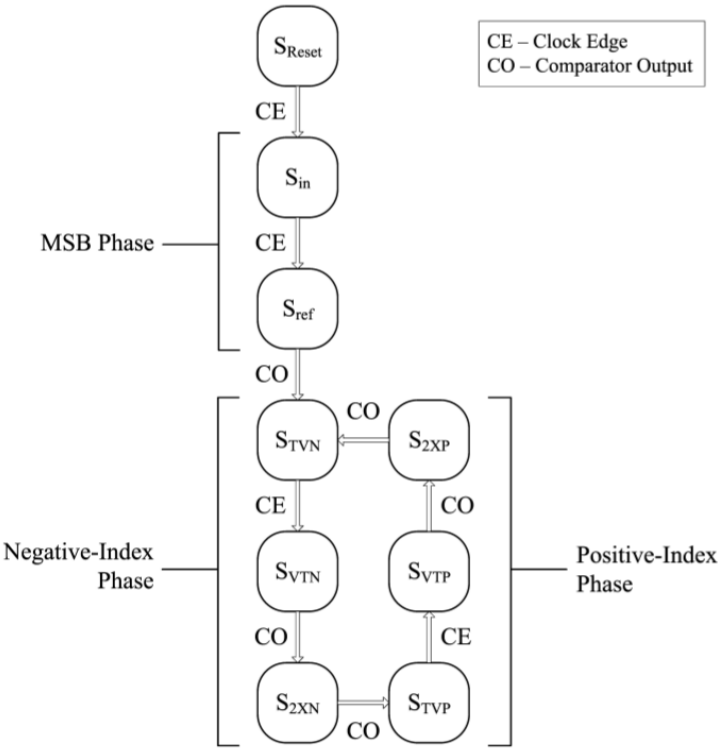

Figure 14: State machine presented in Heemin Yang's thesis.

The ADC propsed by Heemin Yang has a fairly simple state diagram. The transition between states is mostly linear. The only branch in the state machine is in determining whether another bit should be quantized or the desired amount of precision (in this case, 10 bits), has been reached.

As a result of our particular implementation of the LSB output registers (see Section 3.4, there is no need for a distinction between positive-index and negative-index stages, as shown in Figure 15. This modification reduced the complexity of the state machine and the number of required transistors, possibly leading to area and power savings.

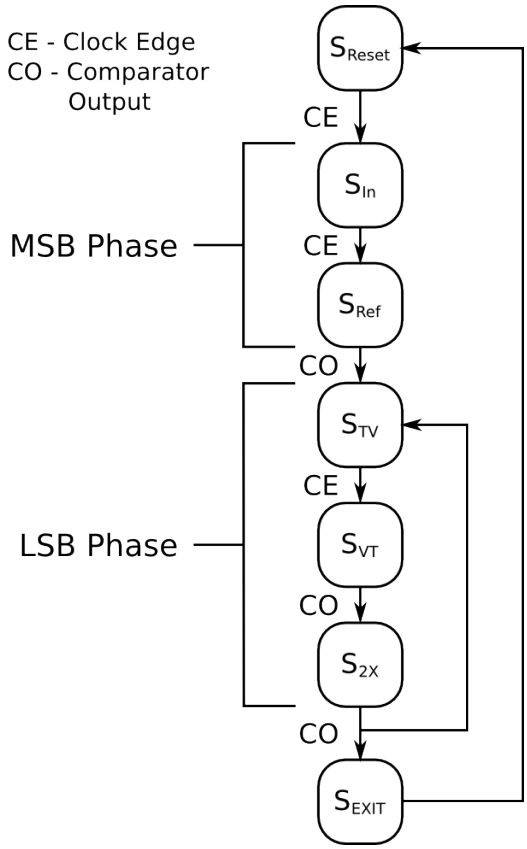

Figure 15: State machine implemented for our ADC.

To implement the state diagram presented in Figure 15, we need a state-holding element (e.g. a static D Flip Flop) and some logic (e.g. logic gates). The state of a state machine can be represented in many different ways—called assignments. One common method—binary assignment—is to assign states a binary value, requiring $\lceil \log_2 n \rceil$ state-holding elements to represent n states. Another common method—one-hot assignment—is to assign each state a single bit in the state string, requiring n state-holding elements to represent n states; the active state is the one represented by a single state-holding element with a logic high output (hence, 'one-hot'). Binary assignment reduces the number of D Flip Flops needed to represent the state machine, but the task of assigning states such that combining the D Flip Flops outputs to result in the correct state machine control signals is very difficult. One-hot assignment requires more D Flip Flops to represent the state machine, but the process of calculating control signals is straightforward and simple—simply use logical combinations of active states (e.g. output a logic high when *State 1* OR *State 2* is logic high). While it is easy to think of generating control signals from a disjunctive normal form (DNF) statement (e.g. a logical OR of a combination of signals), the circuit can be more efficiently implemented with a conjunctive normal form (CNF) statement (e.g. a logical NAND of a combination of signals). By DeMorgan's Law, the statement $a \vee b$ can alternatively be expressed as $\overline{(\overline{a} \wedge \overline{b})}$. By implementing D Flip Flops with a Q and $\overline{Q}$ output, the inverted state outputs can be combined in a NAND gate to produce the same result as a combination of non-inverted state outputs through an OR gate.

One important consideration when designing a state machine is whether the resulting state is always valid. As the initial state of the D Flip Flops is unknown, the initial state representation on power-on may not be valid (e.g. if no D Flip Flop or more than one D Flip Flop has a logic high output). Invalid states may result in damage to the circuit if the wrong control signals interact and/or may prevent the circuit from ever operating as intended (e.g. never having a valid state). Even in the case that only one D Flip Flop has a logic high output on power-on, there is only one desired initial state: the reset state. Thus, there must be a mechanism to reset the state to the desired, valid state on power-on. If the D Flip Flop which represents the reset state resets to high and all other states reset to low upon the application of an external signal (whether from a power-on reset

internal circuit or an external source), the initial state of the state machine can be set. (In our implementation, we will rely on an external signal for the reset due.)

4 Circuit Performance

Simulation of designed schematics was performed using Linear Technology's LTSpice simulator. Figure 16 shows the capacitor voltages V_{C_1} and V_{C_2} through one full sample in response to an input current of twice the reference current. In theory, the ADC should have an output of 100000000000 for this input; instead, the output is 101100100111. The simulation shows that capacitor voltages drop slightly in response to current switching away from a capacitor within the analog switch network. The effects of comparator and state machine delays can be seen in the difference between the capacitor voltages at which a state machine transition occurs.

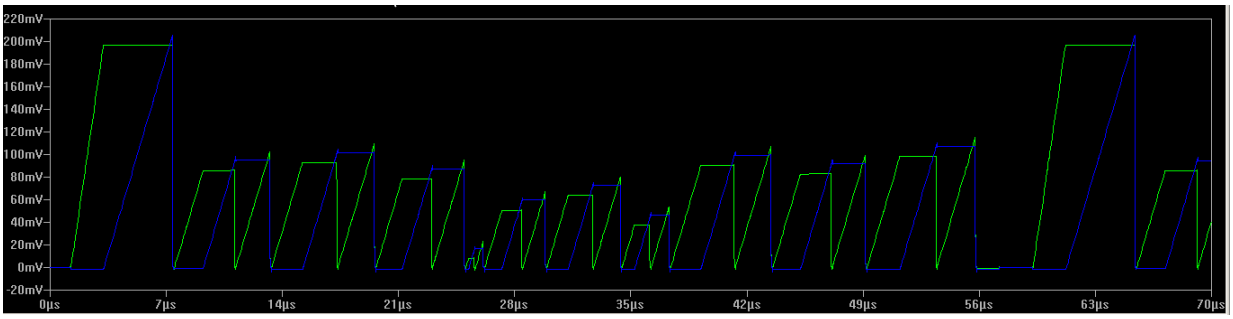

Figure 16: Capacitor voltages V_{C_1} (green) and V_{C_2} (blue) through one full sample in response to an input current of twice the reference current. In theory, the ADC should have an output of 100000000000 for this input; instead, the output is 101100100111.

The behavior of the capacitor voltages indicate that the state machine, comparator, and analog switch network are behaving as expected. The output registers (MSB and LSB) were also confirmed to be working as expected. Unfortunately, the error-cancellation and error-correction techniques employed within our implementation are not sufficient to produce a semblance of a correct output. Section 5 discusses the sources of some of these errors in more detail.

Figure 17 shows the circuit behavior in more detail. As the input current is twice the reference current, we would expect the comparator to trigger exactly 2 clock cycles after C_1 is charged in the dual-slope phase. Indeed, $V_{C_1} = V_{C_2}$ after 2 clock cycles, but the comparator rising edge does not occur until ≈ 180 ns later. The first iteration of the algorithmic phase also takes more than 2 clock cycles to complete the S_{VT} and S_{2X} states despite having an initial residual reasonably distant from the start of the last clock cycle in the S_{TV} state. As the LSB register sub-circuit assumes this error is caused by a different (also frequent) phenomenon, the error is not corrected (see Section 3.3.2 for more details). This same error occurs in the second iteration, leading these two bits to be incorrect (11 instead of 00).

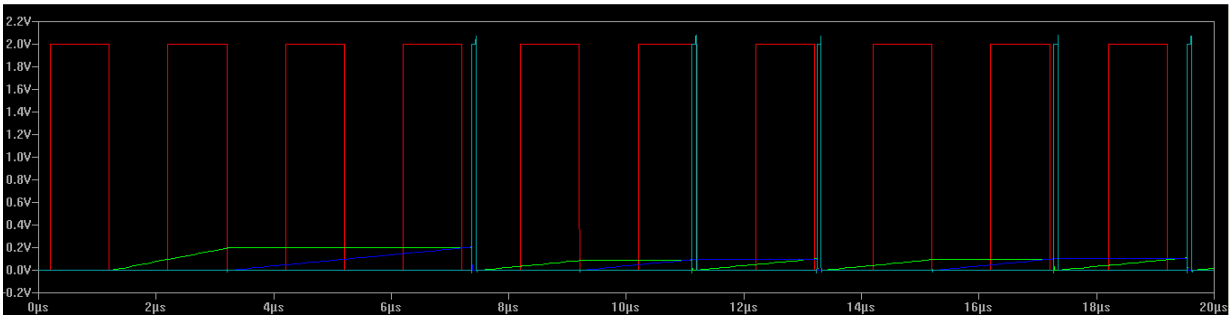

Figure 17: Capacitor voltages V_{C_1} (green) and V_{C_2} (blue), system clock (red), and comparator pulse output (cyan).

5 Areas for improvement

As seen in Section 4, though the algorithm proceeds correctly and all of the individual components are working, the circuit does not accurately digitize analog signals. In most cases, the accuracy of our implementation is limited by the precision of the dual-slope stage (2 MSBs). The causes of the inaccuracies are varied, but ultimately result in temporal delays and inconsistencies in capacitor voltages. We attempted to utilize the various error correction schemes proposed in pg. 41-50 of [1]. Unfortunately, our various attempts at error correction did not yield improved ADC accuracy.

The simplest example of an inaccuracy induced by a temporal delay is in the dual-slope stage. The MSB register circuit described in Section 3.3.1 counts the number of clock transitions which occur before the capacitor charged by the reference current reaches the voltage of the capacitor charged by the input current for one clock cycle. Counting is enabled by the control signal MSB_CLK determined by the state machine. Thus, counting of clock edges only stops after the comparator fires a pulse, the state machine receives the pulse and transitions states, and the updated MSB_CLK signal propagates to the MSB register circuit. In cases where the input current is at a level such that the 2 MSBs of the digitized signal are close to transitioning to a higher value (e.g. $01 \rightarrow 10$), this delay will cause the 2-bit counter to count an extra clock edge.

A similar inaccuracy results from the algorithmic stage when the S_{VT} and S_{2X} stages complete near a clock edge. A delay may cause a 0 clock edge residual to be registered as a 1 clock edge residual; such delays also enable the otherwise impossible possibility of a 1 clock edge residual being registered as a 2 clock edge residual. In the case of a 2 clock edge residual, our implementation assumes that this error results from a S_{TV} stage which is supposed to start near the next falling clock edge, but instead starts soon after the falling clock edge due to a delay. This will result in a 2 clock edge residual which should have been a 0 clock edge residual. As our implementation described in Section 3.3.2 uses a 2-bit counter and the MSB is passed to the shift registers, 2 clock edges (binary: 01) results in a correct value of 0 being passed to the shift registers. The remaining computed bits are then inverted to account for the effective extra inversion of the residual caused by the delay. Unfortunately, we have not found a way to determine whether a 2 clock edge residual results from what should have been a 0 clock edge residual (as described previously) or from a 1 clock edge residual which carried over an extra clock cycle due to delays. The previously described error correction scheme will result in an incorrect bit.

Non-ideal behavior also results from incredibly small residual voltages, as described in pg. 49-50 of [1]. Yang proposes a $1 + \epsilon$ stage after the S_{TV} stage in order to ensure a minimal capacitor voltage. Though this stage would not have been very difficult to implement in our design, we instead focused our time on attempting to resolve the largest sources of error: the temporal delays.

Another inaccuracy results from capacitor voltage drops upon bias current being steered away from the capacitor to a different branch within the analog switch network (see Section 3.1). Though pg. 44-47 suggest that the algorithm can effectively cancel out this switching charge-injection induced error, simulation results of our particular implementation showed otherwise. The amount of charge injection was not consistent across initial capacitor voltages (especially for low capacitor voltages).

In summary, there are many sources of error within the present implementation that lead to the incorrect digitization of input analog current signals. Though [1] outlines the general idea behind many of the possible error-cancellation and error-correction techniques, details of implementation are not given. Our attempts to date at implementing error-cancellation and error-correction techniques have not yet yielded an improvement in performance; more time and analysis would be necessary to produce an ADC with useful levels of precision.

6 Integrated Circuit Layout

After performing schematic capture and simulation in LTSpice, we performed integrated circuit layout in Magic 8.0 for a 0.5 μm CMOS process from ON-Semiconductor through MOSIS. The pinout of the resulting integrated circuit is shown in Table 6. The final pinouts are also displayed graphically in Figure 18. The layout of our final circuit in Magic 8.0 within the padframe is shown in Figure 19.

Pin Number	Node	Pin type	Description
1	V+	Inorpad (output)	Positive output voltage from comparator pre-amp stage, V_{ddA} positive supply, 3.2
2	SHUNT_I_{in}	IOpad, out always enabled	Shunt I_{in} to ground
3	SHUNT_I_{ref}	IOpad, out always enabled	Shunt I_{ref} to ground
5	V_{bias}	Given voltage bias	Bias voltage for comparator circuit, 3.2
8	V_{PWC}	IOpad, out always enabled	Pulse width control output voltage
9	V_{GL}	IOpad, out always enabled	Buffered output of comparator gain/latch stage, 3.2
10	V_{pw1}	Inpad	Sets width of pulse, 3.2.4
11	V_{pw2}	Inpad	Sets width of pulse, 3.2.4
12	Reg2	IOpad, out always enabled	The following store incrementally less significant bits, 3.3.2
13	Reg3	IOpad, out always enabled	
14	Reg4	IOpad, out always enabled	
15	Gnd	Given ground pin	Ground
16	Reg5	IOpad, out always enabled	
17	Reg6	IOpad, out always enabled	
18	Reg7	IOpad, out always enabled	
19	Reg8	IOpad, out always enabled	
20	Reg9	IOpad, out always enabled	
21	Reg10	IOpad, out always enabled	
22	Reg11	IOpad, out always enabled	
23	Reg1	IOpad, out always enabled	Stores second most significant bit, 3.3.1
24	Reg0	IOpad, out always enabled	Stores most significant bit, 3.3.1
25	V_{ddA}	Given pad V_{dd}	Analog supply voltage, 3.2
26	RSTb	Inpad	Resets flip flops to either high or low depending on the type of flip flop
27	Clk	Inpad	Regulates charging up of capacitors
28	S_{exit}	IOpad, out always enabled	Q output of S_{exit} state in state machine, 3.5
29	S_{2x}	IOpad, out always enabled	Q output of S_{2x} state in state machine, 3.5
30	S_{tv}	IOpad, out always enabled	Q output of S_{tv} state in state machine, 3.5
31	S_{tv}	IOpad, out always enabled	Q output of S_{tv} state in state machine, 3.5
32	S_{ref}	IOpad, out always enabled	Q output of S_{ref} state in state machine, 3.5
33	S_{in}	IOpad, out always enabled	Q output of S_{in} state in state machine, 3.5
34	S_{reset}	IOpad, out always enabled	Q output of the S_{reset} state in the state machcine, 3.5
35	V_{ddD}	Given core frame V_{dd}	Digital supply voltage, 3.2
36	CLC1	IOpad, out always enabled	Short C_1 to ground?
37	CLC2	IOpad, out always enabled	Short C_2 to ground?
38	I_{in}	Inorpad (input)	Input current to be digitized
39	I_{ref}	Inorpad (input)	Reference current for ADC operation
40	V-	Inorpad (output)	Negative output voltage from comparator pre-amp stage, V_{ddA} positive supply, 3.2

Table 1: Pinout for the implemented ADC integrated circuit in a 40-poin dual in-line package. Unless otherwise mentioned, the positive supply voltage is V_{ddD}.

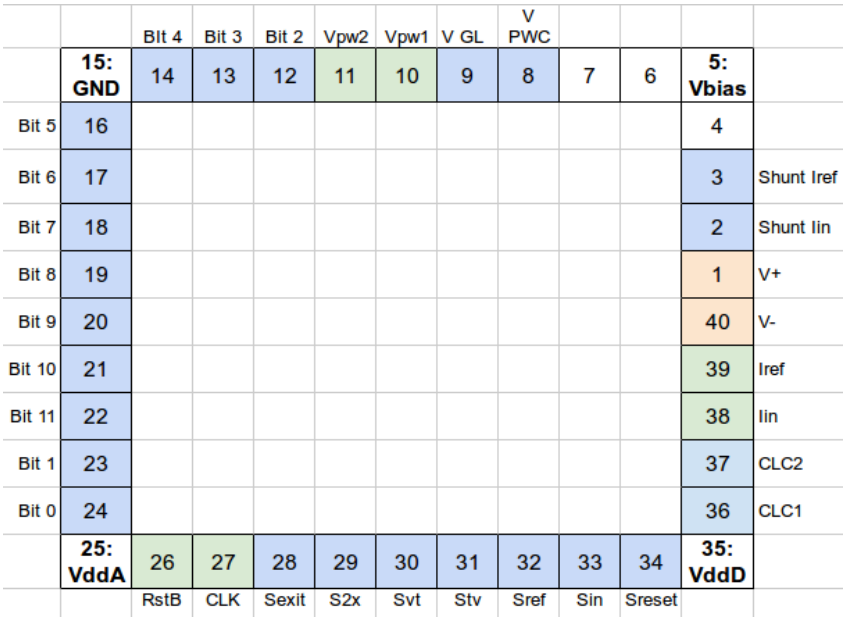

Figure 18: Overview of pinouts and pad types in final design.

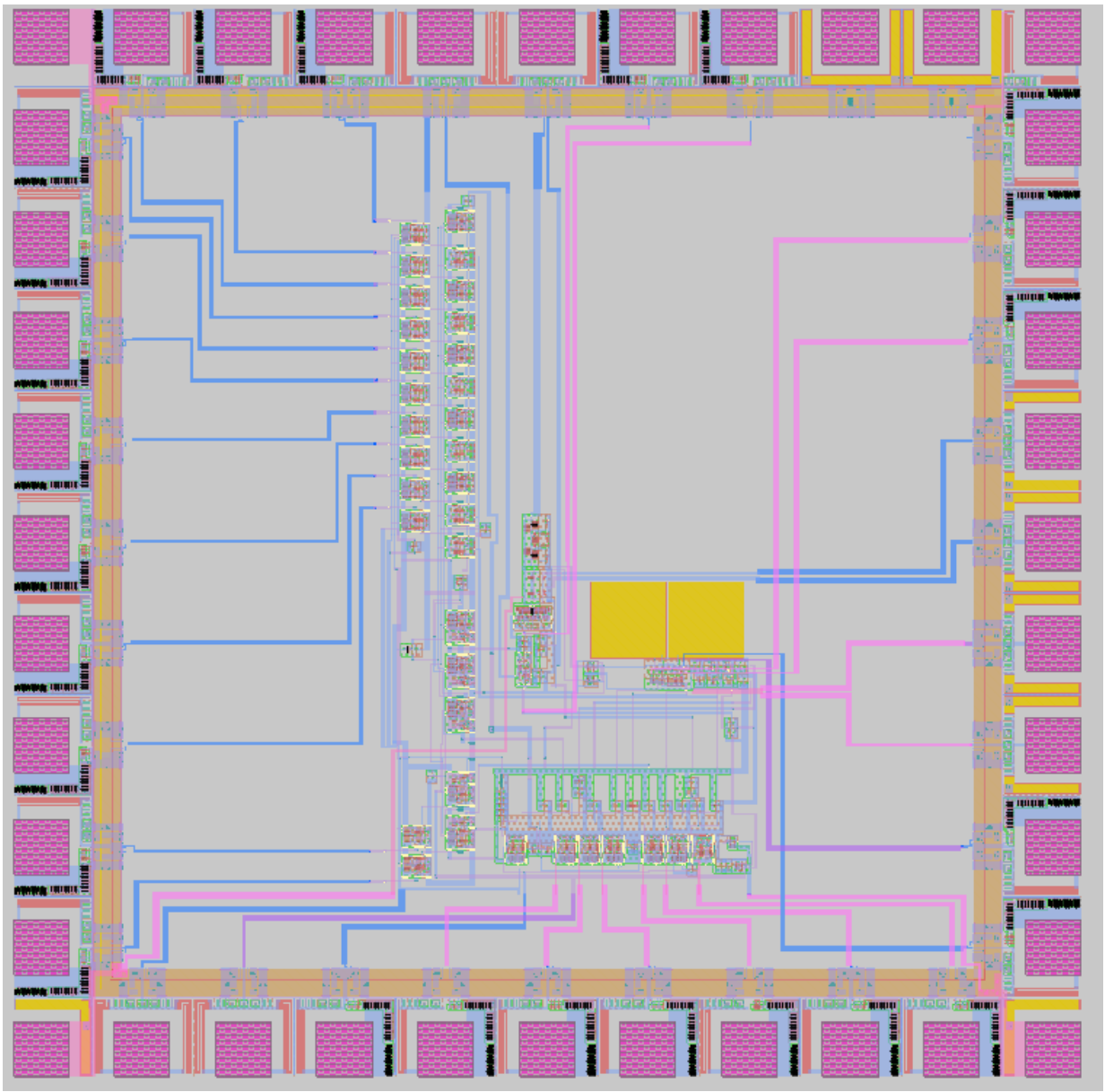

Figure 19: Layout of our circuit in Magic within its padframe.

7 Testing Protocol

Despite the limited usefulness of the output of the implemented ADC, there are still many interesting elements of the circuit's operation which can be tested. The circuit can operate with a wide range of inputs. However, it is important to consider the interplay between clock rate and reference current; if the reference current increases, the internal capacitors will reach the positive supply voltage more quickly, necessitating an increasing clock rate. The duration of the pulses from the comparator should also be shortened as clock rate increases to prevent excessive delays (though, this will not affect performance if the ADC is not accurate to begin with). Please use the following input signals for testing:

- RSTb: **Must be grounded on power-on!** When logic low, this input resets the state machine to a known, valid state. If this input is not logic low upon power-on, the state machine may start in an invalid state which could damage the circuit. Sometime after power on ($> 1\mu$s), this input must be pulled logic high in order for the ADC to operate.
- V_{ddA}: 3.0 V
- V_{ddD}: 2.0 V
- V_{pw1}: 0.7 V
- V_{pw2}: 0.5 V
- V_{bias}: 1.15 V
- Clk: V_{ddD} to ground, 500 kHz square wave, 50% duty cycle
- I_{ref}: 500 nA
- I_{in}: $0 \leq I_{in}$ $4{\cdot}I_{ref}$

These signals have been tested in simulation; please do not attempt to use different signals for operation of the ADC without performing the associated simulation first. Note that all digital logic signals (input and output) use V_{ddD} for the positive supply voltage.

First, before applying power to the circuit, make sure that RSTb will be logic low (ground) upon power-on. RSTb will reset the ADC into a known state before proceeding. Confirm that only the S_{reset} state is active—it should be the only state machine output with a logic high value. The digital output registers should also all have a value of logic low when RSTb is logic low. If these tests pass, RSTb can be set to logic high, allowing the ADC to begin operating. Once the ADC is operating, test whether the state machine is performing as expected (refer to Figure 15 and Section 3.5).

If the state machine is not performing as expected, check the comparator output V_{PWC}. Check the comparator output V_{PWC}—it should output 40 ns pulses within 4 clock cycles (8 μs) during the dual-slope phase (S_{ref}) and within 2 clock cycles (8 μs) during iterations of the algorithmic phase (S_{vt} and S_{2X}). If V_{PWC} does not exhibit this behavior, check the intermediate comparator signal V_{GL}—the buffered output of the gain/latch stage and the input to the pulse-width control circuit. If V_{GL} transitions to logic high, the pulse-width control circuit should output a pulse if it is working correctly. If V_{GL} is not observed to transition to logic high, check the analog inputs to the comparator gain/latch stage (V+ and V-). V+ should increase past V- in both the dual-slope and algorithmic phases of the ADC operation; when this occurs, V_{GL} should transition to logic high if the gain/latch stage is working correctly. V+ and V- should roughly follow the trends of V_{C_1} and V_{C_2} (e.g. when C_2 is charging, V+ should increase when C_1 is the reference capacitor, as in the S_{ref} and S_{vt} phases). If none of the elements of the comparator appear to be working, it is possible that the analog switch network is not operating correctly.

First, check to make sure that capacitors C_1 and C_2 are not being constantly shorted by checking whether pins CLC1 and CLC2 are always logic high. Pins CLC1 and CLC2 should only be logic high in conjunction with a comparator pulse; therefore, these pins should not be logic high if the comparator is not emitting pulses. Next, check whether the input and reference currents are being shunted to ground by checking whether pins SHUNT_I_{in} and SHUNT_I_{ref} are logic high. At least one of the pins SHUNT_I_{in} and SHUNT_I_{ref} should be logic high at all times. If both are logic high, or one is always logic high, this is indicative of incorrect behavior either caused by the state machine control signal logic or the analog switch network logic.

References

[1] Heemin Yang. "Time Based Energy Efficient ADC." Ph.D. Thesis, Massachusetts Institute of Technology. Feb. 2006.

[2] H. Y. Yang and R. Sarpeshkar. "A Bio-Inspired Ultra-Energy-Efficient Analog-to-Digital Converter for Biomedical Applications." *IEEE Transactions on Circuits and Systems-I; Regular Papers*, vol. 53, no. 11, pp. 2349-2356, Nov. 2006.

[3] H. Y. Yang and R. Sarpeshkar. "A Time-Based Energy-Efficient Analog-to-Digital Converter." *IEEE Journal of Solid-state Circuits*, vol. 40, no. 8, pp. 1590-1601, 2005.