On-Line Stability Detectors for Sequential Circuit Elements

Adam Jacobvitz
<ajacobvi@calpoly.edu>

Senior Project

ELECTRICAL ENGINEERING DEPARTMENT

California Polytechnic State University

San Luis Obispo

2009

Contents

1 Introduction **1**

2 Background **2**
2.1 Off-line Testing 2
2.2 On-line Testing 3

3 Requirements **4**

4 Design **5**
4.1 Franco and McCluskey's Stability Checker 5
4.2 Modified Stability Checker 7

5 Development and Construction **10**
5.1 Franco and McCluskey's Stability Checker 10
5.2 Modified Stability Checker 16
5.3 Electric Layouts 19

6 Integration and Test Results **23**
6.1 Franco and McCluskey's Stability Detector 23
6.1.1 Optimizing Setup Time 23
6.1.2 Optimizing Error Detection 24
6.2 Modified Stability Checker 25
6.3 Circuit Sizing 25
6.4 Electric Simulation Results 27

7 Conclusion **30**

Bibliography **30**

8 Appendices **32**
Inverter Characterization Script 32
Franco and McCluskey Batch Script 36
Franco and McCluskey Setup Time Calculator Script 39
Modified Stability Checker Batch Script 43
Modified Stability Checker Setup Time Calculator Script 47

List of Tables

I Off-Line vs On-Line Stability Detectors 3

II MSC Cell Possible Error Codes 9

List of Figures

4.1 Franco and McCluskey's Stability Checker 6
4.2 McCluskey Stability Detector Hold Time Violation 7
4.3 Modified Stability Detector Cell 8
4.4 Modified Stability Detector Cell Output (Hold Violation) 8

5.1 VLSI Design of a Standard Master-Slave Latch 20
5.2 VLSI Design of the Master Portion of Franco's Stability Checker 21
5.3 Full VLSI Design of Franco's Stability Checker 21
5.4 4-Bit Up Counter Based on Franco's Latch 22
5.5 VLSI Layout of Yada's Modified Stability Checker 22

6.1 32nm Transistor Process Setup Times at Varying Temperatures 24
6.2 Setup Time at 20 Deg C as a Function of Error Transistor Sizing and Transistor Feature Size 25
6.3 Franco Detector Error Rate as Dependent on Error Transistor Sizing . . 26
6.4 Franco Detector Error Rate and Setup Time as Dependent on Error Transistor Sizing 27
6.5 MSC Cell Error Detection Rate as Pulldown Circuit Size Increases . . . 28
6.6 Franco Stability Detector Setup Time as a Function of Feature Size . . . 29

Acknowledgments

I would like to thank Dr. John Oliver for guiding me through the process of this study so I could learn not just about detectors, but also how to create reports such as this one. He has been a huge help in guiding me to where I am today.

Abstract

I conducted this study of on-line stability detectors to learn more about stability checking in VLSI circuitry and how I varied the conditions in order to try to find trends on

how well the stability detectors work each set of conditions. I varied the clock speed, temperature, transistor feature size, and sizing of the transistors in both Franco's Stability Checker and Yada's MSC Cell from papers [3] and [5] respectively. I found that the sizing has the greatest impact on both test stability detectors and that both stability detectors can work under a variety of conditions with little to no loss in functionality. I did notice, however, that in general lower temperatures and smaller feature sizes produce better performance under most conditions. For Franco's detector, a smaller error pullup transistor results in better error detection while a larger pullup transistor allows for better setup times. For the MSC Cell, smaller transistors resulted in better performance.

Chapter 1

Introduction

Stability detectors are used in digital VLSI circuits in order to detect what are known as faults. A fault in this context is defined as any result obtained from an element that is different from the result desired when given specified inputs. In other words if we had a two-bit adder, we would expect 00 + 01 = 01. Any result other than 01 is a fault. There are multiple ways in which faults are created. For my senior project I looked at stability detectors designed to detect delay faults. The cause of delay faults will be explained in more depth later in the report. Also of note is that these stability detectors are on-line, which means they function during normal operation of the VLSI circuit and do not require a special testing mode for operation.

Chapter 2

Background

Computer engineering is based on the premise of using transistors chained together to implement certain logical processes. In order to do this, all of the transistors must be working properly. Designing circuits properly can eliminate most errors, but there are still problems that occur due to manufacturing variation and general wear and tear on the circuitry. Since most modern processes create circuity that cannot be seen by the naked eye, we must find more creative ways of detecting and dealing with these errors. To do this requires circuitry known as Built-in Self Test (BIST). BIST circuits are those that are built into systems for the sole purpose of testing. This can be split into two general types of tests, On-Line Tests and Off-Line Tests.

2.1 Off-line Testing

Off-line testing involves taking a circuit and putting it through a series of test vectors designed to test as many states as possible for each transistor to maximize the chances of detecting an error. One such method of applying random test vectors is described in [4]. Off-line testing is primarily to test for intermittent faults and to verify correct circuit operation before shipping off to consumers.

	On-Line Detector	**Off-Line Detector**
States Tested	Only those possible during normal circuit operation	Random assortment, ideally all states are tested
Types of Errors Detected	Errors due to aging equipment, errors due to bad operating conditions, other errors during operation	Intermittent Failures, Errors due to design faults
Major Drawbacks	Requires extra circuitry, theres a chance of faulty error detection during operation	Only can be used in a special testing mode, circuits are too large now to exhaustively test all cases. A special testing mode is required.
Benefits	Extra logic can compensate for runtime errors, old circuits can last longer.	More variations of inputs can be tested, results can be processed outside of the circuit, faults can be detected in a shorter period of time due to the extensive variation in inputs.

Table I: Off-Line vs On-Line Stability Detectors

2.2 On-line Testing

Detecting errors in real time requires on-line error detection. On-line error detector circuits are designed to detect errors that occur during operation such as crosstalk and delay errors. In particular, input stability checkers are designed to detect bad inputs into sequential circuitry such as flip-flops and latches. These are the kinds of circuits that I analyzed in my report.

Chapter 3

Requirements

The requirements for a stability detector are twofold: A stability detector needs to be able to detect the late arrival of an input from a set of combinational logic to a sequential element. Also it needs to be able to detect changes in the input during the setup and hold time since any changes during these times would result in a metastability condition. If both of these conditions are met, then the circuit fulfills its role as a stability detector. Once a stability detector detects a bad state, it needs to be able to generate an error signal so that the rest of the circuit can execute logic to compensate. Since all of this needs to happen in real time, the stability detector must be on-line and ideally not take much extra circuitry on top of what is already there. A successful implementation of a stability detector will allow the circuit to deal with bad states gracefully instead of entering a bad state and forcing the operator to either reset or throw the hardware away completely if the error is permanent.

Chapter 4

Design

The circuits I am analyzing have already been theorized and simulated by researchers in papers [3] and [5]. This paper will take the existing circuits and test different parameters including, but not limited to, variation under temperature, transistor feature size and clock speed. To begin I'll briefly summarize the operation and sizing of the stability detector described in [3].

4.1 Franco and McCluskey's Stability Checker

The Stability Checker design in Figure 4.1 is based off a standard master-slave latch. The major difference between this M-S latch and a normal one is that no changes are allowed to the input of the flip-flop while the clock is high. Any change during this time is detected by the error output. This is done by removing the extra transmission gate that normally blocks inputs into the master stage while the clock is high and adding a single transistor between the power connections of the master inverters and that of the system power. This change makes it so that the circuit is sensitive to inputs while the clock is high. The reason we want to do this is that delay errors can only happen in sequential elements due to the nature of the error (signal propagation after the clock goes high). A MS Latch is an ideal element to have stability detection because storage

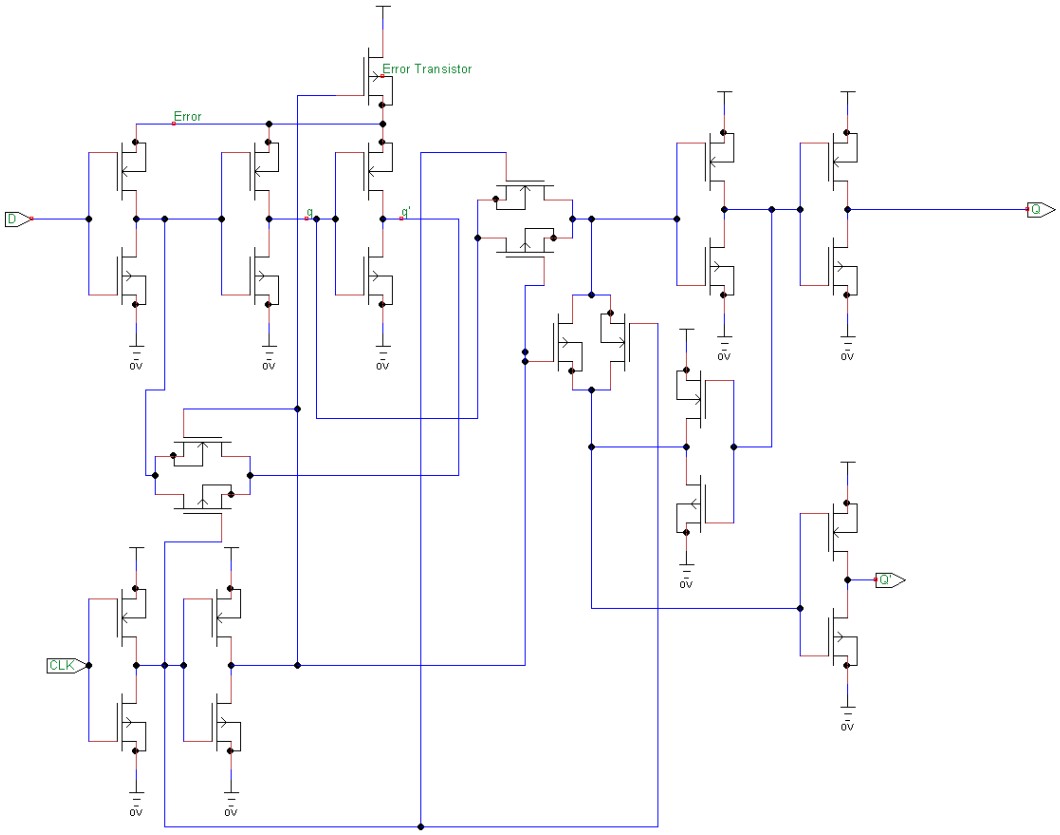

Figure 4.1: Franco and McCluskey's Stability Checker

elements are critical to keeping state between clock cycles, and an MS Latch can be used instead of a flip-flop easily if need be. Also the MS Latch design allows for easy integration of stability detection.

The reason for adding the extra transistor, referred to as the error pullup transistor in the rest of the report, is so that a signal can be generated if the combinational output (or sequential input) changes at any time while the clock is high. If the input changes, then an error signal is generated. The process for the error signal generation is that the error line as labeled in Figure 4.1 will lose charge if any of the inverters below it pull any switching current. If it does, then some circuitry can be added to detect that drop and signify that an error has been detected. A sample hold violation is shown in Figure 4.2.

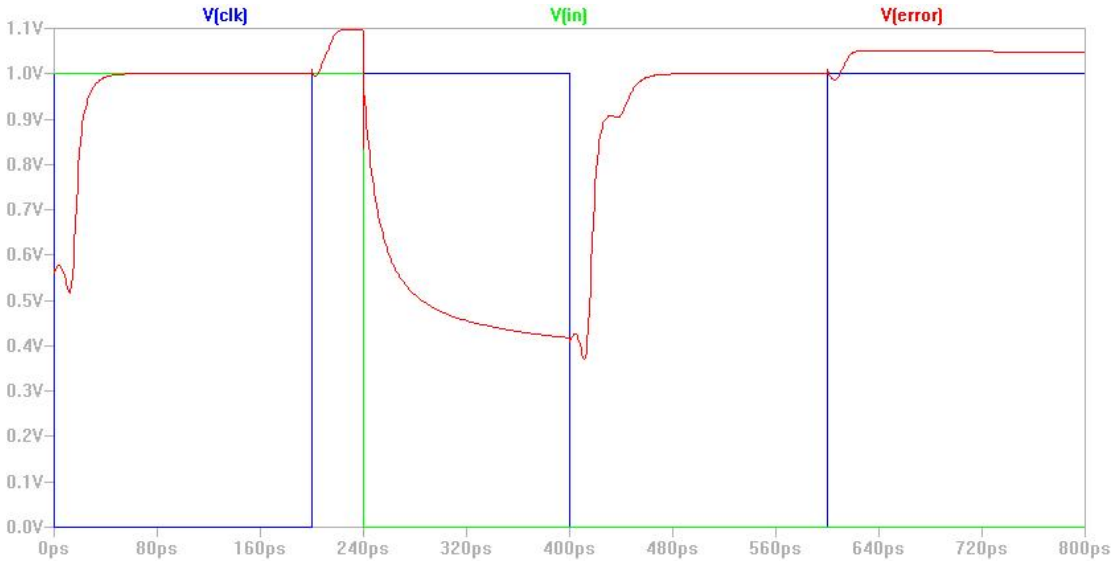

Figure 4.2: McCluskey Stability Detector Hold Time Violation

4.2 Modified Stability Checker

Another method for on-line stability detection involves the creation of an error code. This error code is generated when the combinational output and output of a sequential circuit are different. One generated, additional circuitry can be used to act when an error is detected. One method of generating error codes is proposed in [5].

Error detection is accomplished through the pulldown circuitry. When the clock is high and an error is detected, the pulldown circuit seen in Figure 4.3 will activate and pulldown the sense line. Since the dummy line cannot be pulled down due to the NMOS transistor M_{dd} being tied to ground, the two lines will differ. An example of a hold violation is show in Figure 4.4. The exact magnitude of the difference between the lines depends on the circuit parameters. The *sense* and *dummy* lines are then connected to a sense amplifier, which can then detect differences between the two lines and cause an error signal to be generated. The reason this is called sending an error code is that each line can be assigned a logical 1 or 0 depending on the state of the line (high or low) and a table of possible states can be generated as seen in Table II.

Unlike Franco's stability checker, a MSC Cell could be attached to any sequential circuit

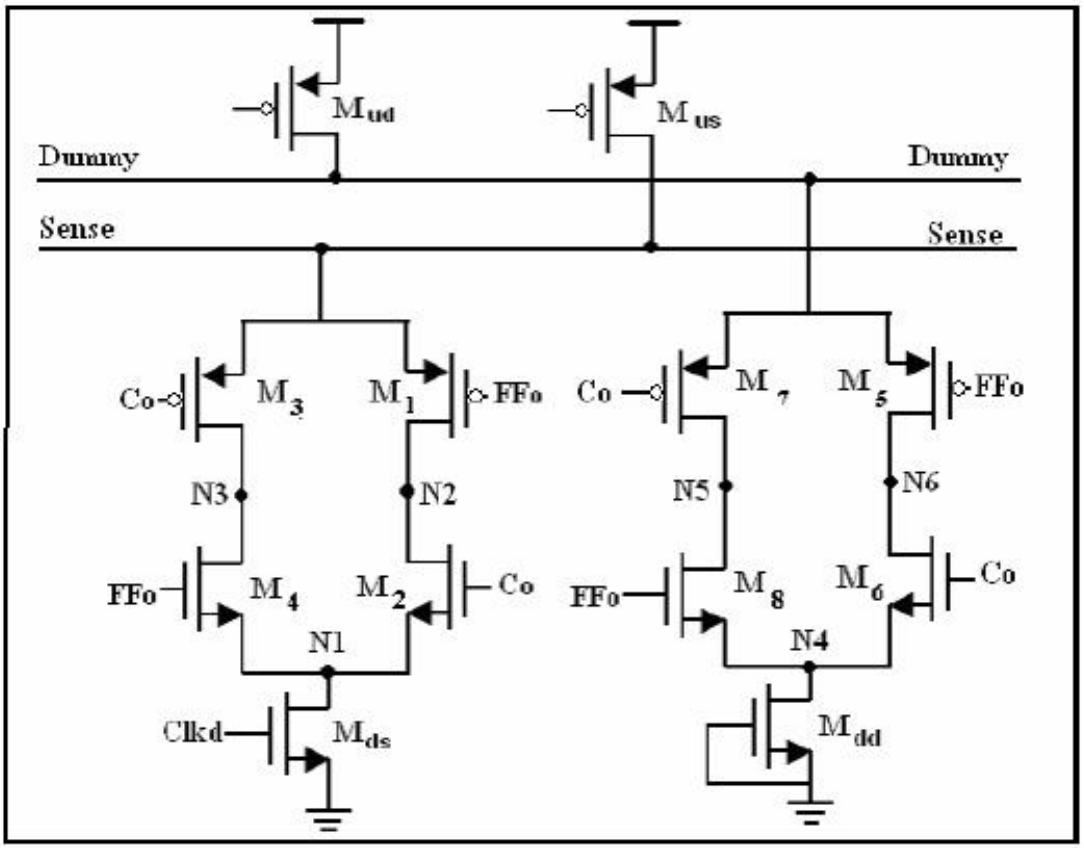

Figure 4.3: Modified Stability Detector Cell

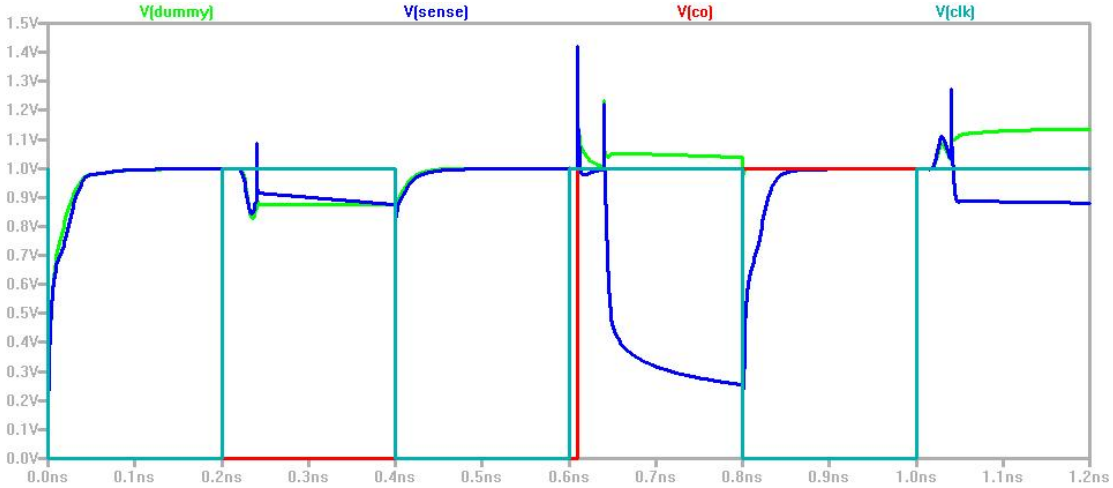

Figure 4.4: Modified Stability Detector Cell Output (Hold Violation)

dummy	*sense*	error output
0	0	0
0	1	1
1	0	1
1	1	0

Table II: MSC Cell Possible Error Codes

element, making it much more versatile. Unfortunately, it is also much bigger since it is self contained. The cell itself contains 12 transistors, and a sense amplifier could contain much more. This means that implementing a MSC Cell system into a circuit would take a lot of overhead. Also, since the easiest way to implement such a system is to attach multiple cells to the same dummy and sense line, routing and loading issues come into play requiring even more work. Finally, since the cell acts as a load on the sequential circuit element and the combinational output, the sizing for the circuits have to be adjusted to compensate most likely requiring more space. All of these sacrifices are much greater compared to that of Franco's latch, but if you need the versatility, it may be worth it.

Chapter 5

Development and Construction

To actually create these circuits would take an amount of time, effort, and funding that is outside the scope of this project. I was, however, able to create VLSI designs in a tool known as Electric (http://www.staticfreesoft.com/) as well as do LTSpice (http://www.linear.com/designtools/software/) simulations. These designs could be sent off to a fabrication facility such as ones provided by MOSIS (http://www.mosis.com/). All transistor models used in this report came from the Predictive Technology Models developed by the Nanoscale Integration and Modeling Group at Arizona State University (http://www.eas.asu.edu/ ptm/) and was developed from [1], [6], and [2].

5.1 Franco and McCluskey's Stability Checker

During this test the voltage input was changed from a logical high to a logical low while the clock was high. During this time the error transistor is off, and the voltage signal on the error line is discharged through the switching current of the flip-flop inverters indicating that the input has changed. This is considered normal operation. Due to the transparent nature of the error detection, this circuit could replace a normal M-S Latch so long as the circuit is designed to not change any combinational outputs while the clock is high.

The first optimization criterion I used was to try and get the fastest setup time possible. The setup time is the minimum time the signal needs to remain stable before the rising clock edge in order to successfully latch the logic state into the flip-flop. If the setup time is violated, the latch has what is called a metastable input, this means the input could be latched as high or low or the circuit may be performing some operation that is not intended.

To measure the setup time, I needed to know how long it would take for an input signal to propagate through the master portion of the M-S latch and reach a level such that it would trigger the slave portion properly when the rising edge hits. This qualifies as the setup time in this circuit. To do this, I executed variations on the following Spice code:

```
*Stability FF
.INCLUDE 32nm_bulk.model1
Vclk clk 0 DC 0V
Vcc  Vcc 0 DC 1V
Vin 1 0 PWL(0ns 1V 1ps 1V 1.00001ps 0V)
*error transistor
M1 5 11 Vcc Vcc pmos L=32nm W=192nm
*inverters
X_inv1 5 0 1   2  Vcc 0 Inverter
X_inv2 5 0 2   3  Vcc 0 Inverter
X_inv3 5 0 3   4  Vcc 0 Inverter
X_tgate1 4 2  8 11 Vcc 0  TGate
X_clkinv1 Vcc 0 clk 8 Vcc 0 Inverter
X_clkinv2 Vcc 0 8 11  Vcc 0 Inverter
* TGate Input Output P_gate N_gate Pwr Gnd
.SUBCKT TGate 1 3 2 4 vcc gnd P=96nm N=48nm
M1 3 2 1 vcc pmos L=32nm W='P'
M2 1 4 3 gnd nmos L=32nm W='N'
.ENDS
* Inverter Vdd Vss Input Output Pwr Gnd
.SUBCKT Inverter  1 4 2 3 vcc gnd P=96nm N=48nm
M1 3 2 1 vcc pmos L=32nm W='P'
M2 3 2 4 gnd nmos L=32nm W='N'
.ENDS
.options TNOM=100 TEMP=100
.print tran V(4)
.tran .01ps 2ns
.END
```

The proceeding Spice code is for a 32nm transistor process at 100 degrees centigrade. I ran this code for 32,45,65,90,130nm processes, 20,60,100,140 and 180 degrees C, and varying the error transistor to be $1\lambda, 2\lambda, 3\lambda, 4\lambda, 5\lambda, 10\lambda, 20\lambda, 50\lambda$. For each run I took a number of sample points and checked to see when the node between the master and slave latches reached a voltage considered to be stable enough to propagate to the next

stage. I did this twice, once for a high-to-low transition, and once for a low-to-high transition and then took the longer of the two as the setup time for that particular run. The results of this are discussed in the Integration and Test Results section.
Once I had the setup time, assuming that the hold time is always shorter than that of the setup time, which is true in a M-S latch, I could calculate the maximum clock speeds for this particular latch. For setting the clock speeds, I used the equation $\frac{1}{(worst\ \ case\ \ setup\ \ time)*2}$ and then rounded down to a more round number. In practice this clock speed would be absurd as pretty much any input time would be a setup violation in the worst case, but it gives a rough estimate of the order of speed for the maximum clock. It just so happened that I also made numerous mistakes at this part involving using the wrong sizing method and not setting the temperature properly in the model file requiring me to start over multiple times until I ended up creating a script that would work and accepted appropriate sizing.
To actually test the latch to check for valid error detection required writing a script with two sections: the first section would be a series of loops in which the model parameters were varied according to the test parameters. For example, here is the script portion for modeling Franco's latch. Essentially it is just LTSpice code, except that instead of hard coding the parameters to test, they are variables that are changed as each loop is executed. The way execution is done is that a spice file is written out to disk, then LTSpice is called in batch mode to generate a raw file. I used a .print tran V(5) statement so only the voltage at node 5 (error line) is written to the raw file. The raw file is then processed by another program called ltsputil.exe, which converts the raw file from a binary format to a human readable CSV format.

```
path = "I:\spice\SeniorProject\CircuitRuns\McCluskeyRuns\"
temperature = Array(20,60,100,140,180)
models = Array(32,45,65,90,130)
'Max speed 10GHz (approx. double longest setup time), Slowest 130nm at 180 deg C
'1x minimum size
'clockspeed = Array(10000000000,5000000000,1000000000)
'1/8x minimum size
clockspeed = Array(375000000,187500000,37500000)
'set the input signal time offset in pico-seconds from when the clock triggers
inputtimeoffset = Array(-1330)
Set fso = CreateObject("Scripting.FileSystemObject")
Set ErrorFile = fso.CreateTextFile("ErrorFile.csv",True)
ErrorFile.WriteLine("Transistor Feature Size,Clock Speed,
Temperature,Setup or Hold Violation (0 = Setup 1 = Hold),Error Detected? (0 = yes 1 = no)")
For l = 0 to UBound(models)
For m = 0 to UBound(clockspeed)
For h = 0 to UBound(temperature)
For n = 0 to UBound(inputtimeoffset)
'clock period in pico-seconds
clockperiod = (1 / clockspeed(m))*1000000000000
tempfile = path & "test.cir"
Set fso = CreateObject("Scripting.FileSystemObject")
Set GuyFile = fso.CreateTextFile(tempfile, True)
GuyFile.WriteLine("*Stability FF")
GuyFile.WriteLine(".INCLUDE " & models(l) & "nm_bulk.model1")
```

```
GuyFile.WriteLine("Vclk clk 0 PULSE(1 0  0ns .1ps .1ps " & 0.5*clockperiod & "ps " & clockperiod & "ps)")
GuyFile.WriteLine("Vcc  Vcc 0 DC 1V")
triggertime = 0.5*clockperiod + inputtimeoffset(n)
temp = triggertime + 0.000001
'Test During the Setup and Hold Times (hold times being defined as the time right after the clock goes high)
GuyFile.WriteLine("Vin 1 0 PWL(0ps 0V " & triggertime & "ps 0V " & temp & "ps 1V)")
GuyFile.WriteLine("*error transistor")
GuyFile.WriteLine("M1 5 11 Vcc Vcc pmos L=" & models(l) * 2 & "nm W=" & models(l)/2*3 & "nm")
GuyFile.WriteLine("*inverters")
GuyFile.WriteLine("X_inv1 5 0 1   2  Vcc 0 Inverter")
GuyFile.WriteLine("X_inv2 5 0 2   3  Vcc 0 Inverter")
GuyFile.WriteLine("X_inv3 5 0 3   4  Vcc 0 Inverter")
GuyFile.WriteLine("X_tgate1 4 2  8 11 Vcc 0  TGate")
GuyFile.WriteLine("X_clkinv1 Vcc 0 clk 8 Vcc 0 Inverter")
GuyFile.WriteLine("X_clkinv2 Vcc 0 8 11  Vcc 0 Inverter")
GuyFile.WriteLine("* TGate Input Output P_gate N_gate Pwr Gnd")
GuyFile.WriteLine(".SUBCKT TGate 1 3 2 4 vcc gnd P=" & models(l)/2*6 & "nm N=" & models(l)/2*3 & "nm")
GuyFile.WriteLine(" M1 3 2 1 vcc pmos L=" & models(l) & "nm W='P'" )
GuyFile.WriteLine(" M2 1 4 3 gnd nmos L=" & models(l) & "nm W='N'" )
GuyFile.WriteLine(".ENDS")
GuyFile.WriteLine("* Inverter Vdd Vss Input Output Pwr Gnd")
GuyFile.WriteLine(".SUBCKT Inverter  1 4 2 3 vcc gnd P=" & models(l)/2*6 & "nm N=" & models(l)/2*3 & "nm")
GuyFile.WriteLine(" M1 3 2 1 vcc pmos L=" & models(l) & "nm W='P'" )
GuyFile.WriteLine(" M2 3 2 4 gnd nmos L=" & models(l) & "nm W='N'" )
GuyFile.WriteLine(".ENDS")
GuyFile.WriteLine(".options TNOM=" & temperature(h) & " TEMP=" & temperature(h))
GuyFile.WriteLine(".print tran V(5)")
If clockspeed(m) = 37500000 Then
GuyFile.WriteLine(".tran 1ps 60ns")
Else
GuyFile.WriteLine(".tran .1ps 10000ps")
End If
GuyFile.WriteLine(".END")
GuyFile.Close
csvfile = path & "" & temperature(h) & ".csv"
Set Command = WScript.CreateObject("WScript.Shell")
cmd = "%comspec% /c sed s/" & chr(34) & "tnom    = 27" & chr(34) & "/tnom=" &
temperature(h) & "/ " & models(l) & "nm_bulk.model > " & models(l) & "nm_bulk.model1"
Command.Run(cmd)
circuit = path & "test.cir"
output = path & "test.out"
raw = path & "test.raw"
cmd = path & "scad3.exe -b " & circuit
Command.Run (cmd)
WScript.Sleep(5000)
'Convert raw to txt
cmd = "ltsputil.exe -x0 test.raw test.out " & chr(34) & "%15.7e" & chr(34) &
" " & chr(34) & "," & chr(34) & " " & chr(34) & "Time,Voltage" & chr(34) & " *"
Command.Run (cmd)
```

The second section of the script consists of test parameters. The data from the runs is extracted from the CSV file and then put into the array MyArray. In this script, I am testing to see if the error line ever goes below 0.7V and if the latch ever recovers from an error. Depending on the results of the tests, different things are written into a CSV file as my results file. At the very end I have a section with code for debugging purposes, basically it saves each CSV generated with the waveforms and organizes them

into a directory hierarchy so that they can be easily located based on the stimuli being tested. This way if there was a problem with the output data I could easily find the test and view the waveform generated without having to rerun LTSpice. The code for this is shown below:

```
'Open Output File
WScript.Sleep(200)
Set GuyFile = fso.OpenTextFile(output,1)
'Read in file into arrFileLines
i = 0
Do Until GuyFile.AtEndOfStream
Redim Preserve arrFileLines(i)
arrFileLines(i) = GuyFile.ReadLine
i = i + 1
Loop
GuyFile.Close

j = i - 1
k = 0
temp = 0
valid = 0
For i = 0 to j
'get rid of excess spaces
arrFileLines(i) = Trim(arrFileLines(i))
Redim Preserve arrValidLines(k)
arrValidLines(k) = arrFileLines(i)
k = k + 1
Next
'FOR DEBUGGING PURPOSES, PRINT OUT arrVALIDLINES
'Set GuyFile = fso.CreateTextFile("temp.txt", True)
'j = k - 1
'for i = 0 to j
'myArray = Split(arrValidLines(i),",")
'myArray(1) = Trim(myArray(1))
'GuyFile.WriteLine(myArray(0) & "," & myArray(1))
'next
'GuyFile.Close
'END DEBUG

'Check to see if it detected the error properly
j = k - 1
k = 0
errordetect = 0
For k = 0 to j
myArray = Split(arrValidLines(k),",")
myArray(1) = Trim(MyArray(1))
If k > 50 Then
'are we below threshold?
If myArray(1) < 0.7 Then
If errordetect = 0 Then
errortime = myArray(0)
errordetect = 1
End If
End If
'Does it reset after finding an error?
If (errordetect = 1) Then
If myArray(1) > 0.5 Then
errordetect = 2
```

```
End If
End If
End If
Next
If errordetect <> 2 Then
'Error not detected
'Lets export all our information to an error file
'Transistor Feature Size,Clock Speed,Temperature,Setup or Hold Violation,Error Detected?
'n = 0 is setup, n = 1 is hold
'Error Detected? 0 is yes, 1 is no
ErrorFile.WriteLine(models(l) & "," & clockspeed(m) & "," & temperature(h) & "," & n & ",1")

Else
'Error detected
ErrorFile.WriteLine(models(l) & "," & clockspeed(m) & "," & temperature(h) & "," & n & ",0")
End If
'Export to a CSV
k = 0
Set GuyFile = fso.CreateTextFile(csvfile, True)
For k = 0 to j
myArray = Split(arrValidLines(k),",")
myArray(1) = Trim(myArray(1))
GuyFile.WriteLine(myArray(0) & "," & myArray(1))
Next
GuyFile.Close
Set GuyFile = Nothing
'make the directory heirarchy
'Feature Size
directory = path & models(l) & "nm_bulk\"
MD(directory)
'Clock Speed
directory = path & models(l) & "nm_bulk\" & clockspeed(m) & "Hz\"
MD(directory)
'Setup or Hold Time Violation
If CDbl(triggertime)  > 0.5*clockperiod Then
'Hold Time Violation
directory =  path & models(l) & "nm_bulk\" & clockspeed(m) & "Hz\" & "Hold_Violation\"
Else
directory =  path & models(l) & "nm_bulk\" & clockspeed(m) & "Hz\" & "Setup_Violation\"
End If
MD(directory)
fso.MoveFile csvfile, directory
fso.DeleteFile(output)
fso.DeleteFile(circuit)
fso.DeleteFile(raw)
fso.DeleteFile(models(l) & "nm_bulk.model1")
Set fso = Nothing
Next
Next
Next
Next
ErrorFile.Close
Set ErrorFile = Nothing


Function MD(strDirName)
     Dim lFSObj
     MD = False
     ' First See if it already exists
     set lFSObj = Wscript.CreateObject("Scripting.FileSystemObject")
If lFSObj.FolderExists(strDirName) Then
MD = True
Else
lFSObj.CreateFolder(strDirName)
```

```
MD = True
End If
     Set lFSObj = Nothing
End Function
```

The following criteria were tested for Franco's Latch:

- Clock Speeds: Max Clock Possible, $\frac{1}{2}$ of Max Clock, $\frac{1}{10}$ of Max Clock
- Temperature: 20 - 180 Degrees Centigrade in steps of 40 degrees
- Bulk PTR Models 32nm, 45nm, 65nm, 90nm, 130nm
- Setup Time Violations and Hold Time Violations
- Error Transistor Widths (Length 1λ): $1\lambda, 2\lambda, 3\lambda, 4\lambda, 5\lambda$
- Error Transistor Lengths (Width 2to1 PtoN Ratio): 1λ, 2λ, 4λ

The reason I only varied the error transistor and not the rest of the circuit's size will also be discussed in the results section. The total number of runs for this circuit was 1200. The actual detection of an error is based off of the discharging of the error line as discussed earlier. Because of this, the threshold that can be considered an error is inherently arbitrary. I chose 0.7 because I found in my runs that it gave a good balance that it was high enough such that given a sufficiently high clock speed, the error line would not drop below this value, and it was high enough such that when an error did occur the line dropped sharply below 0.7. Also 0.7 is a common threshold voltage for transistors, so by putting the gate of a transistor on this line a drop below 0.7 could easily be detected.

5.2 Modified Stability Checker

Where Franco's stability detector is built into the M-S latch itself, Yada took a different approach. Instead of integrating the error detection, he instead opted to attach the error

detection circuit externally. One advantage to this is that the error detection circuitry is completely independent of the actual latch, so that any clocked element could be used. Because of this, I had to construct a M-S latch independently of the error detector in Spice using the following code:

```
*MS Latch
*MSLatch Input Output invOutput Clk Vcc Gnd
.SUBCKT MSLatch 1 7 13 clk Vcc Gnd
X_inv1 Vcc Gnd 1   2   Inverter
X_inv2 Vcc Gnd 3   4   Inverter
X_inv3 Vcc Gnd 5   6   Inverter
X_inv4 Vcc Gnd 6   7   Inverter
X_inv5 Vcc Gnd 4   9   Inverter
X_inv6 Vcc Gnd 6   12  Inverter
X_inv7 Vcc Gnd clk 8   Inverter
X_inv8 Vcc Gnd 8   11  Inverter
X_inv9 Vcc Gnd 12  13  Inverter
X_tgate1 2 3 11 8  TGate
X_tgate2 4 5 8 11  TGate
X_tgate3 9 3 8 11  TGate
X_tgate4 12 5 11 8 TGate
.ends
* Inverter Vcc Gnd Input Output
* TGate Input Output P_gate N_gate
.SUBCKT TGate 1 3 2 4
M1 3 2 1 1 pmos
M2 1 4 3 3 nmos
.ENDS
* Inverter5 Vcc Gnd Input Output
.SUBCKT Inverter  1 4 2 3
M1 3 2 1 1 pmos
M2 3 2 4 4 nmos
.ENDS
```

I then attached a MSC Cell to the end of it and ran all my tests on the complete circuit. The rest of the Spice code in order to accomplish this is shown below:

```
Vcc vcc 0 DC 1V
Vclk clk 0 PULSE(1 0  0ns .1fs .1fs 200ps 400ps)
Vclkd clkd 0 PULSE(1 0  0ns .1fs .1fs 240ps 400ps)
Vco co 0 PWL(0ns 0V 610ps 0V 610.0001ps 1V)
X_mslatch co ffo 1 clk Vcc 0 MSLatch
X_msccell clkd vcc co ffo sense dummy MSCCell
.options TNOM=20 TEMP=20
.op
.probe
.print tran V(dummy) V(sense)
.tran 1ps 1.2ns
.SUBCKT MSCCell clkd vcc co ffo sense dummy
*Dummy Pullup
Mud dummy clkd vcc vcc       pmos L=32nm W=48nm
*Sense Pullup
Mus sense clkd vcc vcc       pmos L=32nm W=48nm
*Sense Pulldown
```

```
Mds 1 clkd 0 0             nmos L=32nm W=96nm
*Dummy Pulldown
Mdd 4 0 0 0               nmos L=32nm W=96nm

*Sense Cell
M1  2  ffo  sense  sense  pmos L=32nm W=96nm
M2  2  co  1  0       nmos L=32nm W=48nm
M3 3 co  sense vcc        pmos L=32nm W=96nm
M4 3 ffo 1 0           nmos L=32nm W=48nm
*Dummy Cell
M5 6 ffo dummy vcc        pmos L=32nm W=96nm
M6 6 co 4 0           nmos L=32nm W=48nm
M7 5 co dummy vcc       pmos L=32nm W=96nm
M8 5 ffo 4 0           nmos L=32nm W=48nm
.ends
```

The first step, like that of in Franco's stability checker, was to calculate the appropriate setup time. Since the MSC cell requires both a synchronous and an asynchronous input, the time it takes to get a valid input is based off of the delayed clock *clkd* instead of the system clock *clk*. To measure this time I measured the time it took after a rising edge for a signal to propagate through the slave portion of the latch. Since this is the segment with the larger load (the MSC cell is a larger load than the slave portion) I doubled this time in order to get a rough estimate for the setup time for the delayed clock.

Using the $\frac{1}{(worst\ \ case\ \ setup\ \ time)*2}$ formula I was then able to get a rough estimate for the maximum clock speed of the delayed clock. To figure out the maximum speed of the system clock I used the formula with the original measurement ($\frac{1}{2}$ of the delayed clock setup time) since the latch is approximately symmetric.

To test the MSC Cell's viability I used the same sensitivity analysis parameters as the Franco checker resulting in another 1200 runs. I also devised my script to run in the same way: I had one section for the spice code, and the other to detect certain events. In this case since the detector works by sensing an error as a divergence in the voltage between the *dummy* and *sense* lines, I detected the maximum error between those lines. For error I used $\frac{sense-dummy}{sense}$ and then rounded to two decimal places. In some situations due to overshoot, the error between the lines actually exceeded 100% so I set a hard cap of a maximum error of 100% on all of my results since any divergence between the lines of more than 100% should in theory be a detected error. In addition

to this, I also set the script to ignore divergences in the two lines due to capacitance since these kinds of errors shouldshould not be detected by a sense amplifier anyway.

5.3 Electric Layouts

In addition to creating Spice simulation files to test the circuits, I also created metal level layouts in a tool known as Electric in order to provide a more accurate simulation to compare against the purely Spice one. Since Franco and McCluskey's stability checker are based off of a standard Master-Slave latch the first element I created was a Master-Slave Latch in Electric to use as a baseline model. This latch can be seen in Figure 5.1. All of the transistors are minimum sized, so at the moment it is not designed to handle larger loads. This could be changed by scaling the output inverters or adding some buffer inverters to the end.

After creating the baseline M-S Latch, I went on to make Franco's Stability Checker in VLSI. The first step was to make the master portion of the latch since it is what makes his latch unique. A layout for it can be seen in Figure 5.2. To get it to fit better I had to rotate it, so the output is on the top and the input d on the bottom.

After creating the master I created a slave portion with both an output Q and an inverted output $\overline{Q}$. The only transistors not minimum sized in this design are the in the clock driving circuit due to the large number of loads. The full layout is shown in Figure 5.3.

To test the functionality of Franco's detector in a circuit, I made a simple 4-bit up counter by stamping out four detectors and connecting the output of each to the *clk* input of the next one. An example of this is shown in Figure 5.4.

In addition to creating Franco's Latch in Electric, I also created the Modified Stability Checker by Yada as can be seen in Figure 5.5.

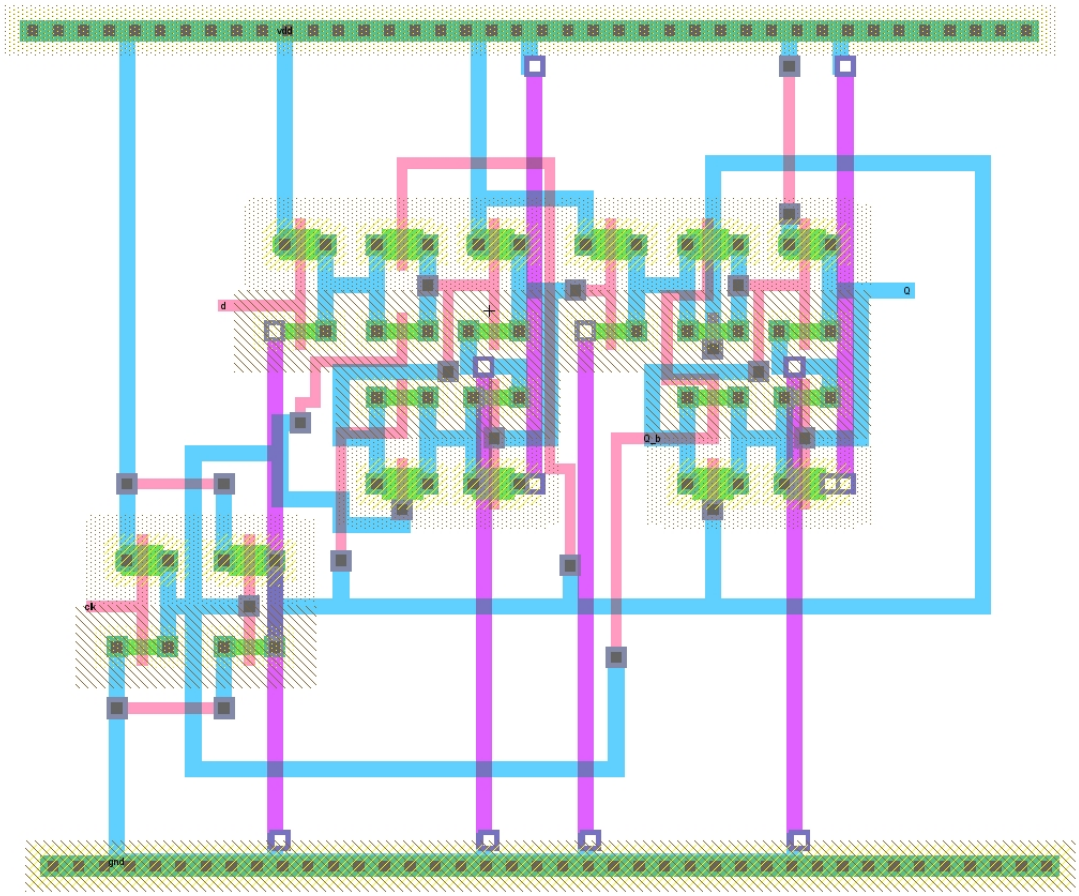

Figure 5.1: VLSI Design of a Standard Master-Slave Latch

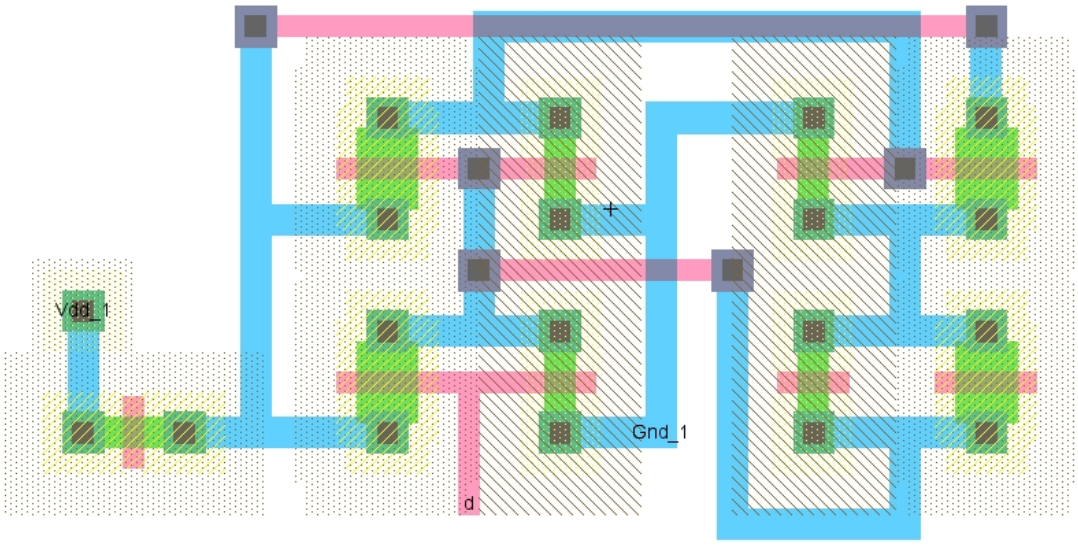

Figure 5.2: VLSI Design of the Master Portion of Franco's Stability Checker

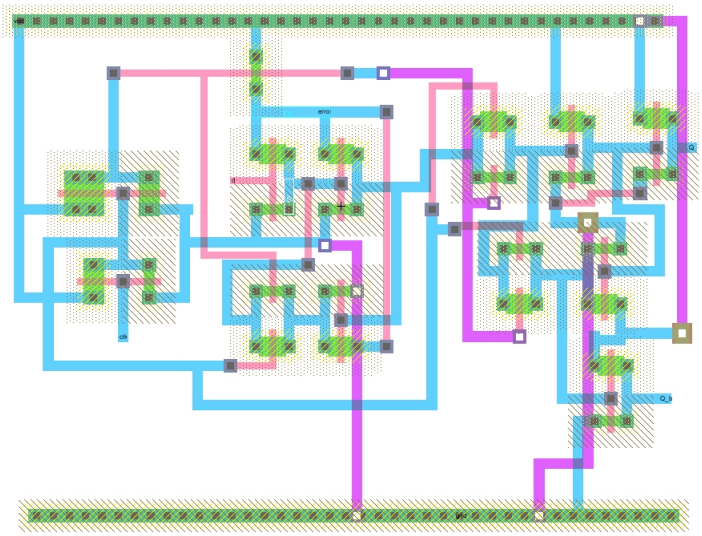

Figure 5.3: Full VLSI Design of Franco's Stability Checker

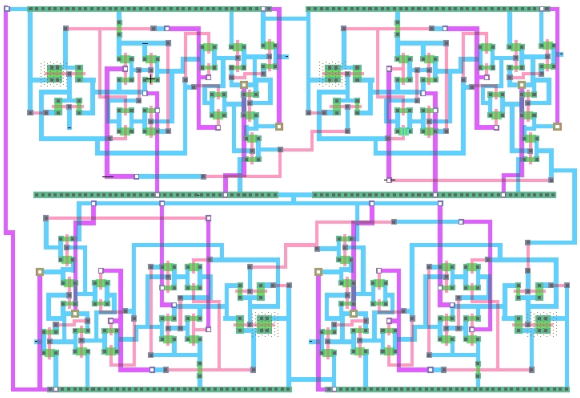

Figure 5.4: 4-Bit Up Counter Based on Franco's Latch

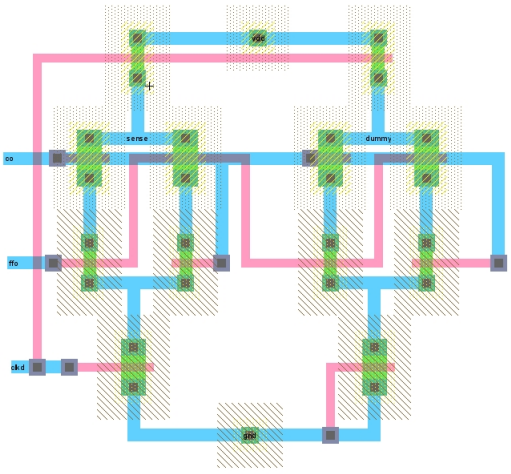

Figure 5.5: VLSI Layout of Yada's Modified Stability Checker

Chapter 6

Integration and Test Results

6.1 Franco and McCluskey's Stability Detector

The first thing I noticed is that the stability detector propagates the signal fastest when all of the inverters in it are minimum sized. This is probably due to the fact that a larger inverter is a larger load, so by keeping the inverters all minimum sized the load sizes stay small. Because of this, I decided to keep the inverter and transmission gate sizes fixed throughout all my runs. The error pullup transistor, on the other hand, had no one best size, and depending on the criteria, it may be better to keep the transistor small, or make it very large.

6.1.1 Optimizing Setup Time

The setup time of the McCluskey circuit depends on how fast the signal can propagate through the master section of the latch as mentioned before. The speed of the propagation in turn depends on how fast the inverters can charge their output nodes. Because the error transistor is the only source of current for all three inverters in the master section of the latch, the larger the error pullup transistor, the faster the signal propagates through the latch. This is shown experimentally in Figure 6.1. After a while the setup time reaches a maximum, most likely due to the inverters themselves.

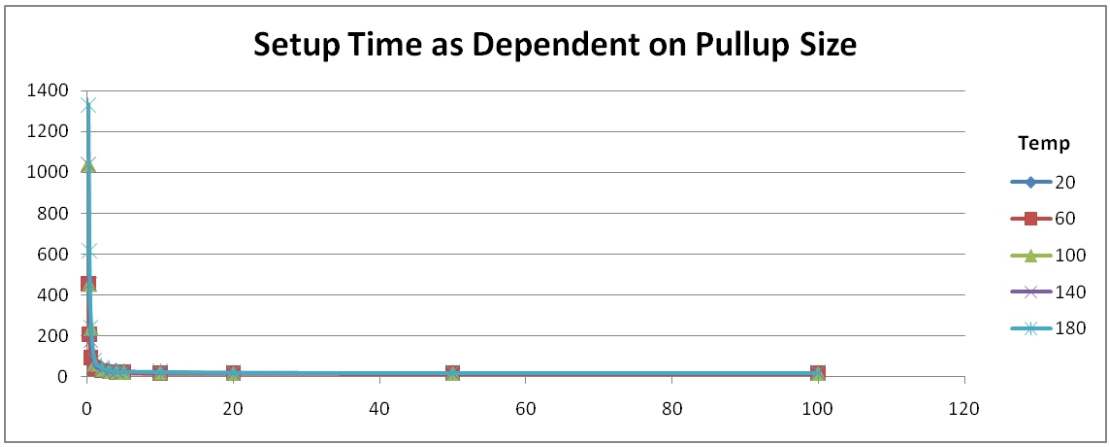

Figure 6.1: 32nm Transistor Process Setup Times at Varying Temperatures

You can also see in Figure 6.1, the overall trend of the setup time is temperature independent. This means that for optimizing setup time, I can completely eliminate the temperature parameter resulting in Figure 6.2. For optimizing setup time, the stronger the pullup transistor the better. The absolute fastest setup time measured was with a 100x error transistor and resulted in a setup time of 50.36ps.

6.1.2 Optimizing Error Detection

During my testing of the setup time, I found that as the pull up transistor increases in size, the error detection rate falls sharply. This is most likely due to the increased capacitance on the drain node as the error transistor increases in size. This increased capacitance is a double edged sword: it takes longer for the charge to leak through the inverters therefore allowing this circuit to be used with a slower clock, but it also takes much longer for the node to discharge. Logically this would mean that a smaller error transistor would allow a faster clock.

In fact, as the error transistor shrinks, the error detection rate improves. Unfortunately, as the transistors shrink, the number of false positives increase as seen in Figure 6.3. A false positive is when the error line dips below the error detection threshold during normal operation. A faster clock may be able to compensate for these kinds of errors

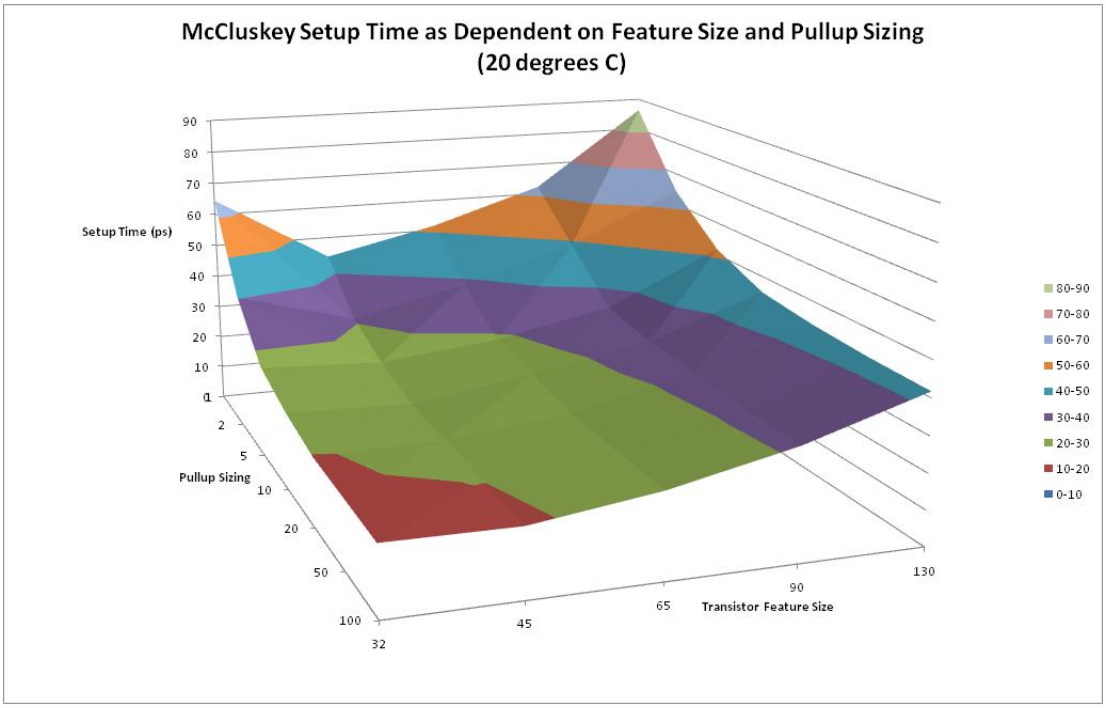

Figure 6.2: Setup Time at 20 Deg C as a Function of Error Transistor Sizing and Transistor Feature Size

so long as the clock is slow enough to compensate for the increased setup time.

6.2 Modified Stability Checker

6.3 Circuit Sizing

To size this circuit I used standard λ sizing rules for the transistor widths. Since the pull up transistors M_{ud} and M_{us} are there to precharge the *dummy* and *sense* lines, I made them both minimum sized. The pull down chain on the other hand needs to pull the *sense* line down as quick as possible. To do this I made the PMOS transistors 18λ and the NMOS transistors 9λ. To insure that the *sense* line would be pulled down as quick as possible, I also sized the inverter chain in the M-S latch slave portion I was using so that the MSC Cell was a 1x load.

Unlike Franco's Detector, I was unable to find any clear cut trends or patterns in the

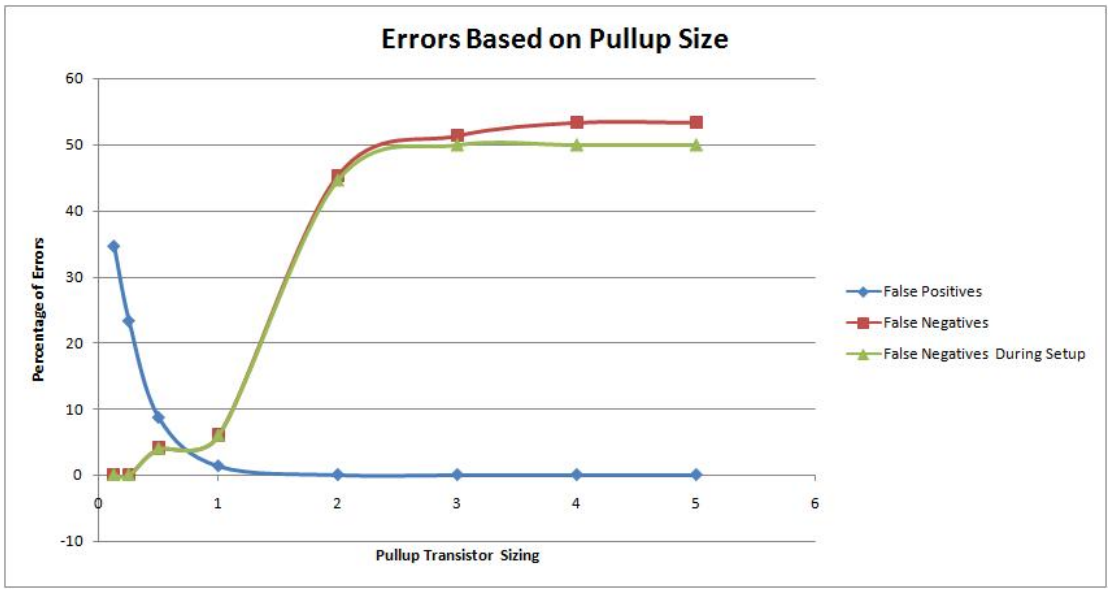

Figure 6.3: Franco Detector Error Rate as Dependent on Error Transistor Sizing

sizing of the MSC Cells in so far as error detection. It appeared the actual difference between the dummy and sense lines tended to get worse at higher temperatures, but there was no pattern that guaranteed this. Also the clock speed and the transistor feature size did not seem to indicate any pattern in error detection rates. Error detection did, however, get worse as the MSC Cell became larger without changing the M-S Latch sizing. Using a threshold of 30% difference to indicate an error was detected, I found that past a 10λ increase in size for the pulldown portion of the MSC Cell, the number of missed errors became very large as can be seen in Figure 6.5. Also at very small λ there was bad error detection, most likely due to the inability of the pulldown circuit to yank the sense line fast enough. As you can see in Figure 6.5 there is a local minima at around $2-5\lambda$ in which the pulldown is balanced enough to detect errors and not too big that the MS Latch cannot drive the circuit well.

My guess is that this is just because as the pulldown circuit increases in size, it just becomes too large of a load for the M-S Latch to turn on in a reasonable time. If the pulldown circuit does not turn on, there is little to no difference in the sense and dummy lines regardless if an error is detected.

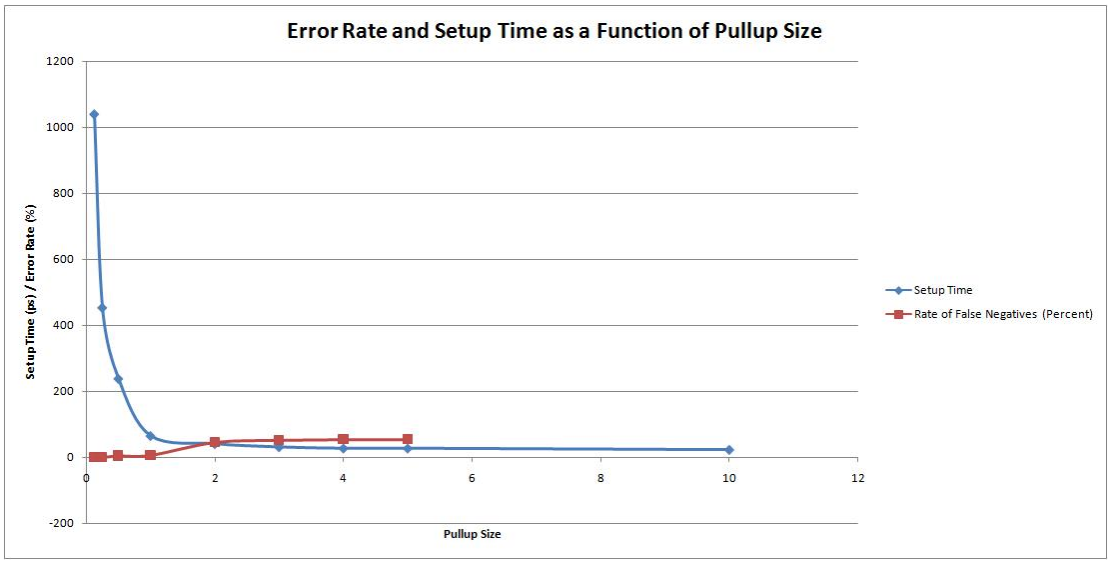

Figure 6.4: Franco Detector Error Rate and Setup Time as Dependent on Error Transistor Sizing

I confirmed this by looking at the setup times required for the MSC Cell. As the cell pulldown circuit reached 100λ, the setup time reached 500ps, a very high figure. The absolute longest setup time occurred at 130nm, 180 degrees C, and 100λ pulldown size. This means that a much longer time is required for a 100λ cell than a 1λ cell to pull down.

6.4 Electric Simulation Results

After creating the circuit layouts in Electric, I exported them to Spice files in order to get a more accurate simulation of the circuit. In the following sections I compare the results of the hand done Spice simulations versus those created in spice using Electric. The setup time of Franco's Checker in Electric exhibits a reverse trend then that of the pure Spice files in regards to the feature size. As the feature size increases, the setup time decreases. The setup time also appears to be independent of temperature. The minimum setup time is at 32nm and is 300pS. This is much larger than that of the simulation files I created, but this is due to the extra lump sum capacitances and

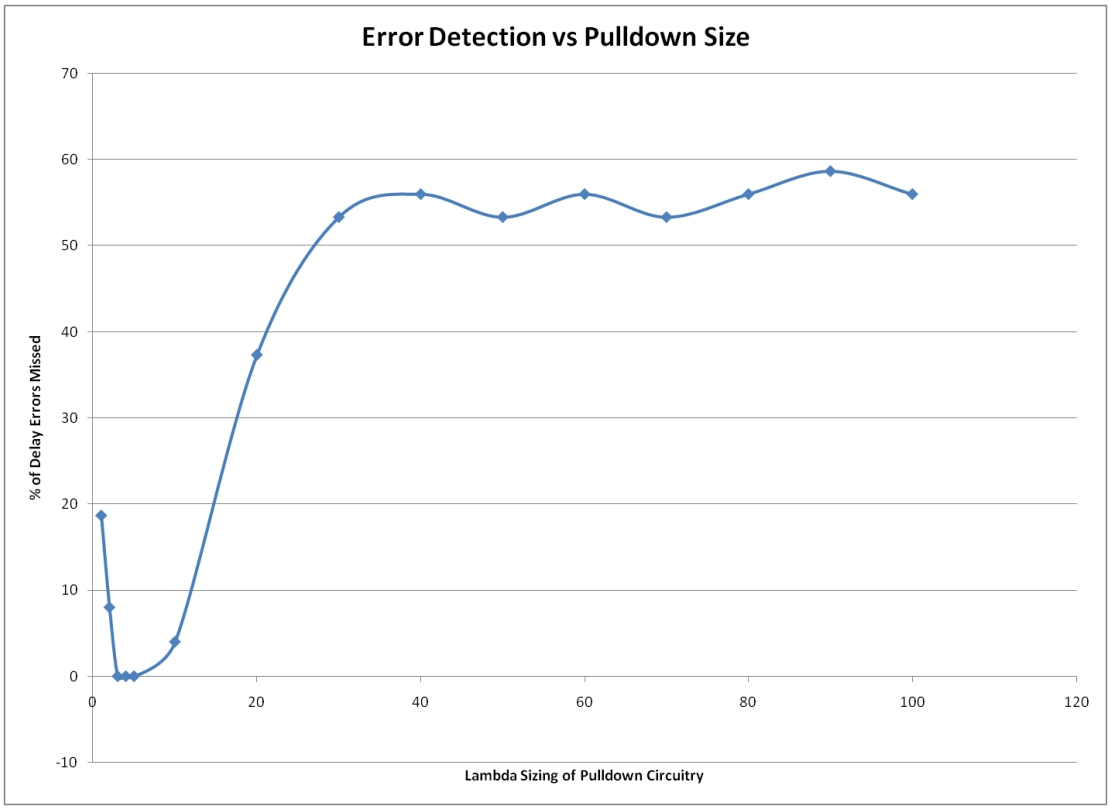

Figure 6.5: MSC Cell Error Detection Rate as Pulldown Circuit Size Increases

resistances on the wires that are ignored in the simulations. A graph depicting this is shown in Figure 6.6.

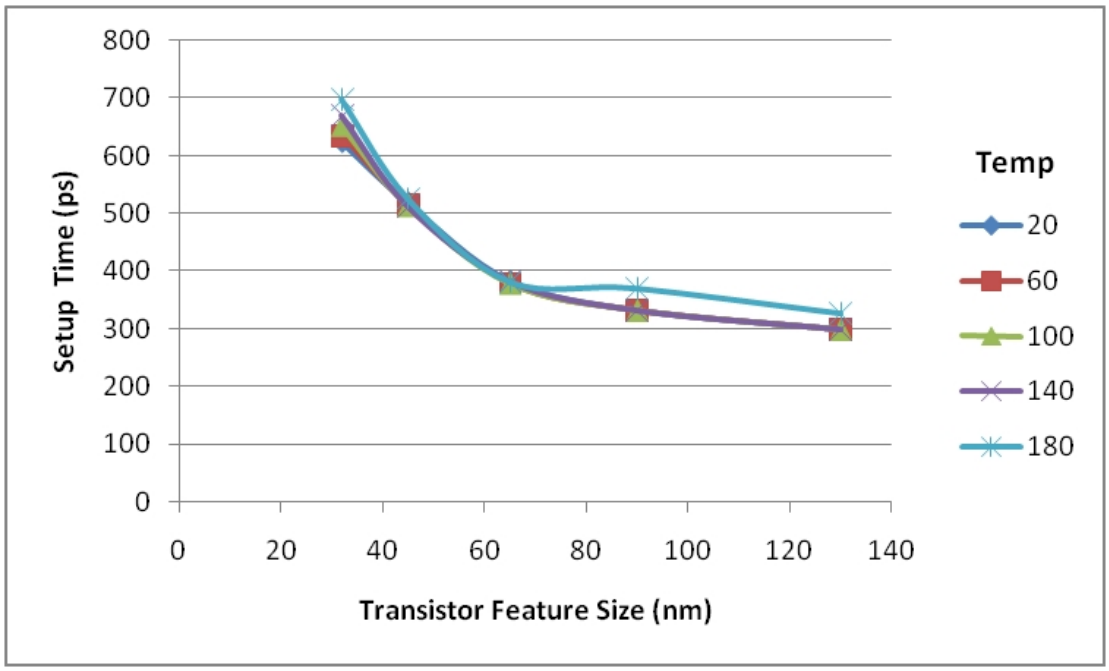

Figure 6.6: Franco Stability Detector Setup Time as a Function of Feature Size

Chapter 7

Conclusion

Fault tolerance is an unfortunately neccesity in modern day circuits. When there are over 200 million transistors in a circuit, it is a near certainty that one or more of the elements in the circuit will fail, and more will fail in the future. To deal with this problem requires the research and development of circuits such as those presented in this report. These stability checkers are very good at finding delay errors. With additional circuitry, they can even be used to correct problems in real time. As presented, Franco's detector is able to handle error detection while the clock is high and actually saves a transistor. Franco's detector also works under a number of temperatures and clock speeds, but when using said detector, it should be sized for worst case as there is a large variation in setup time and error detection as the stimuli change. Yada's MSC Cell is better, exhibiting little to no variation under multiple stimuli in performance, so long as it is sized correctly for the latch.

It is good to remember that error detection does not come free. These detectors take a good deal of routing, for example, the MSC Cell requires a dummy and sense line to each cell and a sense amplifier at the end while Franco's latch requires some sort of threshold detector and a priority encoder. While stability detectors are not ideal solutions to the problem of delay detection, they are functional circuit blocks and can be dropped in easily to any circuit in which unchecked delays are unacceptable.

Bibliography

[1] A. Balijepalli, S. Sinha, and Y. Cao, "Compact modeling of carbon nanotube transistor for early stage process-design exploration," in *ISLPED '07: Proceedings of the 2007 international symposium on Low power electronics and design.* New York, NY, USA: ACM, 2007, pp. 2–7.

[2] Y. Cao, T. Sato, M. Orshansky, D. Sylvester, and C. Hu, "New paradigm of predictive mosfet and interconnect modeling for early circuit simulation," 2000, pp. 201–204.

[3] P. Franco and E. McCluskey, "On-line delay testing of digital circuits," Apr 1994, pp. 167–173.

[4] A. Virupakshia and V. Pratapa Reddy, "A simple random test procedure for detection of single intermittent fault in combinational circuits," *Computers, IEEE Transactions on*, vol. C-32, no. 6, pp. 594–597, June 1983.

[5] S. Yada, B. Amrutur, and R. A. Parekhji, "Modified stability checking for on-line error detection," Jan. 2007, pp. 787–792.

[6] W. Zhao and Y. Cao, "New generation of predictive technology model for sub-45nm design exploration," in *ISQED '06: Proceedings of the 7th International Symposium on Quality Electronic Design.* Washington, DC, USA: IEEE Computer Society, 2006, pp. 585–590.

Chapter 8

Appendices

Inverter Characterization Script

```
Option Explicit
DIM fso, GuyFile,TransFile,TransCSVFile
Dim arrFileLines(),temp
Dim Command
Dim circuit,output,cmd,i,j
Dim arrValidLines()
Dim k
Dim myArray
Dim raw
Dim temperature
Dim h, csvfile,tempfile
Dim l
Dim valid
Dim models,directory
Dim ErrorFile
Dim errordetect
Dim path
Dim errortime
Dim maxvolt, minvolt, vtl, vth, vtlout, vthout
path = "I:\spice\SeniorProject\CircuitRuns\"
temperature = Array(20,60,100,140,180)
models = Array(32,45,65,90,130)

Set fso = CreateObject("Scripting.FileSystemObject")
Set ErrorFile = fso.CreateTextFile("ErrorFile.txt",True)
Set TransFile = fso.CreateTextFile("Transistor_Characteristics.txt",True)
Set TransCSVFile = fso.CreateTextFile("Transistor_Characteristics.csv",True)
TransCSVFile.WriteLine("Transistor Feature Size,Temperature,Max Voltage, Min Voltage, High to Low, Low to High")
For l = 0 to UBound(models)
For h = 0 to UBound(temperature)
tempfile = path & "test.cir"
Set fso = CreateObject("Scripting.FileSystemObject")
Set GuyFile = fso.CreateTextFile(tempfile, True)
GuyFile.WriteLine("*Stability FF")
GuyFile.WriteLine("Vin 2 0 DC 1V")
GuyFile.WriteLine("Vcc 1 0 DC 1V")
```

```
GuyFile.WriteLine("")
GuyFile.WriteLine("M1 3 2 1 1 pmos L=" & models(l) & "nm W=" & models(l)/2*6 & "nm")
GuyFile.WriteLine("M2 3 2 0 0 nmos L=" & models(l) & "nm W=" & models(l)/2*3 & "nm")
GuyFile.WriteLine("C1 3 0 1pF")
GuyFile.WriteLine(".options TNOM=" & temperature(h) & " TEMP=" & temperature(h))
GuyFile.WriteLine(".print dc v(3)")
GuyFile.WriteLine(".dc vin 0 1 0.00001")
GuyFile.WriteLine(".include " & models(l) & "nm_bulk.model1")
GuyFile.WriteLine(".end")
GuyFile.Close
csvfile = path & temperature(h) & ".csv"
Set Command = WScript.CreateObject("WScript.Shell")
cmd = "%comspec% /c sed s/" & chr(34) & "tnom    = 27" & chr(34) & "/tnom=" & temperature(h)
& "/ " & models(l) & "nm_bulk.model > " & models(l) & "nm_bulk.model1"
Command.Run(cmd)
circuit = path & "test.cir"
output = path & "test.out"
raw = path & "test.raw"
cmd = path & "scad3.exe -b " & circuit
Command.Run (cmd)
WScript.Sleep(3500)
'Convert raw to txt
cmd = "ltsputil.exe -x0 test.raw test.out " & chr(34) & "%15.7e" & chr(34) &
" " & chr(34) & "," & chr(34) & " " & chr(34) & "Time,Voltage" & chr(34) & " *"
Command.Run (cmd)
'Open Output File
WScript.Sleep(500)
Set GuyFile = fso.OpenTextFile(output,1)
'Read in file into arrFileLines

i = 0
Do Until GuyFile.AtEndOfStream
Redim Preserve arrFileLines(i)
arrFileLines(i) = GuyFile.ReadLine
i = i + 1
Loop
GuyFile.Close

j = i - 1
k = 0
temp = 0
valid = 0
For i = 0 to j
'get rid of excess spaces
arrFileLines(i) = Trim(arrFileLines(i))
Redim Preserve arrValidLines(k)
arrValidLines(k) = arrFileLines(i)
k = k + 1
Next
'FOR DEBUGGING PURPOSES, PRINT OUT arrVALIDLINES
'Set GuyFile = fso.CreateTextFile("temp.txt", True)
'j = k - 1
'for i = 0 to j
'myArray = Split(arrValidLines(i),",")
'myArray(1) = Trim(myArray(1))
'GuyFile.WriteLine(myArray(0) & "," & myArray(1))
'next
'GuyFile.Close
'END DEBUG

'Check to see if it detected the error properly
j = k - 1
k = 0
errordetect = 0
```

```
For k = 0 to j
myArray = Split(arrValidLines(k),",")
myArray(1) = Trim(MyArray(1))
If k > 50 Then
'are we below threshold?
If myArray(1) < 0.3 Then
If errordetect = 0 Then
errortime = myArray(0)
errordetect = 1
End If
End If
'Does it reset after finding an error?
If (errordetect = 1) Then
If myArray(1) > 0.5 Then
errordetect = 2
End If
End If
End If
Next
If (errordetect = 2) Then
ErrorFile.WriteLine(models(l) & "nm_bulk at " & temperature(h)
& " degrees C Error Detected at " & errortime & " seconds,PASS")
End If
If (errordetect = 0) Then
ErrorFile.WriteLine(models(l) & "nm_bulk at " & temperature(h)
& " degrees C Error Not Detected,FAIL")
End If
If (errordetect = 1) Then
ErrorFile.WriteLine(models(l) & "nm_bulk at "
& temperature(h) & " degrees C Stability Detector Broken,FAIL")
End If
'Export to a CSV
k = 0
Set GuyFile = fso.CreateTextFile(csvfile, True)
For k = 0 to j
myArray = Split(arrValidLines(k),",")
myArray(1) = Trim(myArray(1))
GuyFile.WriteLine(myArray(0) & "," & myArray(1))
Next
'myArray has voltages, 0 = input, 1 = output


'Getting Inverter Characteristics

maxvolt = 0.000
minvolt = 1.000
vtl = 0.000
vth = 0.000
'Casting to Double
maxvolt = CDbl(maxvolt)
minvolt = CDbl(minvolt)
vtl = CDbl(vtl)
vth = CDbl(vth)
'MsgBox IsNumeric(maxvolt)
'MsgBox IsNumeric(minvolt)
'MsgBox IsNumeric(vtl)
'MsgBox IsNumeric(vth)
For k = 0 to j
myArray = Split(arrValidLines(k),",")
myArray(1) = Trim(myArray(1))

myArray(0) = round(myArray(0),3)
myArray(1) = round(myArray(1),3)
'Find the max voltage
```

```
If myArray(1) > maxvolt Then
maxvolt = myArray(1)
End If

'Find the min voltage
If myArray(1) < minvolt Then
minvolt = myArray(1)
End If
Next
'Assuming 10% margins
vthout = 0.9*maxvolt
vtlout = 1.1*minvolt
if minvolt = 0 then
vtlout = 0.2
end if
vthout = round(vthout,3)
vtlout = round(vtlout,3)

For k = 0 to j
myArray = Split(arrValidLines(k),",")
myArray(1) = Trim(myArray(1))

myArray(0) = round(myArray(0),3)
myArray(1) = round(myArray(1),3)

'Find Vtl
If myArray(1) < vtlout*1.05 Then
If myArray(1) > 0.95*vtlout Then
vtl = myArray(0)
End If
End If

'Find Vth
If myArray(1) < vthout*1.05 Then
If myArray(1) > 0.95*vthout Then
vth = myArray(0)
End If
End If
Next
maxvolt = round(maxvolt,3)
minvolt = round(minvolt,3)
vtl = round(vtl,3)
vth = round(vth,3)
TransFile.WriteLine("Transistor Size: " & models(l) & "nm")
TransFile.WriteLine("Temperature: " & temperature(h) & " C")
TransFile.WriteLine("Max Inverter Voltage: " & maxvolt)
TransFile.WriteLine("Min Inverter Voltage: " & minvolt)
TransFile.WriteLine("High to Low Transition Voltage: " & vtl)
TransFile.WriteLine("Low to High Transition Voltage: " & vth)
TransFile.WriteLine("")

TransCSVFile.WriteLine(models(l) & "," & temperature(h)
& "," & maxvolt & "," & minvolt & "," & vtl & "," & vth)
'End Transistor Characteristics
GuyFile.Close
Set GuyFile = Nothing
directory = path & models(l) & "nm_bulk\"
MD(directory)
fso.MoveFile csvfile, directory
fso.DeleteFile(output)
fso.DeleteFile(circuit)
fso.DeleteFile(raw)
fso.DeleteFile(models(l) & "nm_bulk.model1")
Set fso = Nothing
```

```
Next
Next
TransFile.Close
Set TransFile = Nothing
ErrorFile.Close
Set ErrorFile = Nothing
Function MD(strDirName)
     Dim lFSObj
     MD = False
     ' First See if it already exists
     set lFSObj = Wscript.CreateObject("Scripting.FileSystemObject")
If lFSObj.FolderExists(strDirName) Then
MD = True
Else
lFSObj.CreateFolder(strDirName)
MD = True
End If
     Set lFSObj = Nothing
End Function
```

Franco and McCluskey Batch Script

```
Option Explicit
Dim fso, GuyFile
Dim arrFileLines(),temp
Dim Command
Dim circuit,output,cmd,i,j
Dim arrValidLines()
Dim k
Dim myArray
Dim raw
Dim temperature
Dim h, csvfile,tempfile
Dim l,m,n
Dim valid
Dim models,directory
Dim ErrorFile
Dim errordetect
Dim path
Dim errortime
Dim clockspeed,clockperiod,inputtimeoffset
Dim triggertime
path = "I:\spice\SeniorProject\CircuitRuns\McCluskeyRuns\"
temperature = Array(20,60,100,140,180)
models = Array(32,45,65,90,130)
'Max speed 10GHz (approx. double longest setup time), Slowest 130nm at 180 deg C
'1x minimum size
'clockspeed = Array(10000000000,5000000000,1000000000)
'1/8x minimum size
clockspeed = Array(375000000,187500000,37500000)
'set the input signal time offset in pico-seconds from when the clock triggers
inputtimeoffset = Array(-1330)
Set fso = CreateObject("Scripting.FileSystemObject")
Set ErrorFile = fso.CreateTextFile("ErrorFile.csv",True)
ErrorFile.WriteLine("Transistor Feature Size,Clock Speed,Temperature
,Setup or Hold Violation (0 = Setup 1 = Hold),Error Detected? (0 = yes 1 = no)")
For l = 0 to UBound(models)
For m = 0 to UBound(clockspeed)
For h = 0 to UBound(temperature)
For n = 0 to UBound(inputtimeoffset)
'clock period in pico-seconds
clockperiod = (1 / clockspeed(m))*1000000000000
```

```
tempfile = path & "test.cir"
Set fso = CreateObject("Scripting.FileSystemObject")
Set GuyFile = fso.CreateTextFile(tempfile, True)
GuyFile.WriteLine("*Stability FF")
GuyFile.WriteLine(".INCLUDE " & models(l) & "nm_bulk.model1")
GuyFile.WriteLine("Vclk clk 0 PULSE(1 0  0ns .1ps .1ps " & 0.5*clockperiod & "ps " & clockperiod & "ps)")
GuyFile.WriteLine("Vcc  Vcc 0 DC 1V")
triggertime = 0.5*clockperiod + inputtimeoffset(n)
temp = triggertime + 0.000001
'Test During the Setup and Hold Times (hold times being defined as the time right after the clock goes high)
GuyFile.WriteLine("Vin 1 0 PWL(0ps 0V " & triggertime & "ps 0V " & temp & "ps 1V)")
GuyFile.WriteLine("*error transistor")
GuyFile.WriteLine("M1 5 11 Vcc Vcc pmos L=" & models(l) * 2 & "nm W=" & models(l)/2*3 & "nm")
GuyFile.WriteLine("*inverters")
GuyFile.WriteLine("X_inv1 5 0 1   2  Vcc 0 Inverter")
GuyFile.WriteLine("X_inv2 5 0 2   3  Vcc 0 Inverter")
GuyFile.WriteLine("X_inv3 5 0 3   4  Vcc 0 Inverter")
GuyFile.WriteLine("X_tgate1 4 2  8 11 Vcc 0  TGate")
GuyFile.WriteLine("X_clkinv1 Vcc 0 clk 8 Vcc 0 Inverter")
GuyFile.WriteLine("X_clkinv2 Vcc 0 8 11  Vcc 0 Inverter")
GuyFile.WriteLine("* TGate Input Output P_gate N_gate Pwr Gnd")
GuyFile.WriteLine(".SUBCKT TGate 1 3 2 4 vcc gnd P=" & models(l)/2*6 & "nm N=" & models(l)/2*3 & "nm")
GuyFile.WriteLine(" M1 3 2 1 vcc pmos L=" & models(l) & "nm W='P'" )
GuyFile.WriteLine(" M2 1 4 3 gnd nmos L=" & models(l) & "nm W='N'" )
GuyFile.WriteLine(".ENDS")
GuyFile.WriteLine("* Inverter Vdd Vss Input Output Pwr Gnd")
GuyFile.WriteLine(".SUBCKT Inverter  1 4 2 3 vcc gnd P=" & models(l)/2*6 & "nm N=" & models(l)/2*3 & "nm")
GuyFile.WriteLine(" M1 3 2 1 vcc pmos L=" & models(l) & "nm W='P'" )
GuyFile.WriteLine(" M2 3 2 4 gnd nmos L=" & models(l) & "nm W='N'" )
GuyFile.WriteLine(".ENDS")
GuyFile.WriteLine(".options TNOM=" & temperature(h) & " TEMP=" & temperature(h))
GuyFile.WriteLine(".print tran V(5)")
If clockspeed(m) = 37500000 Then
GuyFile.WriteLine(".tran 1ps 60ns")
Else
GuyFile.WriteLine(".tran .1ps 10000ps")
End If
GuyFile.WriteLine(".END")
GuyFile.Close
csvfile = path & "" & temperature(h) & ".csv"
Set Command = WScript.CreateObject("WScript.Shell")
cmd = "%comspec% /c sed s/" & chr(34) & "tnom    = 27" & chr(34) & "/tnom="
& temperature(h) & "/ " & models(l) & "nm_bulk.model > " & models(l) & "nm_bulk.model1"
Command.Run(cmd)
circuit = path & "test.cir"
output = path & "test.out"
raw = path & "test.raw"
cmd = path & "scad3.exe -b " & circuit
Command.Run (cmd)
WScript.Sleep(5000)
'Convert raw to txt
cmd = "ltsputil.exe -x0 test.raw test.out " & chr(34) & "%15.7e" & chr(34)
& " " & chr(34) & "," & chr(34) & " " & chr(34) & "Time,Voltage" & chr(34) & " *"
Command.Run (cmd)
'Open Output File
WScript.Sleep(200)
Set GuyFile = fso.OpenTextFile(output,1)
'Read in file into arrFileLines
i = 0
Do Until GuyFile.AtEndOfStream
Redim Preserve arrFileLines(i)
arrFileLines(i) = GuyFile.ReadLine
i = i + 1
Loop
```

```
GuyFile.Close

j = i - 1
k = 0
temp = 0
valid = 0
For i = 0 to j
'get rid of excess spaces
arrFileLines(i) = Trim(arrFileLines(i))
Redim Preserve arrValidLines(k)
arrValidLines(k) = arrFileLines(i)
k = k + 1
Next
'FOR DEBUGGING PURPOSES, PRINT OUT arrVALIDLINES
'Set GuyFile = fso.CreateTextFile("temp.txt", True)
'j = k - 1
'for i = 0 to j
'myArray = Split(arrValidLines(i),",")
'myArray(1) = Trim(myArray(1))
'GuyFile.WriteLine(myArray(0) & "," & myArray(1))
'next
'GuyFile.Close
'END DEBUG

'Check to see if it detected the error properly
j = k - 1
k = 0
errordetect = 0
For k = 0 to j
myArray = Split(arrValidLines(k),",")
myArray(1) = Trim(MyArray(1))
If k > 50 Then
'are we below threshold?
If myArray(1) < 0.7 Then
If errordetect = 0 Then
errortime = myArray(0)
errordetect = 1
End If
End If
'Does it reset after finding an error?
If (errordetect = 1) Then
If myArray(1) > 0.5 Then
errordetect = 2
End If
End If
End If
Next
If errordetect <> 2 Then
'Error not detected
'Lets export all our information to an error file
'Transistor Feature Size,Clock Speed,Temperature,Setup or Hold Violation,Error Detected?
'n = 0 is setup, n = 1 is hold
'Error Detected? 0 is yes, 1 is no
ErrorFile.WriteLine(models(l) & "," & clockspeed(m) & "," & temperature(h) & "," & n & ",1")

Else
'Error detected
ErrorFile.WriteLine(models(l) & "," & clockspeed(m) & "," & temperature(h) & "," & n & ",0")
End If
'Export to a CSV
k = 0
Set GuyFile = fso.CreateTextFile(csvfile, True)
For k = 0 to j
myArray = Split(arrValidLines(k),",")
```

```
myArray(1) = Trim(myArray(1))
GuyFile.WriteLine(myArray(0) & "," & myArray(1))
Next
GuyFile.Close
Set GuyFile = Nothing
'make the directory heirarchy
'Feature Size
directory = path & models(l) & "nm_bulk\"
MD(directory)
'Clock Speed
directory = path & models(l) & "nm_bulk\" & clockspeed(m) & "Hz\"
MD(directory)
'Setup or Hold Time Violation
If CDbl(triggertime)  > 0.5*clockperiod Then
'Hold Time Violation
directory =  path & models(l) & "nm_bulk\" & clockspeed(m) & "Hz\" & "Hold_Violation\"
Else
directory =  path & models(l) & "nm_bulk\" & clockspeed(m) & "Hz\" & "Setup_Violation\"
End If
MD(directory)
fso.MoveFile csvfile, directory
fso.DeleteFile(output)
fso.DeleteFile(circuit)
fso.DeleteFile(raw)
fso.DeleteFile(models(l) & "nm_bulk.model1")
Set fso = Nothing
Next
Next
Next
Next
ErrorFile.Close
Set ErrorFile = Nothing


Function MD(strDirName)
     Dim lFSObj
     MD = False
     ' First See if it already exists
     set lFSObj = Wscript.CreateObject("Scripting.FileSystemObject")
If lFSObj.FolderExists(strDirName) Then
MD = True
Else
lFSObj.CreateFolder(strDirName)
MD = True
End If
     Set lFSObj = Nothing
End Function
```

Franco and McCluskey Setup Time Calculator Script

```
Option Explicit
Dim fso, GuyFile
Dim arrFileLines(),temp
Dim Command
Dim circuit,output,cmd,i,j
Dim arrValidLines()
Dim k
Dim myArray
Dim raw
Dim temperature
Dim h, csvfile,tempfile
Dim l,m,n
```

```
Dim valid
Dim models,directory
Dim path
Dim errortime
Dim TransFile,OutputFile
Dim vth,vtl,found,setuptime

path = "I:\spice\SeniorProject\CircuitRuns\McCluskeyRuns\"
temperature = Array(20,60,100,140,180)
models = Array(32,45,65,90,130)
Set fso = CreateObject("Scripting.FileSystemObject")
Set OutputFile = fso.CreateTextFile("McCluskey_Setup_Times.csv",True)
OutputFile.WriteLine("Transistor Feature Size (nm),Temperature (C),Setup Time (ps)")
For l = 0 to UBound(models)
For h = 0 to UBound(temperature)
'Check both high to low and low to high transitions
For m = 0 to 1
tempfile = path & "test.cir"
Set fso = CreateObject("Scripting.FileSystemObject")
Set GuyFile = fso.CreateTextFile(tempfile, True)
GuyFile.WriteLine("*Stability FF")
GuyFile.WriteLine(".INCLUDE " & models(l) & "nm_bulk.model1")

GuyFile.WriteLine("Vclk clk 0 DC 0V")
GuyFile.WriteLine("Vcc  Vcc 0 DC 1V")
GuyFile.WriteLine("Vin 1 0 PWL(0ns " & m  & "V 1ps " & m & "V 1.00001ps " & abs(m - 1) & "V)")
GuyFile.WriteLine("*error transistor")
GuyFile.WriteLine("M1 5 11 Vcc Vcc pmos L=" & models(l) & "nm W=" & models(l)/2*3*4 & "nm")
GuyFile.WriteLine("*inverters")
GuyFile.WriteLine("X_inv1 5 0 1   2  Vcc 0 Inverter")
GuyFile.WriteLine("X_inv2 5 0 2   3  Vcc 0 Inverter")
GuyFile.WriteLine("X_inv3 5 0 3   4  Vcc 0 Inverter")
GuyFile.WriteLine("X_tgate1 4 2  8 11 Vcc 0  TGate")
GuyFile.WriteLine("X_clkinv1 Vcc 0 clk 8 Vcc 0 Inverter")
GuyFile.WriteLine("X_clkinv2 Vcc 0 8 11  Vcc 0 Inverter")
GuyFile.WriteLine("* TGate Input Output P_gate N_gate Pwr Gnd")
GuyFile.WriteLine(".SUBCKT TGate 1 3 2 4 vcc gnd P=" & models(l)/2*6 & "nm N=" & models(l)/2*3 & "nm")
GuyFile.WriteLine(" M1 3 2 1 vcc pmos L=" & models(l) & "nm W='P'" )
GuyFile.WriteLine(" M2 1 4 3 gnd nmos L=" & models(l) & "nm W='N'" )
GuyFile.WriteLine(".ENDS")
GuyFile.WriteLine("* Inverter Vdd Vss Input Output Pwr Gnd")
GuyFile.WriteLine(".SUBCKT Inverter  1 4 2 3 vcc gnd P=" & models(l)/2*6 & "nm N=" & models(l)/2*3 & "nm")
GuyFile.WriteLine(" M1 3 2 1 vcc pmos L=" & models(l) & "nm W='P'" )
GuyFile.WriteLine(" M2 3 2 4 gnd nmos L=" & models(l) & "nm W='N'" )
GuyFile.WriteLine(".ENDS")
GuyFile.WriteLine(".options TNOM=" & temperature(h) & " TEMP=" & temperature(h))
GuyFile.WriteLine(".print tran V(4)")
GuyFile.WriteLine(".tran .01ps 2ns")
GuyFile.WriteLine(".END")
GuyFile.Close
csvfile = path & "" & temperature(h) & ".csv"

Set Command = WScript.CreateObject("WScript.Shell")
cmd = "%comspec% /c sed s/" & chr(34) & "tnom    = 27" & chr(34) & "/tnom="
& temperature(h) & "/ " & models(l) & "nm_bulk.model > " & models(l) & "nm_bulk.model1"
        Command.Run(cmd)
circuit = path & "test.cir"
output = path & "test.out"
raw = path & "test.raw"
cmd = path & "scad3.exe -b " & circuit
Command.Run (cmd)
WScript.Sleep(1500)
'Convert raw to txt
cmd = "ltsputil.exe -x0 test.raw test.out " & chr(34) & "%15.7e" & chr(34) & " "
```

```
& chr(34) & "," & chr(34) & " " & chr(34) & "Time,Voltage" & chr(34) & " *"
Command.Run (cmd)
'Open Output File
WScript.Sleep(100)
Set GuyFile = fso.OpenTextFile(output,1)
'Read in file into arrFileLines
i = 0
Do Until GuyFile.AtEndOfStream
Redim Preserve arrFileLines(i)
arrFileLines(i) = GuyFile.ReadLine
i = i + 1
Loop
GuyFile.Close

j = i - 1
k = 0
valid = 0
For i = 0 to j
'get rid of excess spaces
arrFileLines(i) = Trim(arrFileLines(i))
Redim Preserve arrValidLines(k)
arrValidLines(k) = arrFileLines(i)
k = k + 1
Next
'FOR DEBUGGING PURPOSES, PRINT OUT arrVALIDLINES
'Set GuyFile = fso.CreateTextFile("temp.txt", True)
'j = k - 1
'for i = 0 to j
'myArray = Split(arrValidLines(i),",")
'myArray(1) = Trim(myArray(1))
'GuyFile.WriteLine(myArray(0) & "," & myArray(1))
'next
'GuyFile.Close
'END DEBUG

'Measure the Setup Time
j = k - 1
k = 0
'Transistor Characterization Data
'Transistor Feature Size,Temperature,Max Voltage,Min Voltage,High to Low, Low to High
Set TransFile = fso.OpenTextFile("Transistor_Characteristics.csv")
i = 0
Do Until TransFile.AtEndOfStream
Redim Preserve arrFileLines(i)
arrFileLines(i) = TransFile.ReadLine
i = i + 1
Loop
TransFile.Close
'Find Which Line Corresponds to the Transistor and Temp Being Used
'Skip the 1st line cause its all text
For k = 1 to i - 1
temp = split(arrFileLines(k),",")
'msgbox temp(0) & " = " & models(l)
'msgbox temp(1) & " = " & temperature(h)
'Make them numeric
For n = 0 to 5
temp(n) = CDbl(temp(n))
Next
If temp(0) = models(l) Then
If temp(1) = temperature(h) Then
'We got a match!
vtl = temp(4)
vth = temp(5)
End If
```

```
End If
Next

'm = 1 is low to high, m = 0 is high to low
found = 0
setuptime = 0
For k = 0 to j
myArray = Split(arrValidLines(k),",")
myArray(1) = Trim(MyArray(1))
'Convert to pico-seconds
myArray(0) = myArray(0) * 1000000000000
myArray(0) = round(myArray(0),6)
myArray(1) = round(myArray(1),3)
If m = 0 Then
'are we below threshold?
If myArray(1) <  vth Then
If found = 0 Then
found = 1
setuptime = myArray(0) - 1
End If
End If
End If
If m = 1 Then
'are we above threshold?
If myArray(1) > vtl Then
If found = 0 Then
found = 1
'the longer time between the two is going to be the setup time
n = myArray(0) - 1
If n > setuptime Then
setuptime = myArray(0) - 1
End If
End If
End If
End If
Next
If m = 1 Then
OutputFile.WriteLine(models(l) & "," & temperature(h) & "," & setuptime)
End If
fso.DeleteFile(output)
fso.DeleteFile(circuit)
fso.DeleteFile(raw)
fso.DeleteFile(models(l) & "nm_bulk.model1")

Next
Next
Next
OutputFile.Close
Set OutputFile = Nothing
Set fso = Nothing

Function MD(strDirName)
     Dim lFSObj
     MD = False
     ' First See if it already exists
     set lFSObj = Wscript.CreateObject("Scripting.FileSystemObject")
If lFSObj.FolderExists(strDirName) Then
MD = True
Else
lFSObj.CreateFolder(strDirName)
MD = True
End If
     Set lFSObj = Nothing
End Function
```

Modified Stability Checker Batch Script

```
Option Explicit
Dim fso, GuyFile
Dim arrFileLines(),temp
Dim Command
Dim circuit,output,cmd,i,j
Dim arrValidLines()
Dim k
Dim myArray
Dim raw
Dim temperature
Dim h, csvfile,tempfile
Dim l,m,n
Dim valid
Dim models,directory
Dim ErrorFile
Dim path
Dim clockspeed,clockperiod,inputtimeoffset
Dim triggertime
Dim difference,maxdiff
Dim clockdelay
path = "I:\spice\SeniorProject\CircuitRuns\MSCLatchRuns\"
temperature = Array(20,60,100,140,180)
models = Array(32,45,65,90,130)
clockspeed = Array(2500000000,1250000000,250000000)
'clock delay in pico-seconds
clockdelay = 40
'set the input signal time offset in pico-seconds from the 2nd rising edge of the clock
inputtimeoffset = Array(-10,10)
Set fso = CreateObject("Scripting.FileSystemObject")
Set ErrorFile = fso.CreateTextFile("ErrorFile.csv",True)
ErrorFile.WriteLine("Transistor Feature Size,Clock Speed
,Temperature,Setup or Hold Violation (0 = Setup 1 = Hold),Sense and Dummy Error")
For l = 0 to UBound(models)
For m = 0 to UBound(clockspeed)
For h = 0 to UBound(temperature)
For n = 0 to UBound(inputtimeoffset)
'clock period in pico-seconds
clockperiod = (1 / clockspeed(m))*1000000000000
tempfile = path & "test.cir"
Set fso = CreateObject("Scripting.FileSystemObject")
Set GuyFile = fso.CreateTextFile(tempfile, True)
triggertime = 1.5*clockperiod + inputtimeoffset(n)
temp = triggertime + 0.0001
'Test During the Setup and Hold Times (hold times being defined as the time right after the clock goes high)
GuyFile.WriteLine("*MSC with MS Latch")
GuyFile.WriteLine(".INCLUDE " & models(l) & "nm_bulk.model1")
GuyFile.WriteLine("*MSLatch Input Output invOutput Clk Vcc Gnd")
GuyFile.WriteLine("*Inverter Vcc Gnd Input Output")
GuyFile.WriteLine("*TGate Input Output P_gate N_gate")
GuyFile.WriteLine("*MSCCell clk vcc co ffo sense dummy")
'co = combinational out, ffo = flip-flop out
'inputs: co, clk, vcc outputs: sense,dummy
GuyFile.WriteLine("Vcc vcc 0 DC 1V")
GuyFile.WriteLine("Vclk clk 0 PULSE(1 0  0ns .1fs .1fs " & 0.5*clockperiod & "ps " & clockperiod & "ps)")
GuyFile.WriteLine("Vclkd clkd 0 PULSE(1 0  0ns .1fs .1fs " & 0.5*clockperiod+clockdelay & "ps " & clockperiod & "ps)")
GuyFile.WriteLine("Vco co 0 PWL(0ns 0V " & triggertime & "ps 0V " & temp & "ps 1V)")
GuyFile.WriteLine("X_mslatch co ffo 1 clk Vcc 0 MSLatch")
GuyFile.WriteLine("X_msccell clkd vcc co ffo sense dummy MSCCell")
GuyFile.WriteLine(".options TNOM=" & temperature(h) & " TEMP=" & temperature(h))
GuyFile.WriteLine(".op")
GuyFile.WriteLine(".probe")
GuyFile.WriteLine(".print tran V(dummy) V(sense)")
```

```
GuyFile.WriteLine(".tran 1ps 1.2ns")
GuyFile.WriteLine(".SUBCKT MSCCell clkd vcc co ffo sense dummy")
GuyFile.WriteLine("*Dummy Pullup")
GuyFile.WriteLine("Mud dummy clkd vcc vcc pmos L=" & models(l) & "nm W=" & models(l)/2*3 & "nm")
GuyFile.WriteLine("*Sense Pullup")
GuyFile.WriteLine("Mus sense clkd vcc vcc pmos L=" & models(l) & "nm W=" & models(l)/2*3 & "nm")
GuyFile.WriteLine("*Sense Pulldown")
GuyFile.WriteLine("Mds 1 clkd 0 0 nmos L=" & models(l) & "nm W=" & models(l)/2*6 & "nm")
GuyFile.WriteLine("*Dummy Pulldown")
GuyFile.WriteLine("Mdd 4 0 0 0 nmos L=" & models(l) & "nm W=" & models(l)/2*6 & "nm")
GuyFile.WriteLine("")
GuyFile.WriteLine("*Sense Cell")
GuyFile.WriteLine("M1  2  ffo  sense  sense  pmos L=" & models(l) & "nm W=" & models(l)/2*6 & "nm")
GuyFile.WriteLine("M2  2  co  1  0  nmos L=" & models(l) & "nm W=" & models(l)/2*3 & "nm")
GuyFile.WriteLine("M3 3 co  sense vcc pmos L=" & models(l) & "nm W=" & models(l)/2*6 & "nm")
GuyFile.WriteLine("M4 3 ffo 1 0 nmos L=" & models(l) & "nm W=" & models(l)/2*3 & "nm")
GuyFile.WriteLine("*Dummy Cell")
GuyFile.WriteLine("M5 6 ffo dummy vcc pmos L=" & models(l) & "nm W=" & models(l)/2*6 & "nm")
GuyFile.WriteLine("M6 6 co 4 0 nmos L=" & models(l) & "nm W=" & models(l)/2*3 & "nm")
GuyFile.WriteLine("M7 5 co dummy vcc pmos L=" & models(l) & "nm W=" & models(l)/2*6 & "nm")
GuyFile.WriteLine("M8 5 ffo 4 0 nmos L=" & models(l) & "nm W=" & models(l)/2*3 & "nm")
GuyFile.WriteLine(".ends")

GuyFile.WriteLine("*MSLatch Input Output invOutput Clk Vcc Gnd")
GuyFile.WriteLine(".SUBCKT MSLatch 1 7 13 clk Vcc Gnd")
GuyFile.WriteLine("X_inv1 Vcc Gnd 1   2   Inverter")
GuyFile.WriteLine("X_inv2 Vcc Gnd 3   4   Inverter")
GuyFile.WriteLine("X_inv3 Vcc Gnd 5   6   Inverter")
GuyFile.WriteLine("X_inv4 Vcc Gnd 6   7   Inverter")
GuyFile.WriteLine("X_inv5 Vcc Gnd 4   9   Inverter")
GuyFile.WriteLine("X_inv6 Vcc Gnd 6   12  Inverter")
GuyFile.WriteLine("X_inv7 Vcc Gnd clk 8   Inverter")
GuyFile.WriteLine("X_inv8 Vcc Gnd 8   11  Inverter")
GuyFile.WriteLine("X_inv9 Vcc Gnd 12  13  Inverter")
GuyFile.WriteLine("X_tgate1 2 3 11 8  TGate")
GuyFile.WriteLine("X_tgate2 4 5 8 11  TGate")
GuyFile.WriteLine("X_tgate3 9 3 8 11  TGate")
GuyFile.WriteLine("X_tgate4 12 5 11 8 TGate")
GuyFile.WriteLine(".ends")
GuyFile.WriteLine("* Inverter Vcc Gnd Input Output")
GuyFile.WriteLine("* TGate Input Output P_gate N_gate")
GuyFile.WriteLine(".SUBCKT TGate 1 3 2 4")
GuyFile.WriteLine("M1 3 2 1 Vcc pmos L=" & models(l) & "nm W=" & models(l)/2*6 & "nm")
GuyFile.WriteLine("M2 1 4 3 0 nmos L=" & models(l) & "nm W=" & models(l)/2*3 & "nm")
GuyFile.WriteLine(".ENDS")
GuyFile.WriteLine("* Inverter Vcc Gnd Input Output")
GuyFile.WriteLine(".SUBCKT Inverter  1 4 2 3")
GuyFile.WriteLine("M1 3 2 1 Vcc pmos L=" & models(l) & "nm W=" & models(l)/2*6 & "nm")
GuyFile.WriteLine("M2 3 2 4 0 nmos L=" & models(l) & "nm W=" & models(l)/2*3 & "nm")
GuyFile.WriteLine(".ENDS")
GuyFile.WriteLine(".end")
GuyFile.Close

csvfile = path & "" & temperature(h) & ".csv"

Set Command = WScript.CreateObject("WScript.Shell")
cmd = "%comspec% /c sed s/" & chr(34) & "tnom    = 27" & chr(34)
& "/tnom=" & temperature(h) & "/ " & models(l) & "nm_bulk.model > " & models(l) & "nm_bulk.model1"
Command.Run(cmd)

circuit = path & "test.cir"
output = path & "test.out"
raw = path & "test.raw"
cmd = path & "scad3.exe -b " & circuit
```

```
Command.Run (cmd)
WScript.Sleep(3000)
'Convert raw to txt
cmd = "ltsputil.exe -x0 test.raw test.out " & chr(34) & "%15.7e" & chr(34) & " "
& chr(34) & "," & chr(34) & " " & chr(34) & "Time,Voltage" & chr(34) & " *"
Command.Run (cmd)
'Open Output File
WScript.Sleep(200)
Set GuyFile = fso.OpenTextFile(output,1)
'Read in file into arrFileLines
i = 0
Do Until GuyFile.AtEndOfStream
Redim Preserve arrFileLines(i)
arrFileLines(i) = GuyFile.ReadLine
i = i + 1
Loop
GuyFile.Close

j = i - 1
k = 0
temp = 0
valid = 0
For i = 0 to j
'get rid of excess spaces
arrFileLines(i) = Trim(arrFileLines(i))
Redim Preserve arrValidLines(k)
arrValidLines(k) = arrFileLines(i)
k = k + 1
Next
'FOR DEBUGGING PURPOSES, PRINT OUT arrVALIDLINES
'~ Set GuyFile = fso.CreateTextFile("temp.txt", True)
'~ j = k - 1
'~ for i = 0 to j
'~ myArray = Split(arrValidLines(i),",")
'~ myArray(1) = Trim(myArray(1))
'~ GuyFile.WriteLine(myArray(0) & "," & myArray(1) & "," & myArray(2))
'~ next
'~ GuyFile.Close
'END DEBUG

'Check to see if it detected the error properly
j = k - 1
k = 0
'Lets see our differences for debugging
'Set GuyFile = fso.CreateTextFile("Temp.txt",true)
maxdiff = 0
For k = 0 to j
myArray = Split(arrValidLines(k),",")
myArray(1) = Trim(MyArray(1))
myArray(2) = Trim(MyArray(2))
'Subtract dummy from sense and see if theres a difference
difference = (MyArray(1) - MyArray(2)) / MyArray(1)
difference = round(difference,2)
'There is going to be a difference between the sense and dummy lines, but hopefully it wont be much if theres no error
If MyArray(0) > 600*10^(-12) Then
If difference > maxdiff Then
'Store max difference between sense and dummy
maxdiff = difference
'ignore the error line difference if the difference is due to capactance
If MyArray(1) > 0.9 Then
If MyArray(2) > 0.9 Then
difference = 0
End If
End If
```

```
End If
'GuyFile.WriteLine(MyArray(0) & " " & difference)
End If
Next
'Transistor Feature Size,Clock Speed,Temperature,Setup or Hold Violation (0 = Setup 1 = Hold),sense/dummy error
ErrorFile.WriteLine(models(l) & "," & clockspeed(m) & "," & temperature(h) & "," & n & "," & maxdiff)
'GuyFile.Close
'Export to a CSV
k = 0
Set GuyFile = fso.CreateTextFile(csvfile, True)
For k = 0 to j
myArray = Split(arrValidLines(k),",")
myArray(1) = Trim(myArray(1))
GuyFile.WriteLine(myArray(0) & "," & myArray(1) & "," & myArray(2))
Next
GuyFile.Close
Set GuyFile = Nothing
'make the directory heirarchy
'Feature Size
directory = path & models(l) & "nm_bulk\"
MD(directory)
'Clock Speed
directory = path & models(l) & "nm_bulk\" & clockspeed(m) & "Hz\"
MD(directory)
'Setup or Hold Time Violation
If CDbl(triggertime)  > 1.5*clockperiod Then
'Hold Time Violation
directory =  path & models(l) & "nm_bulk\" & clockspeed(m) & "Hz\" & "Hold_Violation\"
Else
directory =  path & models(l) & "nm_bulk\" & clockspeed(m) & "Hz\" & "Setup_Violation\"
End If
MD(directory)
fso.MoveFile csvfile, directory
fso.DeleteFile(output)
fso.DeleteFile(circuit)
fso.DeleteFile(raw)
fso.DeleteFile(models(l) & "nm_bulk.model1")
Set fso = Nothing

Next
Next
Next
Next
ErrorFile.Close
Set ErrorFile = Nothing


Function MD(strDirName)
     Dim lFSObj
     MD = False
     ' First See if it already exists
     set lFSObj = Wscript.CreateObject("Scripting.FileSystemObject")
If lFSObj.FolderExists(strDirName) Then
MD = True
Else
lFSObj.CreateFolder(strDirName)
MD = True
End If
     Set lFSObj = Nothing
End Function
```

Modified Stability Checker Setup Time Calculator Script

```
Option Explicit
Dim fso, GuyFile
Dim arrFileLines(),temp
Dim Command
Dim circuit,output,cmd,i,j
Dim arrValidLines()
Dim k
Dim myArray
Dim raw
Dim temperature
Dim h, csvfile,tempfile
Dim l,m,n
Dim valid
Dim models,directory
Dim path
Dim errortime
Dim OutputFile
Dim found,setuptime

path = "I:\spice\SeniorProject\CircuitRuns\MSCLatchRuns\"
temperature = Array(20,60,100,140,180)
models = Array(32,45,65,90,130)
Set fso = CreateObject("Scripting.FileSystemObject")
Set OutputFile = fso.CreateTextFile("MSLatchwithMSC_Setup_Times.csv",True)
OutputFile.WriteLine("Transistor Feature Size (nm),Temperature (C),Setup Time (ps)")
For l = 0 to UBound(models)
For h = 0 to UBound(temperature)
'Check both high to low and low to high transitions
For m = 0 to 1
tempfile = path & "test.cir"
Set fso = CreateObject("Scripting.FileSystemObject")
Set GuyFile = fso.CreateTextFile(tempfile, True)
GuyFile.WriteLine("*MSC with MS Latch")
GuyFile.WriteLine(".global Vcc")
GuyFile.WriteLine(".INCLUDE " & models(l) & "nm_bulk.model1")
GuyFile.WriteLine("*MSLatch Input Output invOutput Clk Vcc Gnd")
GuyFile.WriteLine("*Inverter Vcc Gnd Input Output")
GuyFile.WriteLine("*TGate Input Output P_gate N_gate")
GuyFile.WriteLine("*MSCCell clk vcc co ffo sense dummy")
GuyFile.WriteLine("Vcc vcc 0 DC 1V")
GuyFile.WriteLine("Vclk clk 0 PULSE(1 0  0ns .1fs .1fs 100ps 200ps)")
GuyFile.WriteLine("Vclkd clkd 0 DC 0V")
GuyFile.WriteLine("Vco co 0 PWL(0ns " & m  & "V 240ps " & m & "V 240.00001ps " & abs(m - 1) & "V)")
GuyFile.WriteLine("X_mslatch co ffo 1 clk Vcc 0 MSLatch")
GuyFile.WriteLine("X_msccell clkd vcc co ffo sense dummy MSCCell")
GuyFile.WriteLine(".options TNOM=" & temperature(h) & " TEMP=" & temperature(h))
GuyFile.WriteLine(".op")
GuyFile.WriteLine(".probe")
GuyFile.WriteLine(".print tran V(ffo)")
GuyFile.WriteLine(".tran 0.001ps 400ps")
GuyFile.WriteLine(".SUBCKT MSCCell clkd vcc co ffo sense dummy")
GuyFile.WriteLine("*Dummy Pullup")
GuyFile.WriteLine("Mud dummy clkd vcc vcc pmos L=" & models(l) & "nm W=" & models(l)/2*6 & "nm")
GuyFile.WriteLine("*Sense Pullup")
GuyFile.WriteLine("Mus sense clkd vcc vcc pmos L=" & models(l) & "nm W=" & models(l)/2*6 & "nm")
GuyFile.WriteLine("*Sense Pulldown")
GuyFile.WriteLine("Mds 1 clkd 0 0 nmos L=" & models(l) & "nm W=" & models(l)/2*3 & "nm")
GuyFile.WriteLine("*Dummy Pulldown")
GuyFile.WriteLine("Mdd 4 0 0 0 nmos L=" & models(l) & "nm W=" & models(l)/2*3 & "nm")
GuyFile.WriteLine("")
GuyFile.WriteLine("*Sense Cell")
GuyFile.WriteLine("M1  2  ffo  sense  sense  pmos L=" & models(l) & "nm W=" & models(l)/2*6 & "nm")
```

```
GuyFile.WriteLine("M2  2  co  1  0  nmos L=" & models(l) & "nm W=" & models(l)/2*3 & "nm")
GuyFile.WriteLine("M3 3 co  sense vcc pmos L=" & models(l) & "nm W=" & models(l)/2*6 & "nm")
GuyFile.WriteLine("M4 3 ffo 1 0 nmos L=" & models(l) & "nm W=" & models(l)/2*3 & "nm")
GuyFile.WriteLine("*Dummy Cell")
GuyFile.WriteLine("M5 6 ffo dummy vcc pmos L=" & models(l) & "nm W=" & models(l)/2*6 & "nm")
GuyFile.WriteLine("M6 6 co 4 0 nmos L=" & models(l) & "nm W=" & models(l)/2*3 & "nm")
GuyFile.WriteLine("M7 5 co dummy vcc pmos L=" & models(l) & "nm W=" & models(l)/2*6 & "nm")
GuyFile.WriteLine("M8 5 ffo 4 0 nmos L=" & models(l) & "nm W=" & models(l)/2*3 & "nm")
GuyFile.WriteLine(".ends")

GuyFile.WriteLine("*MSLatch Input Output invOutput Clk Vcc Gnd")
GuyFile.WriteLine(".SUBCKT MSLatch 1 7 13 clk Vcc Gnd")
GuyFile.WriteLine("X_inv1 Vcc Gnd 1   2   Inverter")
GuyFile.WriteLine("X_inv2 Vcc Gnd 3   4   Inverter")
GuyFile.WriteLine("X_inv3 Vcc Gnd 5   6   Inverter")
GuyFile.WriteLine("X_inv4 Vcc Gnd 6   7   Inverter")
GuyFile.WriteLine("X_inv5 Vcc Gnd 4   9   Inverter")
GuyFile.WriteLine("X_inv6 Vcc Gnd 6   12  Inverter")
GuyFile.WriteLine("X_inv7 Vcc Gnd clk 8   Inverter")
GuyFile.WriteLine("X_inv8 Vcc Gnd 8   11  Inverter")
GuyFile.WriteLine("X_inv9 Vcc Gnd 12  13  Inverter")
GuyFile.WriteLine("X_tgate1 2 3 11 8  TGate")
GuyFile.WriteLine("X_tgate2 4 5 8 11  TGate")
GuyFile.WriteLine("X_tgate3 9 3 8 11  TGate")
GuyFile.WriteLine("X_tgate4 12 5 11 8 TGate")
GuyFile.WriteLine(".ends")
GuyFile.WriteLine("* Inverter Vcc Gnd Input Output")
GuyFile.WriteLine("* TGate Input Output P_gate N_gate")
GuyFile.WriteLine(".SUBCKT TGate 1 3 2 4")
GuyFile.WriteLine("M1 3 2 1 Vcc pmos L=" & models(l) & "nm W=" & models(l)/2*6 & "nm")
GuyFile.WriteLine("M2 1 4 3 0 nmos L=" & models(l) & "nm W=" & models(l)/2*3 & "nm")
GuyFile.WriteLine(".ENDS")
GuyFile.WriteLine("* Inverter Vcc Gnd Input Output")
GuyFile.WriteLine(".SUBCKT Inverter  1 4 2 3")
GuyFile.WriteLine("M1 3 2 1 Vcc pmos L=" & models(l) & "nm W=" & models(l)/2*6 & "nm")
GuyFile.WriteLine("M2 3 2 4 0 nmos L=" & models(l) & "nm W=" & models(l)/2*3 & "nm")
GuyFile.WriteLine(".ENDS")
GuyFile.WriteLine(".end")
GuyFile.Close

csvfile = path & "" & temperature(h) & ".csv"

Set Command = WScript.CreateObject("WScript.Shell")
cmd = "%comspec% /c sed s/" & chr(34) & "tnom    = 27" & chr(34)
& "/tnom=" & temperature(h) & "/ " & models(l) & "nm_bulk.model > " & models(l) & "nm_bulk.model1"
        Command.Run(cmd)
circuit = path & "test.cir"
output = path & "test.out"
raw = path & "test.raw"
cmd = path & "scad3.exe -b " & circuit
Command.Run (cmd)
WScript.Sleep(3000)
'Convert raw to txt
cmd = "ltsputil.exe -x0 test.raw test.out " & chr(34) & "%15.7e" & chr(34)
& " " & chr(34) & "," & chr(34) & " " & chr(34) & "Time,Voltage" & chr(34) & " *"
Command.Run (cmd)
'Open Output File
WScript.Sleep(100)
Set GuyFile = fso.OpenTextFile(output,1)
'Read in file into arrFileLines
i = 0
Do Until GuyFile.AtEndOfStream
Redim Preserve arrFileLines(i)
arrFileLines(i) = GuyFile.ReadLine
```

```
i = i + 1
Loop
GuyFile.Close

j = i - 1
k = 0
valid = 0
For i = 0 to j
'get rid of excess spaces
arrFileLines(i) = Trim(arrFileLines(i))
Redim Preserve arrValidLines(k)
arrValidLines(k) = arrFileLines(i)
k = k + 1
Next
'FOR DEBUGGING PURPOSES, PRINT OUT arrVALIDLINES
'Set GuyFile = fso.CreateTextFile("temp.txt", True)
'j = k - 1
'for i = 0 to j
'myArray = Split(arrValidLines(i),",")
'myArray(1) = Trim(myArray(1))
'GuyFile.WriteLine(myArray(0) & "," & myArray(1))
'next
'GuyFile.Close
'END DEBUG

'Measure the Setup Time
j = k - 1
k = 0


'm = 1 is low to high, m = 0 is high to low
found = 0
If m = 0 Then
setuptime = 0
End If
For k = 0 to j
myArray = Split(arrValidLines(k),",")
myArray(1) = Trim(MyArray(1))
'Convert to pico-seconds
myArray(0) = myArray(0) * 1000000000000
myArray(0) = round(myArray(0),9)
myArray(1) = round(myArray(1),9)
'MsgBox "vtl: " & vtl
'MsgBox "vth: " & vth
If myArray(0) > 300 Then
'MsgBox myArray(0)
If m = 0 Then
'are we above threshold?
'Vth0 = 0.5088 for the worst case
If myArray(1) > 0.5088 Then
If found = 0 Then
found = 1
setuptime = myArray(0) - 300
End If
End If
End If
If m = 1 Then
'are we below threshold?
If myArray(1) <  0.5088 Then
If found = 0 Then
found = 1
n = myArray(0) - 300
'the longer time between the two is going to be the setup time
If n > setuptime Then
```

```
setuptime = myArray(0) - 300
End If
End If
End If
End If
End If
Next

If m = 1 Then
OutputFile.WriteLine(models(l) & "," & temperature(h) & "," & setuptime)
End If
fso.DeleteFile(output)
fso.DeleteFile(circuit)
fso.DeleteFile(raw)
fso.DeleteFile(models(l) & "nm_bulk.model1")


Next
Next
Next
OutputFile.Close
Set OutputFile = Nothing
Set fso = Nothing

Function MD(strDirName)
     Dim lFSObj
     MD = False
     ' First See if it already exists
     set lFSObj = Wscript.CreateObject("Scripting.FileSystemObject")
If lFSObj.FolderExists(strDirName) Then
MD = True
Else
lFSObj.CreateFolder(strDirName)
MD = True
End If
     Set lFSObj = Nothing
End Function
```