To the Graduate Council:

I am submitting herewith a thesis written by Timothy Richard Grundman entitled "Design and Analysis of a Delta Sigma Modulator for a Fractional N Phase Locked Loop Frequency Synthesizer Operating at 2.4 GHz". I have examined the final examination copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering.

_____
Syed K. Islam, Major Professor

We have read this thesis
and recommend its acceptance

_____
Benjamin Blalock

_____
Ethan Farquhar

Accepted for the Council:

Carolyn R. Hodges

Vice Provost and Dean of the

Graduate School

(Original signatures are on file with official student records.)

Design and Analysis of a Delta Sigma Modulator for a Fractional N Phase Locked Loop Frequency Synthesizer Operating at 2.4 GHz

A Thesis
Presented for the
Master of Science Degree
The University of Tennessee, Knoxville

Timothy Richard Grundman
May 2008

ABSTRACT

With the advances in communications, frequency synthesizers are becoming essential for many different circuits and broadcast bands. This need led to the creation of the fractional N frequency synthesizer. This synthesizer has proven itself to be a great invention allowing for many improvements over similar concepts. However, it only reaches the full extents of its capabilities when it is combined with a Delta Sigma modulator. This combined circuit shows great advances in noise performance and frequency resolution.

The fractional N frequency synthesizer is merely an integer N PLL with the ability to change the division ratio. The first attempts to change this ratio used an accumulator which is a first order Delta Sigma modulator. The single order versions were swiftly discarded for higher order models for their better noise performance.

Delta Sigma modulators, or DSM, are separated into two architecture types, MASH and MBSL. MASH modulators are easier to build and unconditionally stable. Unfortunately, they create more noise than MBSL and require more output states for the same division ratio. MBSL modulators create less noise; their noise shaping is more flexible and uses less outputs for the same division. However, MBSL modulators are complex and have some unstable inputs.

This work steps through the design of a MASH 1-1-1 modulator using its basic equations. Also, it covers the implementation of the circuit on a FPGA

board and its testing as part of a frequency synthesizer operating in the 2.4 GHz frequency band. For this work, the Spartan-3 board is used in addition to a PLL built before to create the circuit. The testing is done using a spectrum analyzer to see if the synthesizer creates the right output frequency. Several tests are performed to see the accuracy of the synthesizer over a portion of the frequency band.

The circuit proves to be successful, creating frequencies within one percent of its target values. Initial tests without the DSM were around two percent of the target values. The work goes on to describe future projects. The major one to be the creation of a frequency modulated transmitter by using the synthesizer constructed and adding a digital filter to adjust the data.

TABLE OF CONTENTS

LIST OF FIGURES

1 Introduction

1.1 Introduction

With the advances in communications technology, frequency synthesizers are becoming essential for many different circuits and broadcast bands. This need led to the creation of the fractional N frequency synthesizer. This synthesizer has proven itself to be a great invention allowing for many improvements over similar concepts. However, it only reaches the full extents of its capabilities when it is combined with a Delta Sigma modulator. This combined circuit shows great advances in noise performance and frequency resolution. This ability for frequency resolution makes it essential for the 2.4 GHz band. Considering the specifications for this band especially WLAN and Bluetooth, which are currently the major uses of this frequency band, will prove the worth of this circuit. The details for these come from [1].

1.1.1 Wireless Local Area Networks (WLAN):

WLANs are a very prevalent use of the 2.4 GHz band. They are as their name describes them a wireless version of the local area network. Among the first uses of this technology was to simplify office networks, however this has expanded almost exponentially from where it began. The introduction of Wifi hot spots was a major push for this technology. Also, college campuses are using this to allow students to connect to the Internet without requiring an excessive amount of rewiring across the campus. This technology has also boomed in the home markets as the technology has improved allowing virtually anyone to take

advantage of the benefits.

The major standards pushing this technology are IEEE 802.11 and IEEE 802.11b. These two standards define the specifications for Frequency Hopping Spread Spectrum (FHSS), operating at a maximum of 3 Mbps and Direct Sequence Spread Spectrum (DSSS), offering 11 Mbps products. IEEE 802.11 defines the operations of WLAN systems at frequencies between 2400 and 2483.5 MHz. FHSS WLAN systems utilize 79 possible channels between 2402 and 2480 MHz with a channel spacing of 1 MHz. DSSS WLAN systems utilize 9 (IEEE 802.11) or 11 (IEEE 802.11b) frequency channels with channel spacing of 22 MHz. The development of the technical standards is ongoing with the demands of the technology. FCC intends to increase the channel bandwidth of FHSS systems to 3MHz and 5MHz to enable it to operate at 11 Mbps, allowing for higher speed transfers. The transmitted power is on the order of 100mW (20 dBm). These particular standards are used worldwide and are making a very large market for the 2.4 GHz, however the next technology is used even more prevalently.

1.1.2 Bluetooth

Bluetooth is the creation of a consortium of IT and telecommunication companies to allow for a short range radio link between devices. This technology is also increasing at a roughly exponential rate. The original ideas for it may have been to link together several pieces of office equipment, however in practice it is the technology for individuals that has increased. The easiest example is the ubiquitous Bluetooth headset that practically everyone has with their cellular

phone. However, there are many other applications for this standard.

Bluetooth uses FHSS, 1000 hops/s technology to ensure reliable performance in a noisy environment, supporting both voice and data, up to a data rate of 1 Mbps. The Bluetooth standard uses the same 2402-2480 MHz spectrum as the IEEE 802.11 standards but the transmitted power is significantly less (1mW or 0 dBm). These two technologies alone are enough to justify research into better transmitter equipment at this band of frequencies.

1.2 Research Motivation

One of the big buzzwords in industry right now is system on a chip (SOC). With the current advances by IBM and other foundries, the feature size in silicon has been pushed down into the tens of nanometers range. This allows for a single chip to have an extremely high level of transistors and processing power. However, the major problem with most transmitters is that they often require special processes. Many transmitters require inductors and capacitors to work correctly. Designers can use MOS capacitors for integrated circuits but unfortunately it is extremely hard to create quality inductors in a traditional CMOS process. At the same time ordering a process that uses these devices will cost a lot more than simple CMOS. This requires consideration of several of the other solutions to this problem. One major concern is the antenna. Currently many designers are building their circuitry to do a short range transmission which avoids this particular problem. Many other designers have found that the availability of transistors on current chips allow us to approach this problem with

digital circuitry. With this paper we will have generated a frequency synthesizer that could be implemented on several new processes. This will make a large step towards creating a short range transmitter that could be implemented on any CMOS process.

1.3 Overview

Chapter 2 will cover the design and implementation of the frequency synthesizer that will be added to this work. Chapter 3 will cover the different architectures of Delta Sigma modulators that were considered for this work and their advantages and disadvantages. Chapter 4 will cover the simulated and measured results from the circuits. Chapter 5 will cover the conclusions, achievements, and future work.

2 FRACTIONAL N PHASE LOCKED LOOP FREQUENCY SYNTHESIZER

This work covers the design of a Delta Sigma modulator to interface with a previously constructed fractional N phase locked loop. The combination of these two components creates the fractional N frequency synthesizer which is tested in this work. This circuit is fairly complicated in its construction and builds off of several ideas. The first step in understanding the design is to see how the phase locked loop can be changed into the integer N frequency synthesizer.

2.1 Integer N PLL Frequency Synthesizer

For most engineers in communications the PLL is one of the first circuits that are built. The principles behind its operation are simple however in design it can be difficult. Using feedback, it takes an output signal to the input and locks their phases together. The major pieces of this are a phase Detector, loop filter, and a voltage controlled oscillator (VCO). The operation of this circuit comes from [2] as follows. "The output of the VCO is phase-compared with the reference at the Phase Frequency Detector (PFD). The polarity of the measured phase difference is used to turn on the pump-up or pump-down current source in the charge pump. As a result, some charge is transferred to or taken away from the integrating capacitor in the loop filter. The amount of charge is proportional to the magnitude of the phase difference. This, in turn, results in an adjustment in the

tuning voltage of the VCO so that its phase is retarded or advanced." Now normally the input and output in this fashion would be at exactly the same frequency, so if a divider is added to the feedback loop a divider the output signal frequency will be a multiple of the input signal frequency. The block diagram of the combined PLL system is shown in Figure 2.1 from [3]. TCXO is the crystal oscillator used with the circuit and N is the number chosen to make

$$F_{VCO} = F_x \div N \qquad (1)$$

Of course this circuit has also implemented a predivider to lower the frequency applied to the PLL. Thus, the actual formula for the entire circuit would expand slightly to give the following equation:

$$F_{VCO} = \frac{F_x \div R}{N} \qquad (2)$$

This design will not achieve the specifications for this project. One major

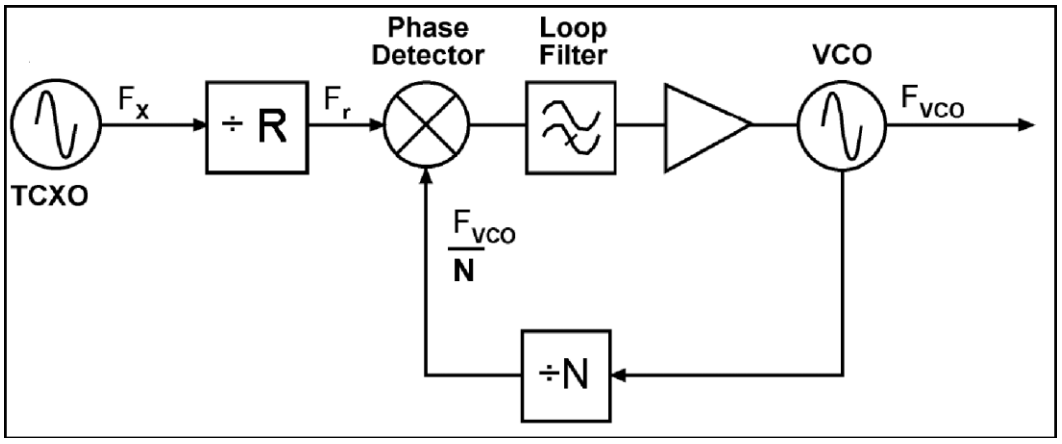

Figure 2.1: A Integer N PLL Synthesizer from [3]

consideration is the phase noise created by the divider. The following equation found in [3] can calculate this value.

$$Phase\ Noise = 20 \times \log(N) dB \quad (3)$$

Phase noise is a major problem in any circuit. For this case since the frequency synthesized is higher than the reference, phase noise will be fairly high. Thus, the phase noise will be an issue to consider. The next consideration is that this circuit can not change its division ratio. For many applications, this is not a problem. However, most synthesizers require the ability to generate more than just one frequency, especially on the Bluetooth band where it will have to be able to change channels. Therefore the circuit will not meet the requirements. Understanding how it functions is the key step to understanding our next circuit and why it has become popular in the communications community.

2.2 Fractional N PLL Frequency Synthesizer

The next development is the creation of the Fractional N PLL Frequency Synthesizer. The earliest examples of this circuit took the integer N PLL and made some small modifications. The idea was to alternate between two nearby division ratios to achieve something like a fractional ratio. To demonstrate this, consider the example above. If 2.45 GHz is chosen as the synthesized frequency the division ratio becomes approximately 24.5. Now the first attempts at fractional division would have just set the divider to divide by 24 half of the time, and then set the division up to 25 for the other half. This trick is called pulse swallowing. Figure 2.2 which comes [4] shows a block diagram of this type of

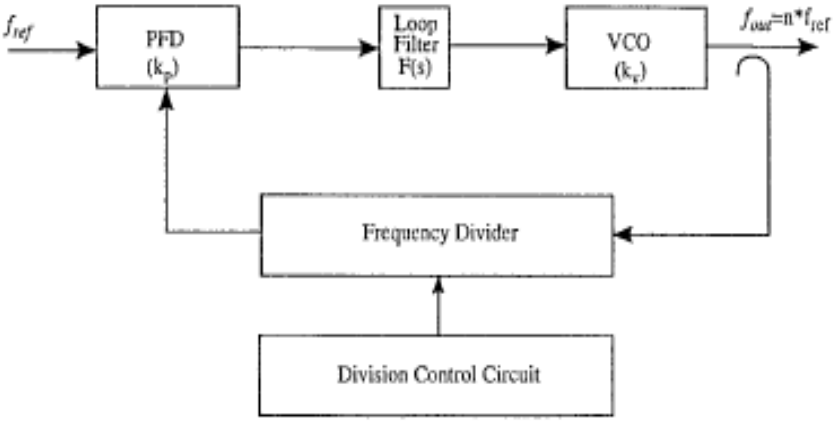

Figure 2.2: Fractional N PLL Synthesizer from [4]

circuit without the actual circuitry to change the division ratio.

This circuit is almost exactly the same as the integer N PLL except for an adjustable divider and some circuitry to control it. The most important difference in the fractional version is the division control circuit, which when studied in detail shows how remarkable it is. The first amount of detail can be shown in Figure 2.3 from [5].

The circuit does not seem remarkable. However, its greatest qualities are seen in the mathematics that can be applied to it. Consider the new equations shown below that come from [6]

$$N_{frac} = N + \frac{K}{2^k} = N + n \qquad (4)$$

$$f_{out} = f_{ref} \times (N + n) \qquad (5)$$

$$K = n \times 2^k \qquad (6)$$

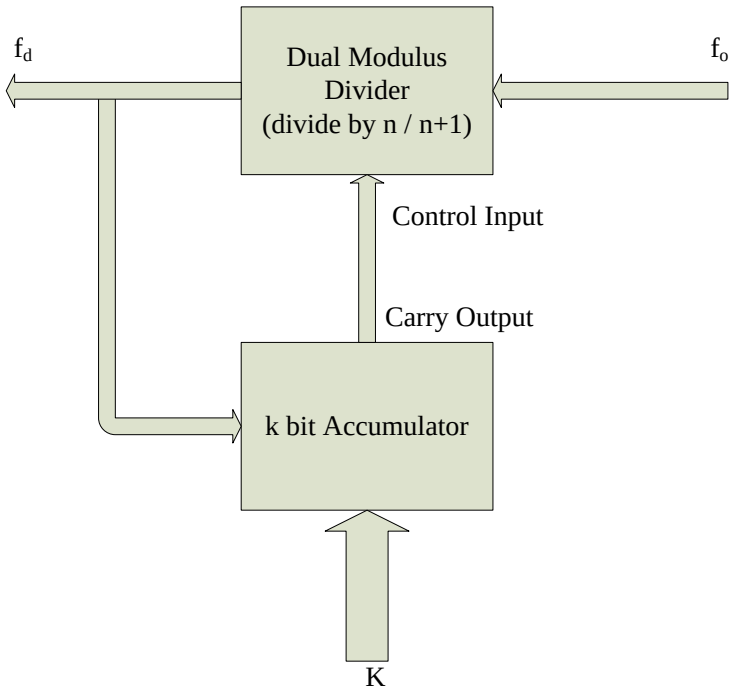

Figure 2.3: Blow up showing division control circuitry from [5]

The variables in these equations are as follows: K is equal to the value of the input to control circuitry, k is the number of bits that K is, N is the integer part of the division ratio, and n is the fractional part of the division. This is very interesting, because it says that the frequency that comes out of the device has a relationship with the input to the control circuitry. For a frequency synthesizer, this is great, because the frequency can be adjusted at any time by merely changing the input word to the control circuitry. This has other implications. Obviously further considerations of this circuit show that it is also ideal for a version of a frequency modulated transmitter. All that would have to be done is to format a signal correctly and feed it into the division circuitry to adjust the

frequency of the output. This makes this a marvelous circuit.

However every marvelous circuit has to have some problems to make it challenging. As stated before one of the easiest methods for creating this circuit is by using the pulse swallowing method. Now the problem with pulse swallowing is well described in this excerpt from [6]:

"In the fractional N loop described above, the VCO is never quite on frequency. That is, it is never an exact integer multiple of the comparison frequency. In one cycle of the comparison frequency the VCO frequency will appear to be high by half the comparison frequency. In the next cycle, the VCO will appear to be low by an equal amount. The loop will therefore attempt to ramp the VCO frequency up, then down in alternate cycles of the phase detector, creating a spur at half the comparison frequency. Because this spur occurs at a fraction of the comparison frequency, it is known as a fractional spur."

The noise power contained in these fractional spurs is obtained from this equation found in [7].

$$Spur = -20 \times \log(\frac{\Delta f}{2 f_m}) dBc \quad (7)$$

Δf comes from the VCO sensitivity and the voltage of any fractional spur. The other variable, f_m, is the frequency the spur is located at. The calculation is a little difficult to do theoretically, however these values are often quite large. There have been several different techniques used in the past to accomplish this task. The next technique that was tried is called phase interpolation. The biggest problem with this method is that it requires a precision digital-to-analog (D/A)

converter or delay generator to make the circuit work which requires fairly precise analog circuitry. The third technique used is called Wheatley Random Jittering. This does not cause as large of spurs as pulse swallowing, but it requires a random number generator. This could have been the best choice; however this type introduces some broad band noise at a fairly large level into the output frequency. Finally, people tried some low order Delta Sigma modulators to attempt the division ratio. These worked without any of the problems that were seen before and can be implemented with entirely digital circuitry. A table showing what has already been stated is shown below (Table 2.1); its information came from [7].

However this type of modulation is not entirely perfect, because it still creates fractional spurs that have the same noise power as stated in the equation above. The example in [7] showing the equation gave a value of -14.4dBc which depending on the value in the carrier could be quite substantial. However, what this modulator does that it moves the fractional spurs to higher frequencies. Thus, the loop filter in the PLL will filter them out and create a very low noise circuit. The next part will discuss the basics of this marvelous circuit and show the different types of architectures that have been created.

Table 2.1: Table showing old Fractional N techniques from [7]

Technique	Pulse Swallowing	Phase Interpolation	Wheatley Random Jittering	∑-Δ Modulated Jittering
Prone to Spurious Frequencies	Yes	No	No	No
Precision Analog Components Required	None	D/A Converter or Delay Generator	None	None
Minimum Complexity of Digital Hardware	1 Accumulator	1 Accumulator	1 Accumulator & Random Number Generator	2 Accumulators
Introduces Broad Band Noise in f_d	No	No	Yes	No

3 THE DELTA SIGMA MODULATOR

3.1 Delta-Sigma Modulator Basics

This section will discuss the operation of the Delta Sigma modulator hereafter referred to as DSM. The easiest way to do this is to start with a first order modulator. An outstanding explanation of this modulation comes from p. 1009 in [8] and will be repeated below:

"As stated previously, the modulator actually provides the quantization in the form of a pulse-density modulated signal. Referred to as sigma-delta or delta-sigma modulation, the density of the pulses represents the average value of the signal over a specific period."

The book was using this to describe how it would work for an analog to digital converter, but it can be applied to frequency synthesizers as well. The randomization is essential for reducing the noise out of the modulator. This circuit is not one that is considered very often, however it is gaining a following in both communications and for signal converters. The major reason that people are switching to this spectacular circuit is its noise performance. Another major reason is the ease of implementation that can be found here. A closer consideration of the circuit will show its full capabilities.

Figure 3.1 shows a block diagram of this circuit, a signal flow, and the digital implementation shown in [9].

Now there is a lot of information contained here that will have to be interpreted. First consider the digital implementation. The circuit shown here is

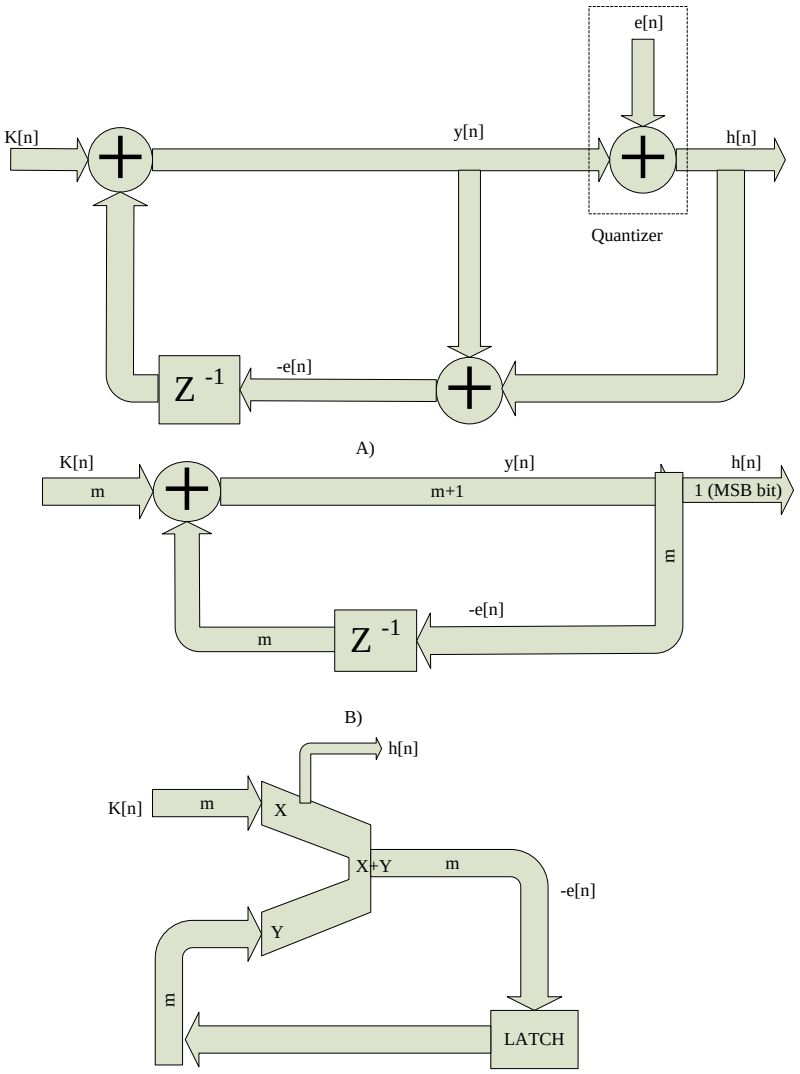

Figure 3.1: a) block diagram b) signal flow c) digital implementation from [9]

called an accumulator. The idea behind it is to take an input and continue adding it to itself until it overflows. This is the first step in understanding the DSM, because the block diagram and the circuit shown in Figure 3.1 are equivalent. The second circuit is there to reinforce this comparison. This allows the creation of an equivalent transfer equation that will be shown below.

$$H[n] = K[n] + e[n] \times (1 - z^{-1}) \quad (8)$$

Here H[n] is the carry bit generated by the accumulator and e[n] is the quantization noise we are creating in the circuit to make a nonlinear circuit give us a linear equation. This equation is not very important at the moment; however it will become essential in the discussion of the higher order versions.

Also consider the noise power in the pass band with this type of modulation. This requires a few equations that come from [9].

$$OSR = f_s \div 2f \quad (9)$$

$$Pn \cong \frac{\Delta^2 \times \pi^2}{36} \times \left(\frac{1}{OSR} \right)^3 \quad (10)$$

$$SNR = \frac{54 \times 2^{2n} \times OSR^3}{\Delta^2 \times \pi^2} \quad (11)$$

These equations show that by increasing the oversampling frequency or the order of the modulator will cause a great increase in the signal to noise ratio. This is very important for both a transmitter and a frequency synthesizer.

The first order modulator is no longer used in frequency synthesizers, because the higher order versions deliver far better noise performance. These higher order versions have been split into two different architectures, MASH and

MBSL. The MASH modulator will be discussed first.

3.2 MASH Architectures

The first major architecture type for a DSM is usually referred to as a MASH structure. MASH stands for multi-stage noise shaping. This was the first DSM architecture created. The idea being that the multiple stages will push any noise generated by the modulator to higher and higher frequencies. There are two reasons why this type of circuit is still very popular. The first is the simplicity of the circuit. Creating any version of this circuit only requires a few adders and registers. The second reason for its popularity is that this circuit is unconditionally stable at any order. These two factors put together make this circuit the choice of many engineers when building a fractional N PLL frequency synthesizer or transmitter.

3.2.1 MASH 1-1-1 and MASH 1-1-1-1

These are the simplest possible DSM architectures. The idea behind them is extremely simple. Take the first order DSM and cascade it until the appropriate order modulator is achieved. Figure 3.2 shows a good picture of the implementation of a third order circuit from [10].

Now reconsider the transfer equation that we created for the first order circuit that will be listed here again. This equation on its own is not special at all.

$$H[n] = K[n] + e[n] \times (1 - z^{-1}) \quad (12)$$

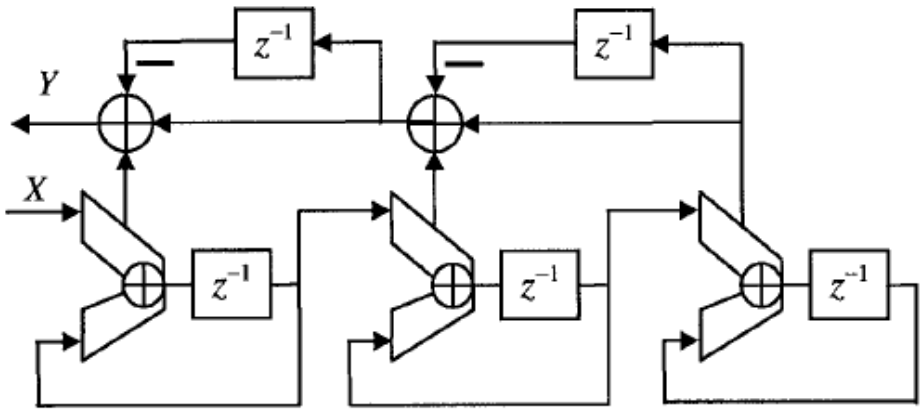

Figure 3.2: MASH 1-1-1 Digital Implementation from [10]

Now apply this equation to each stage of the modulator. These equations will end up looking very similar, but still nothing important.

$$H_1[n] = K[n] + e_1[n] \times (1 - z^{-1}) \quad (13)$$

$$H_2[n] = e_1[n] + e_2[n] \times (1 - z^{-1}) \quad (14)$$

$$H_3[n] = e_2[n] + e_3[n] \times (1 - z^{-1}) \quad (15)$$

The next step is to take this information and attempt to calculate the output.

$$Y[n] = H_1[n] + H_2[n] \times (-z^{-1}) + H_3[n] \times (1 - z^{-1})^2 \quad (16)$$

$$Y[n] = K[n] + e_3[n] \times (1 - z^{-1})^3 \quad (17)$$

The final result is extremely important. Every stage cancels the noise coming into it from the stage before. This result is very important and is why MASH modulators are still used today. The noise generated internally by the circuit is cancelled before ever entering the frequency synthesizer. The fourth order circuit

comes out to the same equations except instead of being multiplied by the transfer function to the third order; it is multiplied to the fourth order. There are two important things to remember about the different order circuits. The first thing is that the order of the modulator has to equal to or less than the order of the loop filter in the PLL. This is done to minimize the noise on the output from the total circuit. The second thing to remember is that the number of output bits for both of these circuits is equal to the order of the modulator. This is just something that has to be remembered when doing the initial design of the circuit. Another important point is that the equations for determining division ratios are the same as for pulse swallowing.

The fourth order circuit is essentially the same as the third order except for an additional accumulator and some more adders. A good example of a fourth order is seen below in Figure 3.3 from [11]

3.2.2 MASH 1-2 and MASH 2-1

These circuits are an attempt to create a hybrid between the MASH and MBSL modulators. The hope was to gain the stability of the MASH circuit and the lower noise of the MBSL and for the most part it was a success. The problem is that some inputs are unstable for this type. Usually about 25 % of the possible inputs are unstable. These occur at the farthest ranges of the frequency band.

Figure 3.4 and Figure 3.5 show the basic idea behind this circuit and come from [12].

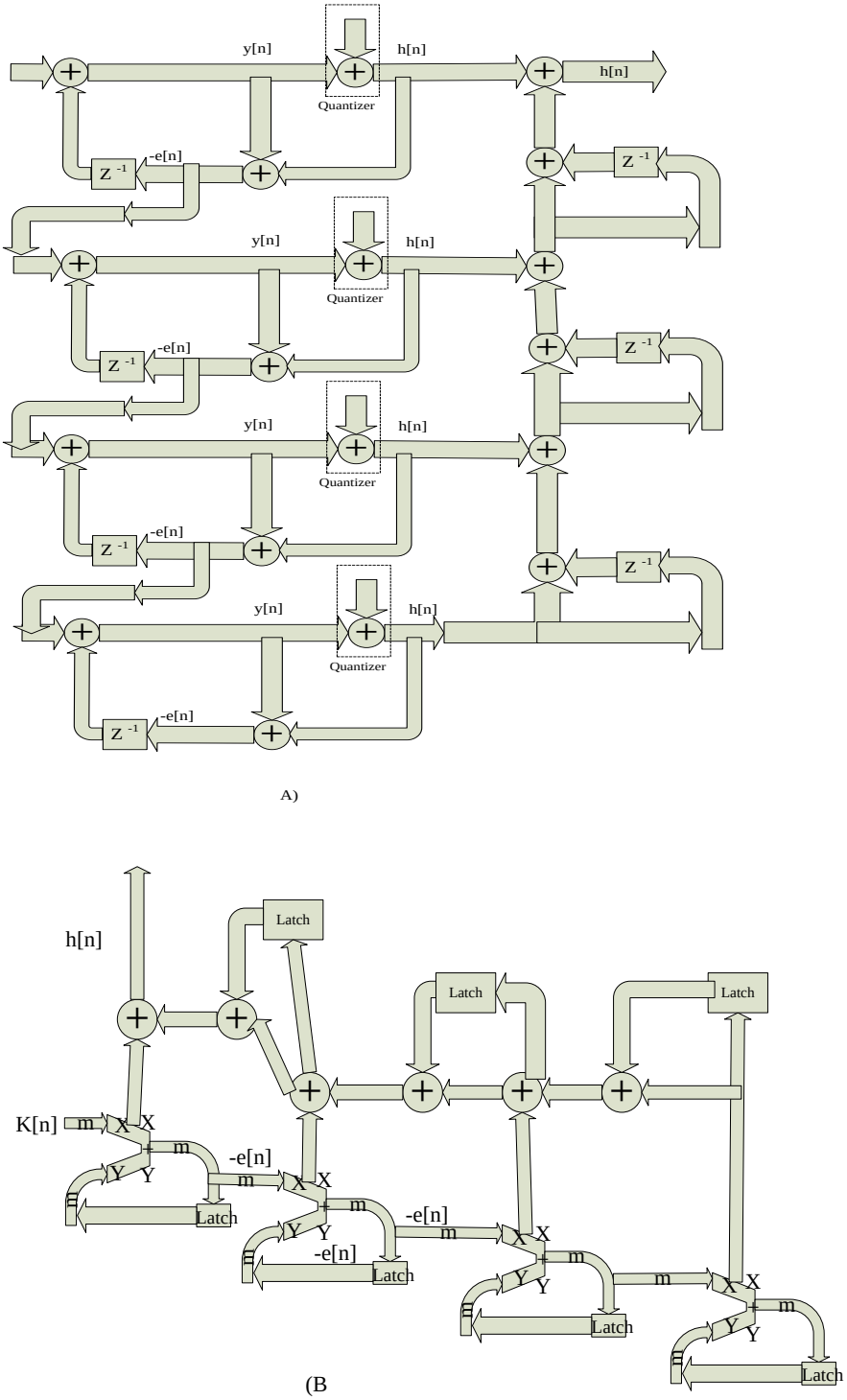

Figure 3.3: a) Block Diagram form and b) Digital Implementation from [11]

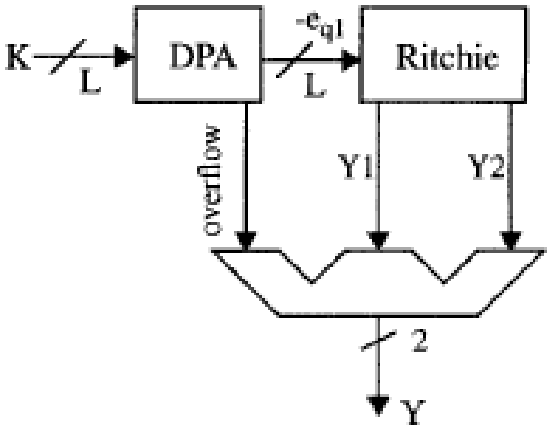

Figure 3.4: Basic diagram for a MASH 1-2 from [12]

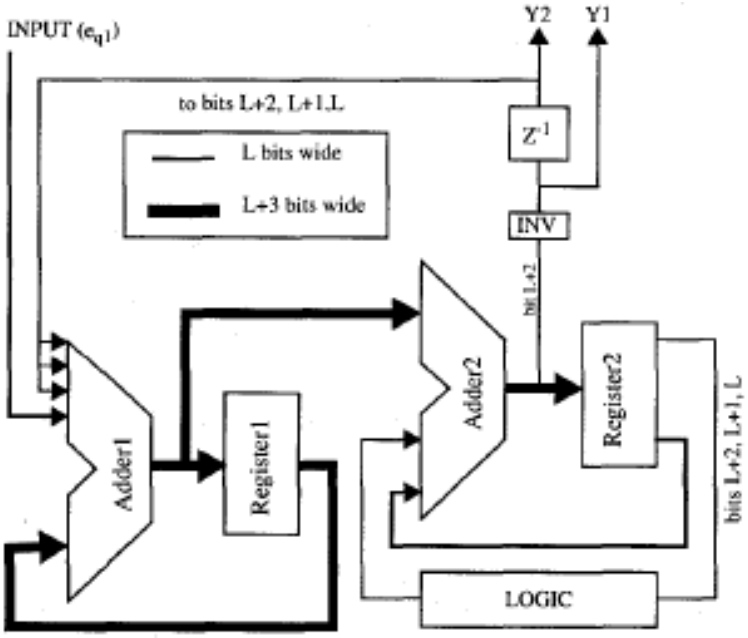

Figure 3.5: Ritchie circuit shown in figure 7 from [12]

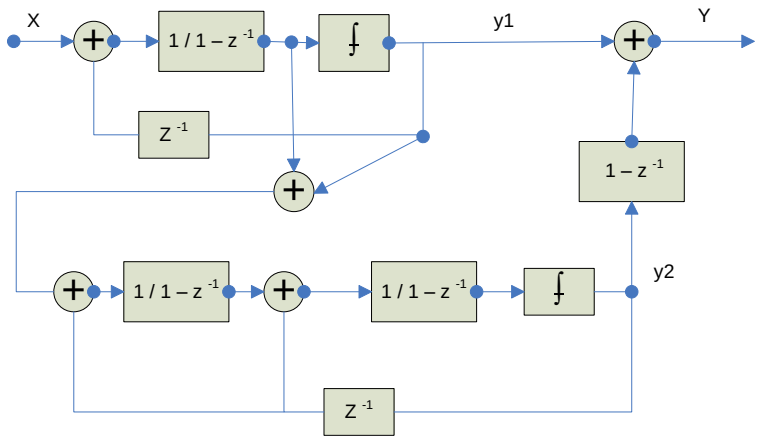

Figure 3.6: Theoretical model of MASH 1-2 SDM from [13]

This circuit is more complex than the MASH 1-1-1, but the ability to reduce the range of outputs makes it more like our other variation of modulators. This modulator seems to be a hybrid between the two types allowing for the unconditional stability of the MASH and the range of the MBSL.

Figure 3.6 shows the theoretical model for the MASH 1-2 from [13].

There is also a MASH 2-1 circuit, however this is merely the circuit shown here except reversed so that the Ritchie circuit is first. The MASH 2-1 reduces the output range, but not by as much as the MASH 1-2. Neither of these circuits appears to be very popular in literature. That is due to the fact that in engineering usually hybrid types of circuits are ignored. Normally design calls to take into account the advantages and disadvantages of differing designs and build circuits that will maximize the advantages and add circuits to lower the disadvantages accordingly.

3.3 MBSL Architectures

The other structure is called MBSL. This stands for mutli-bit single loop. The idea behind this one is instead of doing multiple stages like in the MASH, there is approximately one stage with several bits in the loop. At the same time, this will reduce the stability of the circuit. This circuit does not seem to have a single structure like the MASH models do. In that case, all of the different modulator types were built around cascading several accumulators. However in this case, it seems that the modulators of this type tend to congregate into one of two types, multiple feed forward and multiple feed back. Usually the main difference between them is the number of quantization levels needed for the output. The biggest difference between MASH and MBSL modulators is the output range. The MBSL does not have to jump around as much as the MASH model does to achieve similar fractional spur reduction. This is shown below in Figure 3.7 from [14].

The advantages of this circuit are several. The first is that the noise shaping is more flexible. In a circuit like this some noise proves to be beneficial. The second is that the noise is pushed to higher frequencies than the MASH is able to. This allows for better filtering. The third is that the output does not have to jump around as much as the MASH does to achieve similar fractional spur reduction. This lowers the overall noise of the circuit by reducing the use of the charge pump. The last is that MBSL does better on noise shaping for DC inputs than a MASH does. However, the circuit does have disadvantages. Unlike the

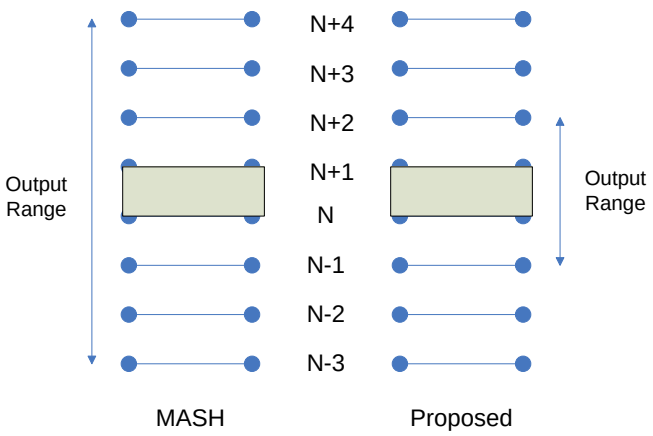

Figure 3.7: Comparison of MASH Output to MBSL Output from [14]

MASH versions the MBSL is not unconditionally stable. Also, the circuit is harder to implement than the MASH versions are.

3.3.1 Multiple Feed Forward Architecture

First, the architecture what is referred to as the multiple feed forward architecture is considered. As shown in Figure 3.8, from [15], the idea is to feed several values forward and multiply them by different amounts.

This figure shows two modulators that are fairly similar. There are a few minor differences, but not enough to classify these two as completely different. A major question is what is dither? Dithering is the process of adding random numbers to the signal in a DSM to make sure that it remains random for constant inputs. If the inputs are constant as in a frequency synthesizer eventually the modulator will have a repetition in the signal. The period of this repetition may be very long but it will be there. This will impact the noise performance. For

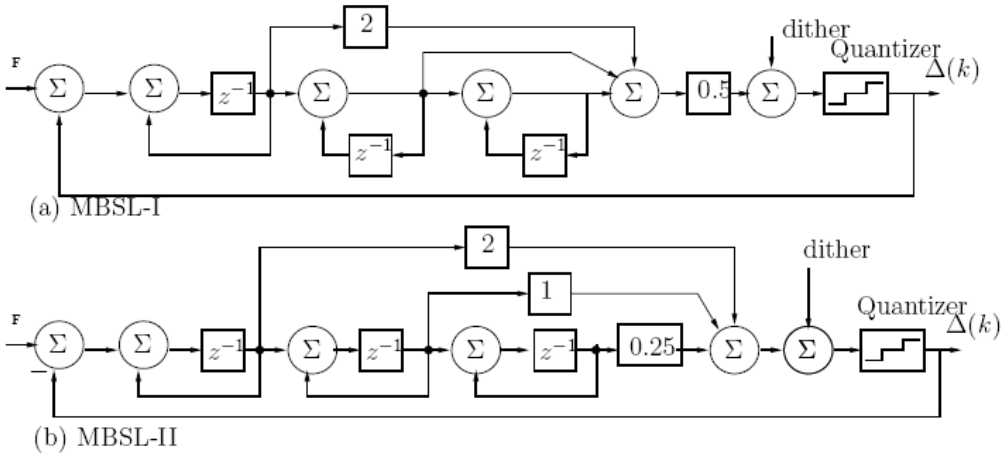

Figure 3.8: Both are examples of feed forward from [15]

any frequency synthesizer design where noise performance is important, dithering will have to be considered.

The major flaw with this type is that there are fewer stable input levels. This is due to feeding the signals forward. Feed back is used in many circuits to help stabilize them and works the same for a modulator.

3.3.2 Multiple Feed Back Architecture

The next type to consider is the multiple feed back architecture. The idea in this case is to have several feed back signals with different multipliers. This circuit in operation is very similar to the feed forward architecture; however the feed back mechanism adds a substantial amount of stability to the circuit. There is still a problem with this circuit and it is that it requires a quantizer with more levels that the feed forward architecture. Figure 3.9 and Figure 3.10 from [16] and [17] show good examples of this particular type of architecture.

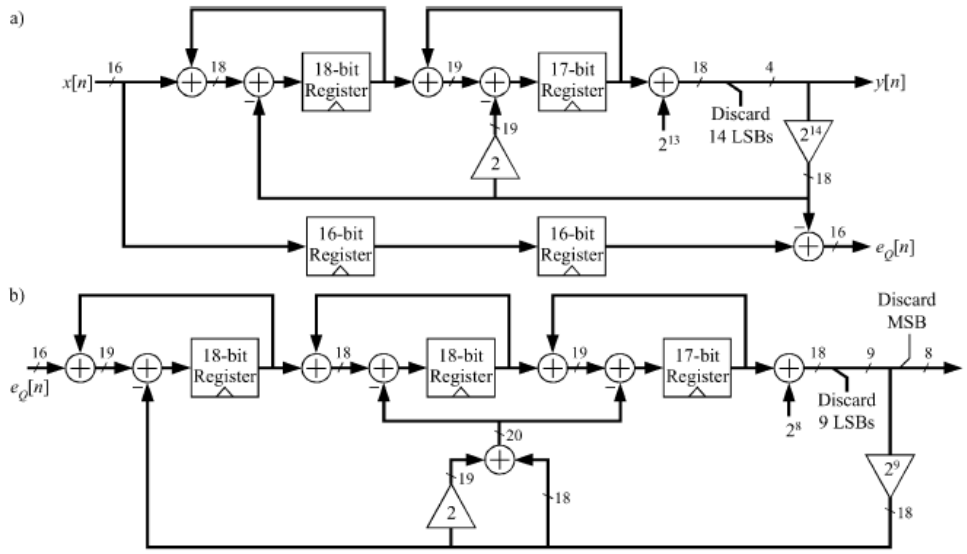

Figure 3.9: A second order and third order feed back from [16]

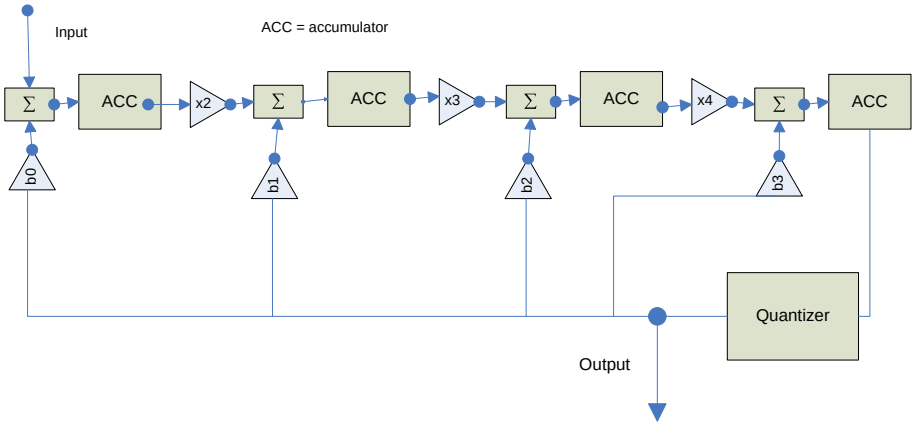

Figure 3.10: Fourth order feed back from [17]

This covers all of the types of DSM. Both major types, MASH and MBSL, have their advantages and disadvantages. MASH models are more stable and easier to implement. MBSL models have lower noise than the MASH models, but they are normally less stable and are much harder to implement. In this particular case, the MASH modulator was chosen because of its ease of implementation. Also, it works fairly well for a transmitter which will be covered in the next section.

3.4 GMSK Filter and Conversion to Transmitter

This section will deal with converting the fractional N frequency synthesizer into a transmitter. As stated before since the job of the Delta Sigma modulator is to constantly adjust the division ratio of the PLL, obviously it will be capable of doing some form of frequency modulation. The two traditional methods are to do either GMSK or FSK. Digital transmission methods are usually more power efficient and allow for the use of cheaper digital signal processing circuits. Figure 3.11 comes from [15] and shows a block diagram of the process.

As can be seen from the diagram, there is not much of a difference between the frequency synthesizer and the transmitter. Most of the effort is put into adjusting the input to the DSM. In order to work as a GMSK filter the data has to be collected and change it from a traditional digital signal where zero is equal to zero volts and one corresponds to five volts. This signal has to be changed to where zero equals negative five volts and one corresponds to five

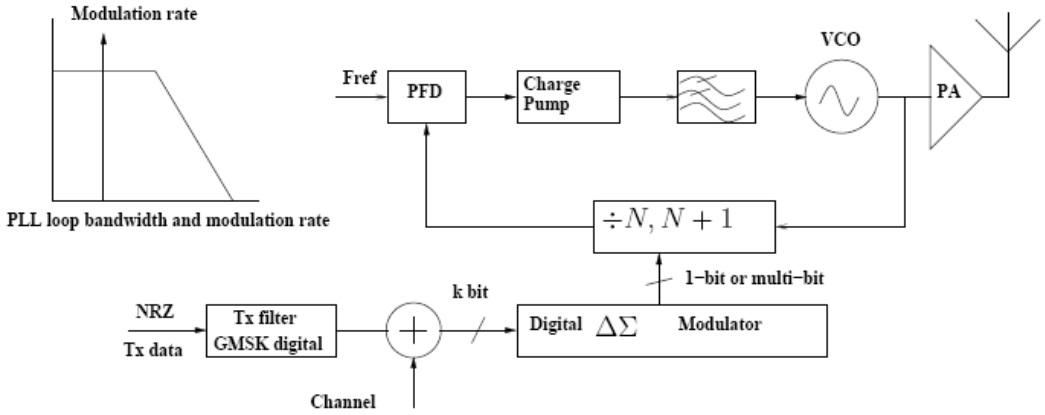

Figure 3.11: Block diagram of fractional n synthesizer from [15]

volts. This is the NRZ, nonreturn to zero, transmission data shown in the above figure. This data then goes through a Gaussian shaped FIR, finite impulse response, digital filter. An example of a filter of this type is shown below in Figure 3.12 from [18].

The output from this particular filter is added to a digital word that would give the correct division ratio to reach a certain frequency. This allows us to set the particular channel to broadcast on. This makes for a very easy way to create a transmitter. At this point everything has been covered to understand the design and the results. The next section will deal with the design and testing of the DSM and frequency synthesizer.

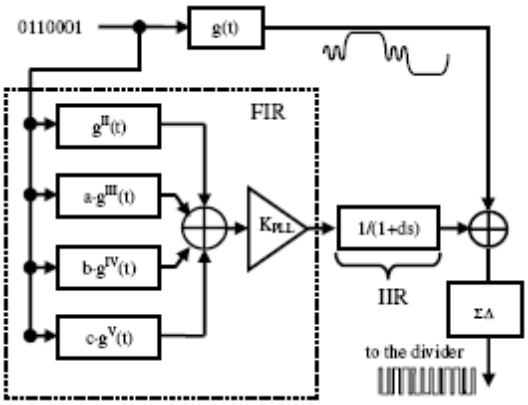

Figure 3.12: Design of a compensated Gaussian Filter from [18]

4 DESIGN OF THE DELTA SIGMA MODULATOR AND TESTING OF FRACTIONAL N FREQUENCY SYNTHESIZER

4.1 Design Choices

The design of the circuit and its operation is described in this chapter. The phase lock loop used for this work was designed and built in a previous work by Rajagopal Vijayaraghavan [19]. This circuit is shown in Figure 4.1 from [19].

This will be the phase locked loop necessary for this work, thus the design of the Delta Sigma modulator will be next. The first decision is the choice of

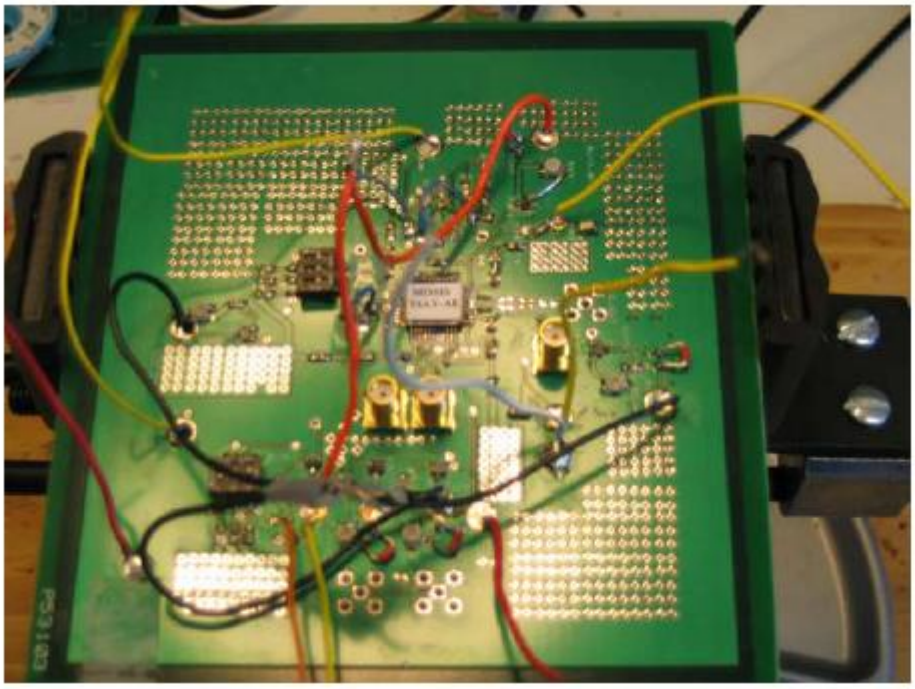

Figure 4.1: Picture of PLL on printed circuit board from [19]

architecture and the order of the modulator. For this work, the MASH architecture was chosen. The MBSL architecture would have provided better noise performance, but it was believed that the ease of design of the MASH architecture would make it easier to guarantee a working circuit. The chosen order for the DSM is three. This is because the input to the divider for the PLL is only three bits. In the MASH architecture style the order is equal to the number of outputs bits from the circuit.

The next stage of the design is to figure out the number of input bits to the DSM. This number is extremely important. To choose this number the equations from [6] that were shown above will be copied here.

$$N_{frac} = N + \frac{K}{2^k} = N + n \qquad (18)$$

$$f_{out} = f_{ref} \times (N + n) \qquad (19)$$

$$K = n \times 2^k \qquad (20)$$

The first equation shows that the division ratio is made of two separate parts, an integer part and a fractional part. The fractional part is defined by the value of the input divided by two with an exponent equal to the number of input bits. Equations 19 and 20 will be essential for selecting the division ratio of the circuit. This will require one more equation to finish the job and it comes from [13].

$$\Delta f = \left(\frac{1}{2^k} \right) \times f_{ref} \qquad (21)$$

The equation listed here has been modified slightly to make it easier to read. The variable used in the paper was M, but was replaced with the equivalent value from the equations listed before, 2^k and Δf is the frequency resolution. This equation must be solved first to finish the design of the modulator, because this will set the number of bits that the modulator will require. F_{ref}, the reference frequency, comes from a crystal oscillator that oscillates at 50 MHz. The next question is the frequency resolution. If the Bluetooth standards are used, the band between 2.4 GHz and 2.4835 GHz is broken into eighty channels. Therefore if the upper limit of the band is subtracted from the lower limit of the band and divided by the number of channels, the frequency resolution should come out to be 1.04375 MHz. The next step is to rearrange the equation above into a form that will give us the k value.

$$k = \frac{\log\left(\frac{f_{ref}}{\Delta f}\right)}{\log 2} \qquad (22)$$

This equation will give the number of bits that is needed for the circuit. When the numbers mentioned above are all plugged in, the k value equals 5.58 bits which will have to be rounded up to six bits. Now the thing is that the number of bits used for the input of the DSM is 24. The original idea was to put a 60 GHz oscillator for fref and the calculations using that value came out to 16 bits for the input word. The extra eight bits were to be used to increase the resolution. The idea behind the design was since it would be implemented on an FPGA board anyway, the extra bits could be added without any added difficulty in design.

Also, if this circuit is used as a frequency transmitter we better resolution will be needed to allow for broadcasting in the channel. The frequency resolution for a transmitter would be significantly less than for the synthesizer. The value of our input word K in terms of f_{out} can be determined from the following equation.

$$f_{out} = f_{ref} \times (N + n) \quad (23)$$

$$K = n \times 2^{k} \quad (24)$$

Using these two equations it is simply a matter of plugging the numbers in and solving. This will have different answers according to what channel is chosen and will have to be converted to a digital base. The calculations would add nothing to this discussion but are helpful when performing the tests. The last question is dithering. For this design, it was decided to not attempt dithering. This process requires complicated circuits for a reasonable gain in noise. However it was believed that the noise performance was second to making a reliable circuit. Therefore, dithering was ignored for this particular work. So, now the equations for the DSM have been solved and the architecture selected which is shown below from [10].

The next step is to implement the design. The choices are whether to use an integrated circuit or to use a field programmable gate array (FPGA). For this work, an FPGA was used. This allowed for easier implementation. Most of the literature chose the same path for the implementation because of time constraints. The fact of the matter is that the creation of this as a digital circuit on chip would require a few weeks whereas the construction of it in VHDL can be accomplished in a matter of hours. The appendix of this work has all of the code implemented

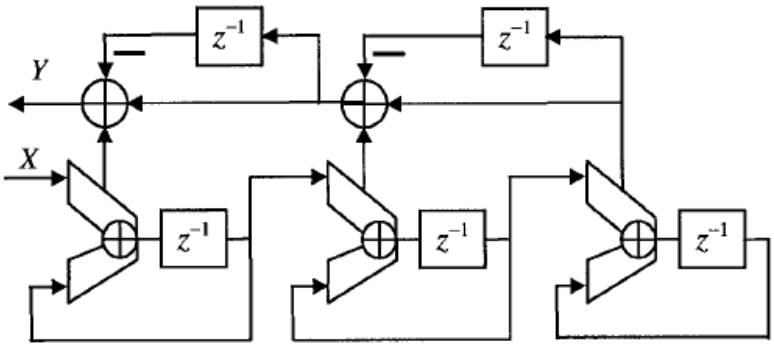

Figure 4.2: Selected Architecture from [10]

for the circuit. Here it will be given a brief overview. DSM is the highest level of the code and implements the circuit seen above in Figure 4.2. The code consists of three twenty-four bit accumulators with three registers of equal size. Then, it sets the size of the carry bits to the same size as the output to make the calculations easier. Next, it performs the same operations seen at the top of the above diagram to create the fractional part of the output. Finally, it adds the fractional part to the integer division ratio which is not shown in the Figure above. The accumulators are generated in accum with accum_package being the file that allows the lower circuits to be called in a similar fashion as C functions. The accumulator code implements a generic function which allows it to be called and then given a size dependent on the application. The adders and register are designed the same way. Adderspecial is designed just to implement the two adders near the output of the adder and were not designed with the generic function. Parts of the code came from [20] which made the design considerably faster. The next step is to see the simulation results.

4.2 Simulation Results

For the simulation results, the early simulations were done in MATLAB showing how the different modulator's output behaves. These outputs will be posted here to give a better understanding of the difference between a third order MASH modulator and a MBSL as far as the output is concerned. This is shown below in Figure 4.3 and 4.4. Both histograms are using a 2.44 GHz output with a 50 MHz input. It is obvious to see the differences between them. The MASH requires the use of more output states than the MBSL does. This fact will be important later in this paper.

The next step was to generate the simulations from the VHDL code. For this case, only the post synthesis simulation is used. The idea being that with the MATLAB simulations the pre synthesis simulations would be equivalent. Pre-synthesis simulations merely implement the code that was typed in. The post-synthesis simulations will be proof that the circuit functions as it should. Post-synthesis adds in the assorted delays and issues that the real board would deal with. Thus, it is a great simulation for checking the validity of the circuit. The results from this simulation are shown below in Figure 4.5.

Figure 4.5 shows that the circuit is working perfectly. The output from the DSM is jumping around as it should and staying in the appropriate areas for the right amount of time. Also, it can be seen that the circuit seems to spend longer times on the value of two which is the N part of the division ratio. This is expected and not something to worry about. The next step is to describe the test

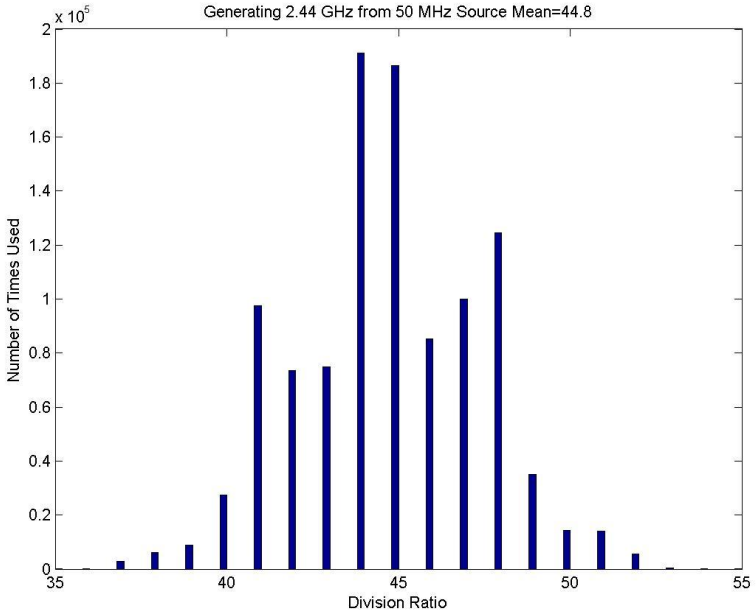

Figure 4.3: Third order MASH modulator output

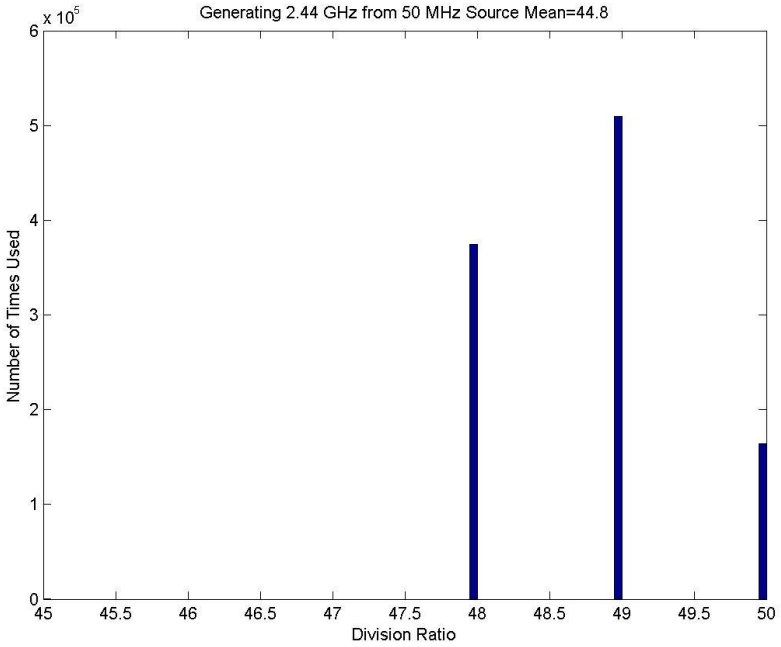

Figure 4.4: Third order MBSL modulator output

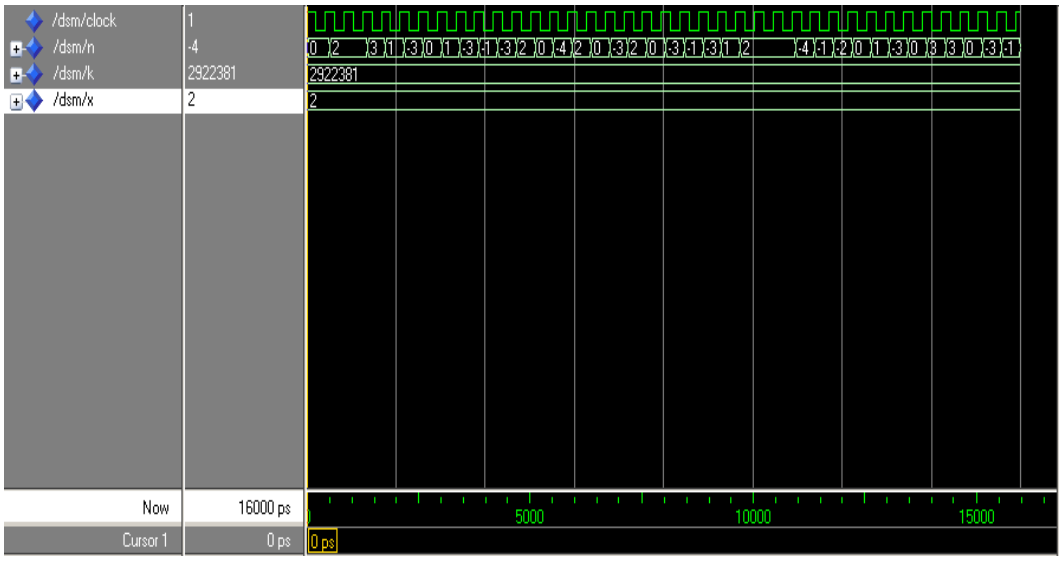

Figure 4.5: Post synthesis test of DSM

setup and check the actual circuit. The test setup will describe the different settings and equipment required to test the circuit.

4.3 Test Setup

The testing in this work is the creation of a prototype fractional N frequency synthesizer. This can be accomplished by combining the phase locked loop created in [19] with the implemented Delta Sigma modulator created in the design section of this work. A block diagram of the prototype frequency synthesizer is shown below in Figure 4.6.

This block diagram shows a fairly standard setup for a fractional n phase locked loop frequency synthesizer. The different blocks are as follows: PFD stands for phase frequency detector. VCO is the voltage controlled oscillator. This circuit actually uses two separate dividers, an ILFD which stands for

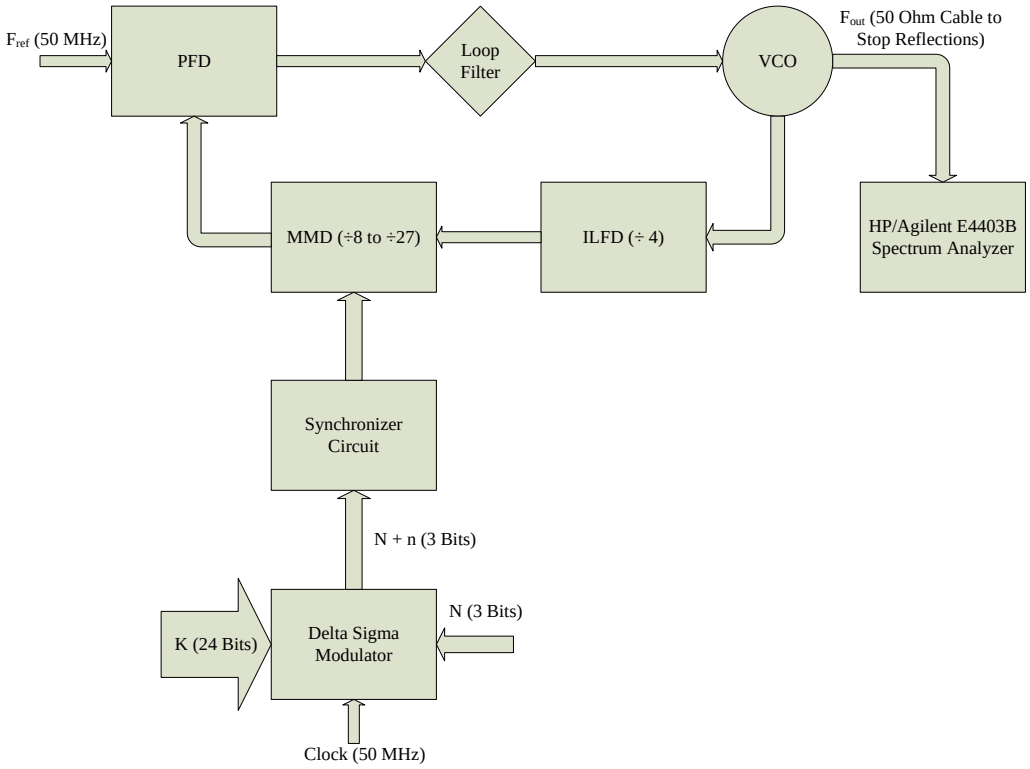

Figure 4.6: Block Diagram of Test Setup

injection locked frequency divider, and an MMD which stands for multi-modulus divider. The MMD is where the Delta Sigma modulator will affect the circuit by changing the division ratio. This particular divider is made up of three cascaded dividers that can change between dividing by two and dividing by three. When this is combined with the ILFD the division ratio will be between 32 and 108. The spectrum analyzer used is the HP/Agilent E4403B Spectrum Analyzer. This particular analyzer has a 50 ohm input and has a frequency range from 9 kHz to 3 GHz. This particular model will work fine for testing, because it will cover the entire 2.4 GHz range. The last piece that has not been discussed is the

synchronizer circuit shown below in Figure 4.7 from [19].

The reason for this circuit is to lower the noise that the Delta Sigma modulator creates. This circuit will make sure that the different output bits from the Delta Sigma modulator appear at the divider at the same time. Without this circuit the charge pump would be activated more often which would increase the noise at the output. Considering the MASH architecture shown in this work will prove the necessity of having this particular circuit in the system. If the first stage output and the third stage output both have to change, the third stage will take longer because of the delays inherent in the circuit. This will create an output state that will be wrong for a very short amount of time, but it would be enough to cause the charge pump to activate and then reactivate to change from the error state. This will surely increase the noise of the circuit.

Another thing to consider is where the test was performed. As stated before, this circuit is going to operate at the 2.4 GHz range. This same frequency band is also used by cell phones and wireless internet access points. A possible problem will arise by performing these tests on a college campus, because both of these sources of noise affecting the circuit will be highly possible. The ideal way to perform these tests would be to make sure that the circuit was shielded from any outside noise. The best way to accomplish this would be to either place the boards into a grounded aluminum box or to do the testing in a Faraday cage. These two methods would work the same by preventing any electrical fields to reach the testing devices. The test setup actually managed some shielding due to

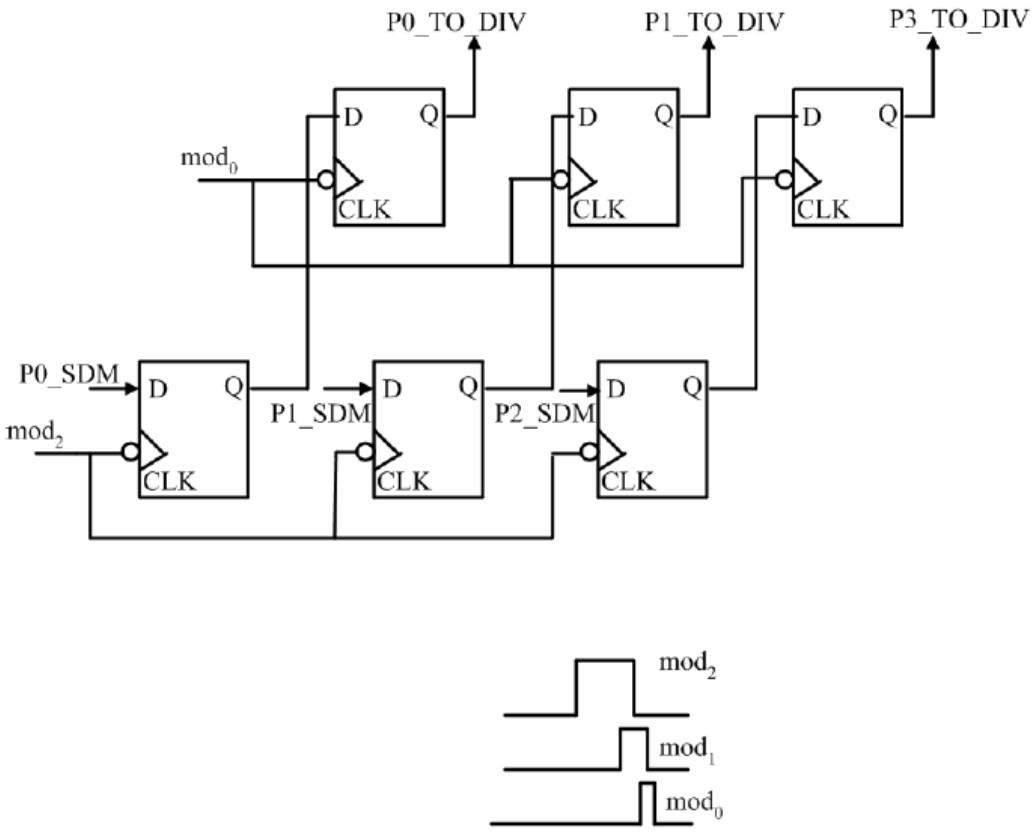

Figure 4.7: Synchronizer circuit on board from [19]

the use of a 50 ohm coaxial cable that connected the output of the phase locked loop to the spectrum analyzer. The testing will show if this is sufficient or if further shielding would have been preferred.

4.4 Test Results

With the architecture of this circuit, the testing required two parts. The first part was to make sure the ILFD achieved its correct locking range. If this was not done the circuit would not function correctly and required a precise setup of the phase locked loop. The next set of tests was to apply inputs to the DSM and check what the output frequency is. This test is the more important one, because it proves the function of the fractional N phase locked loop frequency synthesizer.

The first test required the setup and biasing of just the phase locked loop. This was connected to the spectrum analyzer and a crystal oscillator from ECS Inc. was used to provide a 50 MHz reference frequency. The purpose of this test was to calibrate the loop. The designer of the loop included an extra set of switches to allow for the VCO and the ILFD to be calibrated for the low end of the spectrum. Thus, when the output frequency is equal to 2.4GHz the ILFD is locked and the phase locked loop is in perfect operation. There is another purpose to this test. This will show how much interference is being caused by nearby broadcasts. In the room that the testing was being done, a campus wide WLAN was also broadcasting with sufficient strength to allow for laptops to connect to it at full strength. The output will show if this is an issue or not. The output can be seen below in Figure 4.8.

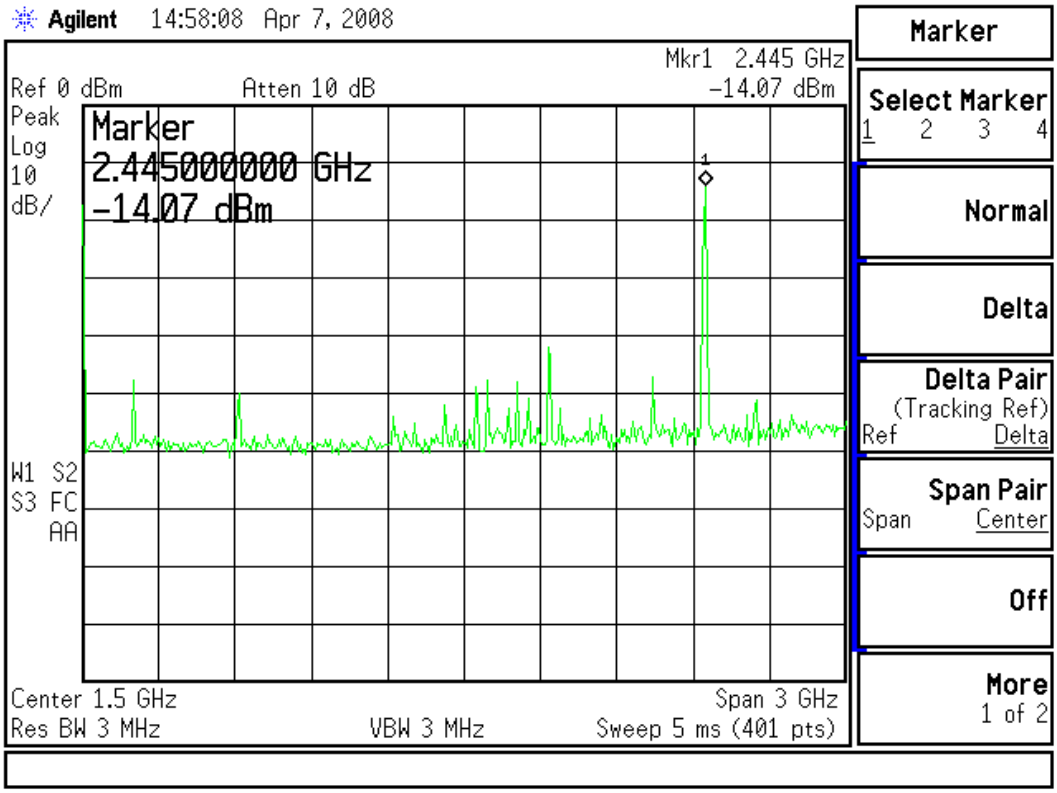

Figure 4.8: Initial Result from phase locked loop with no inputs

The important thing is to cover what these results mean to the design. The first factor is the level that the marker is at -14.07dBm. This result is without the implementation of a power amplifier or antenna, thus is actually fairly high for a circuit without either of these. It proves that the circuit is capable of transmitting, but only over a short distance, something on the order of only a couple of feet. The next factor is the frequency of the spike. In this case, the frequency is 2.445 GHz where the expectation was to achieve only 2.4 GHz. This shows that when the loop is set up as a frequency synthesizer that the final output frequency is going to be off by some percent, for the initial case 1.875 %. This result is actually the best that was achieved over at least twenty different tests. Unfortunately, this will affect all of the other tests using this circuit, but it is unavoidable. The last thing to mention is the spectrum itself. As can be seen in the figure, there is another spike at roughly zero frequency. This suggests that the output will have a DC offset. There is also a cluster of noise around the midpoint of the spectrum at 1.5 GHz to about 2 GHz. It seems like this is noise being picked up by the circuit, however the only two frequencies it works at are 50 MHz and the output frequency. These are too low and too high respectively to be the source of this noise. Also, this is too high of a frequency to be coming from local radio stations which is a very common source of noise like this. The cause of this burst of noise is unknown at this time, but could be determined with more research. However, the most important part of the spectrum is the frequencies around the output. In the 2.4 GHz region, it can be seen that there is very little

interference from the outside sources. It seems like the coaxial cable and transmission lines on the board are shielded enough to minimize the effect. This is good for the results, because the effect of the outside noise can be diminished. It probably is still having an effect, but these results show it to be minimal at best. These results proved that the phase locked loop is working, so the next results will see if the prototype fractional n frequency synthesizer will be effective.

The next sets of tests are to prove the function of the fractional N frequency synthesizer. For this work, the frequencies of 2.44 GHz, 2.48GHz, and 2.46 GHz were chosen to be generated by the overall design. Calculations were done to generate the inputs into the DSM. The clock for the DSM was originally going to come from the crystal oscillator serving as the input into the loop; however its voltage swing was too low to make the board function correctly. Therefore, the crystal oscillator built into the Spartan-3 test board was used as the input clock. Figure 4.9 shows the result of the 2.44 GHz test.

These results will require some discussion to make sense. The first thing to notice is the increase in noise around the spike created by the circuit at 2.415 GHz. This extra noise appears only when the DSM is attached to the circuit and turned on. Therefore, it has to be caused by the charge pump working as the DSM changes the division ratio. This can be proven by looking at the results for just the phase locked loop above in Figure 4.9. It does not show any of this noise and these results were obtained with all of the input wires attached to the phase locked loop board. Thus if any antennas would have been created similar results

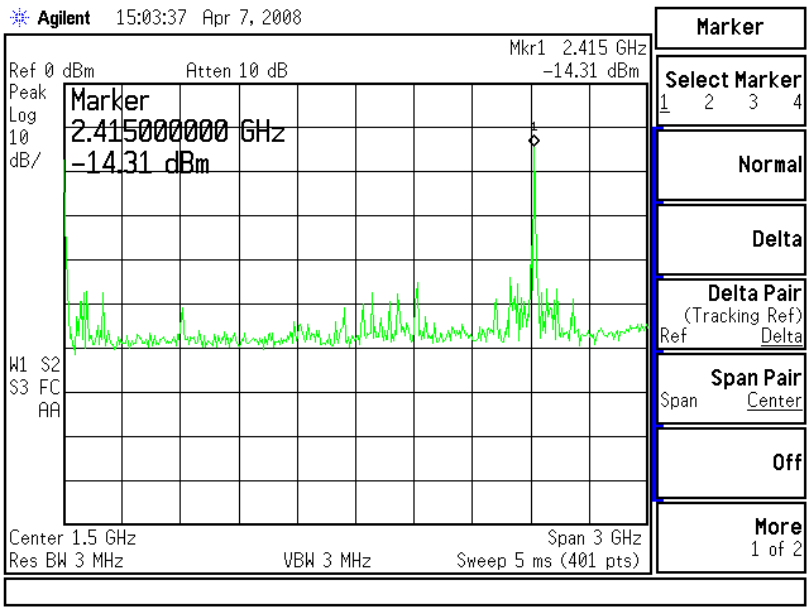

Figure 4.9: Attempt to Generate 2.44 GHz

would be seen on both pictures. The next thing that is noticed from Figure 4.10 is that the output frequency is off. The percentage that it is off is 1.0352 %. The percent off is a little better than the original case, but it can be seen that the original loop is having an effect. The next attempt will be to generate 2.48 GHz which is shown in Figure 4.10.

Figure 4.10 shows a similar spectrum to Figure 4.9. Again it can be seen that the DSM is adding noise around the output frequency, however in this case the output is a little closer this time. The percent error for this case is 0.121 %. This is extremely good considering that the original error was around one percent. This shows an improvement over the last case. Figures 4.11 and 4.12 show the last test. It was noticed when wiring up the DSM for the 2.46 GHz test that the

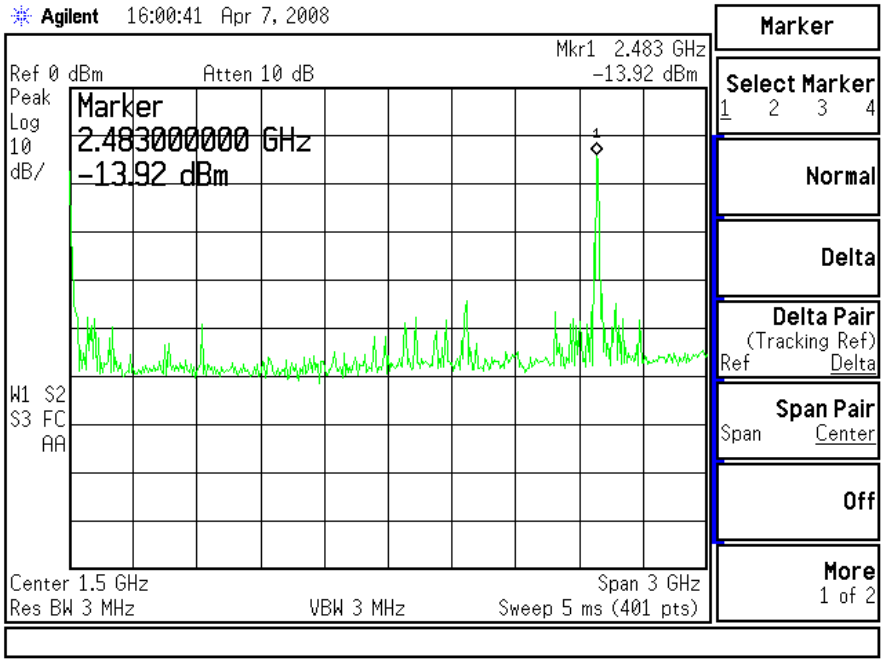

Figure 4.10: Attempt to generate 2.48 GHz

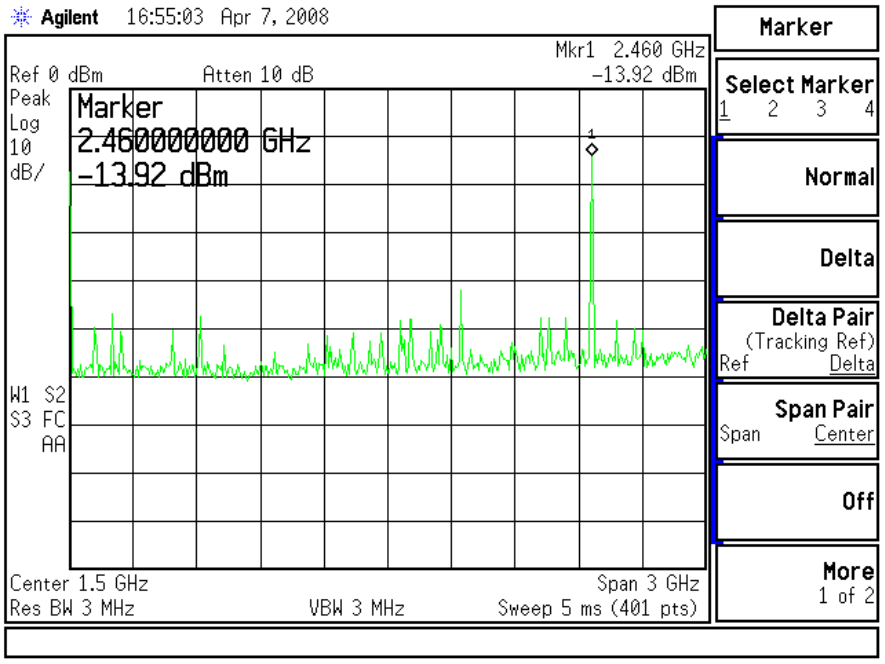

Figure 4.11: Output from PLL before attempting to generate 2.46 GHz

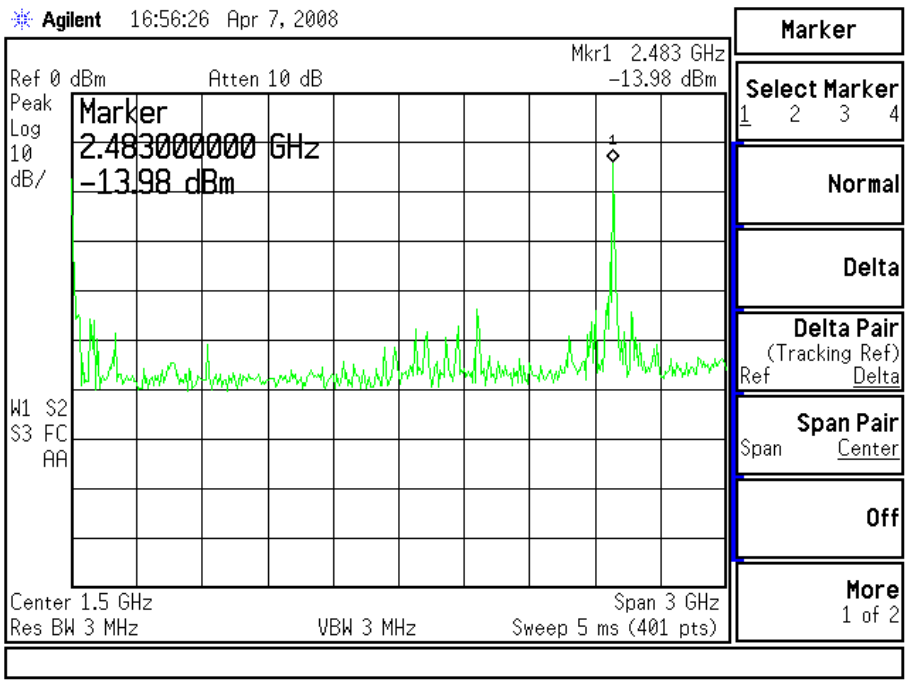

Figure 4.12: DSM connected to PLL to generate 2.46 GHz

initial frequency had shifted from where it was previously. Thus, Figure 4.11 shows the result for just the PLL biased up and with no input from the DSM and Figure 4.12 shows the operation of the entire synthesizer. These results are off as well. The percent error for this case equals 0.935 %. The next section will deal with the conclusions from this work and future work to attempt.

5 CONCLUSION AND FUTURE WORK

5.1 Conclusions and Achievements

In any work eventually the question is asked, is it successful? This work was a complete success. Consider the percent errors calculated in the last section. The phase locked loop by itself was off by 1.8 %. The three cases attempted were each off by less than one percent. This shows an improvement over the original results. A one percent tolerance is acceptable in many applications and would be similar here. When the results from [19] are also considered, it can be seen that the results were had a percent error of roughly two percent for the integer N synthesizer. Therefore, the inclusion of the Delta Sigma modulator was a success.

This work implemented the design of a Delta Sigma modulator. It covered the different architectures available and discerned their strengths and weaknesses. Then, it used the equations available to completely design a version using the MASH architecture. The next step was the implementation and testing of the circuit which were successful. This is a remarkable list of achievements for this work.

5.2 Future Work

The main step for future work is to attempt the transmitter possible from this circuit. Much of the background research for this job is in this paper. Merely the implementation of a digital filter would be left. Another possible job would be to see if there is an appreciable difference between MBSL and MASH

modulators for a transmitter. None of the research found for this paper seemed to consider that the two different modulators might have different characteristics for a transmitter. These would be two major possibilities for continuing this work, however there may be many more.

REFERENCES

[1] “Demand for use of 2.4 GHz”, www.aegis-systems.co.uk/download/ISM2.pdf

[2] “Basics of Dual Fractional-N Synthesizers/PLLs”, www.skyworksinc.com/products_display_item.asp?did=4725.

[3] “Fractional/Integer-N PLL Basics”, focus.ti.com/lit/an/swra029/swra029.pdf

[4] Terrence P. Kenny, Thomas A. D. Riley, Norman M. Filiol, and Miles A. Copeland, “Design and Realization of a Digital ΔΣ Modulator for Fractional-*n* Frequency Synthesis”, *IEEE Transactions on Vehicular Technology,* Vol. 48, pp. 1453-1460, No. 2, March 1999.

[5] Tom A.D. Riley, Miles A. Copeland, Tad A Kwasniewski, “Delta-Sigma Modulation in Fractional-*N* Frequency Synthesis”, *IEEE Journal of Solid-State Circuits,* Vol. 28 pp.553-559, No.5, May 1993

[6] B. D. Muer and M. Steyart, “CMOS Fractional-N Synthesizers, Design for High Spectral Purity and Monolithic Integration”, Kluwer Academic Publishers, 2003, ISBN 1-4020-7387-9.

[7] Gary Appel, “Fractional N Synthesizers”, http://rfdesign.com/images/archive/1100Appel34.pdf

[8] R. Jacob Baker, “CMOS Circuit Design, Layout, and Simulation”, IEEE Press, 2002, ISBN 0-471-22754-4

[9] Bertan Bakkaloglu, Sayfe Kiaei, Bikram Chaudhuri, Delta-sigma frequency synthesizers for wireless applications, Computer Standards & Interfaces 29 (2007) 19 – 30

[10] ∑Δ FRACTIONAL-N PLL SYNTHESIZER

[11] Miicahit Kozak, Izzet Kale, Assaad Borjak, Taoufik Bourdi, “A Pipelined All-Digital Delta-Sigma Modulator for Fractional-N Frequency Synthesis”,

[12] Lizhong Sun, Thieny Lepley", Franck Nozahic**, Arnaud Bellissant**, Tad Kwasniewski and Bany Heim, “Reduced Complexity, High Performance Digital Delta-Sigma Modulator for Fractional-N Frequency Synthesis”

[13] Keliu Shu, Edgar Sanchez-Sinencio, Franco Maloberti, and Udaykiran Eduri, “A Comparative Study of Digital $\sum\Delta$ Modulators for Fractional-N Synthesis”, pp1391-1394

[14] Woogeun Rhee, Bang-Sup Song, and Akbar Ali, “A 1.1-GHz CMOS Fractional-*N* Frequency Synthesizer with a 3-b Third-Order $\Delta\sum$ Modulator”, *IEEE Journal of Solid-State Circuits*, Vol. 35, pp.510-521, No. 10, October 2000.

[15] H. Arora, “Design of 5-Mb/s Fractional-N RF Transimtter in 900 MHz ISM band using GMSK Data Modulation Techniques”, Ph.D Dissertation, Duke Univ, 2005.

[16] Sudhakar Pamarti,Lars Jansson,and Ian Galton, “AWideband 2.4-GHz Delta-Sigma Fractional-N PLL With 1-Mb/s In-Loop Modulation*”, IEEE Journal of Solid-State Circuits*, Vol. 39, No. 1, January 2004

[17] Han-il Lee, Je-Kwang Cho, Kun-Seok Lee, In-Chul Hwang, Tae-Won Ahn, Kyung-Suc Nah, and Byeong-Ha Park, “A Σ-Δ Fractional-N Frequency Synthesizer using a wideband integrated VCO and a fast AFC technique for GSM/GPRS/WCDMA Applications”

[18] Nicola Lofù , Gianfranco Avitabile, Biagio Bisanti, Stefano Cipriani, “Direct Sigma Delta GMSK Modulator Modeling and Design for 2.5G TX Applications”, ISCAS 2004 IV pp. 193-196

[19] Rajagopal Vijayaraghavan, “Process and Temperature Compensated Wideband Injection Locked Frequency Dividers and their Application to Low-power 2.4-GHz Frequency Synthesizers”, Ph.D Dissertation, University of Tennessee – Knoxville, 2007

[20] Stephen Brown, Zvonko Vranesic, “Fundamentals of Digital Logic With VHDL Design”, McGraw Hill, 2000, ISBN 0-07-012591-0

APPENDIX

Accumulator Code

```
LIBRARY ieee;
    USE ieee.std_logic_1164.all;
    USE work.components.all;

    ENTITY accum IS
        PORT( Clock  : IN STD_LOGIC;
              X      : IN STD_LOGIC_VECTOR(23 DOWNTO 0);
              Result : OUT STD_LOGIC_VECTOR(23 DOWNTO 0);
              Cout   : OUT STD_LOGIC );
          END accum;

          ARCHITECTURE Structure OF accum IS
              SIGNAL Sum, Test : STD_LOGIC_VECTOR(23 DOWNTO 0);
              SIGNAL Zero_bit : STD_LOGIC;
            BEGIN
              Zero_bit <= '0';
              adder: addern
                 PORT MAP(Zero_bit, X, Test, Sum, Cout);
              reg: regne
                 PORT MAP(Sum, Clock, Test);
               Result <= Test;
            END Structure;
```

Accumulator Package Code

```
LIBRARY ieee;
    USE ieee.std_logic_1164.all;

    PACKAGE accum_package IS
        COMPONENT accum
        PORT( Clock  : IN STD_LOGIC;
              X      : IN STD_LOGIC_VECTOR(23 DOWNTO 0);
              Result : OUT STD_LOGIC_VECTOR(23 DOWNTO 0);
              Cout   : OUT STD_LOGIC );
          END COMPONENT;
          END accum_package;
```

Addern Code

```
LIBRARY ieee;
    USE ieee.std_logic_1164.all;
    USE work.fulladd_package.all;

    ENTITY addern IS
        PORT (Cin  :IN  STD_LOGIC;
              X,Y  :IN  STD_LOGIC_VECTOR(23 DOWNTO 0);
              S    :OUT STD_LOGIC_VECTOR(23 DOWNTO 0);
              Cout :OUT STD_LOGIC);
    END addern;

    ARCHITECTURE Structure of addern IS
        SIGNAL C : STD_LOGIC_VECTOR(0 TO 24);
        BEGIN
            C(0) <= Cin;
            Generate_label:
            FOR i IN 0 TO 23 GENERATE
            stage: fulladd PORT MAP ( C(i), X(i), Y(i), S(i),
C(i+1) );
        END GENERATE;
        Cout <= C(24);
    END Structure;
```

Adderspecial Code

```
LIBRARY ieee;
    USE ieee.STD_LOGIC_1164.all;
    USE ieee.std_logic_signed.all;

    ENTITY adderspecial IS
        PORT( x, y, z  :IN  STD_LOGIC_VECTOR(2 DOWNTO 0);
              s      :OUT STD_LOGIC_VECTOR(2 DOWNTO 0) );
          END adderspecial;

    ARCHITECTURE Structure OF adderspecial IS
        SIGNAL Sum : STD_LOGIC_VECTOR(3 DOWNTO 0);
        BEGIN
            Sum <= x + y - z;
            S <= SUM(2 DOWNTO 0);
        END Structure;
```

Adderspecial Package Code

```
LIBRARY ieee;
    USE ieee.std_logic_1164.all;
    USE ieee.std_logic_signed.all;

    PACKAGE adderspecial_package IS
        COMPONENT adderspecial
            PORT( x, y, z :IN  STD_LOGIC_VECTOR(2 DOWNTO 0);
                  s       :OUT STD_LOGIC_VECTOR(2 DOWNTO 0) );
              END COMPONENT;
          END adderspecial_package;
```

Components Package Code suggested by [20]

```
LIBRARY ieee;
    USE ieee.std_logic_1164.all;

    PACKAGE components IS

        COMPONENT addern -- 24-bit adder
            PORT ( Cin  : IN STD_LOGIC;
                   X,Y  : IN STD_LOGIC_VECTOR(23 DOWNTO 0);
                   S    : OUT STD_LOGIC_VECTOR(23 DOWNTO 0);
                   Cout : OUT STD_LOGIC);
        END COMPONENT;

        COMPONENT regne --24-bit register
            PORT ( D     : IN STD_LOGIC_VECTOR(23 DOWNTO 0);
                   Clock : IN STD_LOGIC;
                   Q     : OUT STD_LOGIC_VECTOR(23 DOWNTO 0));
        END COMPONENT;

        COMPONENT regs --3-bit register
            PORT ( D     : IN STD_LOGIC_VECTOR(2 DOWNTO 0);
                   Clock : IN STD_LOGIC;
                   Q     : OUT STD_LOGIC_VECTOR(2 DOWNTO 0));
       END COMPONENT;
    END components;
```

DSM Code

```
LIBRARY ieee;
    USE ieee.STD_LOGIC_1164.all;
    USE ieee.std_logic_signed.all;
    USE work.components.all;
    USE work.fulladd_package.all;
    USE work.adderspecial_package.all;
    USE work.accum_package.all;

    ENTITY DSM IS
        PORT ( k     : IN  STD_LOGIC_VECTOR(23 DOWNTO 0);
               Clock : IN STD_LOGIC;
               X     : IN STD_LOGIC_VECTOR (2 DOWNTO 0);
               n     : OUT STD_LOGIC_VECTOR(2 DOWNTO 0) );
       END DSM;

    ARCHITECTURE Structure OF DSM IS
        SIGNAL Sum1, Sum2, Sum3 : STD_LOGIC_VECTOR(23 DOWNTO 0);
        SIGNAL Co1, Co2, Co3 : STD_LOGIC;
        SIGNAL Cout1, Cout2, Cout3 : STD_LOGIC_VECTOR(2 DOWNTO
0);
        SIGNAL DCout3, DCout34, Dcout : STD_LOGIC_VECTOR(2 DOWNTO
0);
        SIGNAL Cout34, final       : STD_LOGIC_VECTOR(2 DOWNTO
0);

        BEGIN
            accum1: accum
            PORT MAP (Clock, k, Sum1, Co1);
            accum2: accum
            PORT MAP (Clock, Sum1, Sum2, Co2);
            accum3: accum
            PORT MAP (Clock, Sum2, Sum3, Co3);
            Cout1(2) <= '0';
            Cout1(1) <= '0';
            Cout1(0) <= Co1;
            Cout2(2) <= '0';
            Cout2(1) <= '0';
            Cout2(0) <= Co2;
            Cout3(2) <= '0';
            Cout3(1) <= '0';
            Cout3(0) <= Co3;
        reg1:regs
        PORT MAP (Cout3, Clock, DCout3);
        adder1: adderspecial
        PORT MAP (Cout2, Cout3, DCout3, Cout34);
        reg2:regs
        PORT MAP (Cout34, Clock, DCout34);
        adder2: adderspecial
        PORT MAP (Cout1, Cout34, Dcout34, final);
            n <= final + X;
    END Structure;
```

Fulladd Code from [20]

```
LIBRARY ieee;
    USE ieee.std_logic_1164.all;

    ENTITY fulladd IS
        PORT( Cin, x, y : IN STD_LOGIC;
              s, Cout   : OUT STD_LOGIC);
    END fulladd;

    ARCHITECTURE LogicFunc OF fulladd IS
        BEGIN
            s <= x XOR y XOR Cin;
            Cout <= (x AND y) OR (x AND Cin) OR (y AND Cin);
        END LogicFunc;
```

Fulladd Package from [20]

```
LIBRARY ieee;
    USE ieee.std_logic_1164.all;

    PACKAGE fulladd_package IS
        COMPONENT fulladd
            PORT( Cin, x, y  :IN  STD_LOGIC;
                  s, Cout    :OUT STD_LOGIC );
              END COMPONENT;
          END fulladd_package;
```

Regne Code suggested from [20]

```
LIBRARY ieee;
    USE ieee.std_logic_1164.all;

    ENTITY regne IS
    PORT( D     : IN  STD_LOGIC_VECTOR(23 DOWNTO 0);
          Clock : IN  STD_LOGIC;
          Q     : OUT STD_LOGIC_VECTOR(23 DOWNTO 0) );
      END regne;

    ARCHITECTURE Behavior OF regne IS
        BEGIN
            PROCESS ( Clock )
                BEGIN
                IF Clock'EVENT AND Clock = '1' THEN
                    Q <= D;
                END IF;
            END PROCESS;
        END Behavior;
```

Regs Code

```
LIBRARY ieee;
    USE ieee.std_logic_1164.all;

    ENTITY regs IS
    PORT( D     : IN  STD_LOGIC_VECTOR(2 DOWNTO 0);
          Clock : IN  STD_LOGIC;
          Q     : OUT STD_LOGIC_VECTOR(2 DOWNTO 0) );
      END regs;

    ARCHITECTURE Behavior OF regs IS
        BEGIN
            PROCESS ( Clock )
                BEGIN
                IF Clock'EVENT AND Clock = '1' THEN
                    Q <= D;
                END IF;
            END PROCESS;
        END Behavior;
```

VITA

Timothy Richard Grundman was born on August 2, 1983, in New Brunswick, NJ. He then moved to Rutledge, TN with his family at the age of four and lived there till completing college. He attended Rutledge High School, then attended University of Tennessee Knoxville for his undergraduate degree in Electrical Engineering.

After obtaining his bachelor's degree, Timothy decided to stay at the University of Tennessee for two years for his Master's degree in Electrical Engineering. During this time, he also worked as a research assistant for Dr. Islam and received four different publications.

Currently, Timothy is hunting a job to make the most of his degree. He is happily married and, in the small amounts of free time that he does have, he reads and enjoys watching movies.