Finite Impulse Response (FIR) Filter Design

Design of a 64-tap 16-bit FIR Filter in Verilog

Jens Daci
M.S. Student in Electrical Engineering
Columbia University
jd3693@columbia.edu

Mateusz Ardito-Proulx
M.S. Student in Electrical Engineering
Columbia University
mma2256@columbia.edu

***Abstract*—In this paper, the design, implementation, and testing of an FIR filter will be discussed. The design of the filter was written in Verilog and tested in QuestaSim simulation. A parameter report was conducted on the filter where the metrics such as power consumption, timing and delay were recorded.**

***Key Terms*—**FIR, Verilog, MATLAB, ASM, PrimeTime

I. Introduction

A FIR (Finite Impulse Response) filter is a filter whose impulse response settles to a discreet value in a finite amount of time. These filters are used in many applications of digital electronics and serves as a steppingstone project for students interested in system level design as well as filter design.

The FIR filter that will be designed as part of this report is a 64-tap 16-bit FIR filter. In other words, the filter would have 64 coefficients and each input would be 16-bits. The input memory of this filter would be 16k of 2B data, whereas the coefficients memory would be 64 of 2B data. The FIR filter should have a fixed-point ALU that can only use one adder and one multiplier in the core. It should also use registers to store data as well as a Finite State Machine (FSM) to manage the control of the data throughout the core. Written in a mathematical notation, the FIR filter should perform the following calculations:

$$y[n] = \sum_{i=0}^{N} b_i * x[n-i]$$

In this notation, $x[n]$ represents the input signal, $y[n]$ represents the output signal, N is the filter order, and b_i are the filter coefficients.

The functionality and testing of the filter is written in Verilog and handled through the extensive QuestaSim simulation software. MATLAB was a tool used in our testing process to generate a golden output to compare to our testbench generated values. After functionality was verified, a netlist was created as well as tested again to generate a cycle-accurate switching activities file and perform a sizing report on the chip area layout. Prime time was used to generate power performance and timing analysis tests. All of these reports are synthesized below and compared to the project constraints given to us in the project metrics outline.

In addition to these requirements, the FIR filter should also be able to have a throughput of $10kS/s$ and the testbench written for the FIR core should load data into the memories, provide a clock signal, and perform the signature analyzer.

II. MATLAB

A. Generating the FIR Output

Before implementing the design using Verilog, the behavior of an FIR filter was created in MATLAB. At first, the code generates 16k (or 16,384) random integers in the range from 0 to 1000. These will serve as the input signal to the FIR filter. Then, the script generates 64 random integers in the same range as the inputs. These will serve as the coefficients needed for the FIR filter. There are 64 coefficients since the requirements are to design a 64-tap FIR filter. After the generation of the random numbers, they are all stored in text files that can be used as testing inputs to the FIR core's testbench.

Now, the script is ready to calculate the output of the filter based on the previously mentioned algorithm. As seen in the equation on the left, the N represents the 64 coefficients, whereas the n represents the 16,384 input entries into the filter. At first, the multiplication between the coefficients and the previous input is performed. Next, the results of the 64 multiplications are added together for each instance. In the end, the output signal is stored in a text file that can be used as a golden output for the FIR core's testbench.

B. Matlab Output Verification

The output from the behavioral code was also tested using a pre-defined FIR filter function in MATLAB. For each of the 16,384 generated outputs, the verification script checks the output of the summation formula with the function's output. This way one is certain that the MATLAB output mentioned in part A is a golden output.

III. Design Diagrams

An ASM (Algorithmic State Machine) is a great way to start translating the design into Verilog. In order for us to use an ASM, we must start by designing the pseudocode for our FIR filter. The pseudocode will serve as the first building block in designing the FIR filter in Verilog. With the help of the pseudocode, the ASM chart will be constructed. From the ASM chart we can build the Datapath Block Diagram and the Control ASM Chart. In the following sub sections these parts of the project are discussed thoroughly.

A. PseudoCode

The following snippet shows the pseudocode for the FIR filter that needs to be designed. The M variable corresponds to the multiplication results, whereas the Y variable corresponds to the output of the FIR filter which is the summation of all of

the products. The loop will run for 16,384 data input entries and the entries will be multiplied by the 64 coefficients of the FIR filter.

```
Y = 0;
M = 0;
for n = 0 to 16383 do
      for i = 0 to 63 do
            M[i] = B[i] * X[n-i];
            Y[n] = Y[n] + M[i];
      end for;
end for;
```

B. *ASM Chart*

The ASM Chart takes the pseudocode and translates it into an implementable FSM that tracks where we are in our program through different states. An ASM chart is a method of describing the sequential operations of a digital system. The ASM shown in Figure 1 below, shows the sequential flow of operations in the FIR filter system. In total, it includes 6 different states. The chart will help guide our creation of the Datapath Block Diagram and Control ASM Chart.

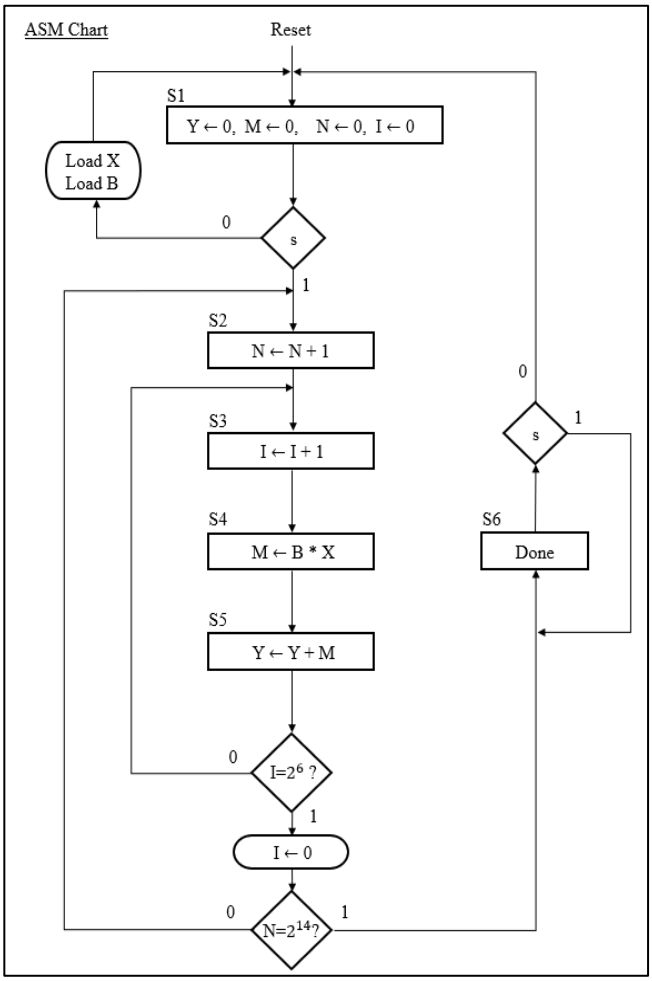


Fig 1. ASM Chart

C. *Datapath Block Diagram*

The Datapath Block Diagram presents us with a system level look at how our data is being manipulated and stored. The registers and counters that will be used for the FIR core are the following:

- Registers X, B – to store the data from both memories
- Register M – to store the multiplication results
- Register Y – to store the output
- Counters N, I – upcounters needed for IMEM and CMEM

The IMEM and CMEM registers hold the values of the inputs and coefficients, respectively. The first inputs are read from their respective registers and are stored in a register before the ALU computations. Next, a multiplexer decides the selection of the operands into the ALU based on a select bit. When the operands at the input of the ALU are ready for computation, a select line is fed into the ALU telling it to either add or multiply. The different selections of the ALU are multiplication (ALUsel = 0) and addition (ALUsel = 1). Afterwards, the output of the ALU goes to another pair of multiplexers to decide if the output we computed is going into the output register Y or the register M for further computations. The figure for the Datapath Block Diagram is shown below:

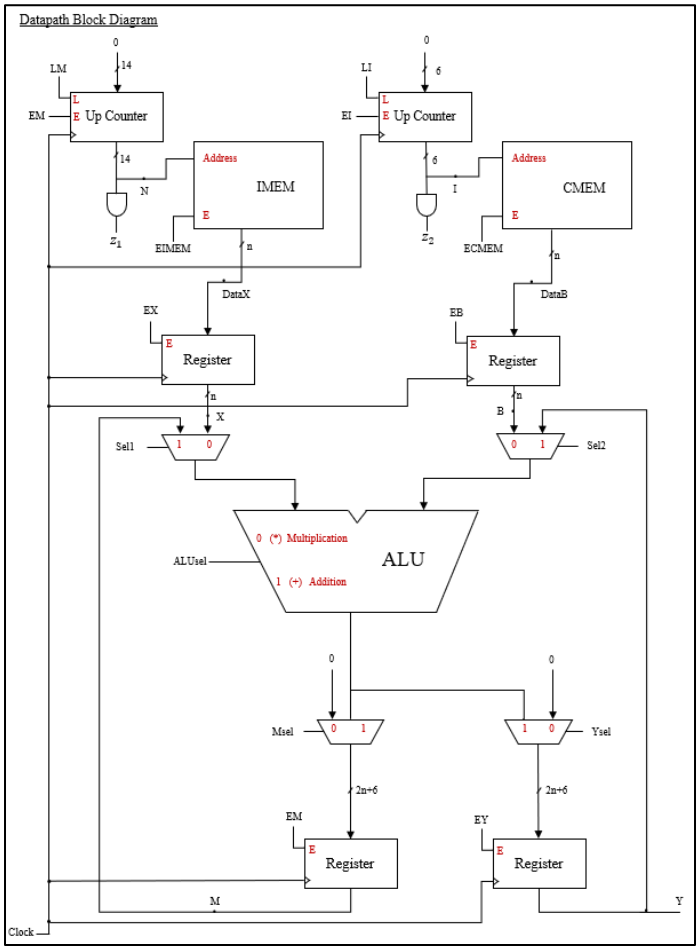


Fig 2. Datapath Block Diagram

D. Control ASM Chart

The Control ASM presents us with the characteristic of important control signals at the different states in the FSM for the system. Each state is associated with a different functionality of the overall FIR filter and needs certain signals to be assigned accordingly. The Control ASM chart is a combination between the ASM Chart found in Figure 1 and Datapath Block Diagram found in Figure 2.

At the first state, all of the registers and upcounters used in the design are initialized to their default mode of operation. The registers are reset, whereas the upcounters are loaded with a value of zero. In this state, the loading of the IMEM and the CMEM memory will also take place. In order to start the calculations for the FIR filter, the start variable, s, must be set to a logic high.

The operations of the FIR filter's core start with State 2 and end at State 5. During these states, through the use of the signals in the ALU, in the registers and in the upcounters, the algorithm is able to calculate the product and sum of the input and the coefficients. Lastly, State 6 will take care of the Done signal.

It is assumed that when the signal name is called in the block, the signal is high for that state, and when the signal name is set equal to zero, it is low. The Control ASM Chart is shown below:

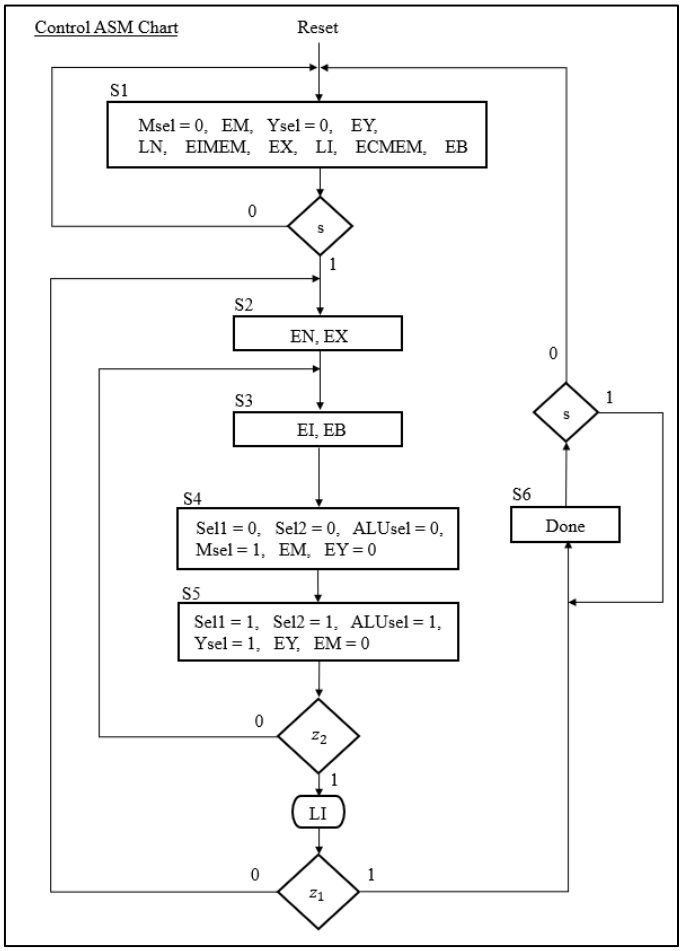


Fig 3. Control ASM Chart

IV. RTL Design

With the completed development of the block diagrams, the RTL design of the FIR filter system will be discussed. In digital circuit design, register-transfer level (RTL) is a design abstraction which models a synchronous digital circuit in terms of the flow of digital signals (data) between hardware registers, and the logical operations performed on those signals. By observing the different diagrams shown in the report, it is clear that the design will be divided into a control circuit and a datapath circuit. Both of these parts of the design will be explained in the next few sections.

A. Control Circuit

The control circuit is the brains of the Finite State Machine where all of the different states are switched, and the output is produced based on the current state. The control circuit in itself is also divided into three different stages which are the state table, the sequential state switching, and the production of the FSM outputs.

The state table keeps track of the next state based on the internal mechanisms of the FIR core. In the FIR algorithm presented in this report, the next state is determined by looking at the present state, the start variable, s, and the upcounter limit check signals $z1$ and $z2$. In addition, the state switching algorithm takes care of the present state of the filter at every positive edge of the clock. In order to reset the calculations and the FIR core, a reset signal is also included that assigns the present state back to State 1.

The FSM outputs section of the control circuit takes care of the input signals to each of the blocks in the system. This section is entirely based on the Control ASM chart seen in Figure 3 on the left. Based on present state of the machine, the algorithm is able to produce the required outputs.

B. Datapath Circuit

In the datapath circuit section of the Verilog code, all of the required digital blocks seen in Figure 2 are instantiated. These include the following:

- 16-bit Register X
- 16-bit Register B
- 38-bit Register M
- 38-bit Register Y
- 14-bit Upcounter N
- 6-bit Upcounter I
- CMEM
- IMEM
- ALU

Besides just the digital blocks mentioned above, the two multiplexers that are at the inputs of the ALU and the other two multiplexers that are at the output of the ALU are also implemented.

Furthermore, the variables $z1$ and $z2$ are assigned a logic high only when the upcounters N and I, respectively, have reached their limit count. In the end, the overall output of the FIR filter, $dout$, is assigned to the truncated 16-bit version of the Y register.

V. Simulation

After writing the Verilog code based on the design diagrams and the pseudocode algorithm, in order to verify the output of the FIR filter core, a testbench was constructed. In the next few sections, the explanation of the testbench as well as the simulation outputs will be presented and discussed.

A. Testbench

In order to thoroughly verify the design, the testbench contains some features that will test every aspect of the FIR core. At first, the testbench instantiates the RTL design and assigns its variables and signals to the input and the output provided in the FIR core.

As mentioned in Part II of this report, the random coefficients generated in the MATLAB behavioral code will be used as inputs to the CMEM memory of the FIR core. In the first 64 iterations, these random coefficients are stored in the CMEM memory by using the enable and write signals provided by the memory compiler.

Moreover, the random inputs generated in MATLAB will also be used as inputs to the IMEM memory of the FIR core. In the next 16,384 iterations, the random signal values are stored in the IMEM memory by using the enable and write signals provided by the memory compiler and the RTL top level code written for the IMEM.

After all of the input data is stored into the memory of the FIR core, the calculations are ready to be performed. To progress into the next state in the FSM algorithm, the start variable, s, is set to logic high. As a result, the testbench now produces the output signal and stores it in a $qsim.out$ file.

At the same time that the output is being produced, the testbench also reads the output text file generated by the MATLAB script mentioned in Part II of this report. As the output is being stored in a file, it is also compared with the results from the MATLAB behavioral code in order to verify the output of the FIR core. In this way, this Signature Analyzer validates the authenticity of the design.

B. Simulation Outputs

The simulation outputs presented in this report are divided into four different sections. In the first section, the QuestaSim simulation of the RTL design will be examined. In the second section, the DC report will generate the gate-level netlist and show the area and the hold and setup time violations. In the third section, the output of the testbench written for the gate-level netlist will be analyzed. Lastly, in the final section, the Prime-Time report will give more insights on the timing and power analysis of the circuit design.

1) QuestaSim Simulation

Running the testbench will produce a QuestaSim waveform that shows the states of the different inputs and outputs throughout the positive and negative edges of the clock signal. As explained in the above section, at first the coefficients data is loaded into the CMEM and the input data is loaded into the IMEM. These operations can be seen in the two figures shown below.

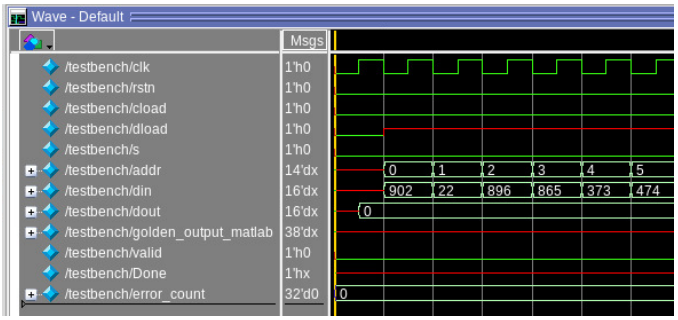


Fig 4. Coefficients Data being loaded into the CMEM

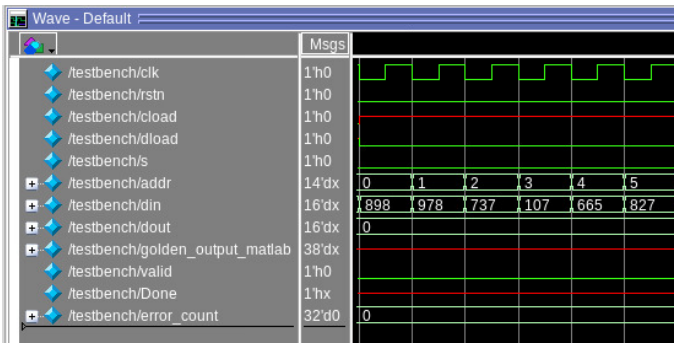


Fig 5. Input Data being loaded into the IMEM

Since the data is loaded successfully in both memories of the FIR core, now the output calculations will be performed. As seen in Figure 6 below, the output, $dout$, seems to produce unknown values when the start variable, s, is being set to logic high. The reading of the golden output from MATLAB as well as the error counting algorithm is done successfully.

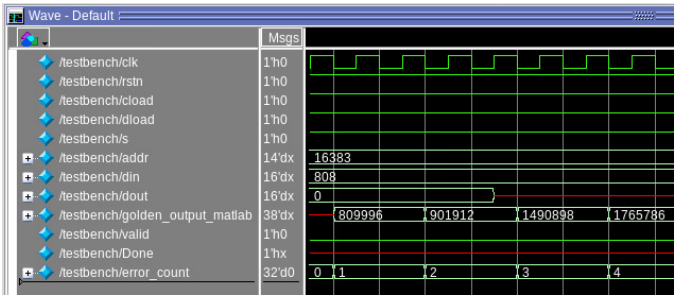


Fig 6. Output Results of the FIR Filter

The report metrics will be based on these results. In future implementations of the FIR core, there will be further efforts to reduce the error.

2) Design Compiler (DC)

Now that the RTL code has been tested, the design will be run through the Design Compiler to produce the gate-level netlist. For the memories, before generating the gate-level netlist, the .lib file from the Memory Compiler was converted into the .db file by using one of the pre-made libraries. The Verilog code generated from the Memory Compiler was treated as a black-box and only the .db file was used as part of the synthesis. The figure below shows a portion of the gate-level netlist created by the Design Compiler.

```
Open   fir.nl.v   ~/Desktop/Project/dc/fir   Save
module fir ( clk, rstn, din, addr, dload, cload, dout, valid, Done, s );
  input [15:0] din;
  input [13:0] addr;
  output [15:0] dout;
  input clk, rstn, dload, cload, s;
  output valid, Done;
  wire   n1000, n1001, n1002, n1003, n1004, n1005, n1006, n1007, n1008, n1009,
         n1010, n1011, n1012, n1013, n1014, n1015, N45, alu_0_N98, alu_0_N97,
         alu_0_N96, alu_0_N95, alu_0_N94, alu_0_N93, alu_0_N92, alu_0_N91,
         alu_0_N90, alu_0_N89, alu_0_N88, alu_0_N87, alu_0_N86, alu_0_N85,
         alu_0_N84, alu_0_N83, alu_0_N82, alu_0_N81, n48, n49, n50, n51, n52,
         n53, n54, n55, n56, n57, n58, n59, n60, n61, n62, n63, n64, n65, n85,
         n86, n87, n88, n89, n90, n91, n92, n93, n94, n95, n96, n97, n98, n99,
         n100, n101, n102, n104, n105, n106, n107, n108, n109, n110, n111,
         n112, n113, n114, n115, n116, n117, n118, n119, n120, n121, n122,
         n123, n124, n125, n126, n127, n128, n129, n130, n131, n132, n133,
         n134, n135, mult_x_19_n1512, mult_x_19_n1511, mult_x_19_n1510,
         mult_x_19_n1509, mult_x_19_n1508, mult_x_19_n1507, mult_x_19_n1506,
         mult_x_19_n1505, mult_x_19_n1504, mult_x_19_n1477, mult_x_19_n1476,
         mult_x_19_n1475, mult_x_19_n1474, mult_x_19_n1473, mult_x_19_n1472,
         mult_x_19_n1471, mult_x_19_n1470, mult_x_19_n1469, mult_x_19_n1445,
         mult_x_19_n1444, mult_x_19_n1443, mult_x_19_n1442, mult_x_19_n1441,
         mult_x_19_n1439, mult_x_19_n1438, mult_x_19_n1437, mult_x_19_n1410,
         mult_x_19_n1409, mult_x_19_n1408, mult_x_19_n1384, mult_x_19_n1383,
         mult_x_19_n1382, mult_x_19_n785, mult_x_19_n782, mult_x_19_n780,
         mult_x_19_n779, mult_x_19_n778, mult_x_19_n777, mult_x_19_n775,
         mult_x_19_n774, mult_x_19_n773, mult_x_19_n772, mult_x_19_n770,
         mult_x_19_n769, mult_x_19_n768, mult_x_19_n765, mult_x_19_n763,
         mult_x_19_n762, mult_x_19_n761, mult_x_19_n758, mult_x_19_n757,
         mult_x_19_n756, mult_x_19_n755, mult_x_19_n754, mult_x_19_n752,
Verilog   Tab Width: 8   Ln 2189, Col 1   INS
```

Fig 7. Gate-Level Netlist of the FIR Core

In addition to the gate-level netlist, the Design Compiler also generates a report file where timing, area and power information can be found. As seen in Figure 8 below, the Design Compiler produced no timing violations since the output mentions that the timing constraints have been MET.

```
Open   fir.dc.rpt   ~/Desktop/Project/dc/fir   Save
U706/Y (NOR2XLTS)                        0.26       3.37 f
U955/Y (CLKBUFX2TS)                      0.25       3.62 f
U956/Y (CLKBUFX2TS)                      0.26       3.88 f
U1016/Y (AOI222XLTS)                     0.38       4.25 r
U643/Y (OAI21XLTS)                       0.22       4.47 f
U411/Y (XOR2XLTS)                        0.34       4.82 r
U1024/Y (NAND2X1TS)                      0.29       5.11 f
U789/Y (OAI21XLTS)                       0.38       5.49 r
U1029/Y (AOI21X1TS)                      0.25       5.73 f
U427/Y (OAI21XLTS)                       0.39       6.12 r
U1033/Y (AOI21X1TS)                      0.27       6.39 f
U424/Y (OAI21XLTS)                       0.40       6.79 r
U1038/Y (AOI21X1TS)                      0.26       7.05 f
U323/Y (OAI21XLTS)                       0.41       7.47 r
U1042/Y (AOI21X1TS)                      0.32       7.79 f
U413/Y (OAI21XLTS)                       0.43       8.21 r
U1079/Y (INVX2TS)                        0.21       8.43 f
U577/Y (OAI21XLTS)                       0.27       8.70 r
U1085/Y (XNOR2X1TS)                      0.29       8.99 f
U576/Y (AO22XLTS)                        0.41       9.41 f
alu_0_ANS_reg_16_/D (DFFQX1TS)           0.00       9.41 f
data arrival time                                   9.41

clock clk (rise edge)                   10.00      10.00
clock network delay (ideal)              0.00      10.00
alu_0_ANS_reg_16_/CK (DFFQX1TS)          0.00      10.00 r
library setup time                      -0.33       9.67
data required time                                  9.67
-----------------------------------------------------------
data required time                                  9.67
data arrival time                                  -9.41
-----------------------------------------------------------
slack (MET)                                         0.26
Plain Text   Tab Width: 8   Ln 172, Col 22   INS
```

Fig 8. Timing Report from Design Compiler

The Design Compiler also produces an area report which can be seen in Figure 9 below. The area report gives more information on the area breakdown and the number of ports, nets, and combinational and sequential cells which can be seen highlighted below.

```
Open   fir.dc.rpt   ~/Desktop/Project/dc/fir   Save
****************************************
Report : area
Design : fir
Version: O-2018.06-SP5-1
Date   : Mon Dec 14 23:18:19 2020
****************************************

Library(s) Used:

    scx3_cmos8rf_lpvt_tt_1p2v_25c (File: /courses/ee6321/share/
ibm13rflpvt/synopsys/scx3_cmos8rf_lpvt_tt_1p2v_25c.db)

Number of ports:                         1247
Number of nets:                          4736
Number of cells:                         2491
Number of combinational cells:           2326
Number of sequential cells:               154
Number of macros/black boxes:               0
Number of buf/inv:                        673
Number of references:                      35

Combinational area:              18730.080454
Buf/Inv area:                     3631.680137
Noncombinational area:            2190.239960
Macro/Black Box area:                0.000000
Net Interconnect area:      undefined  (No wire load specified)

Total cell area:                 20920.320414
Total area:                 undefined

Information: This design contains black box (unknown) components.
(RPT-8)
Plain Text   Tab Width: 8   Ln 154, Col 34   INS
```

Fig 9. Area Report from the Design Compiler

3) Design Compiler Testbench

The gate-level netlist created will now be tested using a modified testbench. The modified testbench contains function calls that store information about the cycle-accurate switching activities. This information will be stored in the $fir.vcd$ file that the testbench will generate.

Besides a change in the testbench code, the $runsim.do$ file will be altered to point to the gate-level netlist and also capture the $fir.syn.sdf$ file generated from the Design Compiler.

The QuestaSim transcript shown in Figure 10, highlights the successful simulation of the gate-level netlist along with the completed SDF back annotation.

```
Open   transcript   ~/Desktop/Project/qsim_dc/fir   Save
without delayed copies of data or reference signals.
#    Time: 0 ps  Iteration: 0  Instance: /testbench File: /
courses/ee6321/share/ibm13rflpvt/verilog/ibm13rflpvt.v Line:
24949
# ** Note: (vsim-3587) SDF Backannotation Successfully
Completed.
#    Time: 0 ps  Iteration: 0  Instance: /testbench File:
test_fir.v
# Entering data into CMEM...
# data_in_cmem:   902
#
# data_in_cmem:    22
#
# data_in_cmem:   896
#
Plain Text   Tab Width: 8   Ln 1196, Col 45   INS
```

Fig 10. SDF Back Annotation

In addition to the transcript, the modified testbench also generates a QuestaSim waveform where the output timing can be analyzed. As seen in Figure 11 below, the results of the gate-level netlist match the results of the original RTL code. However, one can see a delay from the gate-level netlist output that starts at the first clock cycle and continues until the end. This behavior is expected since the modified testbench also takes into account the delay annotation information stored in the $fir.syn.sdf$ file (generated from Design Compiler).

Besides the output delay, glitches in the output can also be seen in the figure below. These glitches might happen because the input of the next cycle might arrive faster than the computation of the result from the previous cycle.

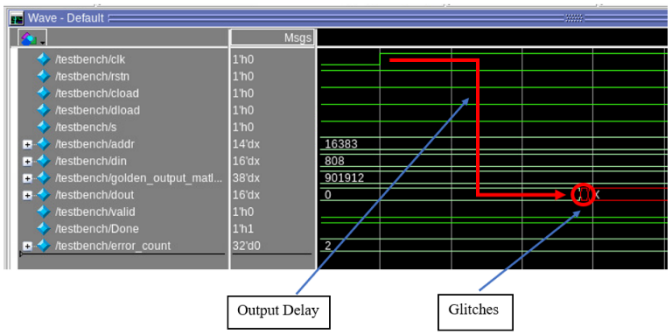


Fig 11. QuestaSim Waveform showing the Output Delay and Glitches

4) Prime Time Report

Running the Prime-Time script will produce a report file which highlights the longest and shortest path delay in the design. As seen in Figure 12 below, the shortest path delay of the circuit is highlighted.

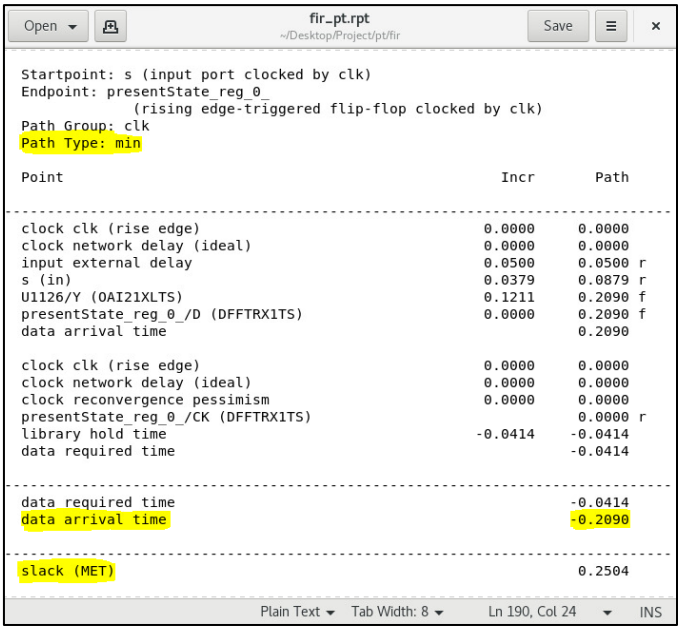


Fig 12. Shortest Path Delay

On the other hand, the maximum path delay can be seen highlighted in Figure 13 below. This number signifies the critical path delay of the circuit.

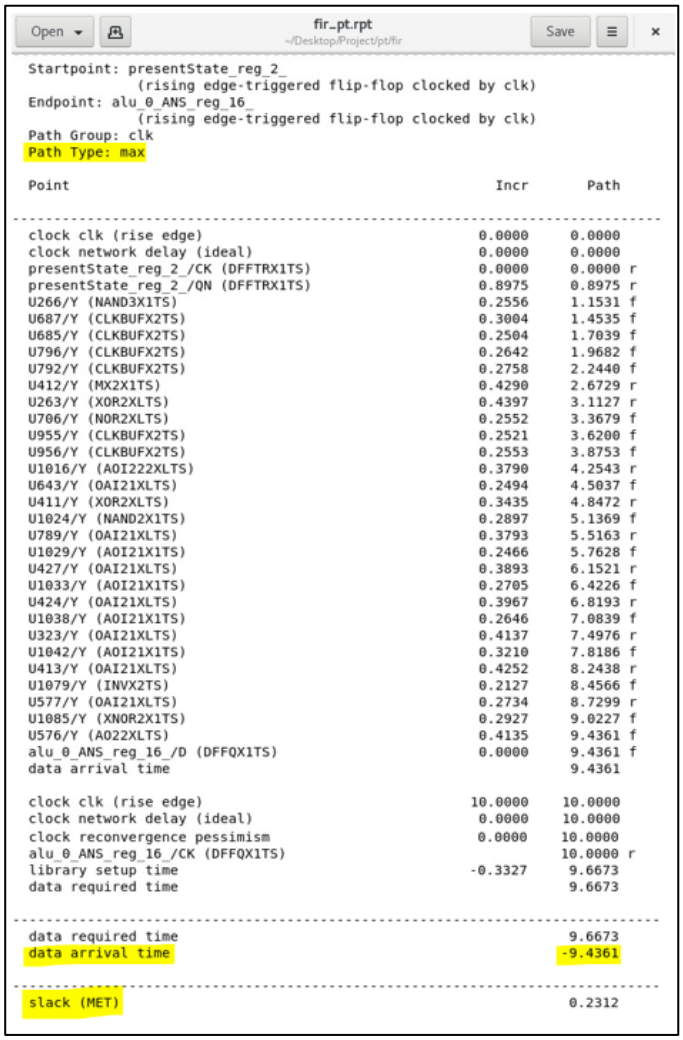
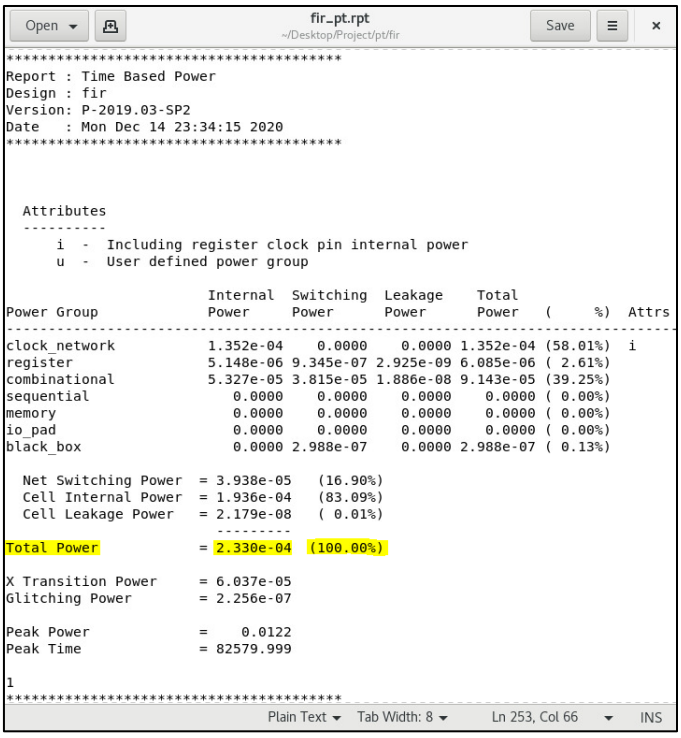


Fig 13. Critical Path Delay

Besides just the shortest and critical path delay, the Prime-Time report also shows the power analysis of the circuit. As seen in the report in Figure 14, the Total Power is the combination of the Net Switching Power, Cell Internal Power and Cell Leakage Power. The black-box power is also included in the calculations.

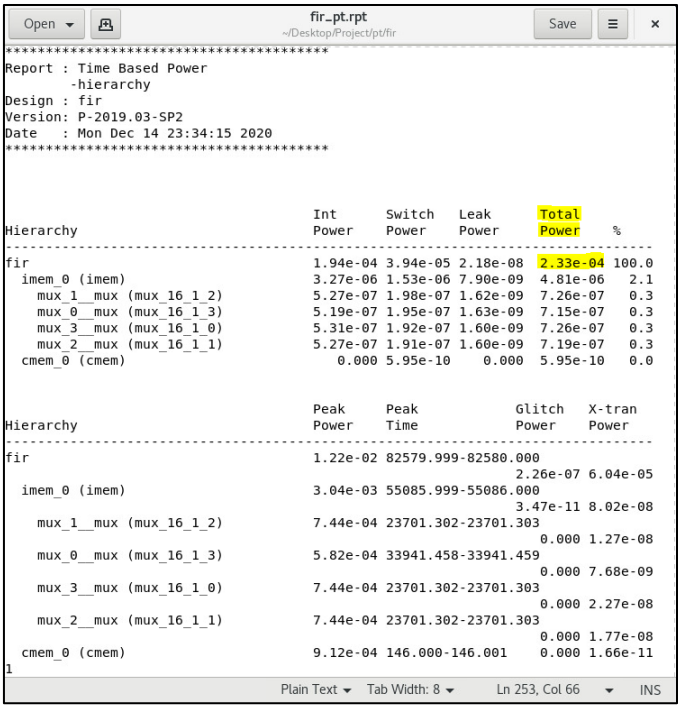
In addition, besides the normal Time-Based Power analysis, the report also shows its Hierarchical version, which can be found in Figure 15.

All of the values highlighted in the Simulation Outputs portion of the report will be analyzed further in the Metrics section. They will also be compared with the requirements set at the beginning.



```
fir_pt.rpt
~/Desktop/Project/pt/fir

****************************************
Report : Time Based Power
Design : fir
Version: P-2019.03-SP2
Date   : Mon Dec 14 23:34:15 2020
****************************************


  Attributes
  ----------
      i  -  Including register clock pin internal power
      u  -  User defined power group

                        Internal  Switching  Leakage    Total
Power Group             Power     Power      Power      Power   (     %)  Attrs
--------------------------------------------------------------------------------
clock_network           1.352e-04    0.0000    0.0000 1.352e-04 (58.01%)  i
register                5.148e-06 9.345e-07 2.925e-09 6.085e-06 ( 2.61%)
combinational           5.327e-05 3.815e-05 1.886e-08 9.143e-05 (39.25%)
sequential                 0.0000    0.0000    0.0000    0.0000 ( 0.00%)
memory                     0.0000    0.0000    0.0000    0.0000 ( 0.00%)
io_pad                     0.0000    0.0000    0.0000    0.0000 ( 0.00%)
black_box                  0.0000 2.988e-07    0.0000 2.988e-07 ( 0.13%)

  Net Switching Power  = 3.938e-05   (16.90%)
  Cell Internal Power  = 1.936e-04   (83.09%)
  Cell Leakage Power   = 2.179e-08   ( 0.01%)
                         ---------
Total Power             = 2.330e-04  (100.00%)

X Transition Power      = 6.037e-05
Glitching Power         = 2.256e-07

Peak Power              =    0.0122
Peak Time               = 82579.999

1
****************************************
Plain Text  Tab Width: 8  Ln 253, Col 66  INS
```

Fig 14. Time Based Power Report



```
fir_pt.rpt
~/Desktop/Project/pt/fir

****************************************
Report : Time Based Power
        -hierarchy
Design : fir
Version: P-2019.03-SP2
Date   : Mon Dec 14 23:34:15 2020
****************************************


                                  Int      Switch   Leak     Total
Hierarchy                         Power    Power    Power    Power    %
-------------------------------------------------------------------------------
fir                               1.94e-04 3.94e-05 2.18e-08 2.33e-04 100.0
  imem_0 (imem)                   3.27e-06 1.53e-06 7.90e-09  4.81e-06   2.1
    mux_1__mux (mux_16_1_2)       5.27e-07 1.98e-07 1.62e-09  7.26e-07   0.3
    mux_0__mux (mux_16_1_3)       5.19e-07 1.95e-07 1.63e-09  7.15e-07   0.3
    mux_3__mux (mux_16_1_0)       5.31e-07 1.92e-07 1.60e-09  7.26e-07   0.3
    mux_2__mux (mux_16_1_1)       5.27e-07 1.91e-07 1.60e-09  7.19e-07   0.3
  cmem_0 (cmem)                      0.000 5.95e-10    0.000  5.95e-10   0.0


                                  Peak     Peak            Glitch   X-tran
Hierarchy                         Power    Time            Power    Power
-------------------------------------------------------------------------------
fir                               1.22e-02 82579.999-82580.000
                                                           2.26e-07 6.04e-05
  imem_0 (imem)                   3.04e-03 55085.999-55086.000
                                                           3.47e-11 8.02e-08
    mux_1__mux (mux_16_1_2)       7.44e-04 23701.302-23701.303
                                                              0.000 1.27e-08
    mux_0__mux (mux_16_1_3)       5.82e-04 33941.458-33941.459
                                                              0.000 7.68e-09
    mux_3__mux (mux_16_1_0)       7.44e-04 23701.302-23701.303
                                                              0.000 2.27e-08
    mux_2__mux (mux_16_1_1)       7.44e-04 23701.302-23701.303
                                                              0.000 1.77e-08
  cmem_0 (cmem)                   9.12e-04 146.000-146.001    0.000 1.66e-11
1
Plain Text  Tab Width: 8  Ln 253, Col 66  INS
```

Fig 15. Hierarchical Time-Based Power Report

VI. Output Analysis (Metrics)

For this FIR filter project, we had to follow given metric constraints in order to deem our project a success. To begin, the core must load all inputs and coefficients from their respective memory registers. It must also use only one adder and one multiplier. The throughput must be $10\,kS/s$. A signature analyzer is needed to verify the data coming out of the core. The extended project metrics that must be met are listed below.

- **Input:** Randomly generated 10k 16-bit real-valued numbers
- **Throughput:** Prime-Time report with design compiler generated sdf annotation
- **Maximum Clock Frequency:** Prime-Time report with design compiler generated. sdf annotation
- **Power Analysis:** Prime-Time report
- **Energy Efficiency:** Prime-Time report with the design compiler generated .sdf and QuestaSim generated .vcd annotations
- **Area:** Design Compiler generated report
- **Accuracy:** RMSE against MATLAB results in 32-bit floating point numbers. Inputs are 16-bit and generated by the random function of MATLAB.

A. *Area*

An area analysis was conducted to show that the area metrics are matched. The total cell area is the physical space occupied by the core and would estimate how big the chip would need to be in order to be manufactured. The report from Figure 9 is listed below:

$$Combinational\ Area = 18{,}730.08\ mm^2$$
$$Buffer\ and\ Inverter\ Area = 3{,}631.68\ mm^2$$
$$Noncombinational\ Area = 2{,}190.24\ mm^2$$

By combining all these values, the total cell area would be:

$$Total\ Cell\ Area = 20{,}920.32\ mm^2$$

B. *Timing*

A timing analysis was conducted to show that the timing metrics were matched. In it, we display the critical path delay and the minimum path delay. Based on Figure 12 and Figure 13 the delays are as follows:

$$Minimum\ Path\ Delay = 0.2090\ ns$$
$$Critical\ Path\ Delay = 9.4361\ ns$$

C. Clock Frequencies

The clock frequency is what determines how fast our circuit operates and is an important parameter when considering the FIR core. We use this frequency to calculate the throughput of our design. The throughput calculation is handled in the next section. Based on the testbench, the half clock period is the following:

$$T_{half} = 5\ ns$$

As a result, the clock period can be calculated as:

$$T = 2\ \times\ T_{half} = 10\ ns$$

We can calculate the frequency of our clock by using the following equation and by plugging in our period:

$$frequency = \frac{1}{T} = \frac{1}{10\ ns} = \frac{1}{10\ \times 10^{-9}} = \ 100\ MHz$$

The maximum frequency is calculated as follows:

$$F_{max} = \frac{1}{T_{min}} = \frac{1}{critical\ path\ delay + \ t_{su} + \ t_{cq}}$$

$$= \frac{1}{9.4361\ ns} = 105.98\ MHz$$

D. Throughput

A throughput analysis was conducted to show that the throughput metrics were matched. Based on the maximum clock frequency from Part C, the throughput can be calculated using the formula below:

$$Throughput = \ F_{max} \times 16\ bits = \ 1.696\ Gbits/s$$

E. Accuracy

An accuracy analysis was conducted to show that the accuracy metrics were matched. An equation is given in the project parameters for how we can calculate the accuracy. The Normalized Root Mean Squared Error (NRMSE) equation is as follows:

$$NRMSE = \frac{RMSE}{y_{max} - y_{min}}$$

Where the equation for RMSE is as follows:

$$\sqrt{\frac{\sum_{i=0}^{n}(y_i - y_i^*)}{n}}$$

Where y_i is your full precision MATLAB result and y_i^* is the quantized result from your RTL simulation. Since our outputs were unable to show in our waveform, we are not able to calculate the accuracy. However, we expect a very high accuracy compared to the golden outputs produced by the MATLAB.

F. Power

The Total Power is the combination of the Net Switching Power, Cell Internal Power and Cell Leakage Power. The black-box power is also included in the calculations. The values for these metrics are as follows:

$$Net\ Switching\ Power = 3.938 \times 10^{-5}\ W$$

$$Cell\ Internal\ Power = \ 1.936 \times 10^{-4}\ W$$

$$Cell\ Leakage\ Power = \ 2.179 \times 10^{-8}\ W$$

By combining all of these values, shown also in Figure 14, the Total Power is the following:

$$Total\ Power = \ 2.330 \times 10^{-4} = 0.233 \times 10^{-3}$$

$$Total\ Power = \ 0.233\ mW$$

G. Energy Efficiency

An energy efficiency analysis was conducted to show that the energy efficiency metrics were matched. The calculation for the energy can be seen in the formula below:

$$Energy = Power\ Consumed \times Critical\ Path\ Delay$$

$$= \ 0.233 \times 10^{-3} \times 9.4361 \times 10^{-9}$$

$$Energy = 2.199\ pJ/S$$

VII. Conclusion

In conclusion, the FIR Filter Core was implemented in MATLAB. The golden output generated from the already written filter function matched with the output generated by the behavioral code. The input file and the coefficients file were created as a result of the MATLAB code, as well as the output. In addition, the pseudocode for the FIR Core was written with the help of the MATLAB algorithm. As a result of the pseudocode, the ASM chart, the Datapath Block Diagram, and the Control ASM were also created. The FIR filter core and its testbench was implemented in Verilog with the waveform generation demonstrating the functionality of the core.

We were successful in reaching most of our metric project parameters. We were able to successfully generate a gate-level netlist from the Design Compiler. The .sdf and .vcd files showed the delay annotation and the cycle-accurate switching activities of the circuit design, respectively. In the end, Prime Time report gave a more complete look at the critical and shortest path in the circuit design as well as a more detailed power analysis.