CPRE 465 FINAL PROJECT

Harith Mohd., Akash Pradhan, Joyal Babu

Akash Pradhan – Streamlining synthesis process M. Harith Arsyad – Verilog Code, Optimizations, and Synthesis Joyal Babu – Report writing

5th December 2022

1. Problem Description

The problem that the USDA is facing is that they want to investigate the relationship between the temperature of a plant and the protein level present in the plant. In order to do this, they need to monitor the temperature of the plants remotely, using temperature sensors and IoT motes. Our task is to design a circuit that can solve this problem for the USDA, allowing them to monitor the temperature of the plants and investigate the relationship between temperature and protein levels. This will require a combination of digital circuit design and algorithm design in order to implement a solution that meets the USDA's requirements. We will

- 1. Start by using an IoT mote to collect temperature readings from the temperature sensor. The mote should be able to communicate with the temperature sensor and collect the readings at regular intervals.
- Next, design a circuit that can receive the temperature readings from the mote and store them in a memory device, such as a shift register or a FIFO buffer. The circuit should be able to store a large number of readings, so that we can calculate the average and standard deviation over a long period of time.
- 3. Once the readings are stored in the memory device, we can use a digital circuit to calculate the average and standard deviation of the readings. This can be done by implementing a simple algorithm that adds up all the readings and divides the result by the number of readings to calculate the average. The standard deviation can be calculated by taking the square root of the sum of the squared differences between each reading and the average, divided by the number of readings.
- 4. Finally, the circuit should be able to display the average and standard deviation of the temperature readings on a display device, such as an LCD screen. This will allow the USDA to monitor the temperature of the plants and investigate the relationship between temperature and protein levels.

In detail regarding the design of the digital circuit the objective is to find the moving average and the moving standard deviation of the last fourteen readings from a temperature sensor.

Moving average is a calculation to analyze the data points by creating a series of averages of different subsets of the full data set. Here we are taking the temperature value the sensor gives as the average temperatures the sensor detects and we are calculating the average temperature of the last 14 inputs from the temperature sensor. The temperature average is given by

$$=$$
 \sum^{n}

 $T avg n - 1_{i=1} T_i$

The standard deviation can be written as

$$\sigma = \left| \frac{\frac{1}{n-1} \sum_{i=1}^{n} |T_i - T_{avg}|^2}{\sum_{i=1}^{n-1} |T_i|^2} = \left| \frac{1}{n-1} \left| \sum_{i=1}^{n} |T_i|^2 \right| - T_{avg}^2 \right|$$

hence since we want the moving average for the last 14 temperature readings if n,14 we take the average and standard deviation of all n readings.

The design specification given are



When RESET is set to 1 all previous temperature readings are forgotten. The temperatures are taken one at a time and the circuit takes in new temperature when the SAMPLE is set to 1. The circuit then takes in one new temperature reading TN and the mode of operation MODE at the

next clock edge. The next few clock cycles are used to calculate the moving average and the moving standard deviation. After the that the circuit will change DONE to 1 and output will be given which can the moving average(if MODE =0) or the moving standard deviation (if MODE =1) given as AVG/SD.

Square root is not easy to be implemented via hardware hence we use the Babylonian method.

$$sum = \sum_{i=1}^{n} T_i \quad n \leq 14$$

$$sum_s = \sum_{i=1}^{n} T_i^{2} \quad n \leq 14$$

$$Moving Average$$

$$Torg = \frac{1}{n} sum = 2 \quad \frac{sum + n > 2}{n}$$

$$decimils of 0.5 or above to next integer$$

$$\frac{\text{Standard Deviation}}{\sigma = \sqrt{\frac{1}{n} \sin n_{-} \sin 1 - 7 \sin 3^{2}}}$$

$$= \sqrt{\frac{1}{n} \sin n_{-} \sin 1 - (\frac{1}{n} \sin n)^{2}}$$

$$= \sqrt{\frac{1}{n} \sin n_{-} \sin 1} = \sqrt{\frac{n \times \sin n_{-} \sin 2 - 3 \sin n^{2}}{n}} \qquad (\frac{\sqrt{n} \times \frac{1}{n} \sin 2 - 3 \sin n^{2}}{n})$$

$$= \sqrt{\frac{1}{2n} \left[\hat{\sigma} + \left(\frac{n \times \sin n_{-} \sin 2 - 3 \sin n^{2}}{\hat{\sigma}} \right) \right]} \qquad \hat{\sigma} = \frac{1}{2} \left[\frac{\hat{\sigma}^{2} + n \sin 2 \sin 2 - 3 \sin n^{2}}{\hat{\sigma}} \right]$$

$$= \frac{1}{2n} \left[\frac{\hat{\sigma}^{2} + n \sin 2 \sin 2 \sin 2 - 3 \sin n^{2}}{\hat{\sigma}} \right] \qquad \hat{\sigma} = \frac{1}{2} \left[\frac{\hat{\sigma}^{2} + n \sin 2 \sin 2 \sin 2 \sin n^{2}}{\hat{\sigma}} \right]$$

$$= \frac{\hat{\sigma}^{2} + n \sin 2 \sin 2 \sin 2 \sin n^{2}}{\hat{\sigma}} = \frac{\hat{\sigma}^{2} n^{2} + n \sin 2 \sin 2 \sin 2 \sin n^{2}}{\hat{\sigma} n^{2}} = \frac{\hat{\sigma}^{2} n^{2} + n \sin 2 \sin 2 \sin 2 \sin n^{2}}{\hat{\sigma} n^{2}} \leq 1$$

2. Brief Description of Team Approach to this Problem

In this project we needed to preserve previous values of temperatures and their running averages. We started with a big memory of registers for storing input temperatures and used accumulator for calculating the sum of temperatures and square of temperatures. These used a lot of adders and multipliers. Then we did division to get the averages and square root using the Netwon-Raphson approximation method. This gave us uneven number of cycles for calculation of running averages and standard deviation. The problem was that we were doing multiple things in sequential manner without maintaining proper cycle accuracy and that is when we changed our design. Next we started with scratch design on paper and made a flow diagram where we found what could be optimized in the process to calculate the required things. We were able to save on multiplication and division hardware by proper insertion of buffers and using the techniques learned in previous labs. We had the following diagrams in our rough sketches which shows our design



Schematics and layouts after optimization :



Timing, Area and Power Reports (by RTL Compiler and Innovus)

	Before Optimization	After Optimization
Clock Period(ns)	4.2	4.2
Slack(ns)	-3.923	0
Area(µm²)	62051.760	11157.48
Average power consumption(mW)	12.761	2.372

	Before optimization	After optimization
Clock period(ns)	4.2	4.2
Slack(ns)	-3.923	0
Area(µm²)	62051.760	11157.48
Average power	4.30942	2.372
consumption(mW)		
Average Energy consumption	33.28*10^-12	4.9831*10^-12
per temperature reading(W)		
Latency	4	4
Throughput	One every 8.12ns,	0.476*10^9, one every 2.1 ns
	0.123*10^9	

4. Verilog Codes of design

This is the final code after optimization:

module NOAA_Module (

input CLK,

input RESET,

input MODE,

input [11:0] TN,

output SAMPLE, output

DONE, output reg [11:0]

AVG_SD);

//Flag wires and regs

wire newData_flag1; //flags for every new data into the process. Waits for the process to be done first then flags for the next set to be processed

reg newData_flag2, newData_flag3, newData_flag4; //registers to delay newData flag to 2, 3, ... cycles

//Clock Components

wire newData_clk1, newData_clk2, newData_clk3; //clocks for every new input to the process. Each clock line adds 1 cycle delay after every new input wire t_avg_clk2; //clock-gating clock for MOVING AVERAGE. 2 cycles after new input to process

wire sd_clk1, sd_clk2, sd_clk3, sd_clk4; //clock-gating clock for STANDARD DEVIATION. 1, 2, 3, ... cycle delay wire mult_clk;

//Multiplier Wires and select wire [31:0]

mult1_w1, mult1_w2; wire [31:0]

mult2_w1, mult2_w2; wire [39:0]

mult1_w3, mult2_w3, mult3_w3;

reg [1:0] mult_sel;

//Division Wires wire [31:0]

div_w1, div_w2; wire

[31:0] div_result;

//Continuos Process Components
reg [15:0] sum; reg [31:0]
sum_sq; reg [31:0] Ti_sq; reg
[31:0] OD_sq; wire [11:0]
oldest_data; reg mode_delay,

mode_delay2; wire [3:0]

counter_n;

//MOVING AVERAGE Components
wire [11:0] t_avg;

//STANDARD DEVIATION Components
reg [31:0] sd_n_sq, sd_n, sum_2;
reg [39:0] sum_sq_n; wire [15:0]
sd_result;

```
reg [15:0] sd_temp1;
```

shift_reg x1(oldest_data, counter_n, newData_flag1, CLK, RESET, TN);

//Processes running every clock cycle
always @(posedge CLK, posedge RESET)
begin
if (RESET) begin mode_delay
<= 0; mode_delay2 <= 0; end
else begin mode_delay <=
MODE; mode_delay2 <=
mode_delay; end
end</pre>

//Processes that need to run every time there is a new input (1st cycle)
//ie. sum(TI), TI^2, OD^2 always @(posedge
newData_clk1, posedge RESET) begin

```
if (RESET) begin
sum <= 0;
Ti_sq <= 0; OD_sq <= 0; end
else begin sum <= sum + TN -
oldest_data; Ti_sq <=
mult1_w3; OD_sq <=
mult2_w3; end
end</pre>
```

//Processes that need to run every time there is a new input (2nd cycle)

//ie. sum_sq always @(posedge newData_clk2, posedge RESET) begin if (RESET) sum_sq <= 0; else sum_sq <= sum_sq + Ti_sq - OD_sq; end

//MOVING AVERAGE

```
assign t_avg = (sum+(counter_n>>1))/counter_n;
```

//STANDARD DEVIATION //2nd Cycle
always @(posedge sd_clk2, posedge
RESET) begin
if (RESET) begin

sum_2 <= 0; sd_n <=

0; end else begin

```
sum_2 <= mult1_w3;
sd_n <= mult2_w3;
end
end
```

```
cha
```

```
//3rd Cycle always @(posedge sd_clk3,
posedge RESET) begin
if (RESET) begin
sd_n_sq <= 0;
sum_sq_n <= 0; end
else begin sd_n_sq <=
mult1_w3; sum_sq_n <=
mult2_w3; end
end
```

```
//4th Cycle always @(posedge sd_clk4,
posedge RESET) begin
if (RESET) begin
sd_temp1 <= 1024; end
else begin sd_temp1
<= sd_result; end
end
//Final Result assign sd_result = (sd_n_sq + sum_sq_n - sum_2 +
sd_n)/(sd_n<<1);</pre>
```

//Mupltipier Select Logic always @(posedge

mult_clk, posedge RESET) begin

```
if (RESET) begin
mult_sel <= 0; end
else begin
if (mult_sel == 2)
mult_sel <= 0;
else
mult_sel <= mult_sel + 1;</pre>
```

end

end

assign mult_clk = newData_clk1 | newData_clk2 | newData_clk3;

always @(posedge DONE, posedge RESET) begin

if (RESET)

AVG_SD <= 0;

else

```
AVG_SD <= div_result;
```

end

//Delay new input flag always

@(negedge CLK, posedge RESET) begin

if (RESET) begin newData_flag2

<= 0; newData_flag3 <= 0;

newData_flag4 <= 0; end else

begin newData_flag2 <=</pre>

newData_flag1; newData_flag3

```
<= newData_flag2;
newData_flag4 <= newData_flag3;</pre>
end
end
//DONE and SAMPLE logic assign
DONE = t avg clk2 | sd clk4;
assign SAMPLE = newData_clk1;
//Multiplier 1 logic
assign mult1 w1 = (mult sel == 0) ? TN : ( (mult sel == 1) ? sum : ((mult sel == 2) ? sd temp1 :
0)); assign mult1_w2 = (mult_sel == 0) ? TN : ( (mult_sel == 1) ? sum : ((mult_sel == 2) ? sd_n :
0));
assign mult1 w3 = mult1 w1 * mult1 w2;
//Multiplier 2 logic
assign mult2_w1 = (mult_sel == 0) ? oldest_data : ( (mult_sel == 1) ? sd_temp1 : ((mult_sel == 2)
? counter n : 0));
assign mult2_w2 = (mult_sel == 0) ? oldest_data : ( (mult_sel == 1) ? mult3_w3 : ((mult_sel ==
2)
? sum_sq : 0));
assign mult2_w3 = mult2_w1 * mult2_w2;
//Multiplier 3 logic
```

```
assign mult3_w3 = counter_n * counter_n;
```

//Division Logic assign div_w1 = mode_delay ? (sd_n_sq + sum_sq_n - sum_2 + sd_n) :
(sum+(counter_n>>1)); assign div_w2 = mode_delay ? (sd_n<<1) : counter_n; assign
div_result = div_w1/div_w2;</pre>

//create clock at every new process delayed by a cycle each assign newData_clk1 = CLK & newData_flag1; //At cycle 1 of new data assign newData_clk2 = CLK & newData_flag2; //At cycle 2 of new data assign newData_clk3 = CLK & newData_flag3; //At cycle 3 of new data assign newData_clk4 = CLK & newData_flag4; //At cycle 3 of new data

//clock-gating clock for MOVING AVERAGE. n cycle delay after new input to process
assign t avg clk2 = newData clk2 & ~MODE; //at cycle 2

//clock-gating clock for STANDARD DEVIATION. n cycle delay after new input to process assign sd_clk1 = newData_clk1 & MODE; //at cycle 1 assign sd_clk2 = newData_clk2 & MODE; //at cycle 2 assign sd_clk3 = newData_clk3 & MODE; //at cycle 3 assign sd_clk4 = newData_clk4 & mode_delay2; //at cycle 4

endmodule

// This is linear queue / FIFO
// The queue length 14 // The
data width is also 12 bits
module shift_reg(output [11:0]
DATAOUT, output reg [3:0]

counter_n, output reg newData_flag1, input clock, input reset, input [11:0] DATAIN);

wire newData_clk1; reg [11:0] memory [13:0]; // the stack is 12 bit wide and 14 locations in size

integer i;

always @(posedge newData_clk1, posedge reset)

begin

if (reset)

begin

for (i = 0; i < 14; i = i + 1) begin

memory[i] <= 0;

end

```
counter_n <= 0;</pre>
```

end

else

begin

end

```
if (counter_n < 14)
```

```
counter_n <= counter_n + 1;
```

end

end

```
always @(negedge clock, posedge reset)
```

begin

if (reset) begin

newData_flag1 <= 0;

end else begin

```
if (memory[0] == DATAIN)
```

```
newData_flag1 <= 0;
```

else

```
newData_flag1 <= 1;</pre>
```

end

end

assign newData_clk1 = clock & newData_flag1;

```
assign DATAOUT = memory[13];
```

endmodule

5. Test results

		Wave	- 1	0
<u>E</u> dit <u>V</u> iew <u>A</u> dd F <u>o</u> rmat <u>T</u> ools Boo <u>k</u> marks	: <u>W</u> indow <u>H</u> elp			_
ve - Default :				+ 5
	- ME 0		_	and the second
A + + A +				
·** 1 ill 12 ill				
. 9. 9 8. 9				
<u>ы</u> фци в				
1++1355				
a co c e sa				
• ** • • • • • • • • • • • • • • • • •	10 02 07			
	Msgs			_
nd_NUAAIOI_MotesinewUata_cik Mh_NUAAIoT_Motesint_cik	101			
to NOAAIoT Motesist cik delav	1'00			
/b NOAA/loT Motes/sd clk delay2				
/b_NOAA/IoT_Motes/newData_clk_delay				
/tb_NOAA/loT_Motes/newData_clk_delay2				
/tb_NOAA/loT_Motes/oldest_data		0 (1590 (2313 (2804 (3003 (1466 (1138 [994 [433 [204 [433 [2415 [1691 [1037 [2571]1916 (2447 [3054		
/tb_NOAA/toT_Motestemp				
/tb_NOAA/loT_Motesidone_flag	1'd0			
/tb_NOAA/loT_Motes/CLK	161	an a han		
ND_NOAANOI_MORSHESEI	100		- 3	
ND_NORMOT_MORSHNUDE	100	1100 1212 1200 1202 1200 1200 1200 1200		
#b_NOAAIoT_Notes1M #b_NOAAIoT_Notes5AMDLE	1/11			
to NOAAIoT MotesDONE	1/00			
Itb NOAAIoT Motes/AVG SD	12'0498			
/tb_NOAA/expectedOutput	12'02428			
/tb_NOAA/IoT_Motest_avg		0 1590 1952 2236 2428 2236 2053 1901 1718 1795 1785 1717 1788 1798 1844 1949 1788 1609 1634 1578 1527 1557 1652 1531 1490 1448 1422 1289 1253 1038		
/tb_NOAA/loT_Motes/pre_temp	24'd15712641	0 12. 24 29 38 34 48 155 72. 169 62 161 57 1558 645. 179. 169 62 161 61 57 1558 645. 179. 169 77 162 160 77 171 71 71 71 71 71 71 71 71 71 71 71		
/tb_NOAAIloT_Motes/SUM_4	4'd3	0 16 115 13 114 110 112 114 115 110 17 12 114 113 15 14 18 115 19 15 115 14 10 15 112 113 12 17 16		
(3)=/tb_NOAA/IoT_Motesisum[3]		السبيحية المراجع المراجع والمراجع		
(2)=tb_NOAAIoT_Motes/sum[2]				
(1)=/tb_NOAA/IoT_Motes/sum(1)				
-> (U)=10_NUAANO1_MOTES/SUM(U)	1			
hb NOLVIOT Meteologister a	1000/0/	0 11990 3993 9100 9110 1111. 112. 133. 137. 101. 110. 124. 233. 250. 272. 250. 225. 220. 213. 214. 206. 204. 119. 100. 119. 14534		
th NOLLINT Motescounter 1	12/1498	0 11 12 19 19 19 19 19 19 19 19 19 19 19 19 19	=	
Ith NOAMOT Motes/sd temp?	12/1498	100 1512 144 164 170 1748 1657 1786 1987 1871 1987 1887 1887 1887 1887		
Ito NOAAloT Motesisd reg1	16'0498	10 1512 1498 1634 1701 1748 1857 1786 1897 1871 1897 1847 1885 1855 1913 1874 1774		
/b NOAA/IoT Motes/sum sqr	28'd15740485	(0) 252) 787] 157] 247] 269] 282 [291] 293] 352] 380 [391] (457] (494] 554] (522] 568] 510] 470] 142] (441] (451] (421] (423] (411] (423] (423] (424] (423] (424] (423] (424] (423] (424] (423] (424]		
/tb_NOAA/loT_Motes/t_avg_sqr		0 1252 1381 1499 1559 1499 1421 (361 1295 1322 1318 (294 1319 1323 1340 1379 1319 1285 1266 1247 1233 (242 (272 1234 1222 (209 1262 1166 1157 11077444	- ال	
/tb_NOAA/loT_Motest_avg_temp	12'd2236	0 11590 11952 1256 12428 12236 12053 (1901 (1718 (1795 (1785 (1717 (1788 1198 1198 1198) 11768)1688)1634)1573)1527 (1557 (1557 (1557 (1490 (1448 (1422 (1289 11253)1038		
• No	uw 2000 ns	na 200 ns 400 ns 600 ns 600 ns 100 ns 1200 ns 1400 ns 1400 ns 1600 ns	2000 ns	111
e Cursor	:1 257 ns	257 ns		
2100 pc				
100 IIS / //ID_NUAA/expecter	ασαφάι			

This is the simulation with initial code with failing test scenarios due to rounding errors and mismatch between cycle for different outputs

Test benches and results

Below shows the successful running of our design with provided test-benches.

Applications Places Transcript			Mon 22:14 • 🛔 🍕 🖸
	Wave		_ = ×
Elle Edit View Add Format Tools Bookmarks Window Help			
📊 Wave - Default			
1) - 2 - 2 → 2 3 + 10 - 10 - 2 → 0 - 44 目 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	L L L X 😫 🕺 🖓		
tatit.a.t			
R. R. 9 G. R			
▶ 5 4 11 11 1			
88.48.52.53			
3+ - +€ - 3+ Search:	🗰 🗜 J - J		
Maga			
			นการการการการการการการการการการการการการก
to_NOAAMODE -No Data- to but			
	Transcrip	vt	_ ¤ ×
Elle Edit View Boogmarks Window Help			
Transcript:			± 🖌 🖈
B. S 6 5 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1			
# End time: 22:12:51 on Dec 05,2022, Elapsed time: 0:00:00 # Errors: 0. Warnings: 0			A
VSIM 90> vsim work.tb_NORA			
# End time: 22:12:55 on Dec 05,2022, Elapsed time: 0:04:05			
# vsim work.tb_NOAA			
# Start time: 22:12:55 on Dec 05,2022			
# Loading work. NOAA Module			
# Loading work.shift_reg			
add wave -position end sim:/tb_NOBA/CLK			
add wave -position end sim:/tb NOAA/MODE			
add wave -position end sim:/tb_NORA/TN			
add wave -position end sim:/tb NORA/SAMPLE			
add wave -position end sin://b NOAA/NVG SD			
add wave -position end sim:/tb_NORA/expectedOutput			
VSIM 99> run			
IF Initiating NUAA for Notes Nodule Testing Phasell Good Luckil			
# ** Note: \$stop ://home/akashkap/ee465/Final protect/sim/tb NOAA.v(67)			
# Time: 6060 ns Iteration: 1 Instance: /tb_NORA			
<pre>[# Break in Nodule th_NOAA at /home/akashkap/ee465/Final_project/sim/th_NOAA.v line 67</pre>			
[kziu Tris]			
Cursor 1 14459 ns			
0 ns to 6033 ns			

Using the 100 values, final simulation run

					Wave									_ = ×
Elle Edit View Add Format Tools Bookmarks Wind	dow <u>H</u> elp													
Wave - Default					2									* *
🖹 - 🚅 🖬 🔅 🚳 🐰 🍡 🛍 ሷ 🔅 O - M	i 😂 🔹 🛱	i 🔉 🕺 🛯 🖬 🛊	• m} k≢ 10000 ns	🕈 🖬 🖻 🖬 🕷 🌐 ()	s 🛍 🖪 🖑									
1 ゆ 1 (土・魚・土														
a.a.g.g.a.a														
N N + M H = = = = = = = = = = = = = = = = = =														
1111111111														
3•••€•3•• i Search: ▼ @b #	8. Ø	0 C C L		1111										
Msgs														
Mb_NCAAVCLK -No Data-		m		un nu									nnt	u u u
/db_NOAAMODE -No Data-														
Ab_NOAA/TN -No Data-	(1383	(3177]	593) 586	1449	2362	(2290	1763	12940	2772	[411) 1767	(1782	1286	2
- //b_NOAA/DONE -No Data-														
AD_NOAVAVG_SD -No Data-	(1383	(905 I	1098 (143)	0	(929	(895	1895 11	1838 1838	1932	[1793	33) 906	1905 (1790	11790	
					Transcrip	ot								_ = ×
File Edit View Bookmarks Window Help														
Transcript :					1100									÷ # >
🛛 🖬 • 🚘 🗟 🕸 🚳 🕉 🖄 🛍 💭 💭 (O • 🛤	8													
# End time: 22:12:51 on Dec 05,2022, Elapsed tim	e: 0:00:00													
VSIM 90> vsim work.tb_NOAA														
# End time: 22:12:55 on Dec 05,2022, Elapsed tim	e: 0:04:05													
# Errors: 0, Warnings: 0														
# Start time: 22:12:55 on Dec 05.2022														
Loading work.tb NOAA														
# Loading work.NOAA_Module														
# Loading work.shift_reg														
add wave -position end sim:/tb_NOAA/CLK														
add wave -position end sim:/tb NOAA/MODE														
add wave -position end sim:/tb NOAA/TN														
add wave -position end sim:/tb_NOAA/SAMPLE														
add wave -position end sim:/tb_NOAA/DONE														
add wave -position end sim:/tb_NOAA/AVG_SD														
and wave -position end sim:/tb_auga/expecteduit	put													
# Initiating NOAA ToT Motes Module Testing Phase	11 Good Tack!!													
# NOAA IOT Notes MODULE PASSED ALL TESTS !! CONGR	ATULATIONS!!													
# ** Note: \$stop : /home/akashkap/ee465/Final	project/sim/tb]	NOAA.v(67)												
Time: 6060 ns Iteration: 1 Instance: /tb	NOAA		-											
sin non-	o/rinal_project/	sim/th_mumA.v line 6	ov											
Lon tool														
Cursor 1 14459 ns														
0 ns to 913 ns														

Zoomed screenshot of the above run.



Using the 30 value data.



Zoomed version of above simulation.

6. Conclusions and Discussion

The learning experience of solving a real world problem using the techniques we learned in the class and lab was an eye opener. The hardest part of the project was verilog coding and optimizing our code to obtain required results(basically incorporating division towards the end in the code). Suggestions would be to use better tools like Cadence NCSIM and Synopsys Verdi for simulation part(or even Xilinx Vivado is better than ModelSim) and different project descriptions should be given to different groups to enhance creativity.