

Phase-Lock Loop (PLL)
and
Testing with LabVIEW

By
Kyle Pierce

NSF-REU at the University of Maine

Summer 2001

Advisor
Dr. David Kotecki

ABSTRACT

The overall goal of this project was to characterize a fabricated phase-lock loop (PLL) that was designed by students in Prof. Kotecki's fall VLSI design class and to set up a testing station with LabVIEW. In order to characterize the PLL, tests were performed as directed by the students' test procedure on digital buffers, inverters, exclusive-or phase detectors, voltage controlled oscillators, passive filters, as well as the entire loop. These tests were performed on the packaged integrated circuit at room temperature with standard lab equipment (multimeter, oscilloscope, voltage supply, and function generator). To aid in testing the custom designed chip a testing platform incorporating LabVIEW was created. The station automated the testing process through the use of GPIB and graphical programming to control function generators and oscilloscopes.

ACKNOWLEDGMENTS

This work has been supported by the National Science Foundation Research Experience for Undergraduates.

Many thanks also go to Dr. David Kotecki, my advisor, Dr. Donald Hummels, Jesse Cousins for his background with the VLSI class and Cadence, and Kannan Sockalingam for his help with LabVIEW.

LabVIEW is a registered trademark of National Instruments, Inc.

Matlab is a registered trademark of The MathWorks, Inc.

TABLE OF CONTENTS

ABSTRACT	i
ACKNOWLEDGMENTS	ii
1 Introduction	1
1.1 Background	1
1.2 Purpose of the Research	1
1.3 Thesis Organization	1
2 LabVIEW	2
2.1 What is LabVIEW?	2
2.2 Graphical Programming	2
2.3 Instrument Control and Data Acquisition	4
2.4 Linear Frequency Sweep	4
3 Phase-Lock Loop	5
3.1 Phase-Lock Loop	5
3.2 Inverter	5
3.3 Digital Buffer	6
3.4 Exclusive Or Phase Detector	7
3.5 Low Pass Filters	8
3.6 Voltage Controlled Oscillator I	8
3.7 Voltage Controlled Oscillator II	9
3.8 Entire Loop	10
4 Conclusions	12
5 Future Work	13
REFERENCES	14
APPENDIX A	15

CHAPTER 1

1.1 Background

As part of ECE547 VLSI Design / Layout, taught by Dr. Kotecki during the fall semester 2000, students designed custom microchips. Four different designs were produced: a Phase-Lock Loop, an Autoranger Circuit, a Proportional Integral Derivative Circuit, and a Sigma-Delta Modulator. These microchips were fabricated by AMI using a 0.5 micron process. For each design five 40 pin packaged chips and five dies were produced. Due to when the chips arrived at the University of Maine very few verification tests were performed.

1.2 Purpose of the Research

The primary goal of this summer research project was to characterize fabricated chips designed by Dr. Kotecki's VLSI class. In order to do this more efficiently both in the present and future a testing platform was set up around LabVIEW with the eventual ability to test both packaged and unpackaged chips. To aid the testing process further programs to do things such as a frequency sweep were also created. As a direct consequence of this research, future students will be able to test their designs easier than ever before.

1.3 Paper Organization

I will start by discussing the benefits of LabVIEW and why it is useful. Then I will explain how I used LabVIEW to set up a testing platform to test custom chips. Finally, I will talk about how well one of the custom designed chips (PLL) performed when tested using the platform.

CHAPTER 2

2.1 What is LabVIEW?

LabVIEW is a graphical programming language used to control instruments and acquire, analyze, and display data. LabVIEW was chosen because it is becoming more and more an industry standard for testing, and is used by several high profile microchip companies including Texas Instruments. One reason LabVIEW is so popular is due to ease of use, flexibility, and portability.

Every LabVIEW program is comprised of two main parts a front panel, user interface, and a block diagram, the programming 'statements'. Once a program is created the user only needs to know how to setup the testing equipment and how to use the front panel (see Appendix A for example front panel).

2.2 Graphical Programming

Traditional computer programs consist of text based statements which follow a linear top to bottom order of execution, such as BASIC. LabVIEW, on the other hand, employs a graphical style of programming much like a circuit schematic. In contrast to the stringent linearity of traditional programming, LabVIEW execution follows one line until a block requiring more inputs is encountered. Then the program goes back and follows the next line until the same result is reached. Once all the data lines reach a given block the block is executed and the flow continues from the executed block.

In order to better understand LabVIEW's programming language and example is necessary (see Figure 1). At the top of the figure is an example program, to calculate the area of a circle with radii 1 thru 5, created with LabVIEW, and at the bottom is the corresponding program written in Matlab. Both programs are comprised of a FOR loop

that sets the radius r to 1, 2, 3, 4, and 5. In Matlab this is the first line and is closed like a parenthesis by the END statement. The equivalent statement in LabVIEW is represented by the dog-eared box which contains an i and N . The 5 tied to the N is the number of times to loop, and i stands for the current iteration number.

Now that the FOR loop has been explained, the next item to discuss is the actual calculation of the area. The second line of the Matlab program is where each area is calculated and stored into an array with index r . Likewise, the area calculation in LabVIEW is carried out inside the FOR box. The $+1$ is needed to add one to the iteration number so that the loop runs from 1 to 5 instead of the default 0 to 4. Once one is added to the current radius the next block squares the radius, and then is multiplied by π . The results are connected to an output array called area.

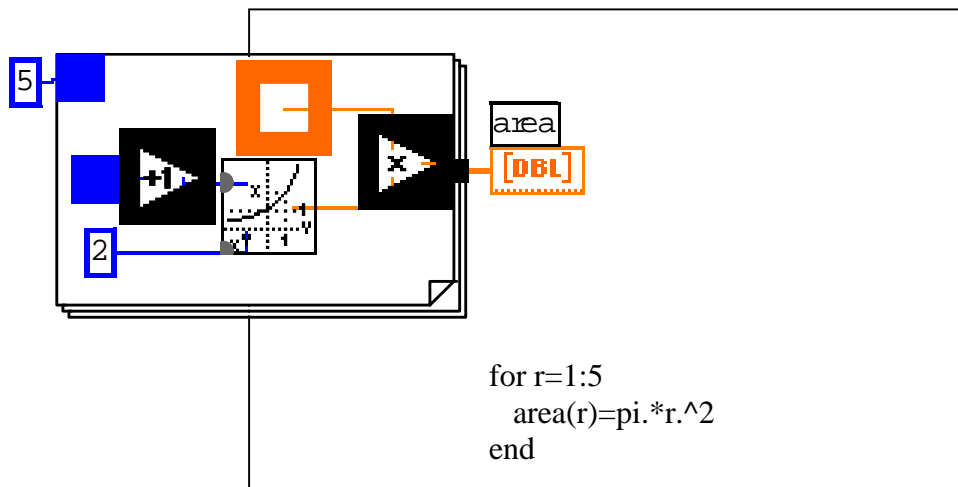


Figure 1

While graphical programming is not as compact and strictly ordered as traditional programming it is able to be better understood by non-programmers than text programs.

While graphical programming is as easy as drag and drop, it may take considerably

longer to program than text because every command must be selected from a menu and connected; however, by using text a programmer can program as fast as one's typing.

2.3 Instrument Control and Data Acquisition

Although LabVIEW has the ability to perform calculations, such as area, like other programming languages, its real power lies in the ability to easily communicate with external instrumentation such as oscilloscopes, function generators, etc. While it is possible to use external instruments with Matlab, the interface Labview provides is extremely user friendly. In order to send a command to an external instrument all that is needed is the command string, telling what to do; a session ID, identifying which device to communicate with; and error messages, for debugging purposes. All these are connected to the write function symbolized by W, as seen in Figure 2. Likewise, reading data from an instrument is almost as easy. The following change: the command becomes a query (usually ending with a question mark), a number is connected telling how much data to receive, and a string is returned (see Figure 2). The ability to communicate with external instruments within a programming environment provides engineers the ability to automate the testing process.

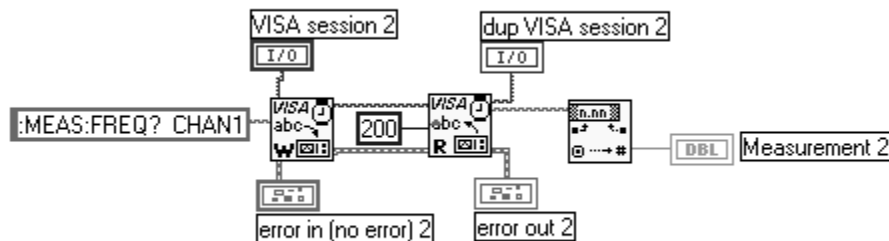


Figure 2

2.4 Linear Frequency Sweep

By combining the ability to read and write to testing equipment with the ability of a program to loop, an automated frequency sweep can be created (Appendix B). In this case a linear frequency sweep was created. The front panel of this program gives the sweep options and displays the data after all the measurements have been gathered.

Although the linear frequency sweep program may look complicated at first, it is actually fairly simple once broken down. In the upper left hand side is the calculations for what the step of the input frequency will be and whether to count up or down. Below that are the initialization and the constant settings for the Agilent 15 MHz arbitrary waveform generator (HP33120) and the Agilent 60 MHz oscilloscope (Ag54621a), only these instruments may be used in order for this program to work properly because the commands are instrument specific. The box to the right is a for loop, which is the main body of the program. This loop systematically sets the input frequency, autoscales the oscilloscope, takes an output frequency measurement, and measures the output voltage peak to peak. After all the data has been collected, it is displayed on the front panel which is represented in the diagram by the orange rectangles on the right.

An automated linear frequency sweep is very useful for test engineers because it allows the engineer to sweep large ranges of frequencies using very small step sizes. Also, since it is automated, someone does not have to tediously scroll through frequencies and record the data. This is beneficial in two ways. First, the program can be run overnight, or something else can be done while the program is running, thereby being more productive. And second, since the data is already on the computer, it can be saved to a file and analyzed later, eliminating data entry.

CHAPTER 3

3.1 Phase-Lock Loop

A phase-lock loop is a very useful circuit in computers and communication. This is because its main purpose is to regenerate a clock signal from data. In the case of a cell phone, only one signal is going to be transmitted to the phone from the tower. In order to synchronize the phone and the tower a clock signal must be part of this signal. In order to do this the clock is encoded with the data. The PLL's job is to rip the clock off the incoming signal. It does this by keeping the output and input at the same frequency and in phase over a certain range, this PLL was designed for around 10MHz. The three main blocks that make up the PLL, phase detector, loop filter, and voltage controlled oscillator can be found in Figure 3 below.

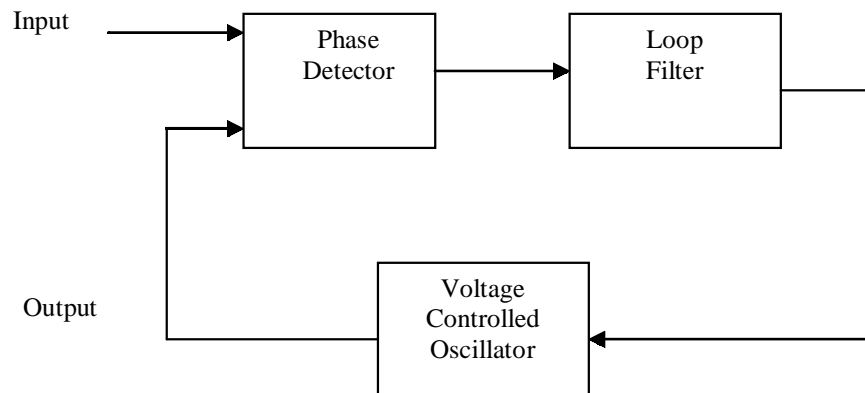


Figure 3

3.2 Inverter

A simple inverter was designed as part of the PLL. The most interesting part of an inverter is at what input voltage does the output switch values. In order to test this, a

voltage source was connected to that part of the chip and very slowly from 5V down to 0V while recording the output voltage along the way.

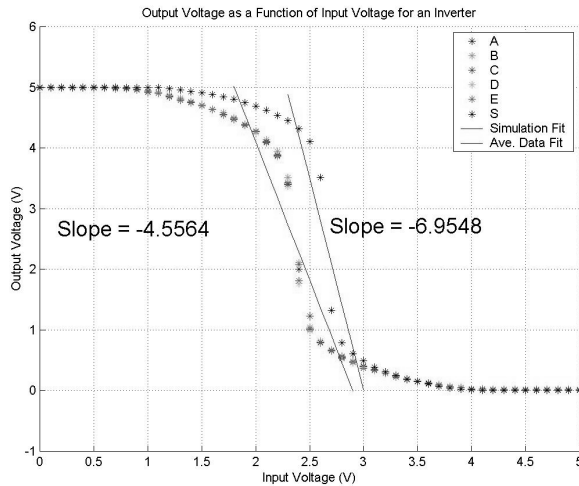


Figure 4

As is illustrated in Figure 4 the actual chips, overlapping each other to form the left hand curve, switch values at about 2.3V whereas the simulated data switches at about 2.6V. The lines and slopes drawn on top of the data indicates how good the inverter is from 4.5V to 0.5V these correspond to the 90% and 10% of possible inputs. The steeper the slope, the better the performance. Overall, all five chips tested were very consistent with each other, as shown by the overlapping, and the result was similar to the expected case.

3.3 Digital Buffer

A digital buffer was also included on the PLL. A digital buffer is meant to take an input voltage and make it either a logical one ('1') or a logical zero ('0'). In doing this, the maximum amount of current can be delivered to the next part of the chip. Like the inverter the interesting part of a digital buffer is at what voltage the output changes.

Ideally everything less than 2.5V would be '0', and everything greater or equal would be '1'.

Once again all five chips were very consistent with one another. If the input was less than 2.6V the output was '0'. Also, if the input was greater than 2.7V a '1' was produced. However, within the range of 2.6V to 2.7V the output oscillated too much to get a stable output. This means that 0.1V out of 5V are invalid, or 98% of all possible input voltages 0V to 5V are valid. Since these middle voltages will probably never be imputed, this design is effective.

3.4 Exclusive Or Phase Detector

The exclusive or phase detector that was included on the chip works like a typical xor gate. The xor gate can be used with bits or with digital signals as in this case. The purpose of this phase detector is to synchronize the input and output signals. One input is the data signal, and the other is the previously gathered signal. The output then is the clock signal which will be at the same frequency at some phase angle due to propagation delay through the chip. All five chips worked because they matched the theoretical results found in Table 1.

Input1	Input2	Out
'0'	'0'	'0'
'0'	'1'	'1'
'1'	'0'	'1'
'1'	'1'	'0'

Table 1

3.5 Low Pass Filters

Another part of the student designed PLL are two low pass filters. The purpose of this component is to filter out any high frequencies, such as harmonics, that inhibit the circuit from operating correctly. The bode plot below shows what one internal simple low pass filter with a resistance of 50K ohms and a 10pf capacitor output.

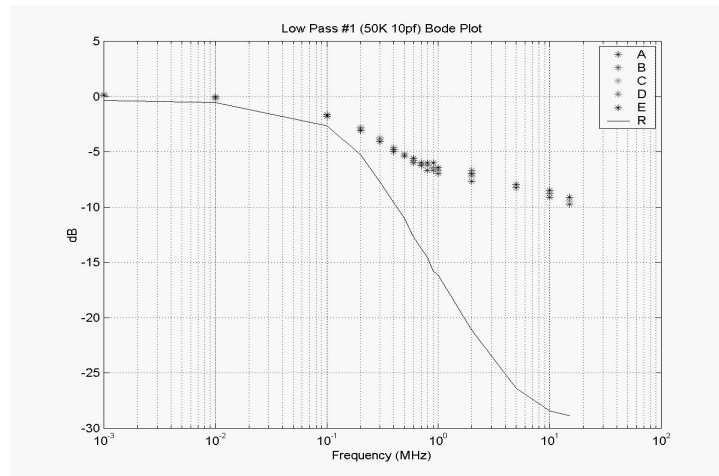


Figure 5

The upper data points are the five chips and the line near the bottom is an external low pass filter. However, the internal filters were not as good as anticipated. Since the internal filters leveled off quicker than desired, external filters were used to test the entire loop as discussed later.

3.6 Voltage Controlled Oscillator I

The fifth component of this custom PLL is a voltage controlled oscillator (VCO), of which there were two types implemented. A VCO is a circuit that outputs a digital square wave, whose frequency is a function of the input voltage. Given that frequency depends on voltage, the easiest case to manage is a linear one, however, that is rarely the case over a given spectrum.

As seen in Figure 6 the data observed on the five chips, the lower data points (upper is the simulated result), resemble a logarithmic function. While the entire function is not linear, input voltages from 1.5V to 2.5V make a pretty good line as shown by the linear regression. Since this particular PLL was designed around 10 MHz, and the VCO is linear above and below the design frequency, VCO I should work fine when integrated into the entire loop.

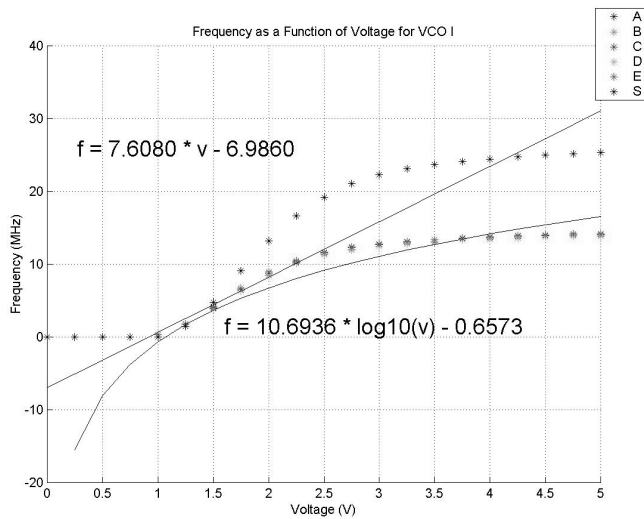


Figure 6

3.7 Voltage Controlled Oscillator II

VCO II works under the same premises as VCO I, but it can be made linear over a larger range through the use of external resistors as parameters. Figure 7 shows several different pairs of resistor values for a single chip. Through observation the longest linearity over the desired range occurs when R_{\min} is a short circuit, and R_{\max} is an open circuit (see black triangles). Due to its wide range, VCO II would seem to be the obvious choice to put into the full loop; however, there exist tradeoffs.

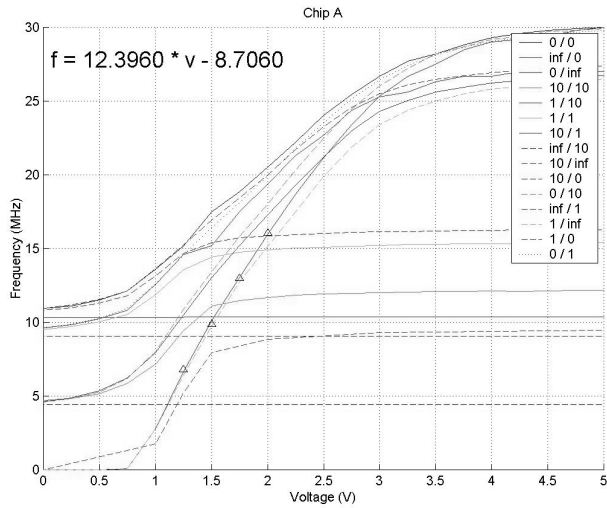


Figure 7

3.8 Entire Loop

Finally, after all the components had been tested individually, the entire loop was tested.

The entire loop consisted of the xor phase detector, connected to a low pass filter (R=100K ohm, C=10pf), which is connected to VCO I. The input goes into the unfilled xor input, and the output is pulled off the xor output before the filter. In order to test each chip, the function generator was set at 10MHz initially in order to get a lock. Then the frequency was increased until it was unstable or gained the wrong frequency (upper limit). Likewise, a lock was gained and then the frequency was decreased until the same thing happened (lower limit). Finally, when at both extremes the frequency would be increased or decreased respectively until they approached 10MHz (upper and lower lock).

As shown in Table 2 all five chips were very similar, and everything turned out close to expected. The locking limits were fairly close to 10MHz. While the upper and lower limits correspond closely with the linear upper and lower limits of VCO I as expected.

(MHz)	A	B	C	D	E
Upper limit	13.6	13.6	13.5	13.6	13.9
Lower limit	2.4	2.3	2.4	2.1	2.6
Upper lock	12.4	12.1	12.2	12.2	12.6
Lower lock	9.7	9.2	9.3	9.5	9.5

Table 2

The next step was to test the complete loop using VCO II instead of VCO I. In theory this should have increased the range over which the PLL would lock. However, there is a tradeoff for range, stability. The first stability issue arose when there was difficulty getting a lock even at 10MHz. This was a problem because this design is able to lock onto harmonic frequencies causing the output frequency to be wrong. And even when a lock was achieved, the PLL would lock on harmonics almost randomly as the frequency was swept.

CHAPTER 4

Conclusions

Throughout the entire testing process all five chips were very consistent with one another. Most of the time they were so close that it was hard to tell one set of data from another without zoom. In addition, the simulations were pretty close to the data gathered. While shifts occurred in several tests, these were probably due to errors with the simulating parasitics. Finally, while the VCO II was too unstable to be tested, the entire loop met the 10MHz specification for lock on when VCO I was used.

CHAPTER 5

Future Work

The next step in developing this testing platform is to incorporate it with a probe station so that dies as well as packaged chips can be tested. Also, the linear frequency sweep can be used as a subroutine so that a logarithmic frequency sweep can be made. Lastly, easier, more automated testing could be done on packaged integrated circuits if a piece of hardware was created to route a given pin to a certain instrument. This would allow input and output pins to be changed using software and a device rather than physically moving the connections.

REFERENCES

- [1] *Agilent 54621A/22A/24A Oscilloscopes and Agilent 54621D/22D Mixed-Signal Oscilloscopes Programmer's Guide*, August 2000
- [2] Atkinson, Brandon. Bethel, Ryan. Silvestre, Conrad. *PLL Group*. 12/19/00
- [3] Bethel, Ryan. *Testing of the Integrated PLL*. 5/17/01
- [4] <http://www.ni.com/labview/what.htm>

APPENDIX A

Linear Frequency Sweep

