A Feed Forward Circuit for Jitter Attenuation on High-Speed Digital Signals

By

Eric Rule

Advised By

Dr. Tina Smilkstein

SENIOR PROJECT

ELECTRICAL ENGINEERING DEPARTMENT

California Polytechnic State University

San Luis Obispo

2015

[This page was intentionally left blank]

TABLE OF CONTENTS

Section *Page*

Appendices

LIST OF TABLES AND FIGURES

Tables *Page*

Figures *Page*

Abstract

In the age of high-speed digital circuitry, there exists a need for clean, precise clock signals. In generating and distributing clock signals throughout a circuit, unwanted jitter can become a serious issue. A common technique for attenuating jitter uses phase-locked-loops to treat the signal, but as the clock frequency increases, so does the cost and complexity of the designs. Following the research completed by Dr. Tina Smilkstein [1], this project examines a purely feed-forward technique for attenuating jitter that is low-complexity and robust, and aims to design an integrated circuit that implements the technique.

I. Introduction

The need for extremely accurate clock signals continues to grow. One issue that can occur when generating and distributing a clock signal throughout a chip is jitter [3] (variation in the periodicity of a signal), and engineers are constantly seeking ways to suppress it. This report proposes and explores an interesting method for removing jitter on clock signals. Initial research only discovered one company that seemed to have commercially available integrated circuits with jitter-removal functionality. Silicon Labs has a product family called "Jitter Attenuating Clocks" [2] that utilize a DPLL technique to feed a clock signal through and remove jitter in the process. They advertise 100 fs RMS jitter on the output, at an output frequency range of 100 to 712.5 MHz Results also came up for small papers describing circuits for jitter removal. The only exhaustive resource found was a paper by Tina Harriet Smilkstein titled "Jitter Reduction on High-Speed Clock Signals" (on UC Berkeley's publications website). The goal of the project is to use the methods described in Dr. Smilkstein's paper to develop a simple, relatively inexpensive circuit that can clean jitter on high-speed clock signals. The general theory is to generate a finite-width pulse at every clock edge, integrate that signal (removing the jitter), and recreating the cleaned clock signal for output.

The methods for jitter removal as described in Dr. Smilkstein's paper have not yet been explored in the real world, and aim to solve a very important issue in modern circuit design. Ideally, this circuit is able to remove 100% of jitter on any digital signal [1]. The theory for this claim is explored in a later section. The simplicity of the theory makes this circuit very attractive as a viable method for cleaning jitter on high-speed digital signals. Of course, when designing a circuit for real-world use, non-idealities come into play. Because of the non-idealities of the circuitry, certain design choices will be taken into consideration and analyzed during implementation. These constraints will be detailed and if possible, resolved when implementing the designs.

Since this project involves designing the circuit using the proprietary IC design suite provided by Cadence [4], some information cannot be disclosed. Integrated circuit design adds a host of possibilities to a circuit design, but also poses new challenges. Some of the design choices will reflect these challenges.

II. Requirements and Specifications

The targeted customer for this design is both commercial and industrial. Circuit is available to be used in hobbyist projects and complex systems to reduce jitter on high-speed digital signals. Some engineering specifications arise from analysis of possible customer needs. The circuit should be able to be used in a relatively wide range of applications, so it should consume a small amount of power, operate on a range of frequencies, and utilize common logic levels. Furthermore, the circuit should be available in as an integrated circuit that may be used in a package such as DIP. Each marketing requirement translates into an engineering specification that meets the IEEE 1233 standards, as described in Table 1.

TABLE 1

REQUIREMENTS AND SPECIFICATIONS FOR JITTER ATTENUATION CIRCUIT

Marketing Requirements	Engineering Specifications	Justification
1	Circuit complies with design rules of CMHV7SF 180nm process.	Passing of various software checks such as DRC and LVS prove circuit complies with process rules.
2	Circuit attenuates jitter on input signal by at least 20%.	Circuit output attenuates acceptable input jitter range by at least 20%.
3	Circuit consumes no more than 500uW.	Across three blocks, the circuit should not consume more than 500uW to ensure customer can use in wide range of applications.
4	Circuit uses 0V to 1.8V logic levels.	Using common logic levels allows circuit to be easily interfaced with by existing circuitry.
5	Circuit should operate at a minimum frequency of 500MHz.	The circuit can operate on a range of high frequencies to ensure interoperability with modern digital circuits.
Marketing Requirements 1. Circuit is available in integrated form. 2. Circuit reduces jitter on input signals. 3. Circuit consumes small amount of power, fit for low-power applications. 4. Circuit easily interfaces to existing hardware. 5. Circuit operates at high frequencies.		

III. Theory

The central theory from this circuit comes from the thesis work of Dr. Tina Smilkstein. The paper written for the thesis describes a simple method that theoretically removes 100% of the jitter on any digital signal; it is broken down and summarized here. Refer to the following equation:

$$tp \times m1 - (T - tp) \times m2 = 0 \; ... \; [1]$$

A digital signal with period *T* can be fed through a pulse generator that produces pulses at every falling or rising edge with a constant pulse width. That is, even with jitter present on the input signal (positive or negative), the generator will always produce a pulse of the same width *tp*. Once there is a pulse train (still containing the jitter), a ramping generator creates a rising slope *m1* for time *tp* when the input pulse is high and a falling slope *m2* for time *tp* when it is low. If a comparator creates a high pulse when the ramping function reaches a certain level, it will output a pulse train of the original frequency, completely free of jitter. The picture below illustrates this fact.

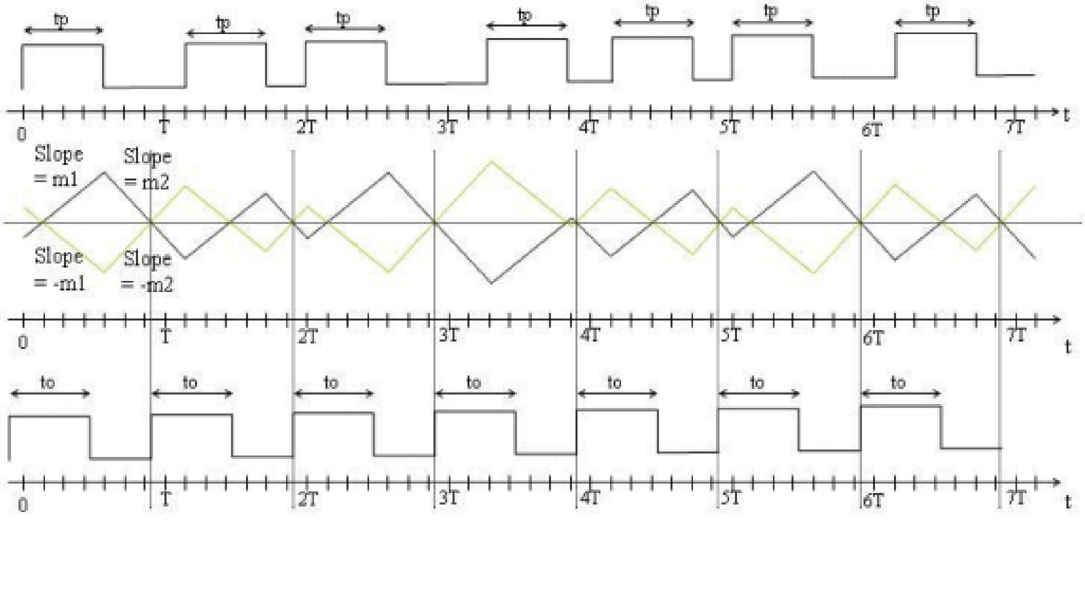

Figure 1: Ideal JAC Theory Illustrated

With the system utilizing differential signals, the ramping waveform will cross its inverse on the falling or rising edge periodically, with the period being the same as described in the equation above, *T*.

This method is not without constraints: the jitter on the input signal must be within a certain threshold, or the ramping signals may never cross, and the output block will never trigger. Furthermore, the system is incredibly sensitive to errors that could spread the ramping signals out too far, for they might not return to an acceptable level, requiring resetting of the system; this situation is illustrated below.

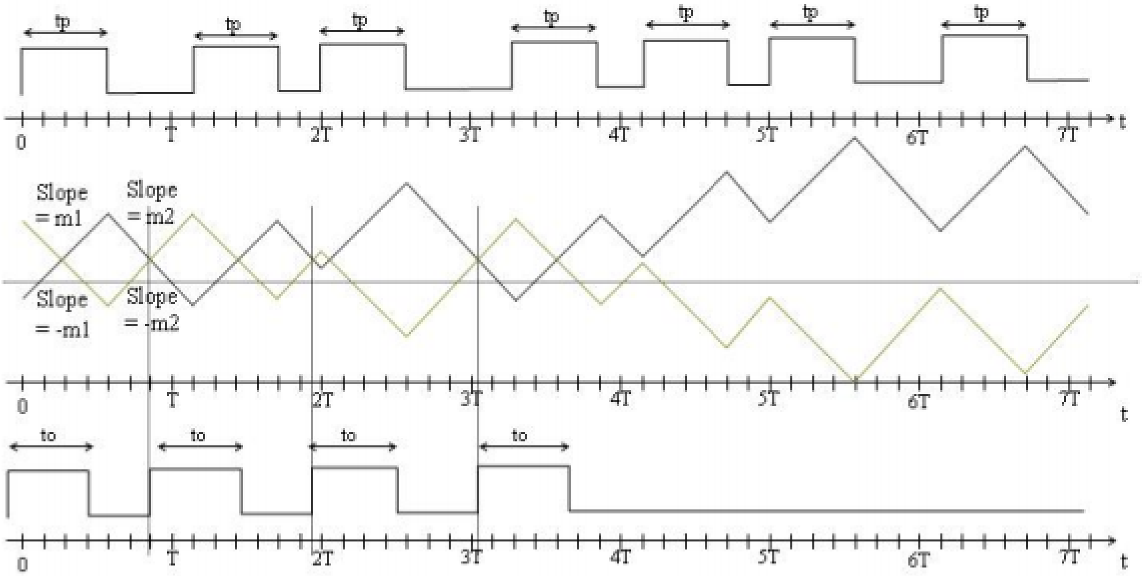

Figure 2: Possible JAC Error

Ideally, the circuit will be able to remove 100% of jitter on any digital signals. However, reality sets in when designing such a circuit, and many non-idealities (intrinsic noise, switching noise, parasitic capacitance and inductance, process variation, etc.) make achieving this theoretical operation extremely difficult. Realistic imperfections and difficulties during design are discussed below.

IV. Functional Decomposition

This section provides an overview of the functional blocks that comprise this circuit. There are three blocks that provide core functionality that will be detailed, along with any sub-blocks that accompany them.

A. High Level Description

The circuit proposed takes a jittery digital signal, and outputs the same signal with the jitter attenuated. It will contain three blocks, and will comprise an entirely feed-forward system. First, a monostable multivibrator generates a pulse of a fixed width every time it receives an edge from the clock signal. It outputs a pulse train that will be processed by an integrator (the step that removes the jitter). Finally, the integrated pulse train will be fed to a comparator circuit that will recreate the ideally jitter-free clock signal.

B. A Picture

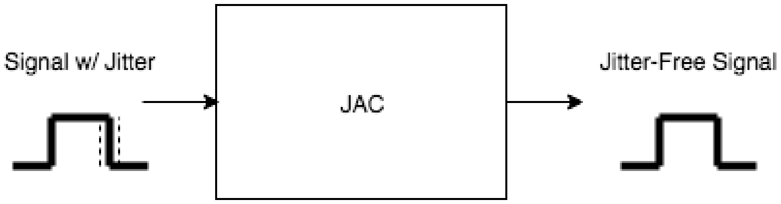

Figure 3: System-Level Block Diagram

This circuit has one input and one output. A digital signal that has unwanted jitter is fed into the circuit, and a signal with the same frequency, logic levels and clean period is output. It should be noted that the design of the internal blocks utilizes differential signaling to eliminate issues common to single-ended systems (i.e. noise), so if the input signal does not have a readily available inverse, a single-to-differential converter is needed.

C. High-Level Block Diagrams of Circuits

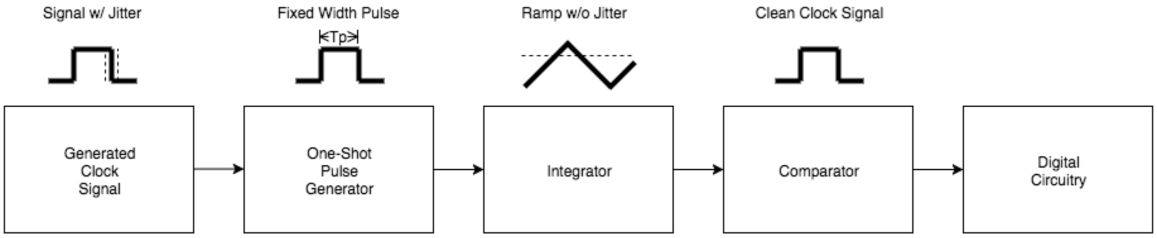

Figure 4: High-Level Block Diagram

There are three core blocks in the JAC. Each block plays an important role in removing the jitter, and there are certain conditions specific to each block that must be met for proper performance. These conditions are discussed below in sections D and E, and affect certain design considerations and decisions when implementing each block in hardware.

D. Detailed Block Diagrams

I. Monostable Block

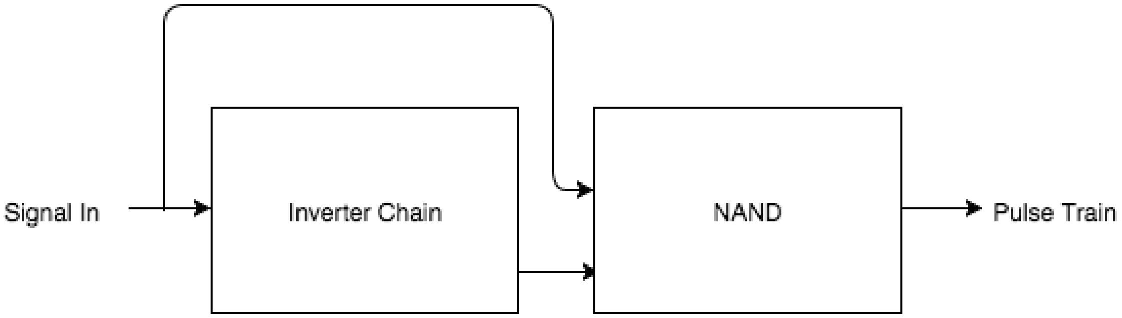

Figure 5: Monostable Block Diagram

Instead of using more sophisticated designs common to commercial monostable multivibrators that involve oscillators and other op-amp-reliant circuitry, a simple gate-based design is proposed. It consists of a NAND gate and a chain of inverters to produce a pulse at the output of the block. The input signal is both fed directly to the NAND gate, as well as through the inverter chain to introduce a delay until the signal pulse is seen on both NAND inputs, thus producing a pulse of duration equal to the combined propagation delays of the inverters. A system constraint appears in this block: the delay (and thus the pulse width) is constant, and will provide different duty cycles at the output for input signals of varying frequencies. For this reason, the system is designed to work most optimally for 500MHz (for now) input signals. Furthermore, the output pulse duration must remain constant for the circuit to work, so dynamic adjustment of the duty cycle is not possible with this current design.

II. Integrator Block

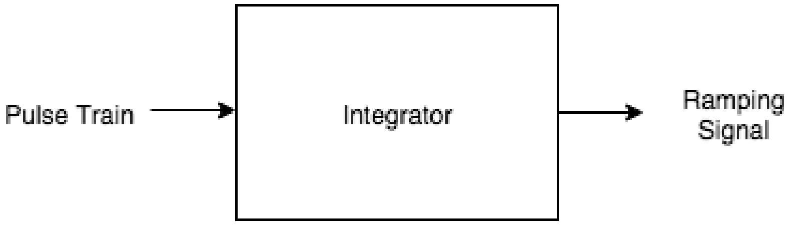

Figure 6: Integrator Block Diagram

The integrator block is the heart of the JAC. It generates a ramping signal at the output: ramping upwards when the input pulse is high, and ramping downwards when the pulse is low. The key of this block is that regardless of the jitter in the pulse train, a comparator can analyze the ramping signal and recreate the digital signal free of jitter. If this system were designed to be single-ended, the ramp would have to be compared with the DC average of the pulse train. However, utilizing differential signaling allows for the comparator to simply take the crossings of the ramp and it's inverse as the trigger for the output regeneration block.

III. Comparator Block

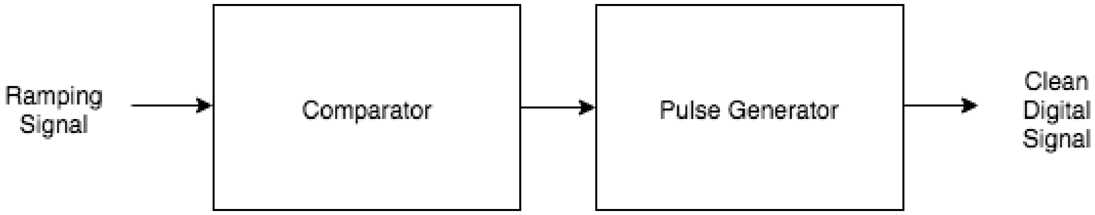

Figure 7: Signal Regeneration Block Diagram

The comparator is set to trigger its output every time it sees a crossing of the differential ramping signal, or that is every time they are equal. Once the comparator sees a crossing, its output goes high, and a pulse is generated at the output of the block. Because the jitter was removed at the integrator stage, this output pulse train should be periodic at the intended frequency.

V. Design and Implementation

A. Individual Circuit Schematics

This section provides information on the implementation of the proposed circuits. Certain design choices such as transistor sizing will be highlighted. Schematics and simulation results are both provided from the Virtuoso design suite.

I. One-Shot

The one-shot pulse generator consists of an inverter chain that is fed into a NAND gate, and the delay of the inverter chain determines the width of the pulse at the output of the gate. Both of these circuits are differentially signaled by design, using MCML (MOS Current Mode Logic). The Virtuoso circuit schematics for these two blocks appear below.

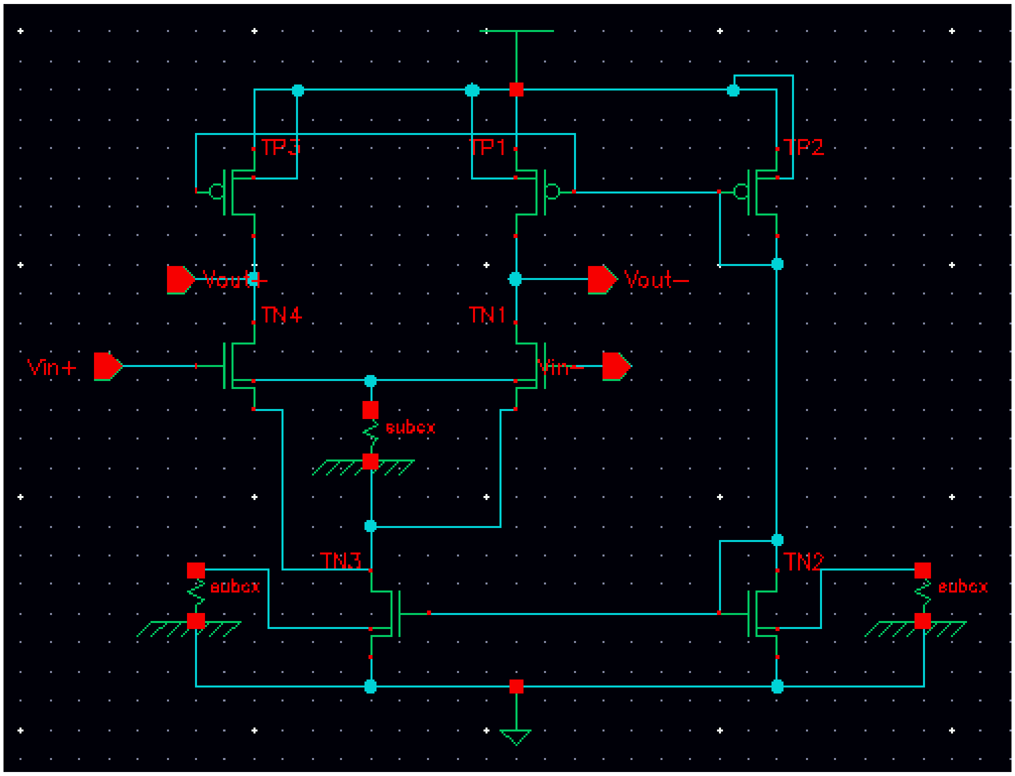

Figure 8: Inverter Schematic

The circuit is a biased differential pair. The lower NMOS transistors provide inputs differential inputs Vin, and the nodes between the NMOS and PMOS provide the inverted outputs, Vout. The current sinking transistor is designed to have larger dimensions to allow for more current to be consumed by the switches, and the switching transistors have much smaller sizing to reduce capacitance. This allows for the transition speed necessary to operate at higher frequencies.

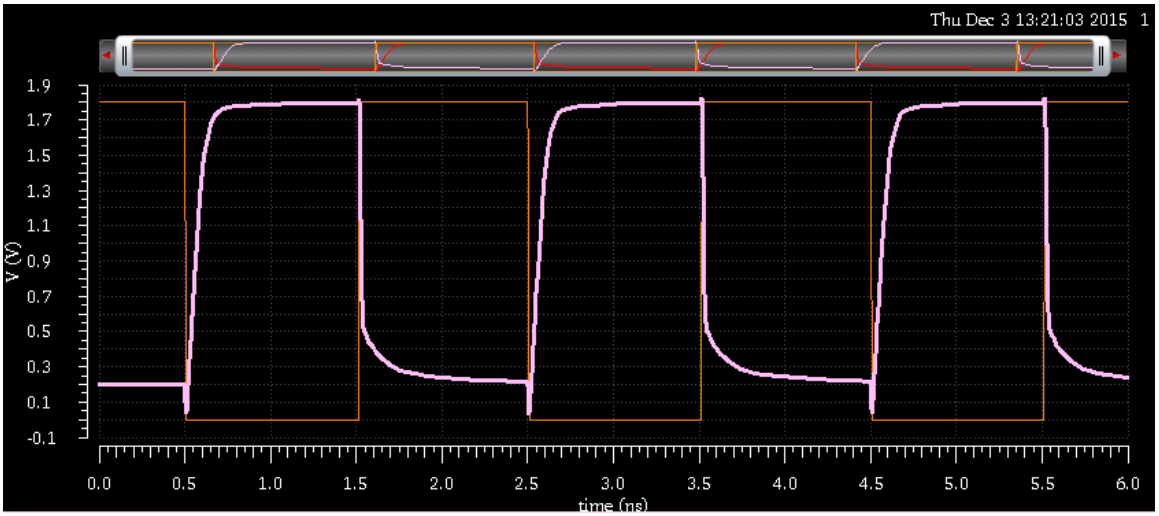

Figure 9: Inverter Simulation

A quick simulation with a pulse input shows the output being inverted properly, with a delay of approximately 20-40 ps. Currently, the rise time seems to be much slower than the fall time, and pull the supply lower than ground before transitioning upwards.

The NAND gate follows the same design logic, utilizing the same biasing circuitry and similar transistor sizing to ensure best interfacing to other blocks. A Virtuoso schematic appears below.

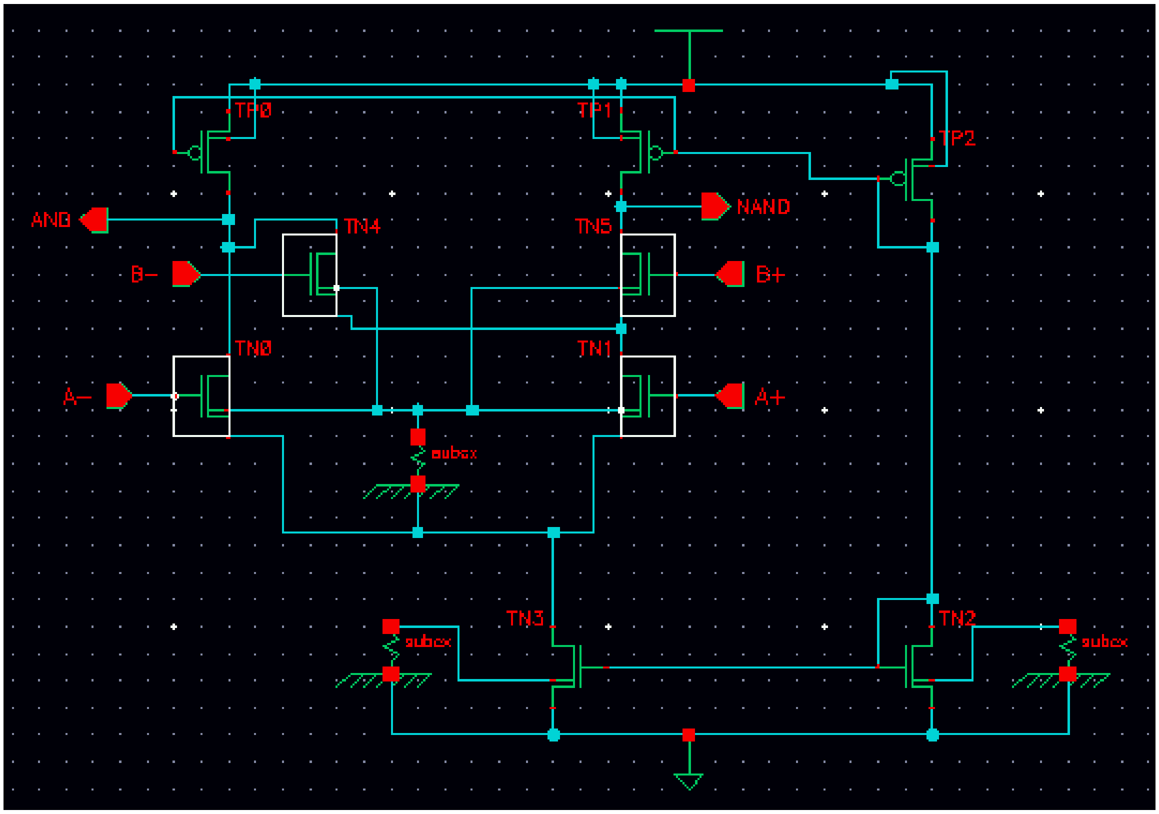

Figure 10: NAND Gate Schematic

The NAND get is a differential circuit, and takes inputs A and B and provides both NAND and AND outputs (inverted NAND). A simulation of the gate shows correct operation of the outputs.

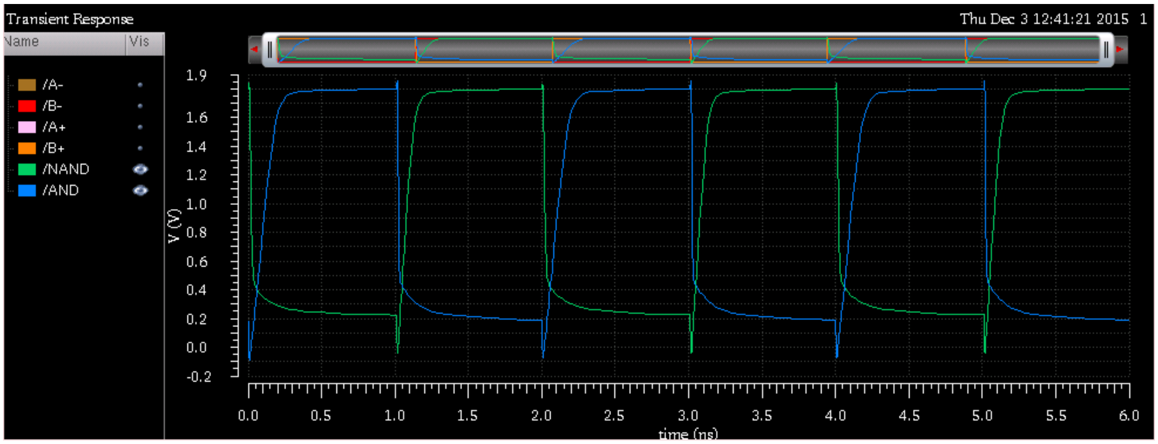

Figure 11: NAND Gate Simulation

Given a logic high pulse on both inputs, the NAND output goes low. Similarly to the inverter, the rising transitions are slower, and pull the supply low upon reaching the threshold voltage. This is most likely due to sizing of the switching transistors, and is being investigated.

To create the pulse generator block, multiple inverters and connected in series and placed in the path of the input to the NAND gate. A schematic of this circuit appears below. Note that the block labeled 'Inverter Chain' is simply eight inverters connected in series to reduce clutter on the Virtuoso design screen.

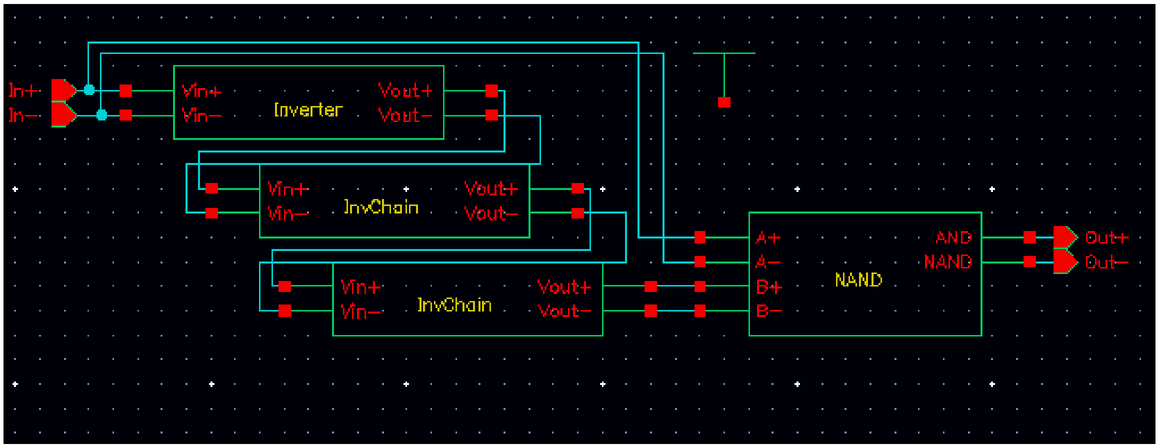

Figure 12: Monostable Pulse Generator Schematic

The delay of each inverter in series adds together to create at the desired pulse width at the output of this block. The results can be seen in simulation.

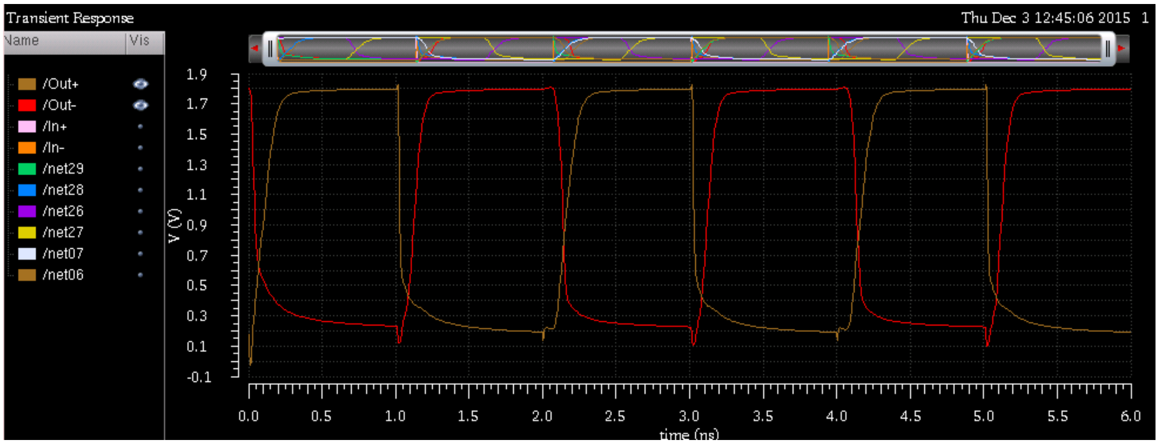

Figure 13: Monostable Simulation

This simulation depicts a pulse train output from a 500MHz input signal. The output has approximately 50% duty cycle at the same frequency. The rise time issues from the inverter and NAND circuits can be seen in this screen; the falling transitions are quite sharp in comparison. The output swing of this block is well behaved; only sacrificing some voltage for the V_{DS} of the conducting transistors.

II. Ramp Generator (Integrator)

The ramp generator is responsible for generating a constant sloped sawtooth function, so the comparator block can regenerate the cleaned clock signal properly. The circuit utilizes similar design techniques, following the biasing circuitry of the previous stage to ensure interfacing. A schematic of the integrator appears below.

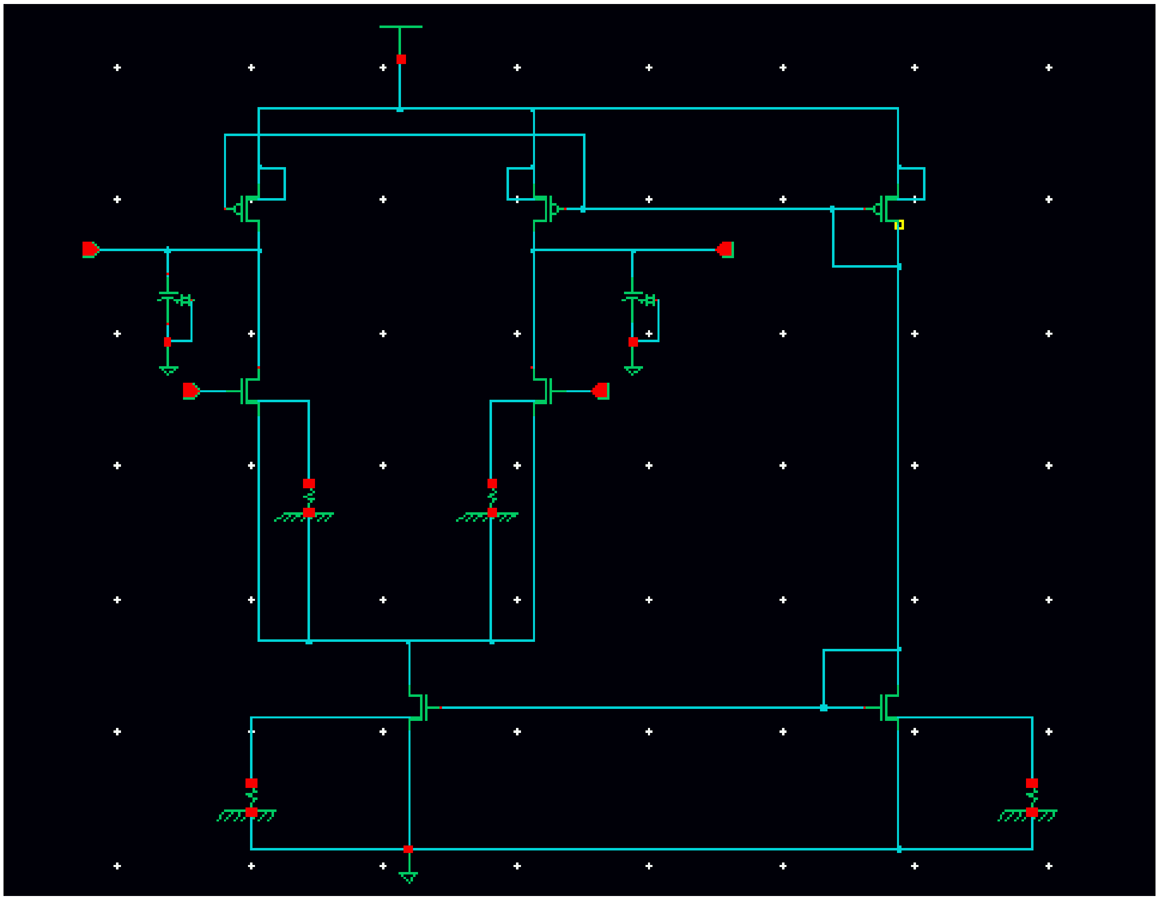

Figure 14: Integrator Schematic

The two capacitors at the output of the differential pair provide differential ramping signal outputs. A simulation with a pulse train input shows the operation of ramp generation. The current sunk into the two branches may be adjusted to generate different slopes.

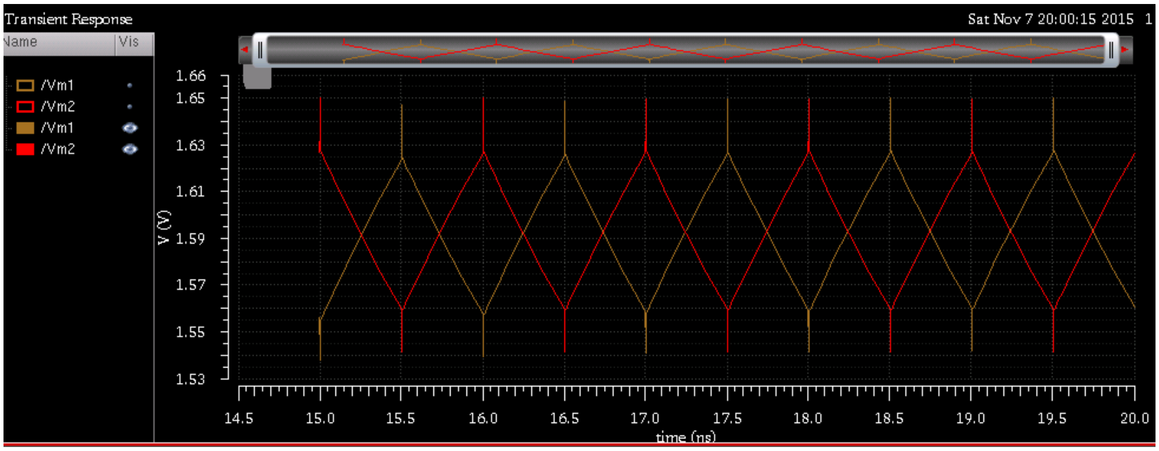

Figure 15: Integrator Simulation

The simulation shows the two outputs ramping up and down with the pulse train, crossing at a constant frequency. It is vital that the output signals of the integrator cross periodically, for the output regeneration relies on this fact. Spikes appear on the transitions of the ramping wave; these are due to a very sharp input wave used for the simulation.

Concerning the Output Swing

In the simulations shown above for the pulse generator block and its various components, one will notice that the output swing is almost the full supply range. Designs that utilize MCML generally design for much smaller output ranges; typically in the 200mV range. For testing purposes, we adjusted the various circuits to utilize a much smaller output swing, but this had little effect on the overall circuit. Adjusting the inverter and NAND circuits to operate on the smaller range allowed for slightly better switching times, but once the output of the monostable block reaches the integrator block, it makes no difference. Shown below are two simulations depicting operation of the monostable being fed into the integrator; one with full output swing, and one with small output swing.

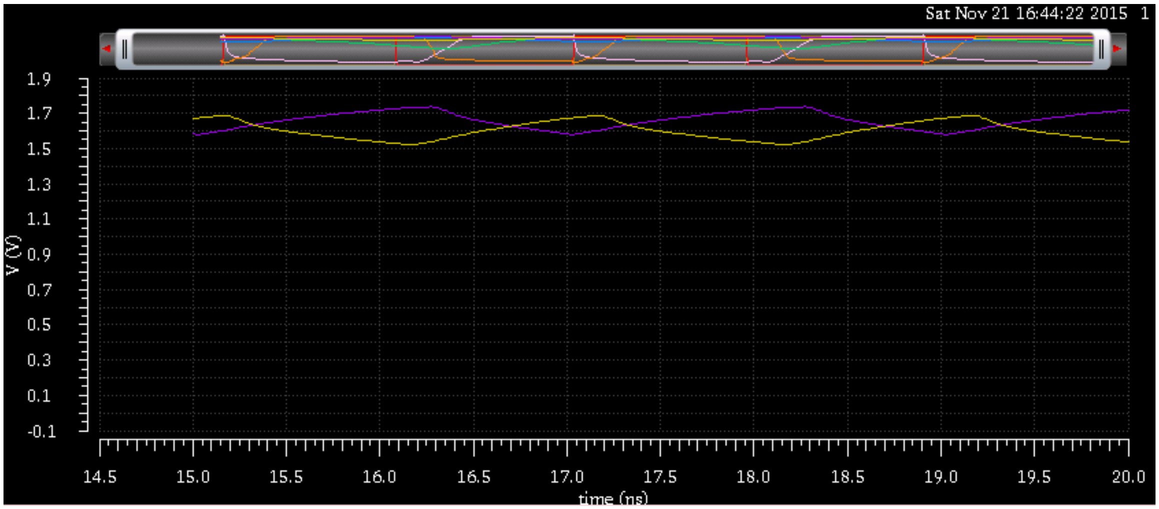

Figure 15.1: Integrator Output with Full Swing Circuitry

This simulation shows the integrator output after being fed the signal from the pulse generator designed with a full supply (~1.6V) output swing. Notice that the output swing is about 300mV, but some curvature appears on the ramping waves.

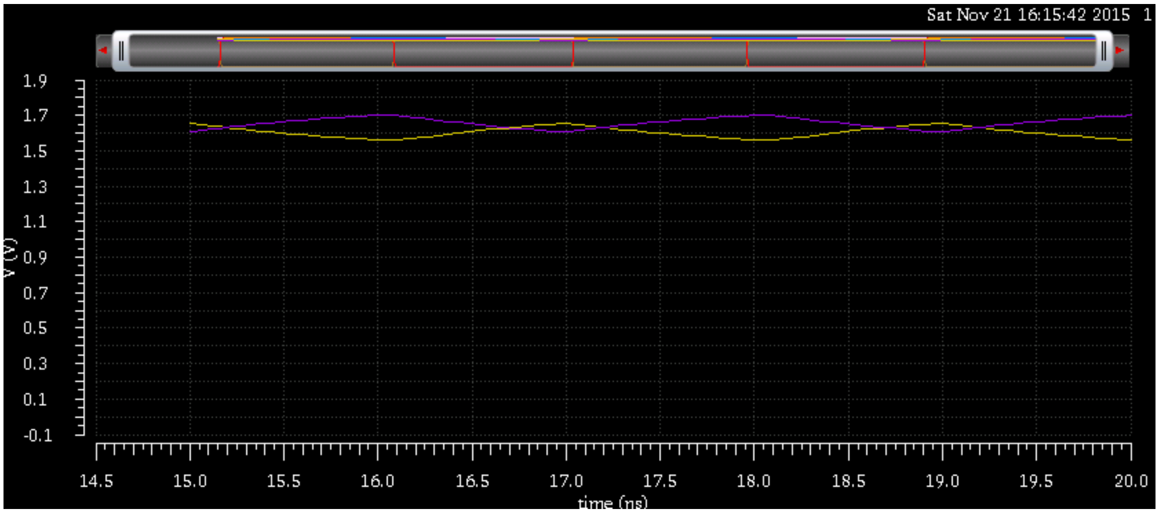

Figure 15.2: Integrator Output with Small Swing Circuitry

This simulation shows the output of the integrator after it has been fed the signal from the pulse generator designed with 200mV output swing. Notice the swing of the integrator output is approximately half (150mV), but there is very little curvature of the ramp. Ultimately, the designs of the inverter/NAND circuits were left to utilize a large output swing because of the extra signal clarity needed to detect the crossings of the integrator at the output regeneration stage. More of this reasoning is discussed in the next section.

First, notice the DC level of the sawtooth waveform is quite high, almost at the positive supply. The circuitry following this stage needed to be biased close to the center of the supply, so a level-shifter, or source-follower, was implemented to bring the level down. The schematic for this circuit is seen below, followed by a simple sine wave simulation showing the level drop.

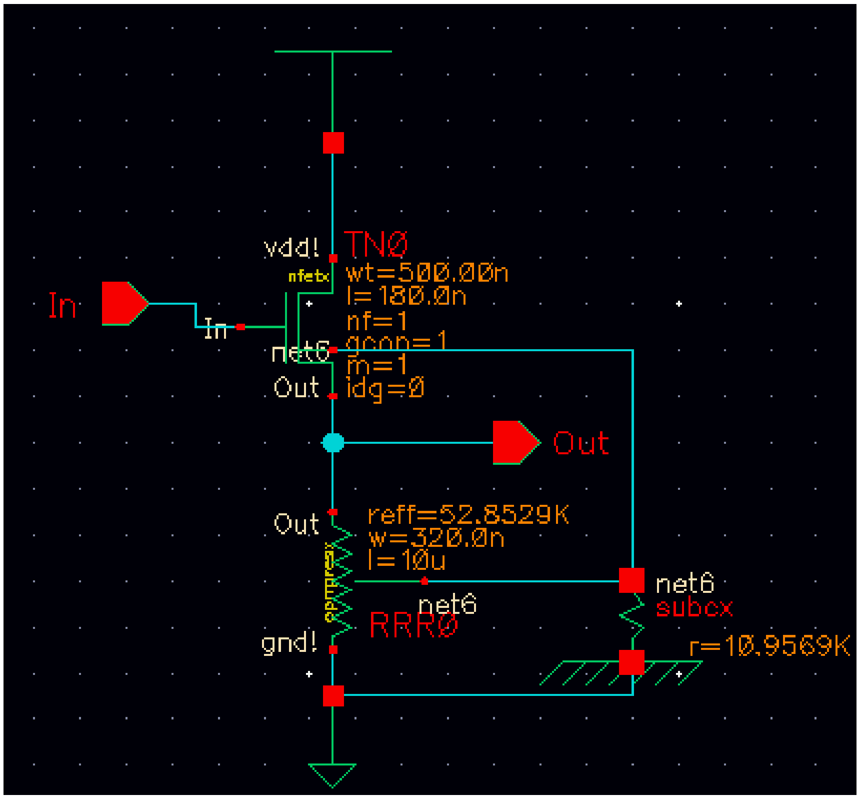

Figure 16: Source Follower Schematic

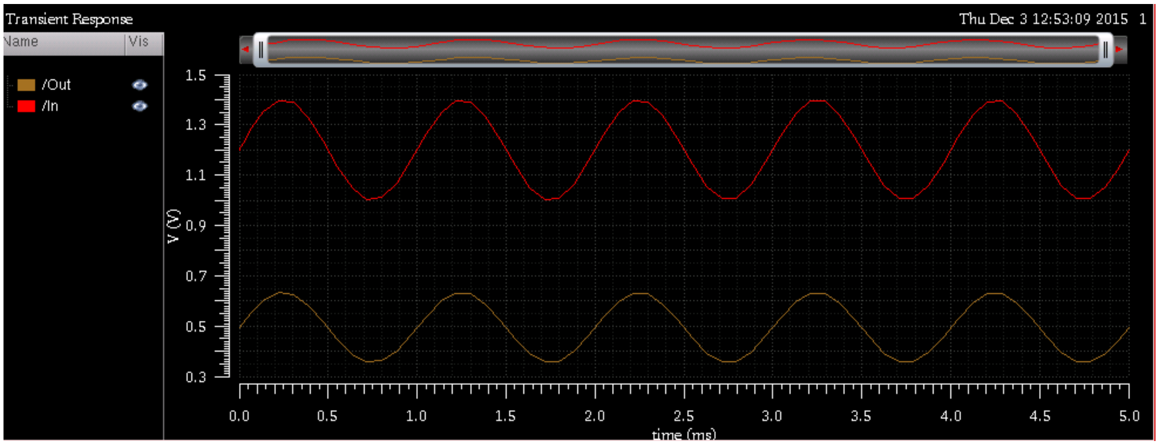

Figure 17: Source Follower Simulation

The signal drops down by a factor of approximately $V_{GS(ON)}$, or 0.55V. This turned out to be just enough to bring the level down to the right value, seen below. Notice that the signal is mostly preserved, but is now biased around 0.8V.

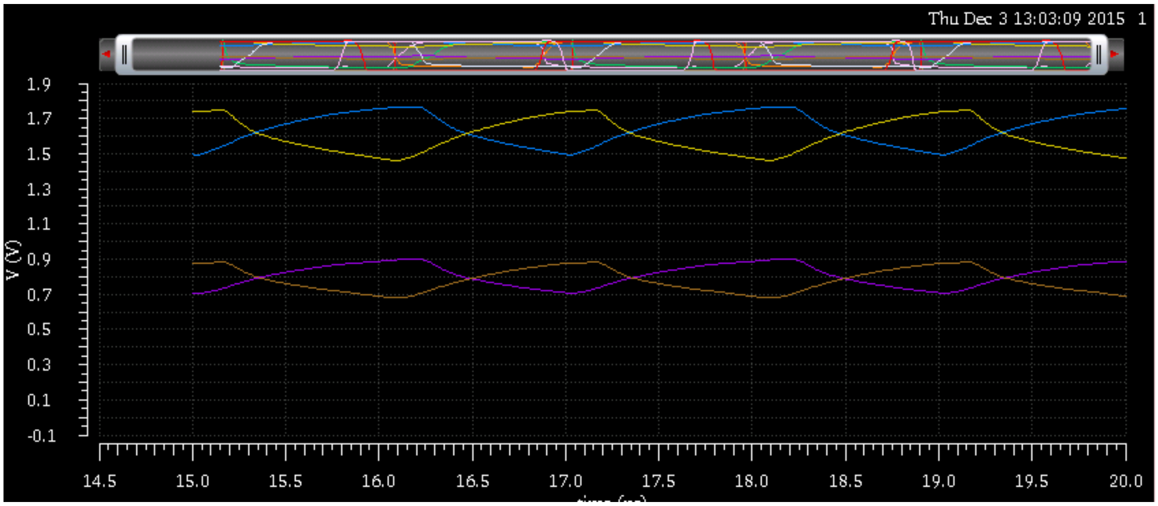

Figure 18: Sawtooth Waveform After Level Shifting

III. Output Regeneration

Once the first two blocks are working together, the third block must be able to regenerate the signal cleanly. In it's simplest form, it is a comparator that triggers when its inputs are equal, and generates a square pulse on each trigger. Ideally, the output of the integrator will have crossings at the desired output frequency, and the output pulse generator will be able to create a perfectly sharp square wave. In reality, this output wave might have some curvature to it, which might not comply with the needs of the following circuitry.

Properly comparing the output of the integrator circuit can get complicated. The output range of the integrator circuit is quite small (~150mVp-p), so the comparator circuit must have a pretty large gain while constantly operating in the proper range. To achieve the correct output (with sufficient gain), the signal is fed through a differential pair and two CMOS inverters.

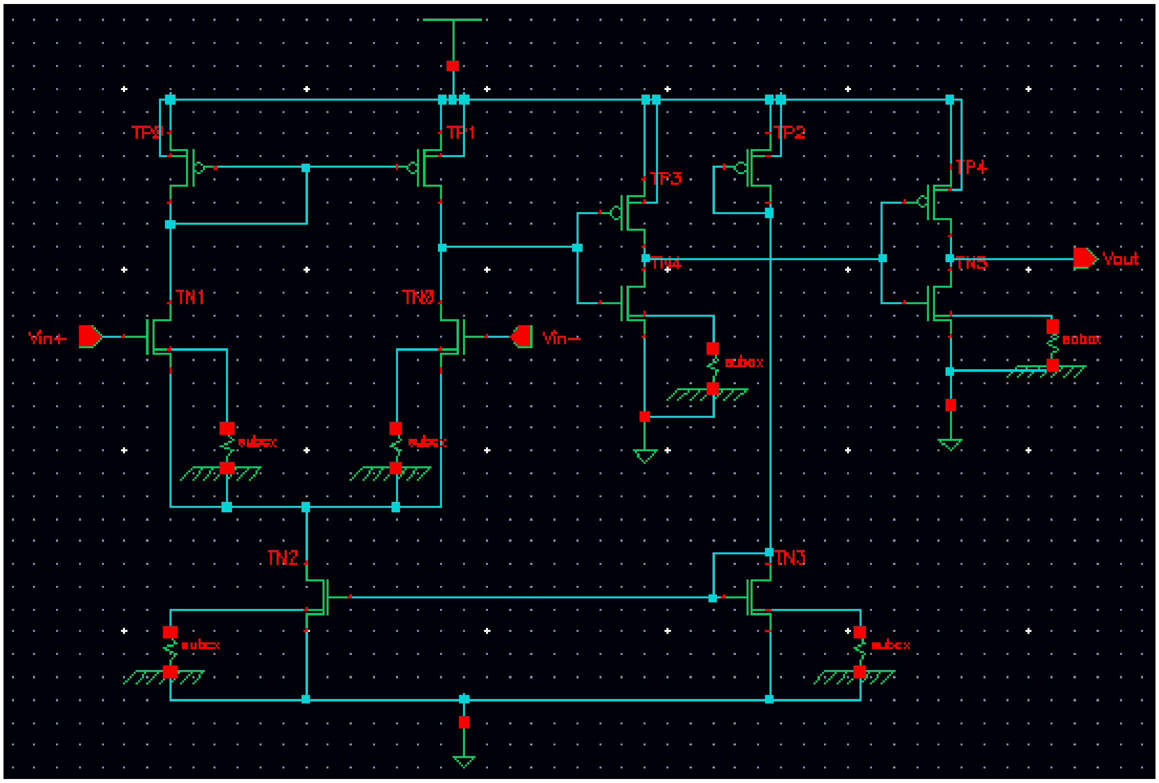

Figure 19: Comparator Schematic

To simulate the correct operation of this circuit, and ideal sawtooth wave (one side of which is shown below in Figure 15) is input into the differential pair. The output waveform shows the non-inverted sawtooth wave being amplified enough to resemble a square wave with sharper transitions. This should be enough amplification to use the circuit as a comparator.

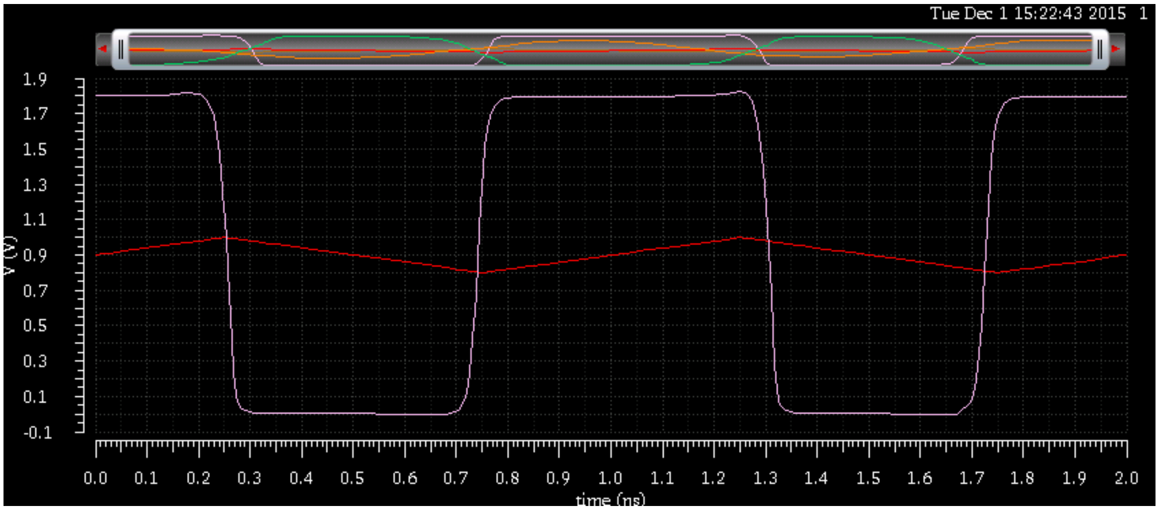

Figure 20: Comparator Simulation

Once the ramping waveform has been transformed into digital signal representing the crossings by the comparator, the same pulse generator used in the first block is used to create the output pulses at the original frequency.

F. Complete Circuit

To complete the project, all three blocks are connected together and tested. This section shows the schematic of the completed circuit, and examines the results of the implementation when connected together. For detailed simulations of the final circuit with jittery input signals, refer to section IV.

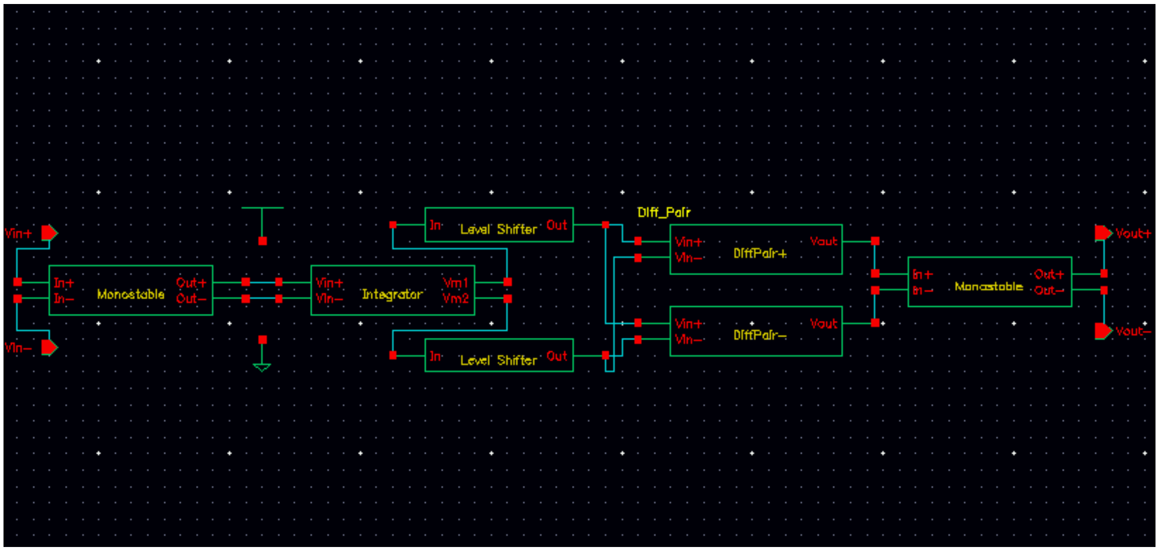

Figure 21: Completed Circuit Schematic

The circuit, from left to right, consists of the first pulse generator (Monostable), which feeds into the ramping waveform generator (Integrator). Then, the opposing differential signals are each fed into the single-ended source followers (Level Shifter) to bring the bias down towards the center. To amplify and compare the signals, there are fed into identical but opposing differential pairs (DiffPair+ and DiffPair-). There needed to be 2 because of the way they were designed. Finally, with the crossings properly compared, the signals are fed into the pulse generator again to produce the output. Notice that each block (or pair of blocks) feeds directly into the following block. This is the feed-forward design philosophy mentioned above. It makes for a very simple circuit, but introduces some other issues, which are discussed at length at the end.

To simulate the basic operation of this circuit, an ideal 500MHz square wave is input to the circuit (1GHz was proving to be too fast for the design). The purpose of this test is to check for things like connectivity, ensuring that circuits aren't loading each other, etc. If the output of the JAC is of the same frequency of the input, then the system works at its most basic level. Shown on the next page is an ideal simulation.

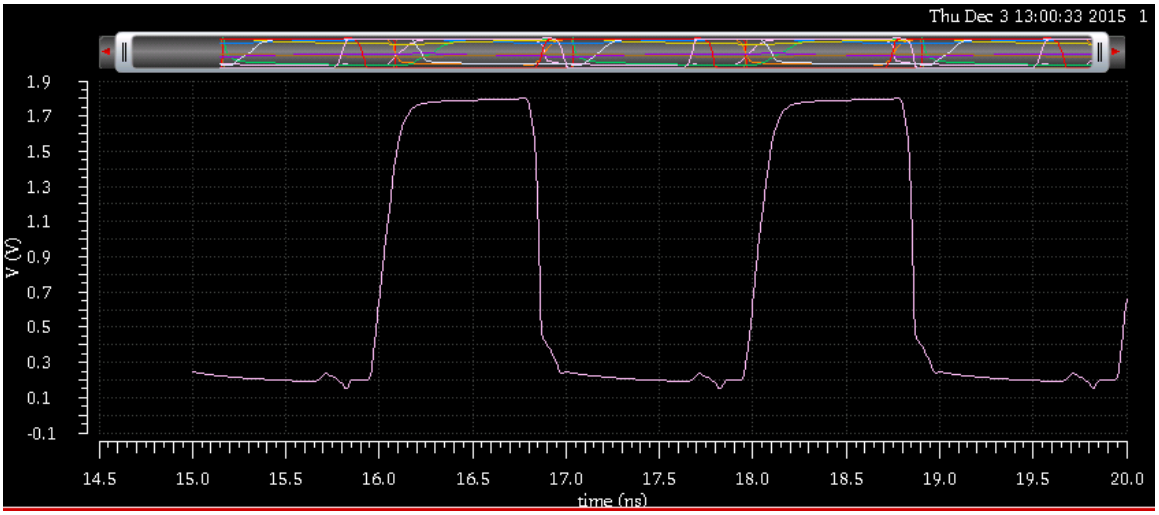

Figure 22: JAC Simulation (Ideal Input)

It is difficult to see, but the waveform shown above has a frequency of (almost) exactly 500MHz.

Concerning Startup Time

Because of the nature of the integrator circuit, the system needs some time to charge up before the output will appear at the end of the circuit. This is because the capacitors on the integrator need enough time to reach their DC steady state value. After some testing, the minimum startup time required for the system is approximately 4ns. This result is illustrated below.

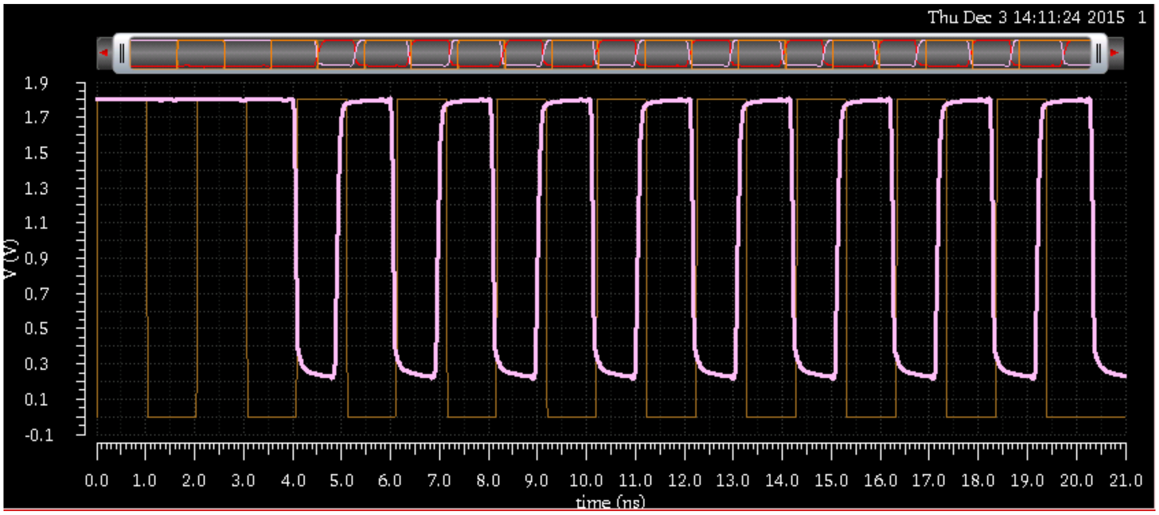

Figure 23: Startup Time of System

VI. Final Testing

In the previous sections, the circuitry was tested in the Spectre simulation environment using jitter-free square wave inputs. To truly test the operation of the circuit, simulations using jittery square wave inputs are detailed below.

A simple method for testing jitter in Spectre is to manually create a piecewise-linear (PWL) waveform. To simulate jitter in the system, two pulses were shifted-one to arrive earlier than expected, and one to arrive later than expected, by 200 picoseconds. A screenshot of the simulated jitter appears below.

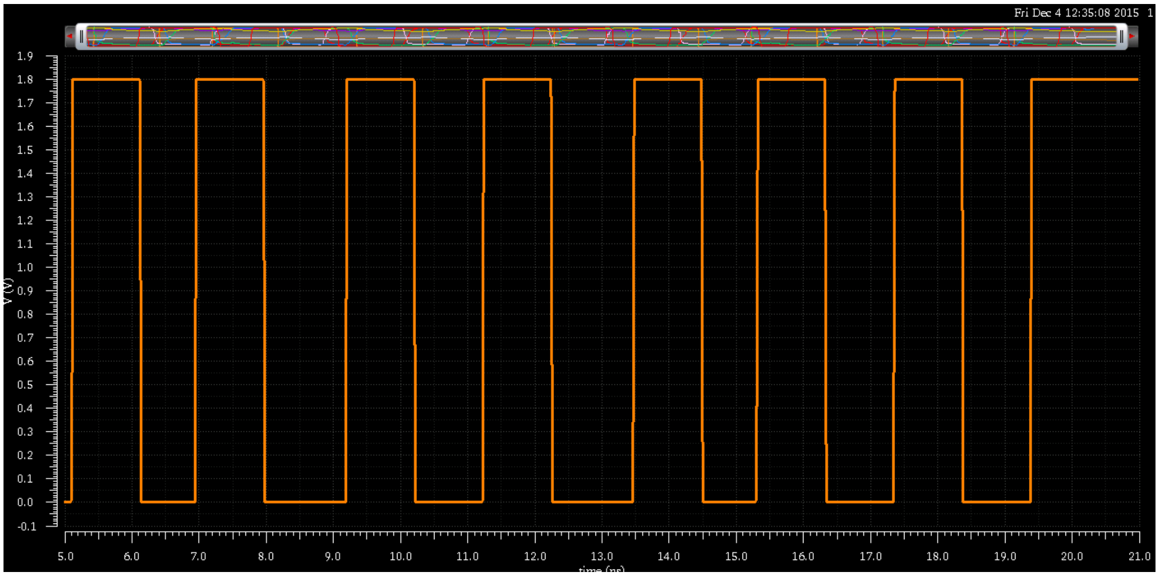

Figure 24: Simulated Jittery Input

Since the original period of the input signal is 1 nanosecond, this simulation tested a 20% positive and negative jitter.

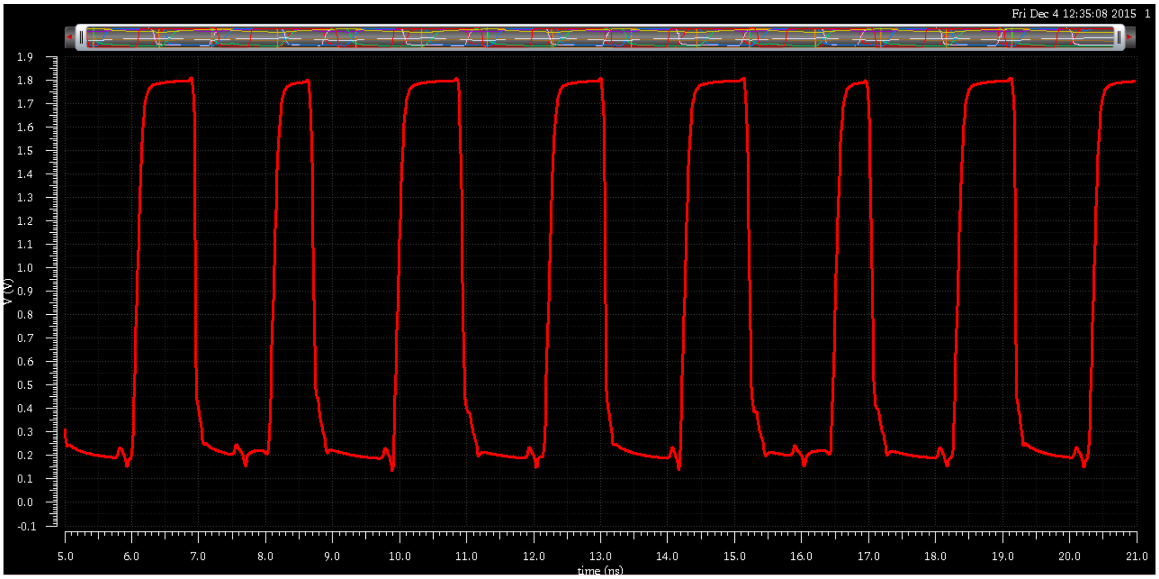

Figure 25: Simulated Jittery Output

Unfortunately, the system did reproduce the original frequency at the output. Instead of producing the output pulse train of the original frequency, the jitter seemed to only distort the corresponding output pulse for the shifted input pulse. Analyzing the rest of the circuit shows this pulse distortion appearing on every single block, all the way back to the output of the first monostable pulse generator. For some reason, the monostable is not outputting a constant pulse width with the jittery input.

Although the output isn't quite right, the circuit seems to be quite close to working. Some of the circuits need to be tweaked and tested further to allow for a more robust performance under the condition of periodic jitter. The amount of jitter simulated on this circuit seems to be a bit large (a digital signal that has periodic noise of 20% is a pretty poor signal). If I had more time, I would like to re-evaluate some of the design choices to see if I could achieve some more stable circuit performances.

VII. Layout

Due to time constraints, and a lack of real benefit from performing layout by hand (completely custom layout), the layout of this circuit uses the automatic tools provided by Cadence. The circuit is fairly complex (compared to previous circuits I developed with the software-CMOS inverters, etc.) so the automatic place and route tool runs the risk of not producing a completely working layout. The result after running the tool and doing some boundary adjustments appears below.

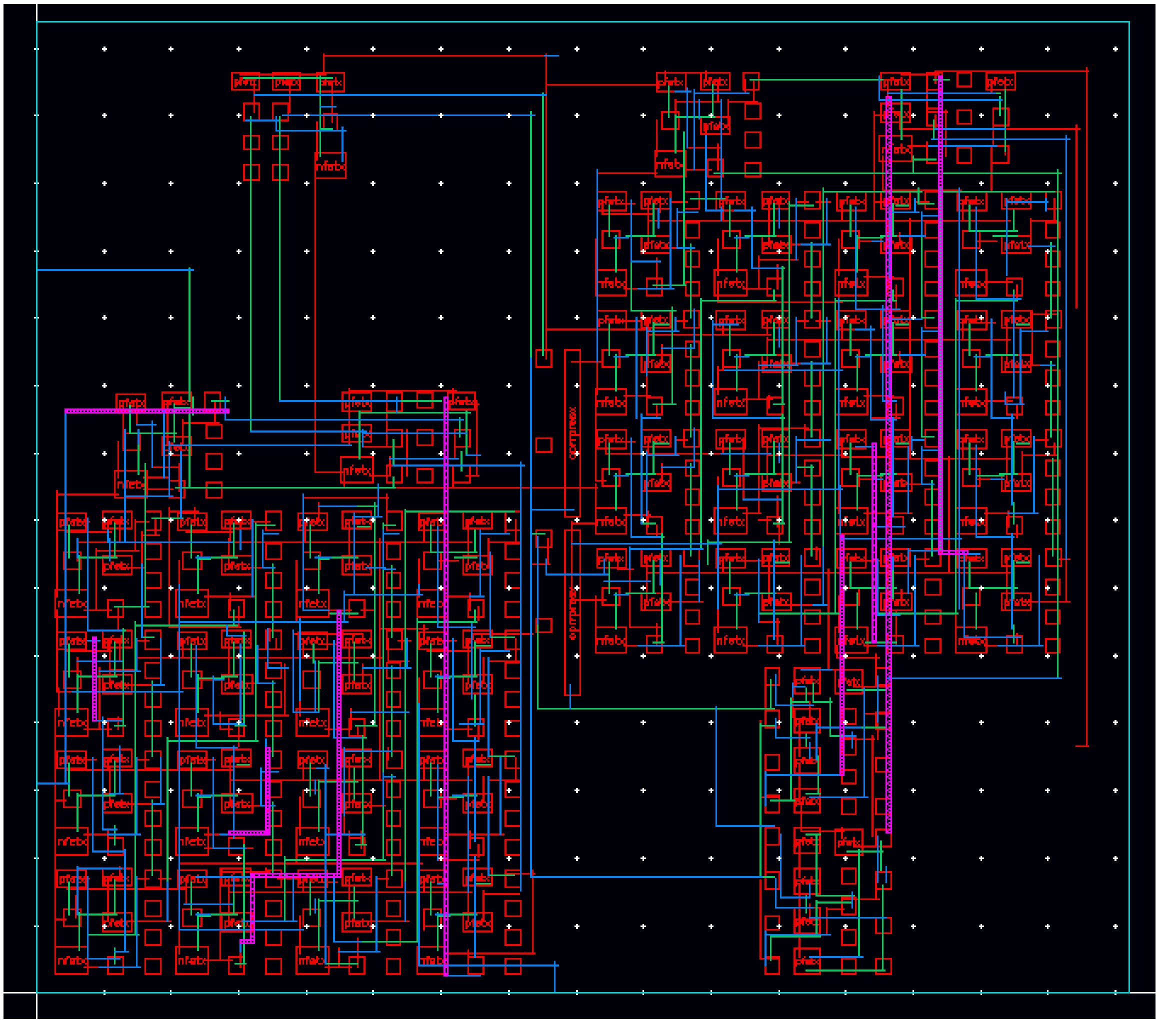

Figure 26: Routed Layout

The automatic P&R tool completed without any errors, but this is farm from an optimized layout. If the circuit were pushed forward towards manufacture, some serious adjustment of this layout would be necessary. Just because a layout doesn't violate any rules, doesn't mean it could not be improved upon.

To move forward with the circuit, one should run DRC (design rules check) and LVS (layout versus schematic) tests to ensure that the layout doesn't violate any strict rules of the process, and that the connections match up with those defined in the schematic view. After running these checks on this circuit and layout, some DRC errors were encountered that would prevent this circuit from being taped out.

To complete the layout process, I/O pads were added to the design, and attached to the four inputs and outputs of the final circuit, as well as the power rails. See below for an image displaying the completed layout.

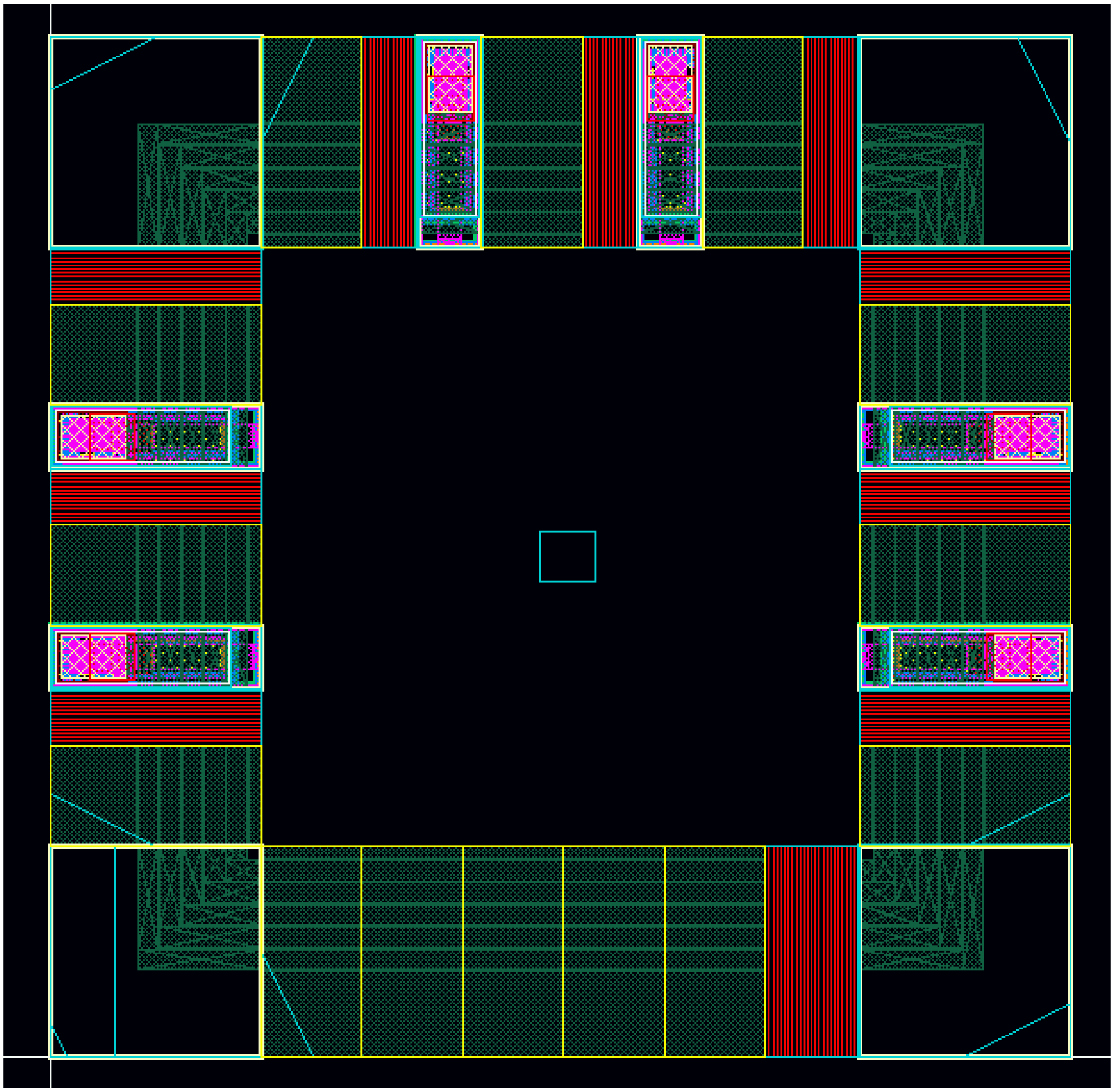

Figure 27: Layout with I/O Pads, Corners, and Fillers

Although the layout ended up not being ready for tape-out (if it were to be taped out), it was an interesting learning experience to see how difficult it can get to perform layout of a circuit with moderate complexity (imagine the layout process of a modern Intel CPU or an Nvidia GPU).

VIII. Analysis of Results and Conclusion

The true difficulty in designing a circuit like this lies in the fact that it is completely feed-forward, without any kind of control circuitry. Each stage relies very heavily on the accuracy of the preceding stage, and any errors accumulated along the signal path can be amplified as they travel through the circuit. From a designer's perspective, this means very tightly controlled design parameters and finely tuned circuits to ensure that each block provides the following block with a signal that is as close to being ideal as possible. This section details some of the more serious issues encountered throughout the design process.

First of all, it was very difficult to design discrete transistor circuitry to operate on a 1.8V supply. Trying to bias the circuitry in the best operating range proves to be pretty difficult when you don't have much DC voltage to work with. Coupled with the small voltage range is the issue of high-frequency performance. As transistors begin to switch at frequencies upwards of 1GHz, they become much more susceptible to switching noise, and the effects parasitic capacitances and inductances become magnified. To effectively drive a circuit at a high frequency, the load capacitance must be sufficiently low.

As I stated above, since the circuit is a purely feed-forward system without any method of self-correction, any error that gets introduced to the system cascades throughout the chain. To make matters more difficult, the pulse generator utilizes a chain of inverters to generate its pulse, so any imperfections of the inverter are amplified at the input to the integrator. Initially, I tried to design the inverter to utilize a small voltage swing of approximately 200mV, but for some reason the gain of the inverter was slightly less than one, and across the inverter chain that is used by the pulse generator, the signal would thin out to be indistinguishable from a DC level. To remedy this, I utilized a large output swing (by sizing the PMOS switching transistors to be larger) on the inverters, and the gain seemed to be very close to one. Since the only control of the duty cycle at the output of the pulse generator is the collective propagation delay of the inverter generator (~20ps per inverter), it becomes difficult to get a nice 50% duty cycle wave at the input of the integrator. At this point in the circuit, tested at 500MHz, the pulse generator was outputting a wave of closer to 35% duty cycle.

Once the signal reaches the integrator, it begins charging/discharging the capacitors at the outputs. With a duty cycle less than 50%, the integrator waveforms do not align as they do in the ideal case. This can make it more difficult to detect crossings, thus limiting the range of jitter that the circuit can effectively attenuate. To make the integrator waveforms larger, the capacitor was sized to reduce the effective capacitance, allowing for more charging/discharging per cycle, until the waveform began to distort.

Once at the comparator stage, the duty cycle of the square wave produced is slightly reduced as well. The original frequency is still detectable, but if too much jitter was present at the input, this amplified waveform can be too distorted to properly trigger the output regeneration monostable. Switching ripples also show up at the output of the final monostable block.

Overall, the circuit didn’t live up to the standards set by the theoretical equations. Of course, this was expected, but I would have liked to tune the circuit a little finer to achieve a more robust performance. VLSI circuit design is difficult, and requires a particular set of skills and a great deal of time. If I had more time, I would like to expand on some of my designs to add feedback to help control the flow of the signal through the circuit. I hope that someone else can continue on with this work and design a set of circuits that can get closer to the theoretical performance of the research.

IX. References

[1] Smilkstein, Dr. Tina Harriet "Jitter Reduction on High-Speed Clock Signals" 6 August, 2007. [Online] Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2007/EECS-2007-96.pdf [Accessed: 22 January 2015].

[2] Silicon Labs. "Jitter Attenuating Clocks". [Online]. Available: http://www.silabs.com/products/timing/clock-generator/si5342-44-45-46-47-49/pages/si5342-44-45-46-47-49.aspx [Accessed: 10 October 2015].

[3] Chandhoke, Baljit, "Basics of Clock Jitter". [Online]. Available: http://www.onsemi.com/pub_link/Collateral/AND8459-D.PDF [Accessed 10 October 2015].

[4] Cadence Design Systems, Inc. [NDA Protected Resources].

[5] Sigenics. "How Much Does a Custom ASIC Design Cost?". [Online]. Available: http://www.sigenics.com/custom-asic-cost-calculator/ [Accessed 22 Nov 2015]

APPENDICES

A. ABET Senior Project Design Analysis

A Feed-Forward Circuit for Jitter Attenuation on High-Speed Digital Signals

Student: Eric Rule

Advisor: Tina Smilkstein

I. Summary of Functional Requirements

The functionality of this circuit is straightforward: Ideally, by feeding a signal with well-defined digital levels through the circuit, any periodic noise (jitter) on the signal is removed at the output. In reality, the circuit attenuates positive or negative jitter up to a certain threshold, over a finite range of input frequencies.

II. Primary Constraints

The most difficult aspect of this project is the context in which it is designed. Using VLSI design software from Cadence Design Systems Inc., an 180nm manufacturing process, and a suite of devices compatible with that process adds an entirely different set of design constraints to the project. Students at Cal Poly often design their circuits in ideal simulation environments such as LTSpice and implement their designs on circuit boards with discrete components. Working with 1.8V FETs and a limited range of resistors, caps, diodes and inductors comes with new challenges. Furthermore, the learning curve of using the complex software suite provided by Cadence eats up a good amount of time, and the software can only be accessed by way of virtual machine while on the campus' secure wireless network. Time constraints added some pressure to the project timeline.

One key aspect of the circuit that adds great difficulty to the project is the feed-forward nature of the system. In many modern circuits, feedback is used to help control the behavior of the signals, often times correcting errors that may present themselves. Having only feed-forward circuits connected together allows for error to be amplified in each block, making it difficult to have a well-behaved output over a wide range of system conditions.

III. Economics

Since this project was completed almost entirely in software (design, simulation, layout), so the costs accrued throughout the project lifecycle are small. The only significant monetary cost of the project is the electricity used by both my personal computer and the computer infrastructure that holds the software and generates the wireless network that is used to access the proprietary software.

There are no foreseeable monetary benefits from the execution of this project. The circuit will not be taped out at a fabrication lab, so it will not be sold to consumers.

This project utilizes various kinds of capital. Obviously, the human capital needed to design, simulate and layout the circuit; the real capital consisting of the computers and computer software used to access and run the software to design, simulate and layout the circuit; and the natural and financial capital involved in generating and providing the electrical energy used to run the machines used for this project.

IV. Commercial Concerns

If this circuit were to be manufactured on a commercial basis, a deal would have to be struck with a fabrication company. Many companies only take chip orders in *very* large amounts to make the manufacturing of a chip financially beneficial. If a large enough market were found, a deal with a foundry might be made to produce a run of 1,000 units for initial testing. Different foundries have different rates that also depend on the complexity of the design, so an initial run could cost anywhere from $5,000 to $15,000.

Assuming the product caught on and had a large enough demand, some serious capital would be necessary to move forward. Costs accrue at many different points during the lifecycle of an integrated circuit. After specification, design and development, prototyping and testing are all completed, a relatively simple ASIC can cost upwards of $250,000. Once the circuit is moved into production, assuming that the yield of a given process is relatively high (~95%), an individual ASIC in a simple kind of packaging can be priced around $5 per chip, with a manufacturing cost of around $0.15 for each chip. A run of 100,000 could net approximately $485,000 gross profit.

Trying to estimate costs for a commercial integrated circuit without consulting foundries is extremely difficult, as there are a *huge* amount of factors that can affect the end finances.

V. Environmental Impacts

The most prevalent environmental impacts of this circuit come from the manufacturing process used to fabricate the chip. Foundries use many different kinds of chemicals to etch circuits onto wafers, and by law, must dispose of those chemicals without damaging the environment. There are companies that have developed technologies to treat wastewater from foundries to help reduce environmental impact and reuse some chemicals, but a foundry will always have harmful waste as a byproduct of its manufacturing process.

VI. Manufacturability

As mentioned previously, the challenges of manufacturing an integrated circuit are varied and large in number. There is an entire suite of costs required in designing, developing and prototyping a chip. If the circuit passes the initial checks, there are separate costs of manufacturing the chip on a large scale. Furthermore, market timing is of the utmost importance. Releasing a product too late can cause the company to miss out on a large amount of revenue. Bringing a new integrated circuit from an idea to market requires years of time, millions of dollars, and as such carries high risk from a business perspective. It is said that semiconductor businesses make their money on improving already existing product lines, not on brand new chips.

VII. Sustainability

Since this circuit will be used in conjunction with an existing piece of hardware, any sustainability concerns for this circuit carry over to the host system. The only concern in maintaining this circuit over time is the actual lifecycle of the part. Assuming that a given part is not defective (not within a margin of expected faulty devices), the effective life span of an integrated circuit can realistically be anywhere from 10 to 100 years long. The only significant resources consumed by the operation of this circuit are the various natural energy resources consumed in generated electrical energy such as coal or oil. To improve the environmental impact this circuit has, the host system it is used in should utilize some kind of clean or renewable energy resource such as wind or solar power.

VIII. Ethical

I don't see any inherent ethical dilemmas in the manufacturing or use of this circuit. There is a possibility that the circuit could end up in a larger system that is used to commit some kind of crime. A user cannot directly harm his or herself, nor can they directly harm another person with this circuit.

If the product were to be manufactured on a large scale, I would select a fabrication company that doesn't make use of unethical business practices or work in unethical conditions.

IX. Health and Safety

The integrated circuit has incredibly small physical dimensions (1.5mm^2 area), so it is not physically hazardous to a user. It operates on low-voltage, low power supplies, and only consumes microwatts of electrical power, and poses no electric shock risk to the user. Whatever system the circuit ends up in might have other unpredictable physical or electrical risks.

X. Social and Political

This product will not be used by any general population, and will have minimal impact on societal or political environments, local, national, or multinational. The targeted customer bases for this circuit are technicians and engineers who might find use in improving their circuitry with this device. There are a few stakeholders in the scope of this project: I, the designer; the company who manufactures the chips, the customer who purchases the chips, and the consumers who benefit from systems that the chip inhabits.

XI. Development

An entire host of new tools from Cadence Design Systems, Inc. needed to be learned to complete this project. Coming in with no previous VLSI design experience, it was an interesting experience to see the world through the lens of those who design integrated circuits for a living. To be able to utilize the tools properly, I had to first learn about the CMHV7SF 180nm process and the different FETs and passives that are available for that particular process.

The Virtuoso design environment is a very complex tool, and comes with an incredible range options for the user to ensure maximum flexibility in the design process. Unfortunately, more flexibility means more complexity, and there was a short learning curve to understand the file system of Virtuoso, the schematic entry tool, the symbol creation tool, and the library management tool. Once I learned how to enter and edit schematics and join them with other circuit schematics, I encountered the simulation tool that Cadence integrates into their software stack, Spectre. Spectre is an incredibly powerful simulation tool that provides the designer with a wide range of options for testing a circuit out, many of which I didn’t even get to touch. This project was completed with their schematic entry tool and Spectre simulator, so I got the most experience with it. There are various other ways to design circuits, such as using a hardware description language to describe a digital circuit, and using their synthesis tools to generate logic.

Beyond designing and simulating the circuits, I needed to learn how to perform circuit layout. Cadence gives the user a few options for layout, generally falling under three categories: full custom layout, semi-custom layout, and automatic layout. Full custom layout involves drawing out the actual physical dimensions of the materials in silicon, and provides the most control over the layout, but is very time and labor intensive. There are other methods for laying out a circuit that speed up the process by sacrificing some control to the program, which is what this project utilized.

Aside from having to learn how to use the tools from Cadence, I had to adapt my circuit design skills to a low-voltage, high frequency application. It's hard to anticipate how difficult it actually is to design transistor circuits to operate on high frequencies (MHz-GHz) with such a small supply voltage (1.8V). This proved to be the most difficult part of the entire project, but the most rewarding. Designing circuits at the transistor level and simulating them with the various parasitic effects is a difficult task.