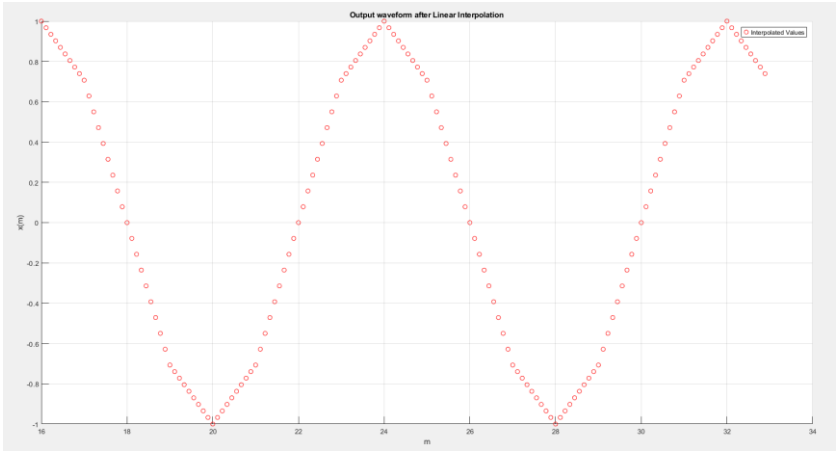
DSP in VLSI

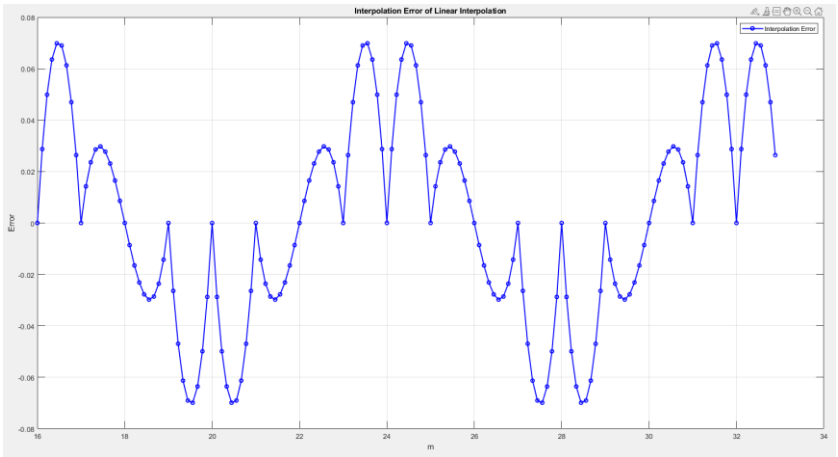
HW3

電子所 ICS 組, R13943015, 張根齊

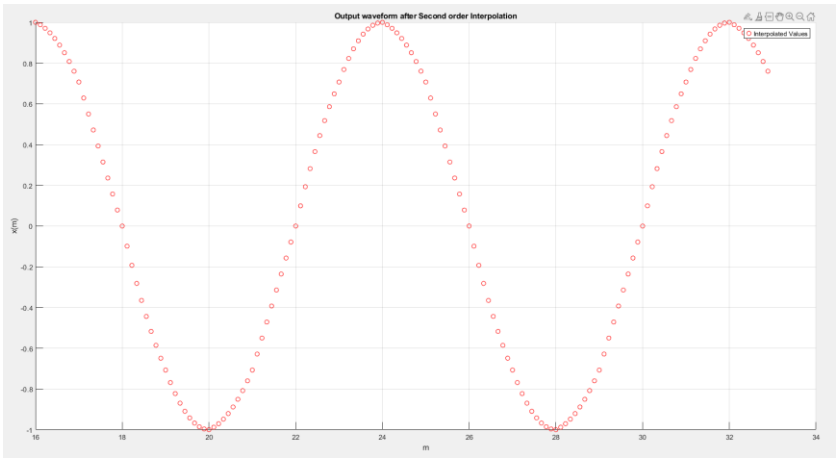
1. (Step 1) Show the output waveform after interpolation using linear interpolator, second-order polynomial interpolator, and piecewise parabolic interpolator to interpolate the sampled waveform in the region of 16 ≤ m ≤ 32 with μ = 0, 1/9 , 2/9 , … 8/9 . In addition, draw the error in the region of 16 ≤ m ≤ 32 with μ = 0, 1/9 , 2/9 , … 8/9 . (30%)

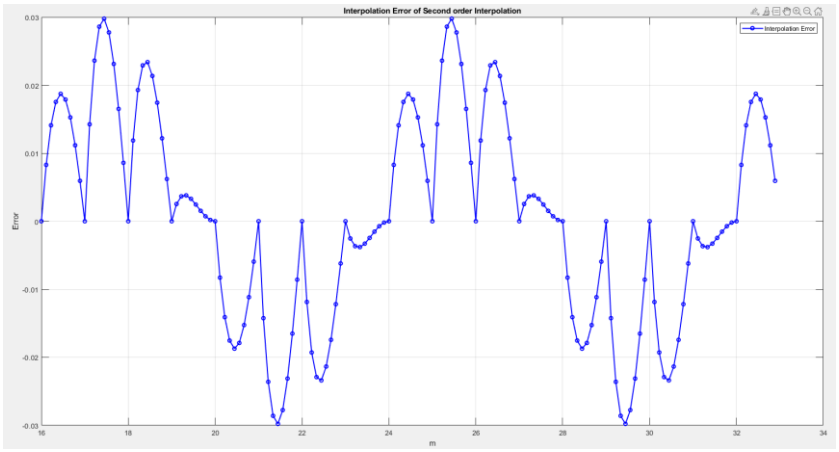
1-1. Linear interpolator output waveform and error



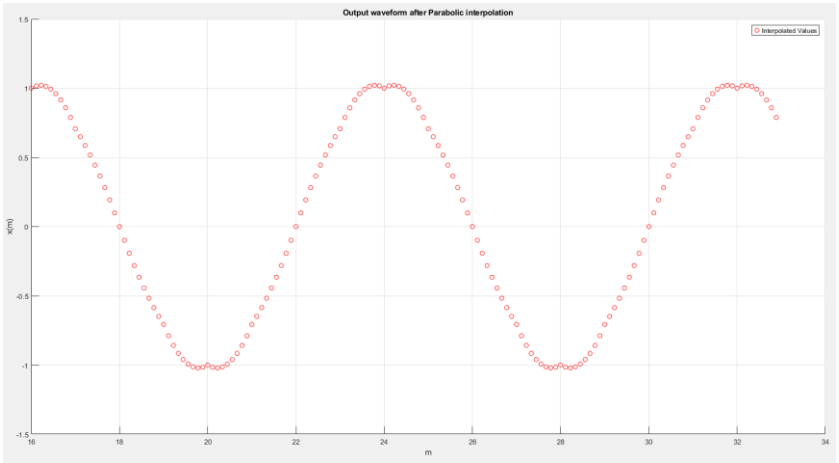


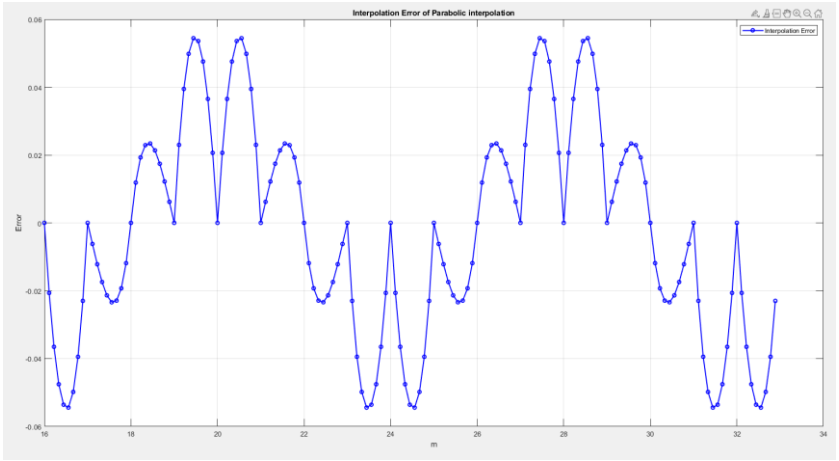
1-2. Second-order polynomial interpolator output waveform and error





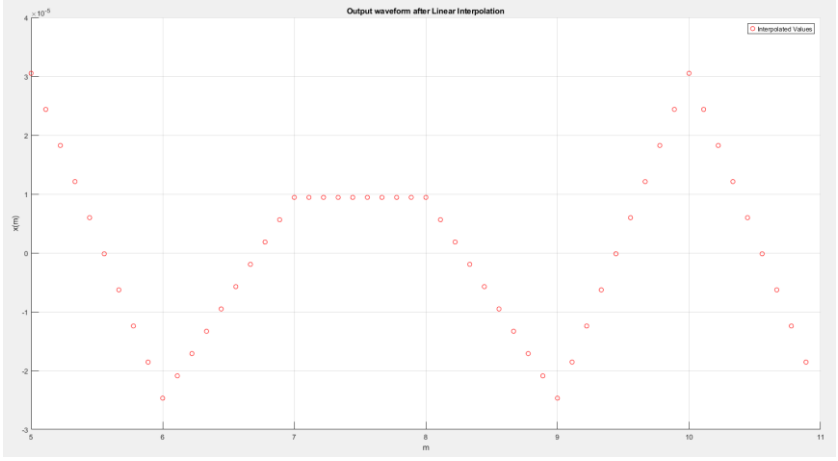
1-3. piecewise parabolic interpolator output waveform and error

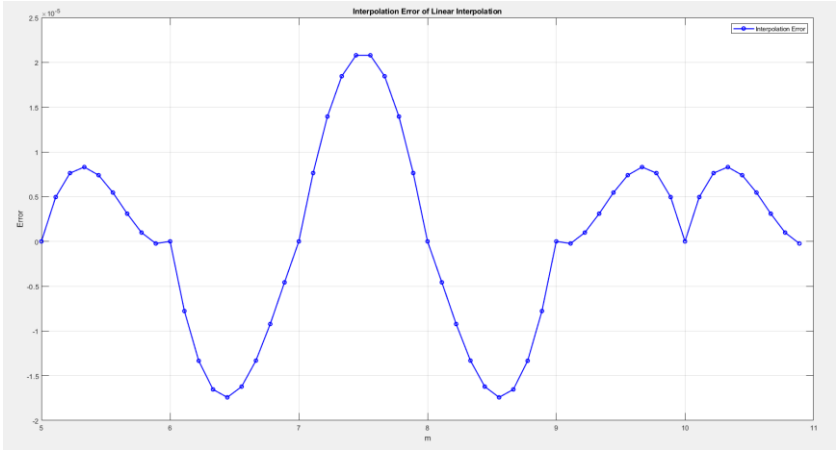




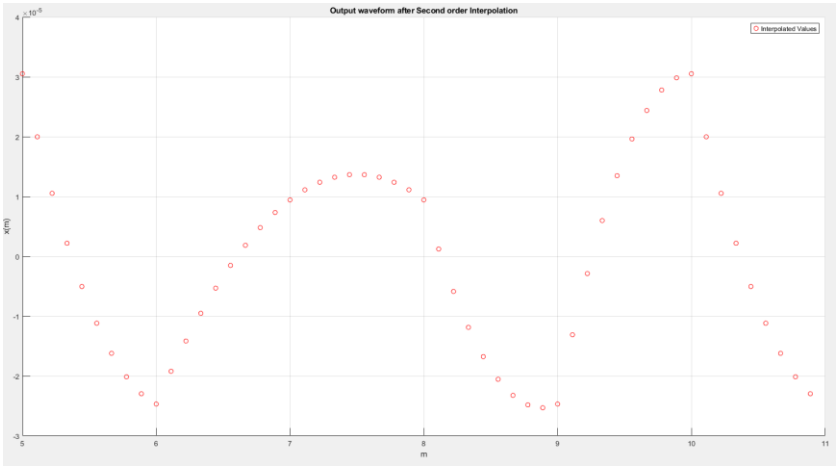
2. (Step 2) Show the output waveform after interpolation using linear interpolator, second-order polynomial interpolator, and piecewise parabolic interpolator to interpolate the sampled waveform in the region of 5 ≤ m ≤ 10 with μ =0, 1/9, 2/9, …8/9. In addition, draw the error in the region of 5 ≤ m ≤ 10 with μ =0, 1/9, 2/9, …8/9.

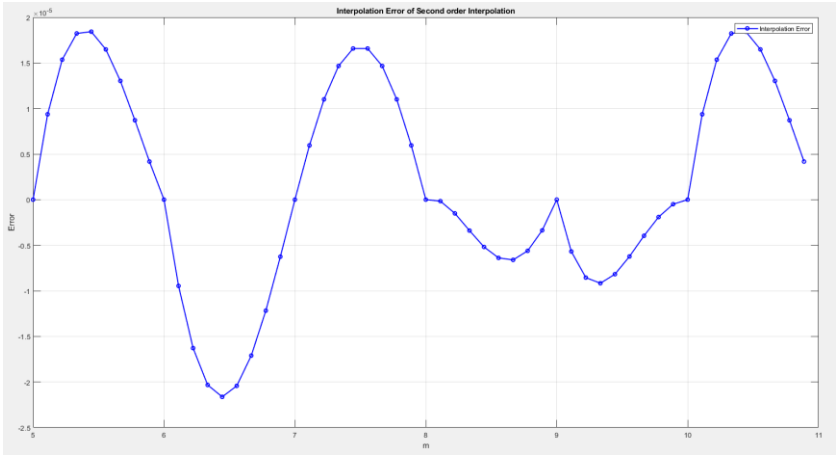
2-1. Linear interpolator output waveform and error



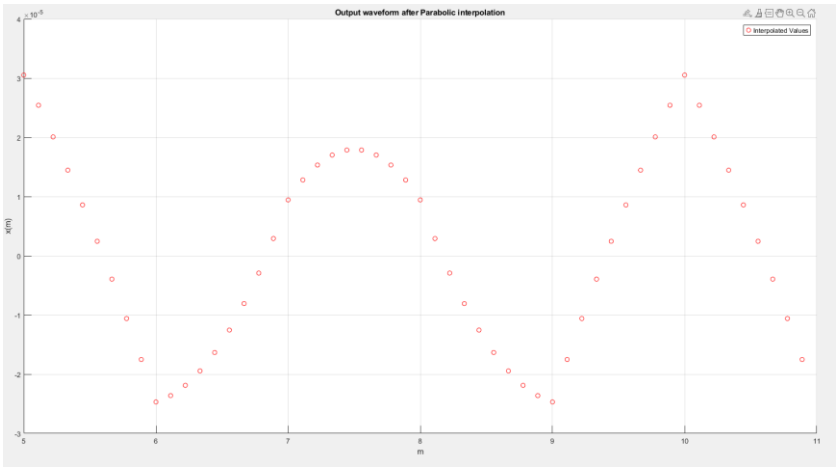


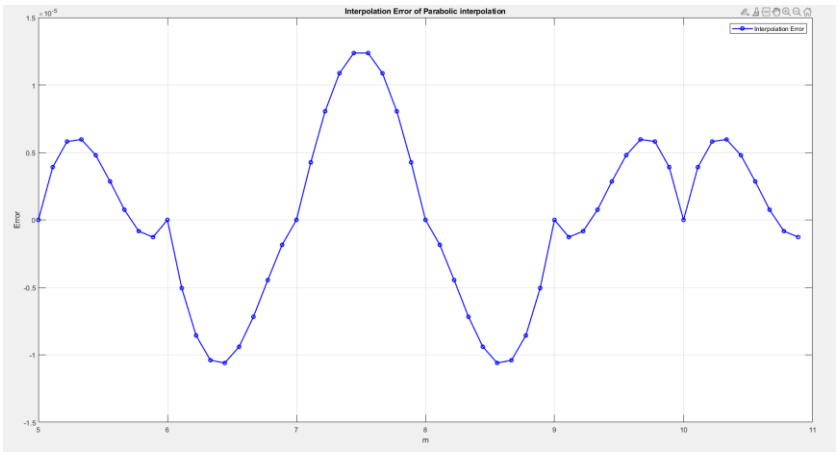
2-2. Second-order polynomial interpolator output waveform and error





2-3. piecewise parabolic interpolator output waveform and error



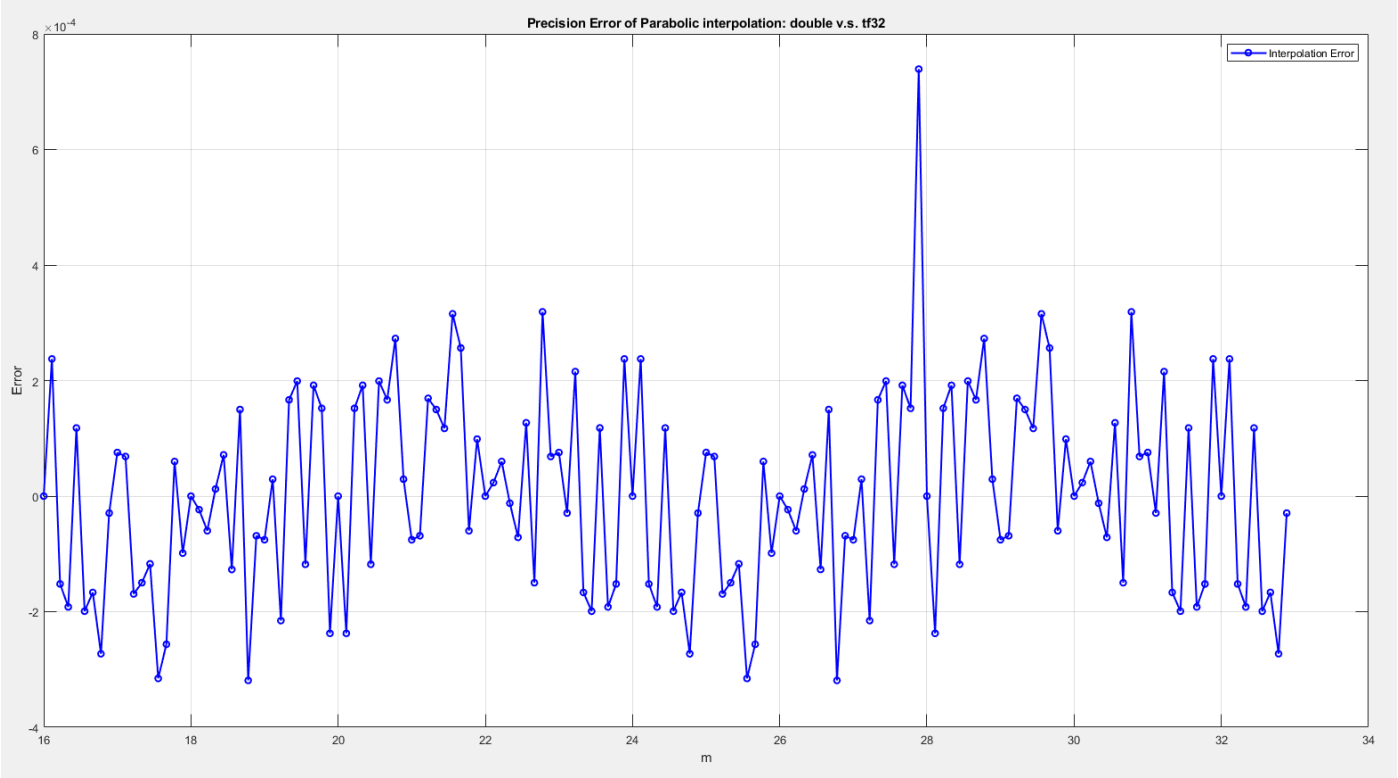


3. (Step 3) Please show the results that are calculated by your bit-true model for the following operands and operators. Express results both in decimal and binary representation. (S, E, F mean the sign bit, exponent field, and fraction field. All are given in binary.)

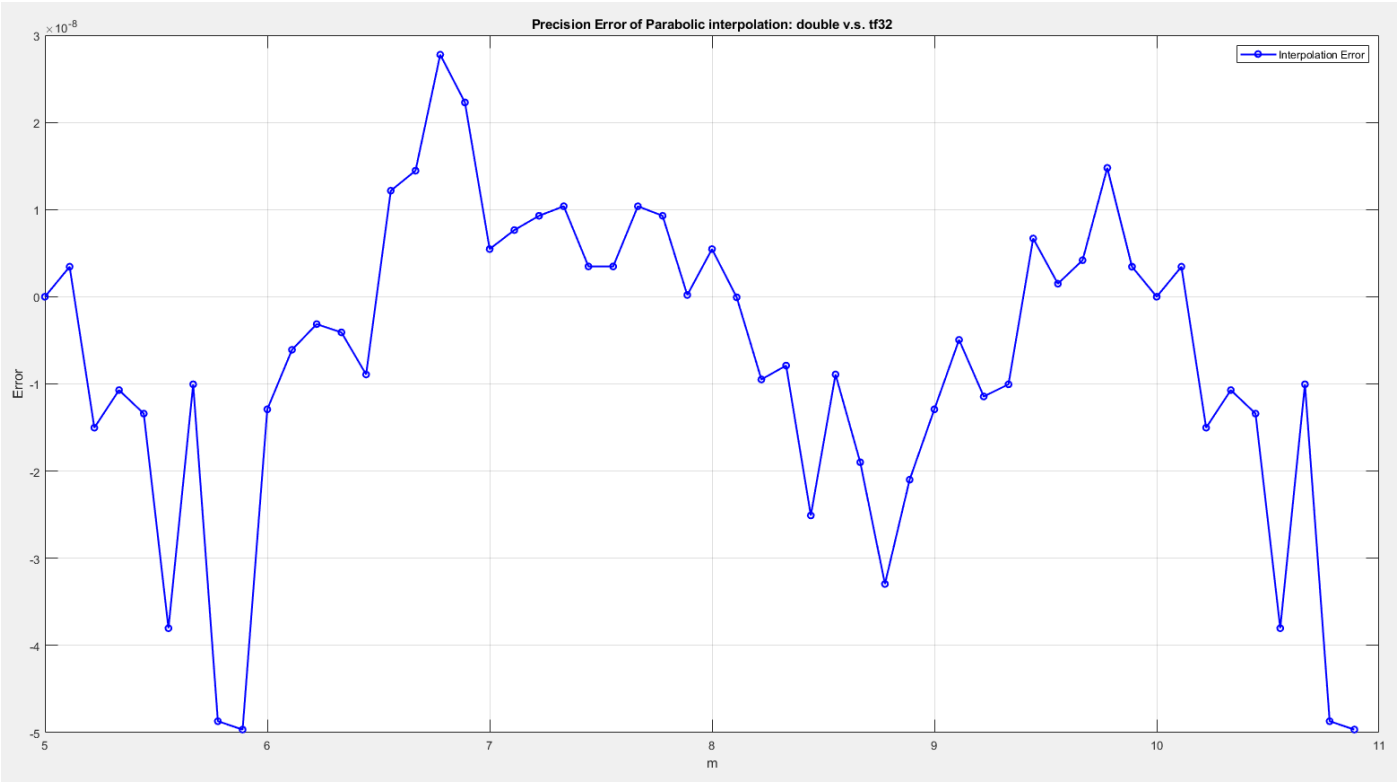
Operation	Operand 1	Operand 2	Result
Addition	S: 1 E: 10000011 F: 0000000111 Decimal: - 16.109375	S: 0 E: 10000011 F:0000000101 Decimal: 16.078125	S: 0 E: 01111010 F:0000000000 Decimal: -0.03125
Addition	S: 1 E: 10000011 F: 0000000111 Decimal: -16.109375	S: 1 E: 10010011 F: 0000000101 Decimal: -1053696.0	S: 1 E: 10010011 F: 0000000101 Decimal: -1053696.0
Multiplication	S: 1 E: 00100011 F: 1100000111 Decimal: -3.547902E-28	S: 0 E: 10000011 F: 1111000101 Decimal: 31.078125	S: 1 E: 00101000 F: 1011010011 Decimal: -1.102512E-26
Multiplication	S: 0 E: 01100011 F: 0011110111 Decimal: 4.623871E-9	S: 0 E: 10000011 F: 0011001111 Decimal: 19.234375	S: 0 E: 01100111 F: 0111111000 Decimal: 8.8941306E-8

4. (Step 4) Show the interpolation differences of the piecewise parabolic interpolator using TF32 and double-precision (default precision of Matlab) arithmetic operations. Draw the figures indicating difference of interpolated results caused by data format for inputs $x1[m]$ in the region of 16 ≤ m ≤ 32 with μ =0, 1/9, 2/9, … 8/9. and $x2[m]$ in the region of 5 ≤ m ≤ 10 with μ = 0, 1/9, 2/9, … 8/9.

4.1 $x1[m]$ error between double precision and TF32:



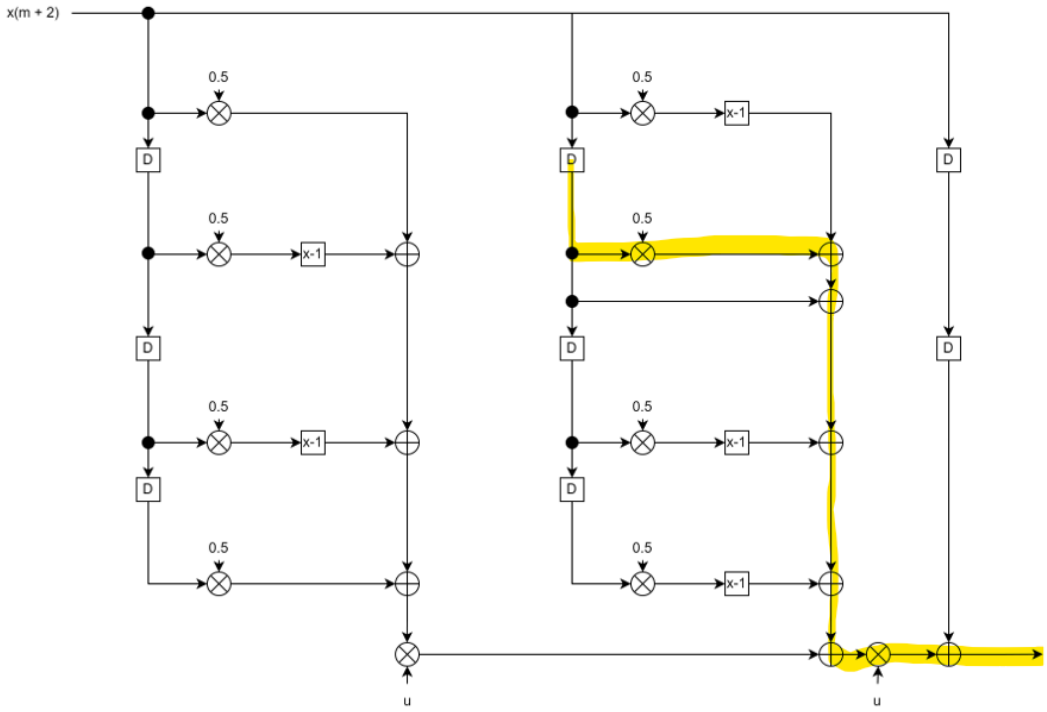
4.2 $x2[m]$ error between double precision and TF32:


Precision Error of Parabolic interpolation: double v.s. tf32
×10^-8
3
2
1
0
-1
-2
-3
-4
-5
Error
5
6
7
8
9
10
11
m
Interpolation Error

5. (Step 5) Draw the complete block diagram of your first version implementation according to Fig. 4. (5%) List two design approaches of hardware sharing (The precision of the datapath must be TF32 and can not be changed.) (10%) Draw the block diagram after your improvement. (5%) Indicate the critical path (5%)

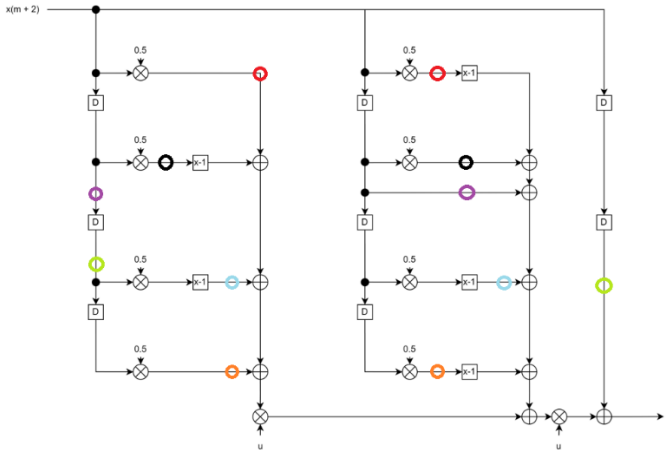
5.1 First version implementation:

Critical path mark as yellow. (tf32 adder delay may longer than tf32 multiplier)

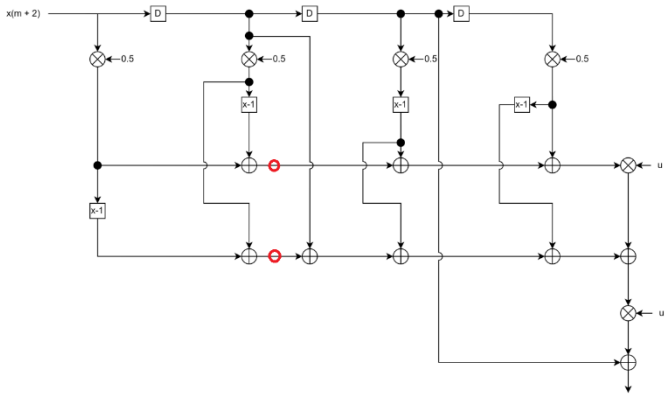


5.2 List two design approaches of hardware sharing

5.2.1 First approach



From the original design (as shown in the figure above), we can see that the nodes marked with the same color are common terms, which can be shared in hardware. By applying hardware sharing to the following parts, we can obtain the result shown in the figure below.

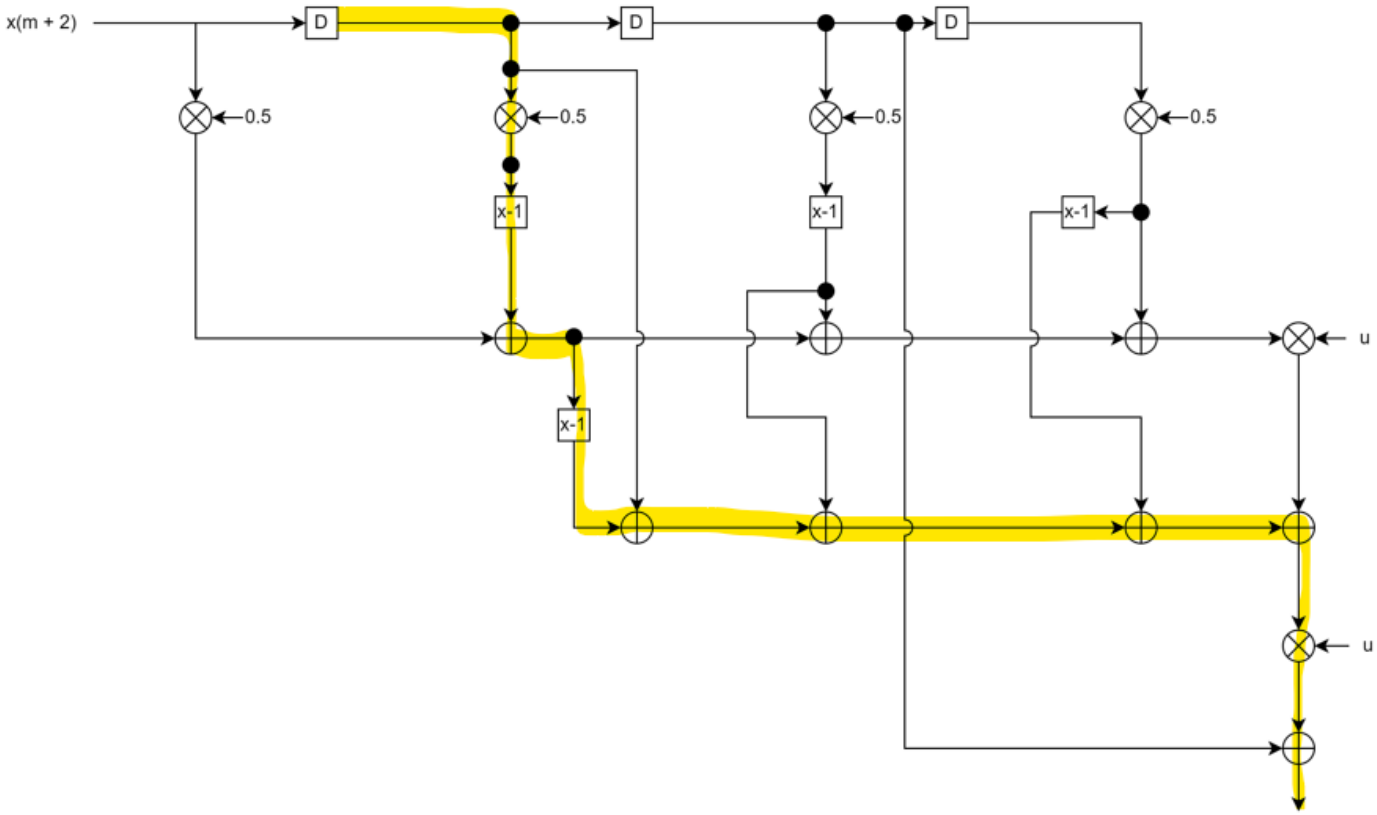


5.2.2 Second approach

From the hardware sharing ver1 design (as shown in the figure above), we can see that the nodes marked with the same color only differ by a minus sign, so this part can also be shared in hardware. The result is shown below (5.3).

5.3 Block diagram after improvement

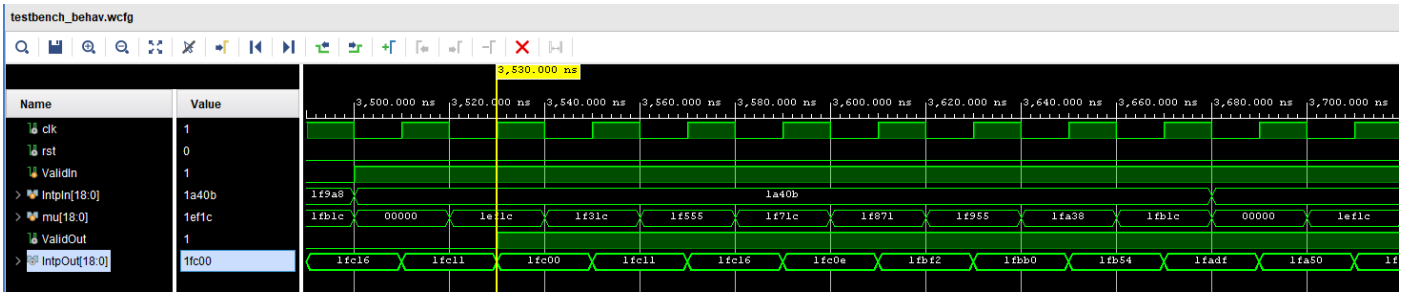
Critical path mark as yellow. (tf32 adder delay may longer than tf32 multiplier)



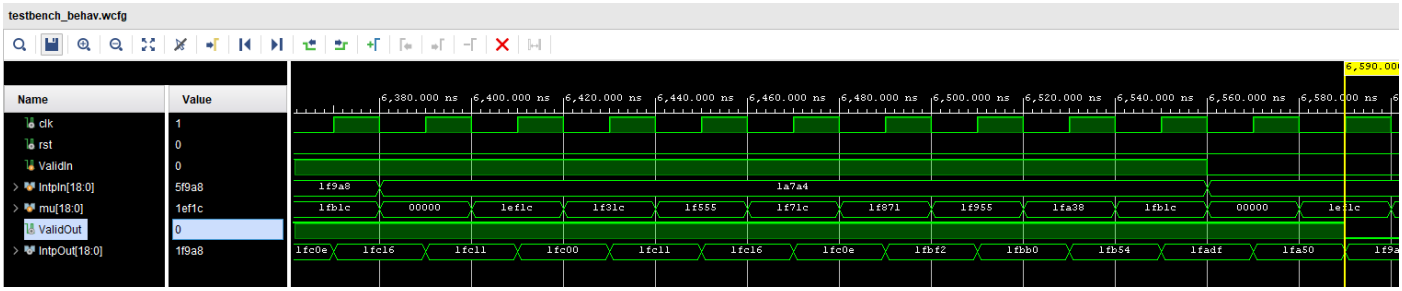
6. (Step 6) Write HDL to implement your piece-wise parabolic interpolator of Farrow structure after reduction. Please note that your input to be interpolated will change every 9 clock cycles and your μ value will change every clock cycle. Show the first 9 (μ = 0, 1/9, 2/9, … 8/9) and the last 9 (μ = 0, 1/9, 2/9, … 8/9) interpolated outputs of $x1[m]$ in the region of $16 \leq m \leq 32$ with μ = 0, 1/9, 2/9, … 8/9 and $x2[m]$ in the region of $5 \leq m \leq 10$ in the timing diagram of behavior and post-synthesis simulations.

6.1 $x1[m]$ behavior simulations:

6.1.1 first 9:

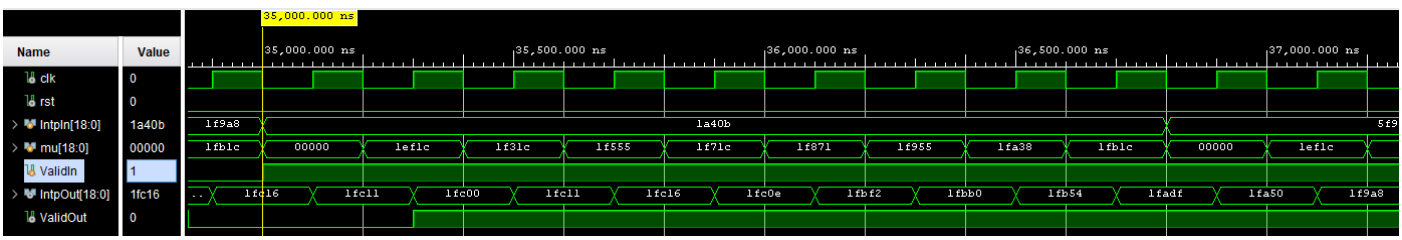


6.1.2 last 9:

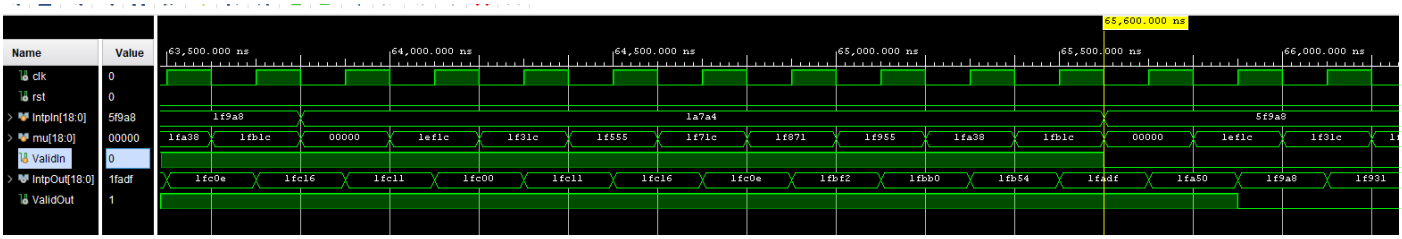


6.2 $x1[m]$ post-synthesis simulations:

6.2.1 first 9:

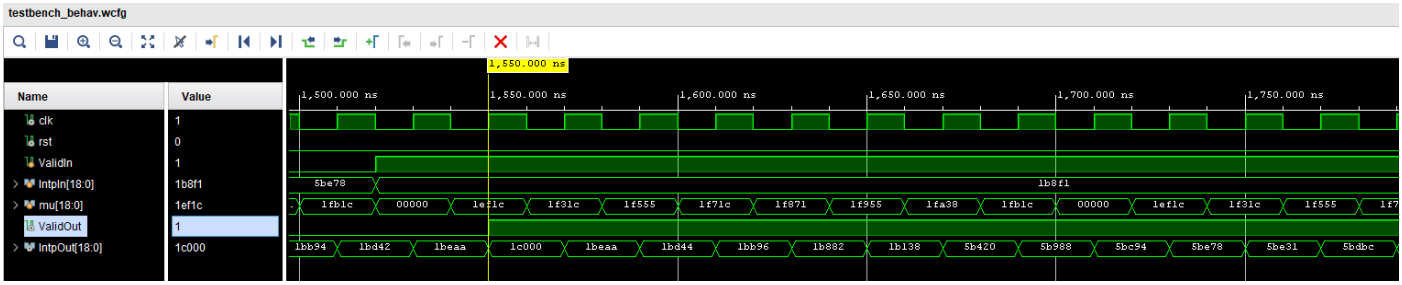


6.2.2 last 9:

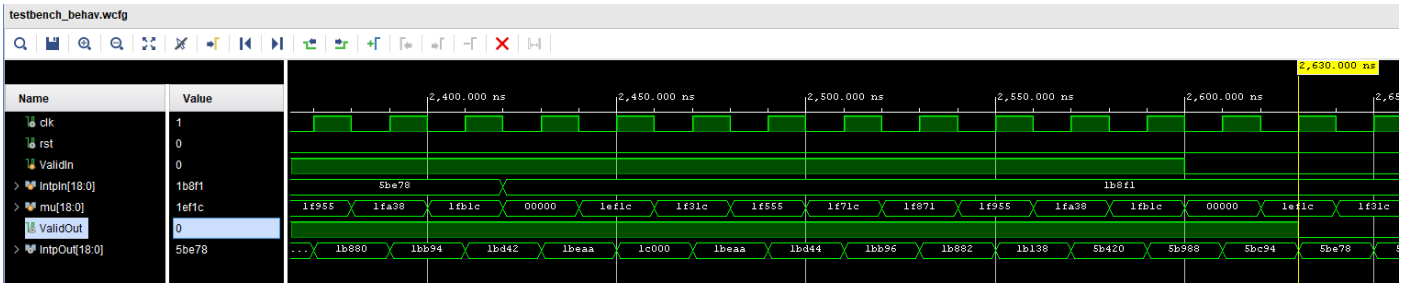


6.3 $x2[m]$ behavior simulations:

6.3.1 first 9:

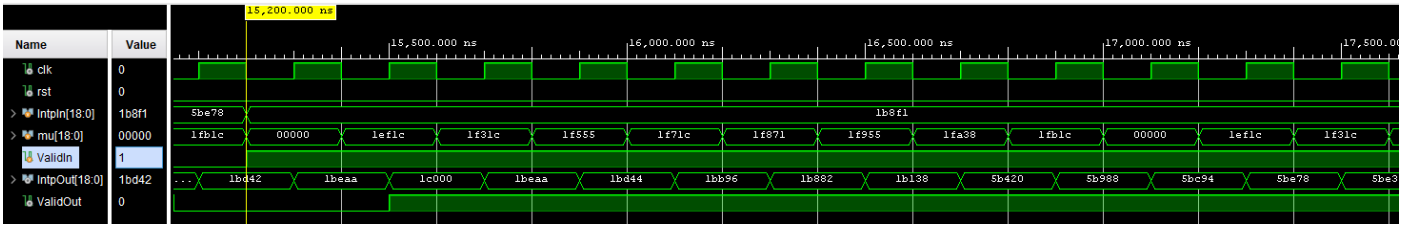


6.3.2 last 9:

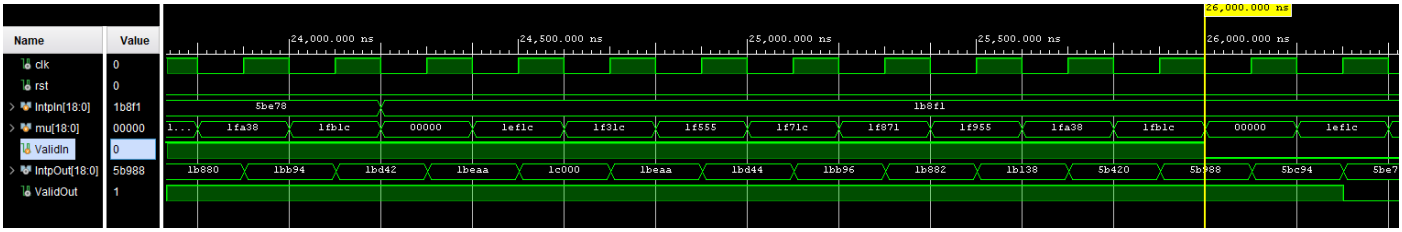


6.4 $x2[m]$ post-synthesis simulations:

6.4.1 first 9:



6.4.2 last 9:



7. (Step 6) Show your timing report to verify the critical path of your block diagram in Q5. (10%) List your timing constraint for post-synthesis simulation. (5%)

7.1. critical path from delay_element0 (x(m+1)) to output and only contain a multiplier (DSP48E1)

Summary

Name	Path 1
Slack	20.158ns
Source	delay_element0_reg[8]/C (rising edge-triggered cell FDCE clocked by clk {rise@0.000ns fall@100.000ns period=200.000ns})
Destination	output_FF_reg[10]/D (rising edge-triggered cell FDCE clocked by clk {rise@0.000ns fall@100.000ns period=200.000ns})
Path Group	clk
Path Type	Setup (Max at Slow Process Corner)
Requirement	200.000ns (clk rise@200.000ns - clk rise@0.000ns)
Data Path Delay	179.690ns (logic 62.228ns (34.631%) route 117.462ns (65.369%))
Logic Levels	332 (CARRY4=146 DSP48E1=1 LUT2=13 LUT3=4 LUT4=19 LUT5=53 LUT6=96)
Clock Path Skew	-0.145ns
Clock Un...rtainty	0.035ns

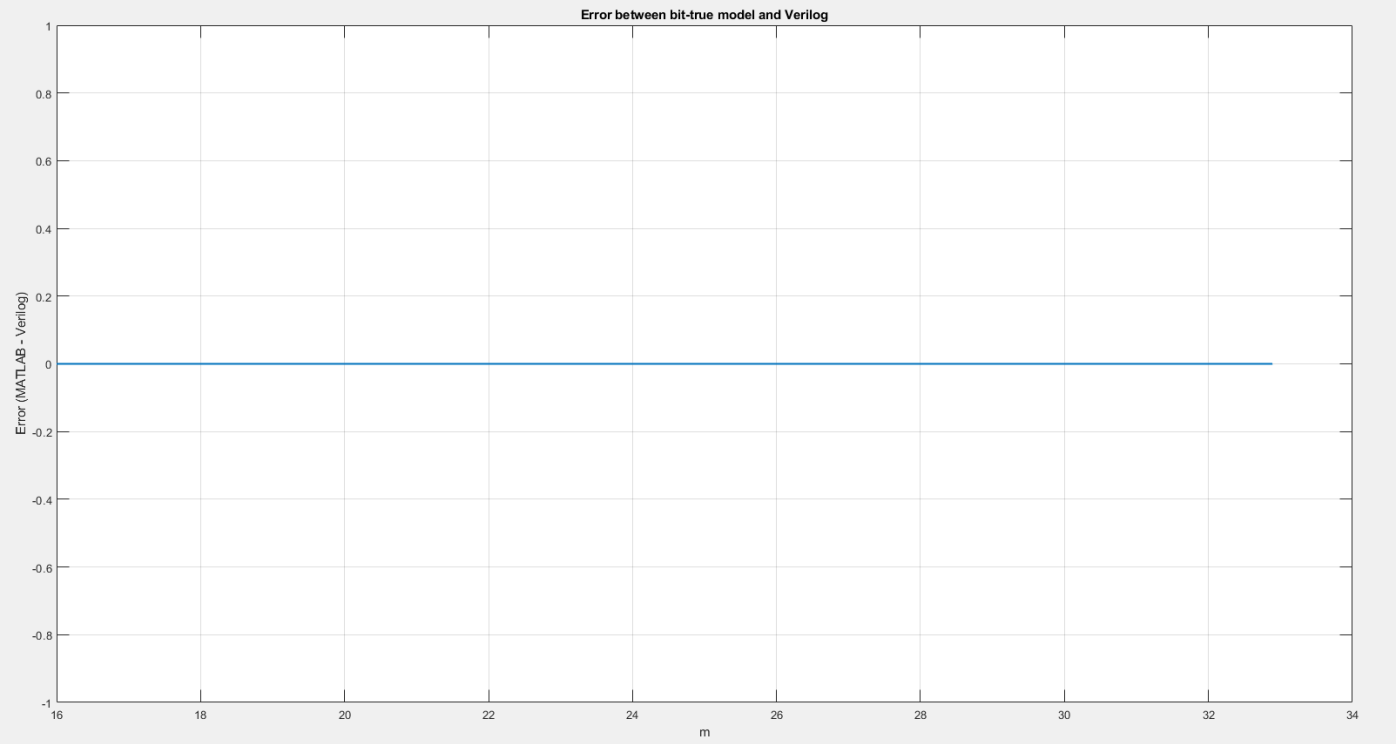
7.2 timing constraint

```
create_clock -period 200.000 -name clk -waveform {0.000 100.000} -add [get_ports clk]
set_input_delay -clock [get_clocks *] -add_delay 100.000 [get_ports -filter { NAME =~  "*" && DIRECTION == "IN" }]
set_output_delay -clock [get_clocks *] -add_delay 1.000 [get_ports -filter { NAME =~  "*" && DIRECTION == "OUT" }]

```

8. (Step 6) Change your bit-true model to the new architecture after your improvement. Also depict all the errors of interpolated outputs of $x1[m]$ in the region of 16 ≤ m ≤ 32 with μ = 0,1/9,2/9, …8/9 and $x2[m]$ in the region of 5 ≤ m≤ 10 with μ = 0,1/9,2/9, …8/9. between the Verilog outputs and bit-true model (20%)

8.1 $x1[m]$: error is 0.



8.2 $x2[m]$: error is 0.