EE 465: Final Project

NOAA Module Report

Professor: Dr. Ahamed Maruf
TA: Shuo Xie

Abraham Steenhoek, Samuel Ferguson, Sriram Nithin Gopal Lanka

11-21-2020

Introduction

In the agricultural space, embedded systems are always in high demand. This is due to their customizability to fit many different problems, and their modularity to work with other systems. In this project, we are given the task to calculate the moving average and moving standard deviation of the current temperature. Our module is provided the current temperature from an external source (i.e. a sensor from another system) and will transmit the moving average or standard deviation using an external transmitter. Our task is to build the module that performs the calculations to determine the moving average and standard deviation.

Design Description

We expressed the moving average the moving standard deviation as follows:

$N = \min(n, 14)$ (where n is the number of samples taken since reset)

$$T_{avg} = \frac{1}{N}\sum_{i=1}^{n} T_i$$

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{n}(T_i - T_{avg})^2} = \sqrt{\frac{1}{N}\left(\sum_{i=1}^{n}(T_i^2)\right) - T_{avg}^2}$$

We used one iteration of the Babylonian method to square root of the variance to find the standard deviation in the equation above.

$$\sqrt{V} \cong \frac{1}{2}\left(\hat{\sigma} + \frac{V}{\hat{\sigma}}\right)$$

Where $\hat{\sigma}$ is a guess of $\sqrt{V}$. Using these given equations, we went through the following derivation to obtain a workable expression of the moving standard deviation:

$$\sqrt{V} = \sigma \qquad T_{sum} = \sum_{i=1}^{n} T_i \qquad T_{sqrd_{sum}} = \sum_{i=1}^{n}(T_i^2)$$

$$\sigma \cong \frac{1}{2}\left(\hat{\sigma} + \frac{V}{\hat{\sigma}}\right)$$

$$\sigma = \frac{1}{2}\left(\hat{\sigma} + \frac{\frac{1}{N}\left(\sum_{i=1}^{n}(T_i^2)\right) - T_{avg}^2}{\hat{\sigma}}\right) = \frac{1}{2}\left(\frac{\hat{\sigma}^2 + \frac{1}{N}\left(T_{sqrd_{sum}}\right) - \left(\frac{1}{N}\right)^2 (T_{sum})^2}{\hat{\sigma}}\right)$$

$$\sigma = \frac{1}{2}\left(\left(\frac{1}{N^2}\right)\left(\frac{1}{\hat{\sigma}}\right) * \left(\hat{\sigma}^2 + \frac{1}{N}\left(T_{sqrd_{sum}}\right) - \left(\frac{1}{N}\right)^2 (T_{sum})^2\right)\right)$$

$$\sigma = \frac{\left(N^2 * \hat{\sigma}^2 + \left(N * T_{sqrd_{sum}}\right) - (T_{sum})^2\right)}{2 * N^2 * \hat{\sigma}}$$

Using this reworked equation, we can calculate the moving standard deviation without having to calculate the variance if we've been given an initial $\hat{\sigma}$ value. Additionally, this minimizes the number of divisions for our calculation to 1 division, making it possible to delay the division until the end of the calculation to avoid propagating roundoff errors.

Schematics

In our design, we had 3 separate modules: a register file, a calculator for the standard deviation numerator and denominator, and a top-level module to tie the two modules together and perform the division necessary to get the output.

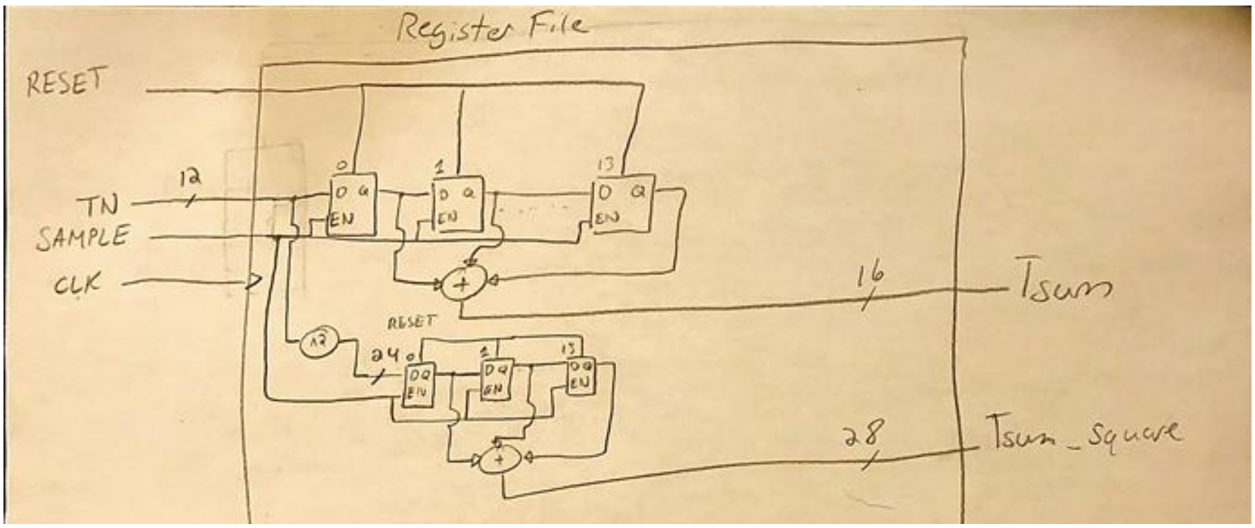

Figure 1: Schematic of register file

The register file stores 14 sampled temperature in the order that they were sampled in. On the rising CLK edge, when the SAMPLE signal is high, the module will store the incoming temperature in DFF0, and transfer the value in DFF0 into DFF1, until DFF13 has been filled. Since we only keep track of 14 temperatures, the value in DFF13 is discarded when a new temperature is measured. This same procedure is used to store the square of each incoming temperature. The sum of all measured temps, and squared temps, is calculated and updated every time a new sample is taken as well.

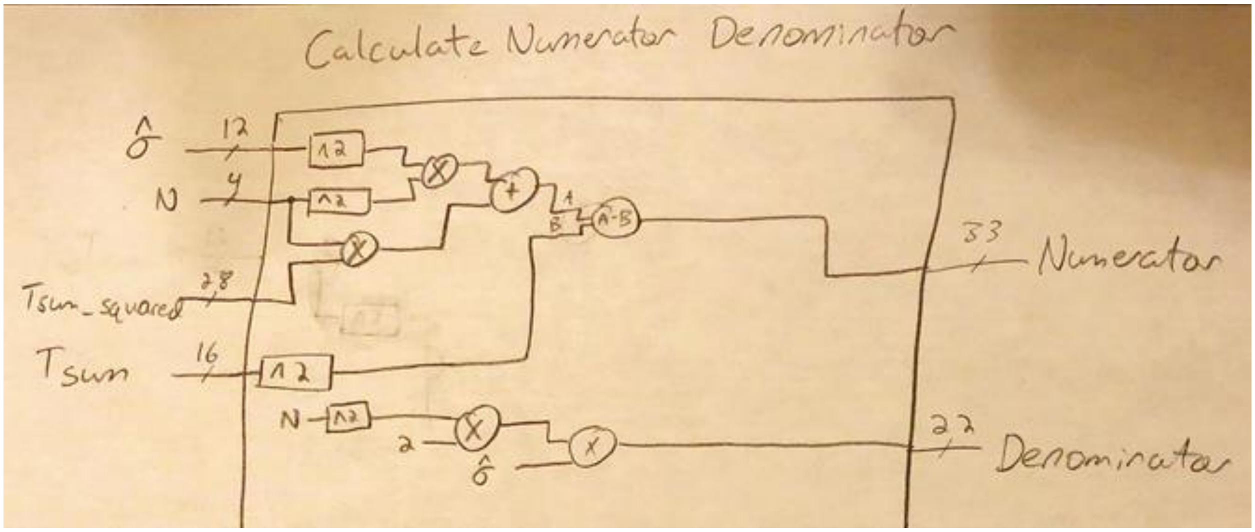

Figure 2: Schematic of Standard Deviation numerator/denominator calculator

The standard deviation numerator and denominator calculator receives the necessary inputs to calculate the numerator:

$$\sigma_{numerator} = \left(N^2 * \hat{\sigma}^2 + \left(N * T_{sqrd_{sum}}\right) - (T_{sum})^2\right)$$

And the denominator:

$$\sigma_{denominator} = 2 * N^2 * \hat{\sigma}$$

We opted for a design that does not calculate on a CLK signal, in order to try and provide simplicity when integrating into the NOAA module by not having to worry about another clock signal to coordinate.

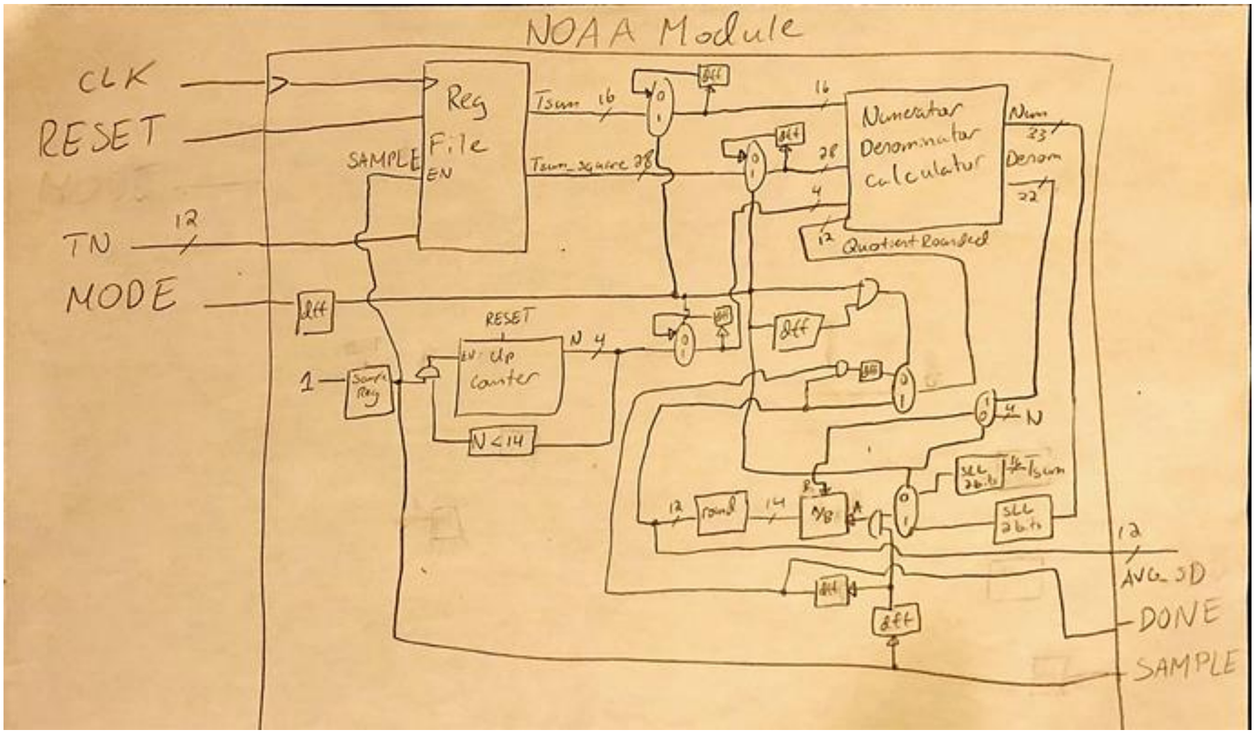

Figure 3: Schematic of top-level module

In designing this NOAA module, we decided to break the calculation into two stages. This gave us room to perform some simple switching optimizations which will be described later.

Because the SAMPLE signal is placed in a register, and the register file also uses registers and works off of SAMPLE, we know that with our design, it will take at least 2 clock cycles to calculate the first output. However, any output can be calculated in 1 clock cycle after the first output. This means that the user can input new temperatures on every clock cycle, and the correct output will be generated 2 clock cycles later. Since the **calc_state[]** reg is only used to provide the correct state upon reset, it is not elaborated in the schematic. After the first 2 clock cycles, the **calc_state[]** variable will always be 3 (2'b11), so it is basically ignored after reset.

As a new temperature sample is taken the **N** value is incremented until it reaches 14.

The 2 clock cycles represent the 2 stages in the design: (1) loading the variables needed to calculate the output, and (2) calculating the output, and updating the $\hat{\sigma}$ value if necessary. Because it will take 2 clock cycles to calculate the first output, the mode must be preserved for 2 clock cycles, hence the use of the mode1 and mode2 regs.

In the first stage, the numerator and denominator are chosen based on the mode requested, which is **mode1**. We can select the appropriate numerator/denominator with a mux using **mode1** as the select signal. If **mode1** = 1, then the numerator and denominator will come from the standard deviation numerator/denominator calculator, otherwise the numerator is the sum of the temperatures (**Tsum**) and the denominator is the number of samples taken(**N**).

In order to make sure the answer was rounded correctly, we left-shifted the numerator value by 1 bit and checked the LSB of the left-shifted value. We rounded up if the LSB = 1 by adding 1 to the top 12 bits of the left-shifted value. While this isn't a perfect solution, for our use case it was the simplest to implement.

Since we only want to update the $\hat{\boldsymbol{\sigma}}$ value when we have calculated a new standard deviation, we only want to replace it when the **mode2** = 1, because **mode2** is the saved mode from when the initial request was made. Additionally, we do not want to calculate a new standard deviation unless the user requests one, so the inputs used to calculate $\boldsymbol{\sigma}$: [$\hat{\boldsymbol{\sigma}}$**, N, Tsum, Tsum_square**], should not be updated unless the mode for that request is 1. This can be done by multiplexing these values with the **mode1** signal, and by multiplexing the $\hat{\boldsymbol{\sigma}}$ value with AND(**mode1**, **mode2**). The reason that $\hat{\boldsymbol{\sigma}}$ needs a special mux is if case the user requests multiple standard deviations in a row, the $\hat{\boldsymbol{\sigma}}$ must be updated only after a standard deviation has been calculated first, which is when **mode2** is set to one, which is a clock cycle behind **mode1**. This gives us the added benefit of saving switching power, because the standard deviation is only being calculated when it is requested, not every clock cycle.

When a new standard deviation has been calculated, we save that value as the $\hat{\boldsymbol{\sigma}}$ for our next standard deviation calculation. Since the $\hat{\boldsymbol{\sigma}}$ value is seen as a "guess" of the standard deviation, it seemed to us that using the previous $\boldsymbol{\sigma}$ was the most accurate guess we could make without creating some other kind of external formula. This saved us from having to calculate the variance that was given in the report as well, which could have required extra divisions.

RTL Compiler Reports

RTL synthesis has been performed on the *NOAA_module.v* file and the resulting schematics with timing, power and area report are illustrated below.

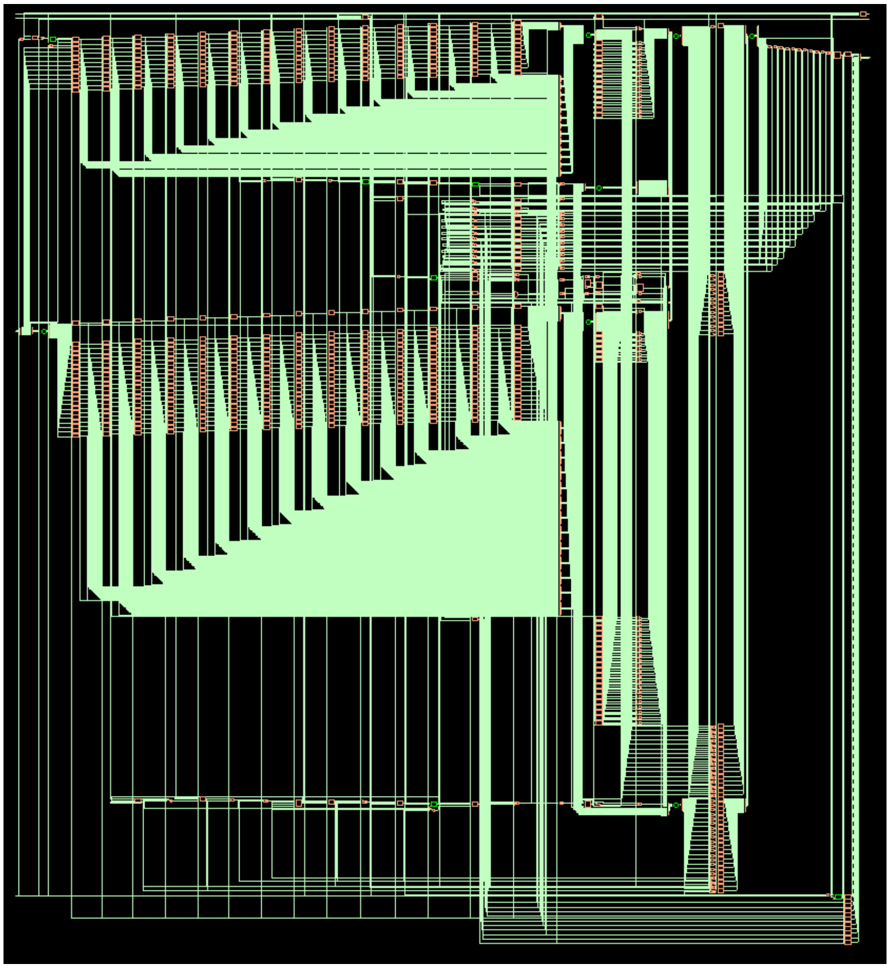

Figure 4 RTL synthesized circuit schematic

```
  g12238/ZN                     MAOI222D1       3  4.5  150  +115    45860 F
  g12222/A                                                 +0    45860
  g12222/ZN                     MAOI222D1       3  4.5  171  +126    45986 R
  g12187/B1                                                +0    45986
  g12187/ZN                     MAOI22D1        3  3.1   91  +106    46092 R
  g12176/A1                                                +0    46092
  g12176/ZN                     NR2D1           1  0.8   47   +26    46118 F
  g12171/C                                                 +0    46118
  g12171/ZN                     AOI221D0        2  2.4  234  +180    46297 R
  g12144/B1                                                +0    46297
  g12144/ZN                     MAOI22D1        4  4.7  176  +126    46423 R
  g12013/B1                                                +0    46423
  g12013/ZN                     MOAI22D1        2  3.1   95   +94    46517 R
  g11974/B                                                 +0    46517
  g11974/ZN                     MAOI222D1       2  3.3  122  +110    46627 F
  g11968/A                                                 +0    46627
  g11968/ZN                     MAOI222D1       2  3.4  149  +108    46735 R
  g11964/A                                                 +0    46735
  g11964/ZN                     MAOI222D1       2  3.3  122   +99    46835 F
  g11962/A                                                 +0    46835
  g11962/ZN                     MAOI222D1       2  3.4  149  +108    46942 R
  g11960/A                                                 +0    46942
  g11960/ZN                     MAOI222D1       2  3.3  122   +99    47042 F
  g11958/A                                                 +0    47042
  g11958/ZN                     MAOI222D1       2  3.3  148  +107    47149 R
  g11956/A                                                 +0    47149
  g11956/ZN                     MAOI222D1       2  3.3  123   +99    47248 F
  g11954/A                                                 +0    47248
  g11954/ZN                     MAOI222D1       2  3.3  148  +107    47355 R
  g13553/A1                                                +0    47355
  g13553/Z                      XOR3D1          1  1.2   32  +198    47552 R
csa_tree_calc_num_sub_12_64_groupi/out_0[29]
g2988/A1                                                   +0    47552
g2988/Z                         AN2XD1          1  1.0   34   +45    47597 R
numerator_store_reg[31]/D    <<<  DFKCNQD1                 +0    47597
numerator_store_reg[31]/CP      setup                      0  +118    47715 R
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(clock CLK)                     capture                         1000000 R
-------------------------------------------------------------------------------
Cost Group   : 'CLK' (path_group 'CLK')
Timing slack :  952285ps
Start-point  : denominator_store_reg[19]/CP
End-point    : numerator_store_reg[31]/D
```

Figure 5 RTL timing report

```
============================================================
  Generated by:           Genus(TM) Synthesis Solution 19.10-p002_1
  Generated on:           Nov 17 2020  06:38:20 pm
  Module:                 NOAA_module
  Technology library:     tcbn65gpluswc 121
  Operating conditions:   WCCOM (balanced_tree)
  Wireload mode:          segmented
  Area mode:              timing library
============================================================

              Instance                           Module              Cell Count  Cell Area  Net Area   Total Area    Wireload
---------------------------------------------------------------------------------------------------------------------------------
NOAA_module                                                                6519  20789.280     0.000    20789.280 ZeroWireload (S)
  div_49_35                          divide_unsigned                       2551   5763.960     0.000     5763.960 ZeroWireload (S)
  csa_tree_calc_num_sub_12_64_groupi csa_tree_calc_num_sub_12_64_group_2   1365   3919.320     0.000     3919.320 ZeroWireload (S)
  reg_file_csa_tree_add_87_154_groupi csa_tree_add_87_154_group_2           684   2729.880     0.000     2729.880 ZeroWireload (S)
  reg_file_csa_tree_add_86_95_groupi csa_tree_add_86_95_group_2             348   1428.840     0.000     1428.840 ZeroWireload (S)
  calc_num_mul_13_33                 mult_unsigned_193                      266    758.160     0.000      758.160 ZeroWireload (S)
  square_calc_num_mul_12_31          square_unsigned_2652_2654              250    670.320     0.000      670.320 ZeroWireload (S)
  reg_file_square_mul_68_17          square_unsigned_2652_2654_203          269    669.960     0.000      669.960 ZeroWireload (S)
```

Figure 6 RTL area report

```
Instance: /NOAA_module
Power Unit: W
PDB Frames: /stim#0/frame#0
  -------------------------------------------------------------------------
    Category         Leakage     Internal    Switching        Total    Row%
  -------------------------------------------------------------------------
      memory     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
    register     3.06004e-05  4.16089e-06  1.73359e-07  3.49346e-05  18.48%
       latch     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
       logic     1.38162e-04  8.67777e-06  6.57516e-06  1.53415e-04  81.15%
        bbox     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
       clock     2.46059e-07  4.58224e-08  3.98796e-07  6.90677e-07   0.37%
         pad     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
          pm     0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00   0.00%
  -------------------------------------------------------------------------
    Subtotal     1.69008e-04  1.28845e-05  7.14732e-06  1.89040e-04 100.00%
  Percentage          89.40%        6.82%        3.78%      100.00% 100.00%
  -------------------------------------------------------------------------
```

Figure 7 RTL power report

Clock Period (ns)	Slack (ns)	Area (um^2)	Average Power (mW)
1000	952.285	20790	0.189

Minimum Clock Period Synthesis at 55ns

```
--------------------------------------------------
Cost Group   : 'CLK' (path_group 'CLK')
Timing slack :      64ps
Start-point  : denominator_store_reg[19]/CP
End-point    : numerator_store_reg[32]/D

(p) : Instance is preserved but may be resized

legacy_genus:/>
```

64ps is negligible slack with a clock period of tens of nanoseconds.

Minimum clock period: 55 ns

Maximum clock frequency: 18.18MHz

Can take 18 million samples a second.

Must wait two clock cycles after initialization for output.

Innovus Reports

After performing the RTL synthesis, we have done encounter layout/Innovus layout and the resultant power, area & timing reports are shown below.

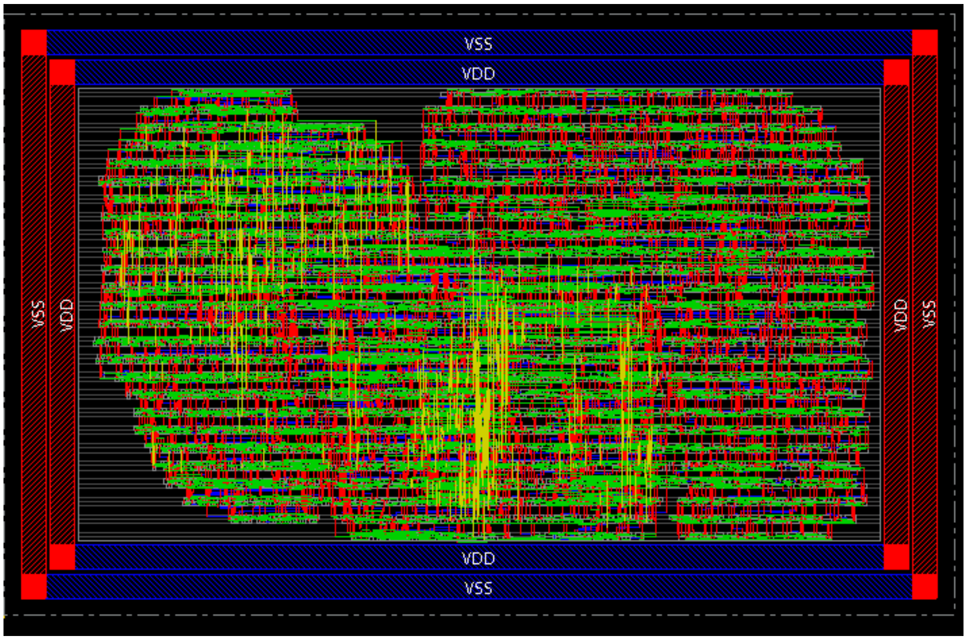

Figure 8 Encounter synthesized layout

```
    Hinst Name                       Module Name                         Inst Count          Total Area
-----------------------------------------------------------------------------------------------------
NOAA_module                                                                    6548           20883.240
 RC_CG_HIER_INST0                    RC_CG_MOD                                    1               6.480
 RC_CG_HIER_INST1                    RC_CG_MOD_1                                  1               6.480
 RC_CG_HIER_INST2                    RC_CG_MOD_2                                  1               6.480
 RC_CG_HIER_INST3                    RC_CG_MOD_3                                  1               6.480
 RC_CG_HIER_INST4                    RC_CG_MOD_4                                  1               6.480
 calc_num_mul_13_33                  mult_unsigned_193                          266             758.160
 csa_tree_calc_num_sub_12_64_groupi  csa_tree_calc_num_sub_12_64_group_2       1367            3924.360
 div_49_35                           divide_unsigned                           2568            5815.440
 reg_file_RC_CG_HIER_INST5           RC_CG_MOD_5                                  1               6.480
 reg_file_csa_tree_add_86_95_groupi  csa_tree_add_86_95_group_2                 348            1428.840
 reg_file_csa_tree_add_87_154_groupi csa_tree_add_87_154_group_2                684            2729.880
 reg_file_square_mul_68_17           square_unsigned_2652_2654_203              269             669.960
 square_calc_num_mul_12_31           square_unsigned_2652_2654                  250             670.320
innovus 4>
```

Figure 9 Encounter area report

```
Total Power
-----------------------------------------------------------------------------------------
Total Internal Power:        0.01325094            6.8099%
Total Switching Power:       0.01009579            5.1884%
Total Leakage Power:         0.17123678           88.0017%
Total Power:                 0.19458351
-----------------------------------------------------------------------------------------


Group                           Internal   Switching     Leakage       Total  Percentage
                                Power      Power         Power         Power  (%)
-----------------------------------------------------------------------------------------
Sequential                      0.004052   0.0003459     0.02919     0.03359       17.26
Macro                                  0           0           0           0           0
IO                                     0           0           0           0           0
Combinational                   0.009153     0.00975      0.1418      0.1607       82.59
Clock (Combinational)                  0           0           0           0           0
Clock (Sequential)              4.57e-05           0   0.0002436   0.0002893      0.1487
-----------------------------------------------------------------------------------------
Total                            0.01325      0.0101      0.1712      0.1946         100
-----------------------------------------------------------------------------------------


Rail                  Voltage   Internal   Switching     Leakage       Total  Percentage
                                Power      Power         Power         Power  (%)
-----------------------------------------------------------------------------------------
VDD                       0.9    0.01325      0.0101      0.1712      0.1946         100


Clock                           Internal   Switching     Leakage       Total  Percentage
                                Power      Power         Power         Power  (%)
-----------------------------------------------------------------------------------------
CLK                             4.57e-05           0   0.0002436   0.0002893      0.1487
-----------------------------------------------------------------------------------------
Total                           4.57e-05           0   0.0002436   0.0002893      0.1487
-----------------------------------------------------------------------------------------
Clock: CLK
Clock Period: 1.000000 usec
Clock Toggle Rate:     2.0000 Mhz
Clock Static Probability:  0.5000


-----------------------------------------------------------------------------------------
*       Power Distribution Summary:
*              Highest Average Power: csa_tree_calc_num_sub_12_64_groupi/g13721 (CMPE42D1):        0.0002812
*              Highest Leakage Power: reg_file_csa_tree_add_86_95_groupi/g4533 (CMPE42D1):         0.000241
*                Total Cap:      3.32571e-11 F
*                Total instances in design:  6548
*                Total instances in design with no power:     0
*                Total instances in design with no activty:     0

*                Total Fillers and Decap:     0
-----------------------------------------------------------------------------------------
```

Figure 10 Encounter power report

```
###############################################################
#  Generated by:      Cadence Innovus 19.10-p002_1
#  OS:                Linux x86_64(Host ID vlinux-35.ece.iastate.edu)
#  Generated on:      Tue Nov 17 19:00:52 2020
#  Design:            NOAA_module
#  Command:           report_timing
###############################################################
Path 1: MET Setup Check with Pin numerator_store_reg[31]/CP
Endpoint:   numerator_store_reg[31]/D   (^) checked with  leading edge of 'CLK'
Beginpoint: denominator_store_reg[19]/Q (v) triggered by  leading edge of 'CLK'
Path Groups: {CLK}
Analysis View: typical_view
Other End Arrival Time          0.000
- Setup                         0.121
+ Phase Shift                1000.000
= Required Time               999.879
- Arrival Time                 55.470
= Slack Time                  944.409
     Clock Rise Edge                 0.000
     + Clock Network Latency (Ideal) 0.000
     = Beginpoint Arrival Time       0.000
```

Figure 11 Encounter Timing report

Clock Period (ns)	Slack (ns)	Area (um^2)	Average Power (mW)
1000	944.409	20884	0.289

Test Results

We tested our register file by giving it a series of temperature inputs and ensuring that the module calculated the correct Tsum and Tsum_squared values. We added a terminal input to show the expected and the actual Tsum and Tsum_squared values.

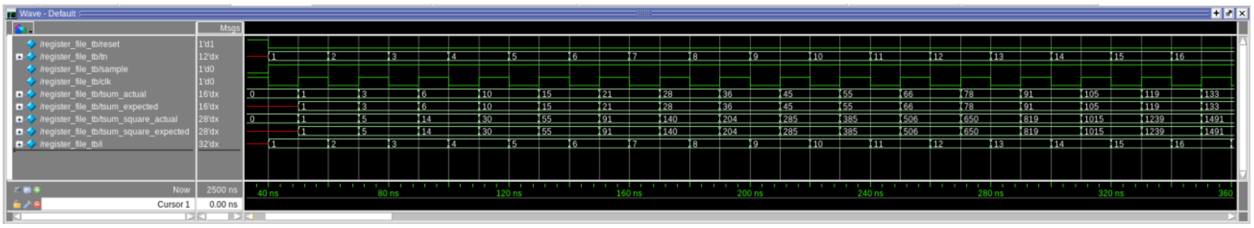

Figure 12: Modelsim Waveforms for register_file_tb.v

```
# Running tests for register_file module
# Tsum_actual=   x | Tsum_expected=   x | Tsum_square_actual=      x | Tsum_square_expected=      x
# Tsum_actual=   1 | Tsum_expected=   1 | Tsum_square_actual=      1 | Tsum_square_expected=      1
# Tsum_actual=   3 | Tsum_expected=   3 | Tsum_square_actual=      5 | Tsum_square_expected=      5
# Tsum_actual=   6 | Tsum_expected=   6 | Tsum_square_actual=     14 | Tsum_square_expected=     14
# Tsum_actual=  10 | Tsum_expected=  10 | Tsum_square_actual=     30 | Tsum_square_expected=     30
# Tsum_actual=  15 | Tsum_expected=  15 | Tsum_square_actual=     55 | Tsum_square_expected=     55
# Tsum_actual=  21 | Tsum_expected=  21 | Tsum_square_actual=     91 | Tsum_square_expected=     91
# Tsum_actual=  28 | Tsum_expected=  28 | Tsum_square_actual=    140 | Tsum_square_expected=    140
# Tsum_actual=  36 | Tsum_expected=  36 | Tsum_square_actual=    204 | Tsum_square_expected=    204
# Tsum_actual=  45 | Tsum_expected=  45 | Tsum_square_actual=    285 | Tsum_square_expected=    285
# Tsum_actual=  55 | Tsum_expected=  55 | Tsum_square_actual=    385 | Tsum_square_expected=    385
# Tsum_actual=  66 | Tsum_expected=  66 | Tsum_square_actual=    506 | Tsum_square_expected=    506
# Tsum_actual=  78 | Tsum_expected=  78 | Tsum_square_actual=    650 | Tsum_square_expected=    650
# Tsum_actual=  91 | Tsum_expected=  91 | Tsum_square_actual=    819 | Tsum_square_expected=    819
# Tsum_actual= 105 | Tsum_expected= 105 | Tsum_square_actual=   1015 | Tsum_square_expected=   1015
# Tsum_actual= 119 | Tsum_expected= 119 | Tsum_square_actual=   1239 | Tsum_square_expected=   1239
# Tsum_actual= 133 | Tsum_expected= 133 | Tsum_square_actual=   1491 | Tsum_square_expected=   1491
# All tests for register_file module passed
```

Since the given testbench did not operate correctly, we made our own testbench using the same data as the test cases given to us, just inserted directly into the Verilog testbench file (source code shown at the end of the report).

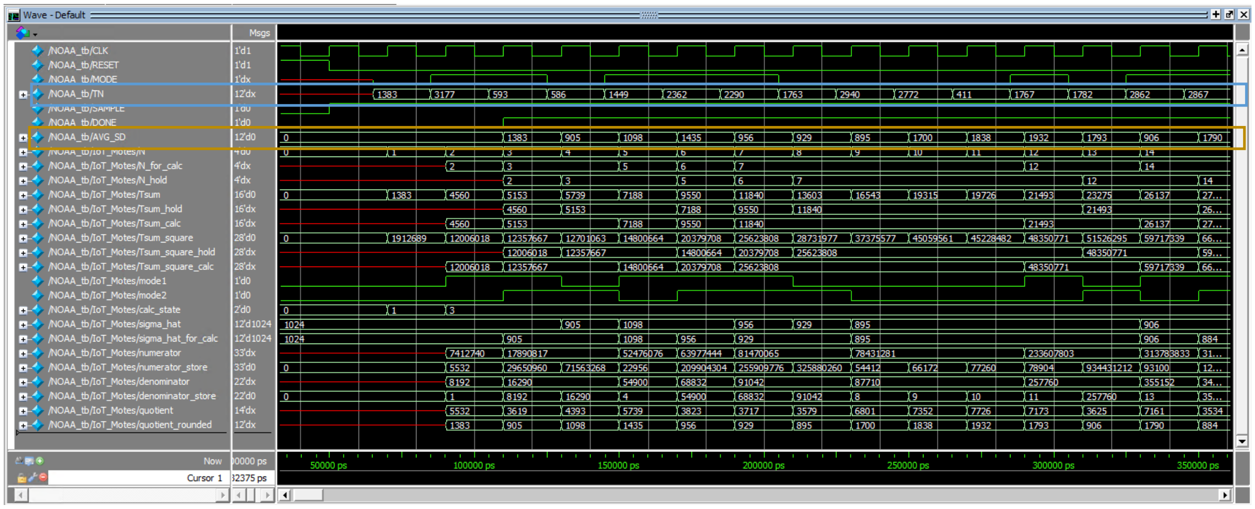

Figure 13: Modelsim Waveforms from 100 sample test case (NOAA_TEST_DATA_100_n.xlsx)

Here we can see from the waveform that the module generates the expected output (highlighted in yellow) 2 clock cycles after the input (highlighted in blue) was originally given.

We also added an output to the terminal, showing the expected value and the actual value for AVG_SD:

```
# AVG_SD (actual) = 1383 | AVG_SD (expected) = 1383
# AVG_SD (actual) =  905 | AVG_SD (expected) =  905
# AVG_SD (actual) = 1098 | AVG_SD (expected) = 1098
# AVG_SD (actual) = 1435 | AVG_SD (expected) = 1435
# AVG_SD (actual) =  956 | AVG_SD (expected) =  956
# AVG_SD (actual) =  929 | AVG_SD (expected) =  929
# AVG_SD (actual) =  895 | AVG_SD (expected) =  895
# AVG_SD (actual) = 1700 | AVG_SD (expected) = 1700
# AVG_SD (actual) = 1838 | AVG_SD (expected) = 1838
# AVG_SD (actual) = 1932 | AVG_SD (expected) = 1932
# AVG_SD (actual) = 1793 | AVG_SD (expected) = 1793
# AVG_SD (actual) =  906 | AVG_SD (expected) =  906
# AVG_SD (actual) = 1790 | AVG_SD (expected) = 1790
# AVG_SD (actual) =  884 | AVG_SD (expected) =  884
# AVG_SD (actual) =  908 | AVG_SD (expected) =  908
# AVG_SD (actual) = 1770 | AVG_SD (expected) = 1770
# AVG_SD (actual) = 1872 | AVG_SD (expected) = 1872
# AVG_SD (actual) =  912 | AVG_SD (expected) =  912
# AVG_SD (actual) = 1959 | AVG_SD (expected) = 1959
# AVG_SD (actual) = 1948 | AVG_SD (expected) = 1948
# AVG_SD (actual) =  942 | AVG_SD (expected) =  942
# AVG_SD (actual) =  969 | AVG_SD (expected) =  969
# AVG_SD (actual) =  962 | AVG_SD (expected) =  962
# AVG_SD (actual) = 1825 | AVG_SD (expected) = 1825
# AVG_SD (actual) = 2018 | AVG_SD (expected) = 2018
# AVG_SD (actual) = 2007 | AVG_SD (expected) = 2007
# AVG_SD (actual) = 2000 | AVG_SD (expected) = 2000
# AVG_SD (actual) =  937 | AVG_SD (expected) =  937
# AVG_SD (actual) = 1964 | AVG_SD (expected) = 1964
# AVG_SD (actual) =  815 | AVG_SD (expected) =  815
# AVG_SD (actual) =  852 | AVG_SD (expected) =  852
# AVG_SD (actual) =  702 | AVG_SD (expected) =  702
# AVG_SD (actual) =  706 | AVG_SD (expected) =  706
# AVG_SD (actual) =  832 | AVG_SD (expected) =  832
# AVG_SD (actual) =  822 | AVG_SD (expected) =  822
```

```
# AVG_SD (actual) =  922 | AVG_SD (expected) =  922
# AVG_SD (actual) =  889 | AVG_SD (expected) =  889
# AVG_SD (actual) =  913 | AVG_SD (expected) =  913
# AVG_SD (actual) = 1697 | AVG_SD (expected) = 1697
# AVG_SD (actual) = 1699 | AVG_SD (expected) = 1699
# AVG_SD (actual) = 1799 | AVG_SD (expected) = 1799
# AVG_SD (actual) = 1607 | AVG_SD (expected) = 1607
# AVG_SD (actual) = 1559 | AVG_SD (expected) = 1559
# AVG_SD (actual) = 1052 | AVG_SD (expected) = 1052
# AVG_SD (actual) =  988 | AVG_SD (expected) =  988
# AVG_SD (actual) = 1709 | AVG_SD (expected) = 1709
# AVG_SD (actual) = 1691 | AVG_SD (expected) = 1691
# AVG_SD (actual) = 1812 | AVG_SD (expected) = 1812
# AVG_SD (actual) = 1734 | AVG_SD (expected) = 1734
# AVG_SD (actual) =  965 | AVG_SD (expected) =  965
# AVG_SD (actual) = 1876 | AVG_SD (expected) = 1876
# AVG_SD (actual) = 1806 | AVG_SD (expected) = 1806
# AVG_SD (actual) =  944 | AVG_SD (expected) =  944
# AVG_SD (actual) = 2098 | AVG_SD (expected) = 2098
# AVG_SD (actual) = 1914 | AVG_SD (expected) = 1914
# AVG_SD (actual) = 1898 | AVG_SD (expected) = 1898
# AVG_SD (actual) = 1835 | AVG_SD (expected) = 1835
# AVG_SD (actual) = 1052 | AVG_SD (expected) = 1052
# AVG_SD (actual) = 1091 | AVG_SD (expected) = 1091
# AVG_SD (actual) = 1003 | AVG_SD (expected) = 1003
# AVG_SD (actual) = 1054 | AVG_SD (expected) = 1054
# AVG_SD (actual) = 1104 | AVG_SD (expected) = 1104
# AVG_SD (actual) = 1112 | AVG_SD (expected) = 1112
# AVG_SD (actual) = 1112 | AVG_SD (expected) = 1112
# AVG_SD (actual) = 1038 | AVG_SD (expected) = 1038
# AVG_SD (actual) = 1440 | AVG_SD (expected) = 1440
# AVG_SD (actual) =  961 | AVG_SD (expected) =  961
# AVG_SD (actual) =  901 | AVG_SD (expected) =  901
# AVG_SD (actual) =  943 | AVG_SD (expected) =  943
# AVG_SD (actual) = 1502 | AVG_SD (expected) = 1502
# AVG_SD (actual) = 1005 | AVG_SD (expected) = 1005
# AVG_SD (actual) = 1030 | AVG_SD (expected) = 1030
# AVG_SD (actual) = 1028 | AVG_SD (expected) = 1028
# AVG_SD (actual) = 1755 | AVG_SD (expected) = 1755
# AVG_SD (actual) =  961 | AVG_SD (expected) =  961
# AVG_SD (actual) =  957 | AVG_SD (expected) =  957
# AVG_SD (actual) =  960 | AVG_SD (expected) =  960
# AVG_SD (actual) = 1016 | AVG_SD (expected) = 1016
# AVG_SD (actual) = 1002 | AVG_SD (expected) = 1002
# AVG_SD (actual) = 1055 | AVG_SD (expected) = 1055
# AVG_SD (actual) = 1439 | AVG_SD (expected) = 1439
# AVG_SD (actual) = 1552 | AVG_SD (expected) = 1552
# AVG_SD (actual) = 1448 | AVG_SD (expected) = 1448
# AVG_SD (actual) = 1260 | AVG_SD (expected) = 1260
# AVG_SD (actual) = 1368 | AVG_SD (expected) = 1368
# AVG_SD (actual) = 1281 | AVG_SD (expected) = 1281
# AVG_SD (actual) = 1244 | AVG_SD (expected) = 1244
# AVG_SD (actual) =  775 | AVG_SD (expected) =  775
# AVG_SD (actual) =  764 | AVG_SD (expected) =  764
# AVG_SD (actual) = 1467 | AVG_SD (expected) = 1467
# AVG_SD (actual) =  856 | AVG_SD (expected) =  856
# AVG_SD (actual) = 1685 | AVG_SD (expected) = 1685
# AVG_SD (actual) =  880 | AVG_SD (expected) =  880
# AVG_SD (actual) =  816 | AVG_SD (expected) =  816
# AVG_SD (actual) =  940 | AVG_SD (expected) =  940
# AVG_SD (actual) =  950 | AVG_SD (expected) =  950
```

```
# AVG_SD (actual) =  951 | AVG_SD (expected) =  951
# AVG_SD (actual) = 1852 | AVG_SD (expected) = 1852
# AVG_SD (actual) =  918 | AVG_SD (expected) =  918
# AVG_SD (actual) =  909 | AVG_SD (expected) =  909
# All tests passed.
# ** Note: $stop    : /home/astee/ee465/FinalProject/NOAA_tb.v(51)
#    Time: 2130 ns  Iteration: 1  Instance: /NOAA_tb50;
```

Conclusion

Difficulties and learning experiences

Overall, this project was a successful exercise in understanding a problem, come up with a solution, and implement that solution in software through teamwork. The Babylonian method of calculating the standard deviation was a new formula that we learned through this project. We learned that by combining the multiple expressions for standard deviation, we could come up with a much more workable formula that fit our use case very well.

As far as difficulties, we learned in this project that it most certainly would have made the software development process smoother if we drew the schematic first, and then implemented the schematic in Verilog. In our approach, we wrote the software first, and then drew a schematic based on our Verilog code. Drawing the schematic first would have been a much more systematic approach, where we could have ironed out details ahead of time and made debugging the Verilog code much easier.

Feedback on project experience

Overall, this project was a good exercise in using HDL to solve a practical problem. The project left a lot of room for teammates to coordinate their efforts and come up with a unique design to solve the given problem.

As far as how the project could improve, I think it would be helpful to have a more gradual ramp up to the difficulty that this project presents in preceding labs. I felt like this project was a much more complex task than anything we had done in labs previously. Although they may not have had anything to do with the project directly, I think it would help bring more success if we had implemented more sophisticated HDL designs in labs before this.

Source Code

register_file.v

```
`timescale 1ns/1ps

module register_file(RESET, TN, SAMPLE, CLK, Tsum, Tsum_square );
input RESET;
input [11:0] TN;
input SAMPLE;
input CLK;

output [15:0] Tsum;
output [27:0] Tsum_square;

reg[11:0] TN1, TN2, TN3, TN4, TN5, TN6, TN7;
reg[11:0] TN8, TN9, TN10, TN11, TN12, TN13, TN14;

reg[23:0] TN1_sqr, TN2_sqr, TN3_sqr, TN4_sqr, TN5_sqr, TN6_sqr, TN7_sqr;
reg[23:0] TN8_sqr, TN9_sqr, TN10_sqr, TN11_sqr, TN12_sqr, TN13_sqr, TN14_sqr;

always @ (posedge CLK)
begin
    if(RESET == 1'b1) begin
    TN1 <= 1'b0;
    TN2 <= 1'b0;
    TN3 <= 1'b0;
    TN4 <= 1'b0;
    TN5 <= 1'b0;
    TN6 <= 1'b0;
    TN7 <= 1'b0;
    TN8 <= 1'b0;
    TN9 <= 1'b0;
    TN10 <= 1'b0;
    TN11 <= 1'b0;
    TN12 <= 1'b0;
    TN13 <= 1'b0;
    TN14 <= 1'b0;

    TN1_sqr <= 1'b0;
    TN2_sqr <= 1'b0;
    TN3_sqr <= 1'b0;
    TN4_sqr <= 1'b0;
    TN5_sqr <= 1'b0;
    TN6_sqr <= 1'b0;
    TN7_sqr <= 1'b0;
```

```
    TN8_sqr <= 1'b0;
    TN9_sqr <= 1'b0;
    TN10_sqr <= 1'b0;
    TN11_sqr <= 1'b0;
    TN12_sqr <= 1'b0;
    TN13_sqr <= 1'b0;
    TN14_sqr <= 1'b0;
    end
    else begin
        if( SAMPLE ) begin
            TN1 <= TN;
            TN2 <= TN1;
            TN3 <= TN2;
            TN4 <= TN3;
            TN5 <= TN4;
            TN6 <= TN5;
            TN7 <= TN6;
            TN8 <= TN7;
            TN9 <= TN8;
            TN10 <= TN9;
            TN11 <= TN10;
            TN12 <= TN11;
            TN13 <= TN12;
            TN14 <= TN13;

            TN1_sqr <= TN**2;
            TN2_sqr <= TN1_sqr;
            TN3_sqr <= TN2_sqr;
            TN4_sqr <= TN3_sqr;
            TN5_sqr <= TN4_sqr;
            TN6_sqr <= TN5_sqr;
            TN7_sqr <= TN6_sqr;
            TN8_sqr <= TN7_sqr;
            TN9_sqr <= TN8_sqr;
            TN10_sqr <= TN9_sqr;
            TN11_sqr <= TN10_sqr;
            TN12_sqr <= TN11_sqr;
            TN13_sqr <= TN12_sqr;
            TN14_sqr <= TN13_sqr;
        end
    end
end

assign Tsum = TN1 + TN2 + TN3 + TN4 + TN5 + TN6 + TN7 + TN8 + TN9 + TN10 + TN11 +
 TN12 + TN13 + TN14;
```

```
assign Tsum_square = TN1_sqr + TN2_sqr + TN3_sqr + TN4_sqr + TN5_sqr + TN6_sqr +
TN7_sqr + TN8_sqr + TN9_sqr + TN10_sqr + TN11_sqr + TN12_sqr + TN13_sqr + TN14_sq
r;

endmodule
```

register_file_tb.v

```
`include "register_file.v"

`timescale 1ns/1ns

module register_file_tb;

reg reset;
reg [11:0] tn;
reg sample;
reg clk;
wire [15:0] tsum_actual;
reg [15:0] tsum_expected;
wire [27:0] tsum_square_actual;
reg [27:0] tsum_square_expected;

always #10 clk <= ~clk;

integer i;

initial begin
    $display("Running tests for register_file module");
    $display("Tsum_actual=%d | Tsum_expected=%d | Tsum_square_actual=%d | Tsum_sq
uare_expected=%d",
    tsum_actual,
    tsum_expected,
    tsum_square_actual,
    tsum_square_expected
    );

    clk = 0;
    reset = 1;
    sample = 0;
    #40
    reset = 0;
    sample = 1;
```

```
    for ( i = 1; i <= 16; i = i + 1 ) begin
        tn = i;
        #10
        case ( i )
            1: begin
                tsum_expected = 1;
                tsum_square_expected = 1**2;
            end
            2: begin
                tsum_expected = 1+2;
                tsum_square_expected = 1**2 + 2**2;
            end
            3: begin
                tsum_expected = 1+2+3;
                tsum_square_expected = 1**2 + 2**2 + 3**2;
            end
            4: begin
                tsum_expected = 1+2+3+4;
                tsum_square_expected = 1**2 + 2**2 + 3**2 + 4**2;
            end
            5: begin
                tsum_expected = 1+2+3+4+5;
                tsum_square_expected = 1**2 + 2**2 + 3**2 + 4**2 + 5**2;
            end
            6: begin
                tsum_expected = 1+2+3+4+5+6;
                tsum_square_expected = 1**2 + 2**2 + 3**2 + 4**2 + 5**2 + 6**2;
            end
            7: begin
                tsum_expected = 1+2+3+4+5+6+7;
                tsum_square_expected = 1**2 + 2**2 + 3**2 + 4**2 + 5**2 + 6**2 +
7**2;
            end
            8: begin
                tsum_expected = 1+2+3+4+5+6+7+8;
                tsum_square_expected = 1**2 + 2**2 + 3**2 + 4**2 + 5**2 + 6**2 +
7**2 + 8**2;
            end
            9: begin
                tsum_expected = 1+2+3+4+5+6+7+8+9;
                tsum_square_expected = 1**2 + 2**2 + 3**2 + 4**2 + 5**2 + 6**2 +
7**2 + 8**2 + 9**2;
            end
            10: begin
                tsum_expected = 1+2+3+4+5+6+7+8+9+10;
```

```
                tsum_square_expected = 1**2 + 2**2 + 3**2 + 4**2 + 5**2 + 6**2 +
7**2 + 8**2 + 9**2 + 10**2;
            end
            11: begin
                tsum_expected = 1+2+3+4+5+6+7+8+9+10+11;
                tsum_square_expected = 1**2 + 2**2 + 3**2 + 4**2 + 5**2 + 6**2 +
7**2 + 8**2 + 9**2 + 10**2 + 11**2;
            end
            12: begin
                tsum_expected = 1+2+3+4+5+6+7+8+9+10+11+12;
                tsum_square_expected = 1**2 + 2**2 + 3**2 + 4**2 + 5**2 + 6**2 +
7**2 + 8**2 + 9**2 + 10**2 + 11**2 + 12**2;
            end
            13: begin
                tsum_expected = 1+2+3+4+5+6+7+8+9+10+11+12+13;
                tsum_square_expected = 1**2 + 2**2 + 3**2 + 4**2 + 5**2 + 6**2 +
7**2 + 8**2 + 9**2 + 10**2 + 11**2 + 12**2 + 13**2;
            end
            14: begin
                tsum_expected = 1+2+3+4+5+6+7+8+9+10+11+12+13+14;
                tsum_square_expected = 1**2 + 2**2 + 3**2 + 4**2 + 5**2 + 6**2 +
7**2 + 8**2 + 9**2 + 10**2 + 11**2 + 12**2 + 13**2 + 14**2;
            end
            15: begin
                tsum_expected = 2+3+4+5+6+7+8+9+10+11+12+13+14+15;
                tsum_square_expected = 2**2 + 3**2 + 4**2 + 5**2 + 6**2 + 7**2 +
8**2 + 9**2 + 10**2 + 11**2 + 12**2 + 13**2 + 14**2 + 15**2;
            end
            16: begin
                tsum_expected = 3+4+5+6+7+8+9+10+11+12+13+14+15+16;
                tsum_square_expected = 3**2 + 4**2 + 5**2 + 6**2 + 7**2 + 8**2 +
9**2 + 10**2 + 11**2 + 12**2 + 13**2 + 14**2 + 15**2 + 16**2;
            end
        endcase
        #5

        $display("Tsum_actual=%d | Tsum_expected=%d | Tsum_square_actual=%d | Tsu
m_square_expected=%d",
        tsum_actual,
        tsum_expected,
        tsum_square_actual,
        tsum_square_expected
        );
```

```
        if ( tsum_expected != tsum_actual || tsum_square_expected != tsum_square_
actual ) begin
            $error( "Wrong output for iteration: %d", i);
            $stop;
        end
        #5;
    end

    $display( "All tests for register_file module passed" );
end


register_file reg_file(
    .RESET(reset),
    .TN(tn),
    .SAMPLE(sample),
    .CLK(clk),
    .Tsum(tsum_actual),
    .Tsum_square(tsum_square_actual)
    );

endmodule
```

calculate_numerator_denominator.v

```
`timescale 1ns/1ns

module calculate_numerator_denominator(
    input [11:0] sigma_hat,
    input [3:0] N,
    input [15:0] Tsum,
    input [27:0] Tsum_square,
    output [32:0] numerator, // can be up to 33 bits large
    output [21:0] denominator
);

assign numerator = ((sigma_hat**2) * (N**2)) + (N*Tsum_square) - (Tsum**2);
assign denominator = 2 * (N**2) * sigma_hat;

endmodule
```

calculate_numerator_denominator_tb.v

NOAA_module.v

```
`include "register_file.v"
`include "calculate_numerator_denominator.v"

`timescale 1ns/1ns

module NOAA_module(
    input RESET,
    input MODE,
    input [11:0] TN,
    input CLK,
    output reg SAMPLE,
    output reg DONE, // Need to put calc_num and calc_denom on the clk to set DON
E appropriately (maybe)
    output reg [11:0] AVG_SD
);

reg [3:0] N;
wire [3:0] N_for_calc;
reg [3:0] N_hold;

wire [15:0] Tsum;
reg [15:0] Tsum_hold;
wire [15:0] Tsum_calc;

wire [27:0] Tsum_square;
reg [27:0] Tsum_square_hold;
wire [27:0] Tsum_square_calc;

reg mode1; // used to transfer keep the mode that was given with the input
reg mode2;
reg [1:0] calc_state; // in what stage of the calculation pipeline are we in
//(basically used when the module is reset to get the pipeline going)

reg [11:0] sigma_hat;
wire [11:0] sigma_hat_for_calc;

wire [32:0] numerator;
reg [32:0] numerator_store;
wire [21:0] denominator;
reg [21:0] denominator_store;

wire [1:0] quotient; // needs to be 13 bits instead of 12 bits to leave room for
rounding
wire [11:0] quotient_rounded;
```

```
// This is an idea to save switching for these regs, idk if it'll work
assign N_for_calc = mode1 ? N : N_hold;
assign Tsum_calc = ( mode1 ) ? Tsum : Tsum_hold;
assign Tsum_square_calc = ( mode1 ) ? Tsum_square : Tsum_square_hold;

assign quotient = numerator_store / denominator_store;
assign quotient_rounded[11:0] = quotient[0]?(quotient[12:1]+1):quotient[12:1]; //
 round up if needed
assign sigma_hat_for_calc = ( mode1 && mode2 ) ? quotient_rounded : sigma_hat; //
 use the latest sigma_hat if we've calculated a two stddev's in a row

always @ ( posedge CLK )
begin
    if ( RESET )
    begin
        SAMPLE <= 1'b0;
        DONE <= 1'b0;
        sigma_hat <= 12'b010000000000; // 32deg F
        AVG_SD <= 0;
        N <= 0;

        calc_state <= 0;

        numerator_store <= 0;
        denominator_store <= 0;

        mode1 <= 0;
        mode2 <= 0;
    end
    else
    begin
        SAMPLE <= 1'b1; // should be able to sample every CLK cycle
        if ( SAMPLE )
        begin
            if (N < 14) begin
                N <= N + 1;
            end

            Tsum_hold <= Tsum_calc;
            Tsum_square_hold <= Tsum_square_calc;

            N_hold <= N_for_calc;

            mode1 <= MODE;
```

```
            calc_state[0] <= 1;
        end

        if ( calc_state[0] ) // data available in first stage
        begin
            numerator_store <= ( mode1 ) ? ( numerator << 1 ) : ( Tsum << 1 ); //
 not sure why I have to multiply the numerator by 4, but whatever
            denominator_store <= ( mode1 ) ? ( denominator ) : ( N );

            calc_state[1] <= 1;
            mode2 <= mode1;
        end

        if ( calc_state[1] ) // data available in the second stage
        begin
            if ( mode2 == 1'b1 ) // update sigma_hat if we've calculated a new st
ddev
                sigma_hat <= quotient_rounded;

            AVG_SD <= quotient_rounded; // perform the division
            DONE <= 1;
        end
        else
            DONE <= 0; // no new output ready yet
    end
end

register_file reg_file(
    .RESET( RESET ),
    .TN( TN ),
    .SAMPLE( SAMPLE ),
    .CLK( CLK ),
    .Tsum( Tsum ),
    .Tsum_square( Tsum_square )
);

calculate_numerator_denominator calc_num(
    .N( N_for_calc ),
    .sigma_hat( sigma_hat_for_calc ),
    .Tsum( Tsum_calc ),
    .Tsum_square( Tsum_square_calc ),
    .numerator( numerator ),
    .denominator( denominator )
);
```

```
endmodule
```

register_file_tb.v

```
`include "register_file.v"

`timescale 1ns/1ns

module register_file_tb;

reg reset;
reg [11:0] tn;
reg sample;
reg clk;
wire [15:0] tsum_actual;
reg [15:0] tsum_expected;
wire [27:0] tsum_square_actual;
reg [27:0] tsum_square_expected;

always #10 clk <= ~clk;

integer i;

initial begin
    $display("Running tests for register_file module");
    $display("Tsum_actual=%d | Tsum_expected=%d | Tsum_square_actual=%d | Tsum_sq
uare_expected=%d",
    tsum_actual,
    tsum_expected,
    tsum_square_actual,
    tsum_square_expected
    );

    clk = 0;
    reset = 1;
    sample = 0;
    #40
    reset = 0;
    sample = 1;

    for ( i = 1; i <= 16; i = i + 1 ) begin
        tn = i;
        #10
        case ( i )
            1: begin
```

```
                tsum_expected = 1;
                tsum_square_expected = 1**2;
            end
            2: begin
                tsum_expected = 1+2;
                tsum_square_expected = 1**2 + 2**2;
            end
            3: begin
                tsum_expected = 1+2+3;
                tsum_square_expected = 1**2 + 2**2 + 3**2;
            end
            4: begin
                tsum_expected = 1+2+3+4;
                tsum_square_expected = 1**2 + 2**2 + 3**2 + 4**2;
            end
            5: begin
                tsum_expected = 1+2+3+4+5;
                tsum_square_expected = 1**2 + 2**2 + 3**2 + 4**2 + 5**2;
            end
            6: begin
                tsum_expected = 1+2+3+4+5+6;
                tsum_square_expected = 1**2 + 2**2 + 3**2 + 4**2 + 5**2 + 6**2;
            end
            7: begin
                tsum_expected = 1+2+3+4+5+6+7;
                tsum_square_expected = 1**2 + 2**2 + 3**2 + 4**2 + 5**2 + 6**2 +
7**2;
            end
            8: begin
                tsum_expected = 1+2+3+4+5+6+7+8;
                tsum_square_expected = 1**2 + 2**2 + 3**2 + 4**2 + 5**2 + 6**2 +
7**2 + 8**2;
            end
            9: begin
                tsum_expected = 1+2+3+4+5+6+7+8+9;
                tsum_square_expected = 1**2 + 2**2 + 3**2 + 4**2 + 5**2 + 6**2 +
7**2 + 8**2 + 9**2;
            end
            10: begin
                tsum_expected = 1+2+3+4+5+6+7+8+9+10;
                tsum_square_expected = 1**2 + 2**2 + 3**2 + 4**2 + 5**2 + 6**2 +
7**2 + 8**2 + 9**2 + 10**2;
            end
            11: begin
                tsum_expected = 1+2+3+4+5+6+7+8+9+10+11;
```

```
                tsum_square_expected = 1**2 + 2**2 + 3**2 + 4**2 + 5**2 + 6**2 +
7**2 + 8**2 + 9**2 + 10**2 + 11**2;
            end
            12: begin
                tsum_expected = 1+2+3+4+5+6+7+8+9+10+11+12;
                tsum_square_expected = 1**2 + 2**2 + 3**2 + 4**2 + 5**2 + 6**2 +
7**2 + 8**2 + 9**2 + 10**2 + 11**2 + 12**2;
            end
            13: begin
                tsum_expected = 1+2+3+4+5+6+7+8+9+10+11+12+13;
                tsum_square_expected = 1**2 + 2**2 + 3**2 + 4**2 + 5**2 + 6**2 +
7**2 + 8**2 + 9**2 + 10**2 + 11**2 + 12**2 + 13**2;
            end
            14: begin
                tsum_expected = 1+2+3+4+5+6+7+8+9+10+11+12+13+14;
                tsum_square_expected = 1**2 + 2**2 + 3**2 + 4**2 + 5**2 + 6**2 +
7**2 + 8**2 + 9**2 + 10**2 + 11**2 + 12**2 + 13**2 + 14**2;
            end
            15: begin
                tsum_expected = 2+3+4+5+6+7+8+9+10+11+12+13+14+15;
                tsum_square_expected = 2**2 + 3**2 + 4**2 + 5**2 + 6**2 + 7**2 +
8**2 + 9**2 + 10**2 + 11**2 + 12**2 + 13**2 + 14**2 + 15**2;
            end
            16: begin
                tsum_expected = 3+4+5+6+7+8+9+10+11+12+13+14+15+16;
                tsum_square_expected = 3**2 + 4**2 + 5**2 + 6**2 + 7**2 + 8**2 +
9**2 + 10**2 + 11**2 + 12**2 + 13**2 + 14**2 + 15**2 + 16**2;
            end
        endcase
        #5

        $display("Tsum_actual=%d | Tsum_expected=%d | Tsum_square_actual=%d | Tsu
m_square_expected=%d",
        tsum_actual,
        tsum_expected,
        tsum_square_actual,
        tsum_square_expected
        );

        if ( tsum_expected != tsum_actual || tsum_square_expected != tsum_square_
actual ) begin
            $error( "Wrong output for iteration: %d", i);
            $stop;
        end
        #5;
```

```
    end

    $display( "All tests for register_file module passed" );
end
register_file reg_file(
    .RESET(reset),
    .TN(tn),
    .SAMPLE(sample),
    .CLK(clk),
    .Tsum(tsum_actual),
    .Tsum_square(tsum_square_actual)
    );

endmodule
```

NOAA_module_tb.v

```
`timescale 1ns/1ns

module NOAA_tb();

  // Ports of __NOAA_Module__
  reg       CLK, RESET, MODE;
  reg       [11:0] TN;
  wire      SAMPLE, DONE;
  wire      [11:0] AVG_SD;

  reg [11:0] result_expected;
  reg [11:0] result_expected_0;
  reg [11:0] result_expected_1;
  reg [11:0] result_expected_2;

  initial begin
    CLK = 0;
    RESET = 1;
    result_expected = 0;
    result_expected_0 = 0;
    result_expected_1 = 0;
    result_expected_2 = 0;
  end


  initial begin
    #50
    RESET = 0;
```

```
  end

  always #10
  begin
    CLK = ~CLK;
  end

  always @ (posedge CLK)
  begin
    result_expected = result_expected_0;
    result_expected_0 = result_expected_1;
    result_expected_1 = result_expected_2;

    if ( DONE )
    begin
      $display( "AVG_SD (actual) = %d | AVG_SD (expected) = %d", AVG_SD, result_e
xpected );
      if ( AVG_SD != result_expected )
      begin
        $error( "Wrong output generated!" );
        $stop;
      end
    end
  end

// copied numbers from NOAA_Test_Data_30_n.xlsx
  initial begin // numbers from test dataset of 100
    #65
    TN = 1383; MODE =   0; result_expected_2 =  1383;
    #20
    TN = 3177; MODE =   1; result_expected_2 =  905;
    #20
    TN = 593; MODE =  1; result_expected_2 =  1098;
    #20
    TN = 586; MODE =  0; result_expected_2 =  1435;
    #20
    TN = 1449; MODE =   1; result_expected_2 =  956;
    #20
    TN = 2362; MODE =   1; result_expected_2 =  929;
    #20
    TN = 2290; MODE =   1; result_expected_2 =  895;
    #20
    TN = 1763; MODE =   0; result_expected_2 =  1700;
    #20
    TN = 2940; MODE =   0; result_expected_2 =  1838;
```

```
#20
TN = 2772; MODE =   0; result_expected_2 =  1932;
#20
TN = 411; MODE =  0; result_expected_2 =  1793;
#20
TN = 1767; MODE =   1; result_expected_2 =  906;
#20
TN = 1782; MODE =   0; result_expected_2 =  1790;
#20
TN = 2862; MODE =   1; result_expected_2 =  884;
#20
TN = 2867; MODE =   1; result_expected_2 =  908;
#20
TN = 329; MODE =  0; result_expected_2 =  1770;
#20
TN = 2022; MODE =   0; result_expected_2 =  1872;
#20
TN = 269; MODE =  1; result_expected_2 =  912;
#20
TN = 2993; MODE =   0; result_expected_2 =  1959;
#20
TN = 2211; MODE =   0; result_expected_2 =  1948;
#20
TN = 1829; MODE =   1; result_expected_2 =  942;
#20
TN = 2821; MODE =   1; result_expected_2 =  969;
#20
TN = 984; MODE =  1; result_expected_2 =  962;
#20
TN = 2398; MODE =   0; result_expected_2 =  1825;
#20
TN = 3115; MODE =   0; result_expected_2 =  2018;
#20
TN = 1613; MODE =   0; result_expected_2 =  2007;
#20
TN = 1691; MODE =   0; result_expected_2 =  2000;
#20
TN = 3156; MODE =   1; result_expected_2 =  937;
#20
TN = 2062; MODE =   0; result_expected_2 =  1964;
#20
TN = 1396; MODE =   1; result_expected_2 =  815;
#20
TN = 3105; MODE =   1; result_expected_2 =  852;
#20
```

```
    TN = 1884; MODE =   1; result_expected_2 =  702;
    #20
    TN = 1136; MODE =   1; result_expected_2 =  706;
    #20
    TN = 446; MODE =  1; result_expected_2 =  832;
    #20
    TN = 2113; MODE =   1; result_expected_2 =  822;
    #20
    TN = 124; MODE =  1; result_expected_2 =  922;
    #20
    TN = 1982; MODE =   1; result_expected_2 =  889;
    #20
    TN = 2814; MODE =   1; result_expected_2 =  913;
    #20
    TN = 234; MODE =  0; result_expected_2 =  1697;
    #20
    TN = 1643; MODE =   0; result_expected_2 =  1699;
    #20
    TN = 3087; MODE =   0; result_expected_2 =  1799;
    #20
    TN = 476; MODE =  0; result_expected_2 =  1607;
    #20
    TN = 1388; MODE =   0; result_expected_2 =  1559;
    #20
    TN = 3003; MODE =   1; result_expected_2 =  1052;
    #20
    TN = 2354; MODE =   1; result_expected_2 =  988;
    #20
    TN = 3132; MODE =   0; result_expected_2 =  1709;
    #20
    TN = 876; MODE =  0; result_expected_2 =  1691;
    #20
    TN = 2139; MODE =   0; result_expected_2 =  1812;
    #20
    TN = 1026; MODE =   0; result_expected_2 =  1734;
    #20
    TN = 894; MODE =  1; result_expected_2 =  965;
    #20
    TN = 3195; MODE =   0; result_expected_2 =  1876;
    #20
    TN = 1834; MODE =   0; result_expected_2 =  1806;
    #20
    TN = 3067; MODE =   1; result_expected_2 =  944;
    #20
    TN = 2897; MODE =   0; result_expected_2 =  2098;
```

```
    #20
    TN = 517; MODE =  0; result_expected_2 =  1914;
    #20
    TN = 252; MODE =  0; result_expected_2 =  1898;
    #20
    TN = 501; MODE =  0; result_expected_2 =  1835;
    #20
    TN = 1086; MODE =   1; result_expected_2 =  1052;
    #20
    TN = 265; MODE =  1; result_expected_2 =  1091;
    #20
    TN = 1244; MODE =   1; result_expected_2 =  1003;
    #20
    TN = 40; MODE =   1; result_expected_2 =  1054;
    #20
    TN = 2831; MODE =   1; result_expected_2 =  1104;
    #20
    TN = 2097; MODE =   1; result_expected_2 =  1112;
    #20
    TN = 881; MODE =  1; result_expected_2 =  1112;
    #20
    TN = 2309; MODE =   1; result_expected_2 =  1038;
    #20
    TN = 2167; MODE =   0; result_expected_2 =  1440;
    #20
    TN = 1897; MODE =   1; result_expected_2 =  961;
    #20
    TN = 186; MODE =  1; result_expected_2 =  901;
    #20
    TN = 2506; MODE =   1; result_expected_2 =  943;
    #20
    TN = 3019; MODE =   0; result_expected_2 =  1502;
    #20
    TN = 328; MODE =  1; result_expected_2 =  1005;
    #20
    TN = 2532; MODE =   1; result_expected_2 =  1030;
    #20
    TN = 3103; MODE =   1; result_expected_2 =  1028;
    #20
    TN = 670; MODE =  0; result_expected_2 =  1755;
    #20
    TN = 1308; MODE =   1; result_expected_2 =  961;
    #20
    TN = 740; MODE =  1; result_expected_2 =  957;
    #20
```

```
    TN = 2196; MODE =   1; result_expected_2 =  960;
    #20
    TN = 218; MODE =  1; result_expected_2 =  1016;
    #20
    TN = 1246; MODE =   1; result_expected_2 =  1002;
    #20
    TN = 121; MODE =  1; result_expected_2 =  1055;
    #20
    TN = 1979; MODE =   0; result_expected_2 =  1439;
    #20
    TN = 1764; MODE =   0; result_expected_2 =  1552;
    #20
    TN = 1041; MODE =   0; result_expected_2 =  1448;
    #20
    TN = 393; MODE =  0; result_expected_2 =  1260;
    #20
    TN = 1834; MODE =   0; result_expected_2 =  1368;
    #20
    TN = 1324; MODE =   0; result_expected_2 =  1281;
    #20
    TN = 2587; MODE =   0; result_expected_2 =  1244;
    #20
    TN = 943; MODE =  1; result_expected_2 =  775;
    #20
    TN = 2227; MODE =   1; result_expected_2 =  764;
    #20
    TN = 2659; MODE =   0; result_expected_2 =  1467;
    #20
    TN = 2632; MODE =   1; result_expected_2 =  856;
    #20
    TN = 2837; MODE =   0; result_expected_2 =  1685;
    #20
    TN = 2875; MODE =   1; result_expected_2 =  880;
    #20
    TN = 674; MODE =  1; result_expected_2 =  816;
    #20
    TN = 58; MODE =   1; result_expected_2 =  940;
    #20
    TN = 2429; MODE =   1; result_expected_2 =  950;
    #20
    TN = 1035; MODE =   1; result_expected_2 =  951;
    #20
    TN = 1818; MODE =   0; result_expected_2 =  1852;
    #20
    TN = 2943; MODE =   1; result_expected_2 =  918;
```

```
    #20
    TN = 1528; MODE =   1; result_expected_2 =  909;
    #20;

  end

  NOAA_module IoT_Motes(
                  .CLK(CLK),
                  .RESET(RESET),
                  .MODE(MODE),
                  .TN(TN),
                  .SAMPLE(SAMPLE),
                  .DONE(DONE),
                  .AVG_SD(AVG_SD)
                  );
endmodule
```