

EE445-Homework 3

Part a:

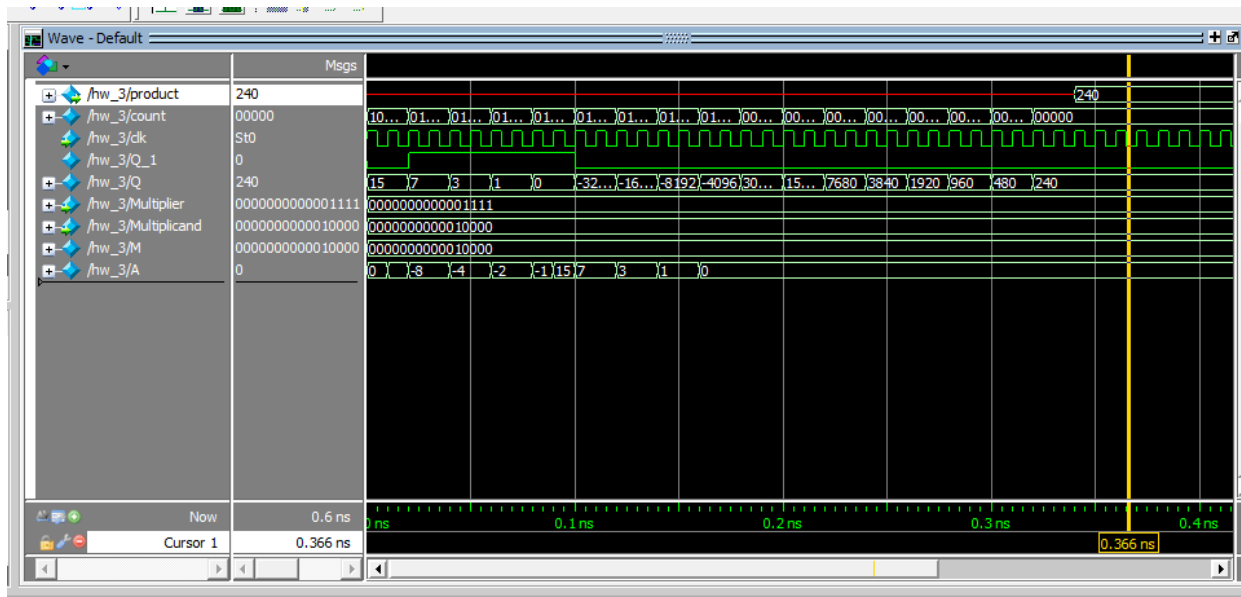


Figure 1: Simulation result of booth's multiplication algorithm implementation

I put 15 to the multiplier, multiplicand is equal to 16. Result is 240 as shown in the product register.

Part b:

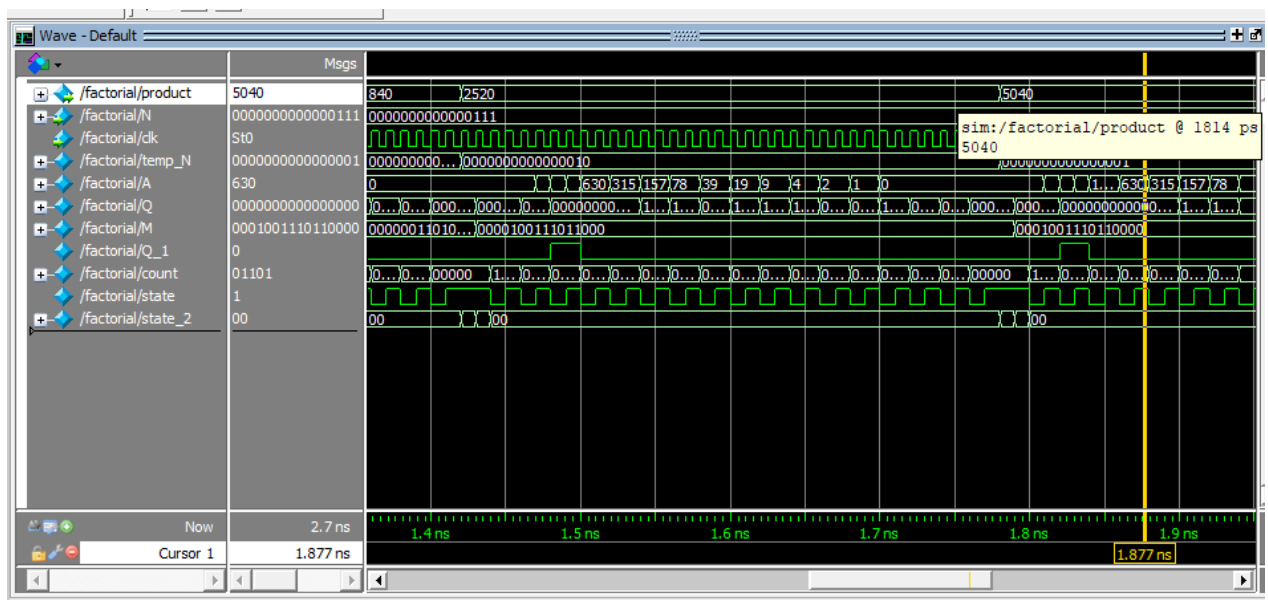


Figure 2: Factorial algorithm simulation result that is implemented based on booth's multiplication algorithm. Given value is 7 and result is 5040 as shown in the product register.

Part c:

Exponential module:

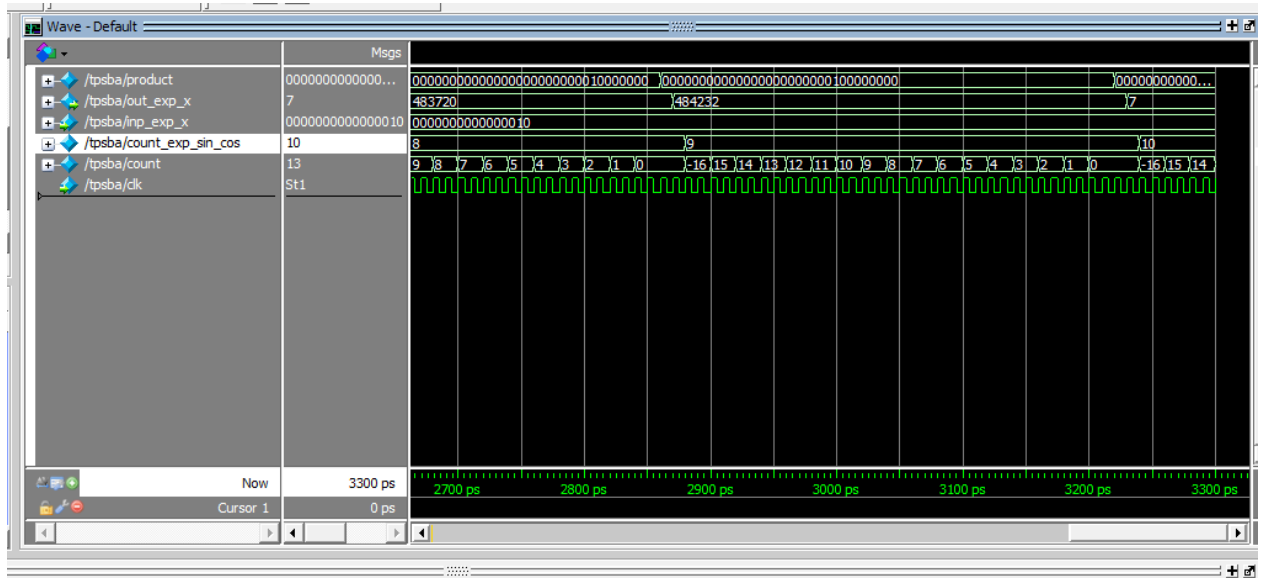


Figure 3: for exp (2), result is shown in the out_exp_x register

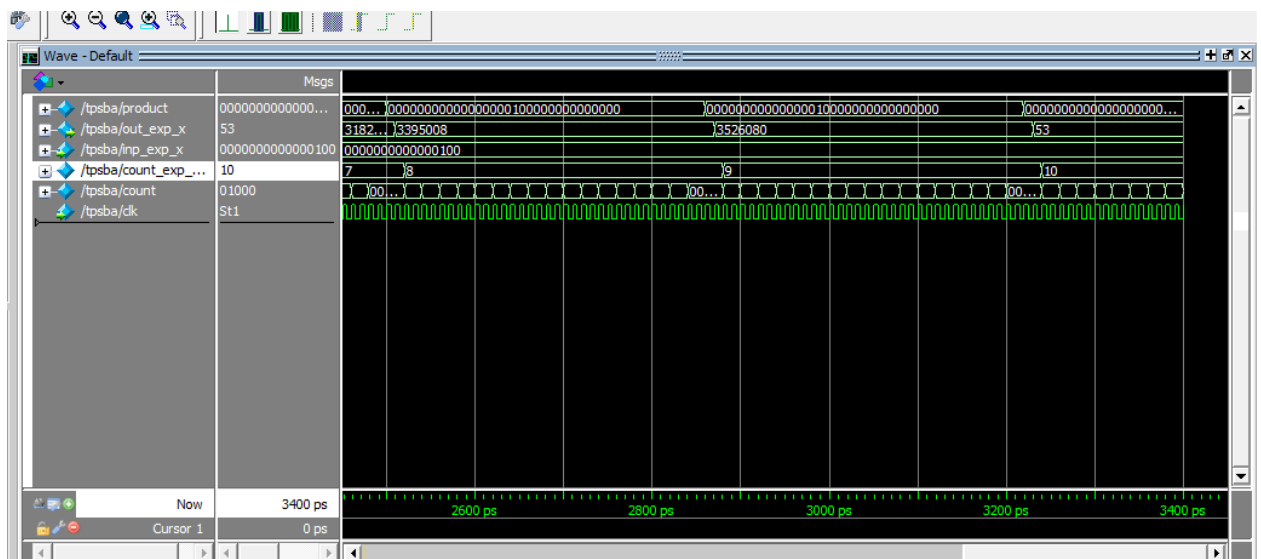


Figure 4: for exp (4), result is shown in the out_exp_x register

There has occurred some problem related with booth's algorithm of multiplication. At some point 16 bit M and Q register's have not enough bit for big numbers. It is especially observed in the below modules. For the $x=2$ case, when shift 16 bit at last stage, we lost the fractional part of the result. Also as we are carrying out processes due to the limited range of registers we also lost some of the fractional values. For example $x=4$ case, result has to be 54. However, since we lost some of the fractional parts while carrying out lots of recursive function. Therefore, result is not exactly the same, but very close.

Sine module:

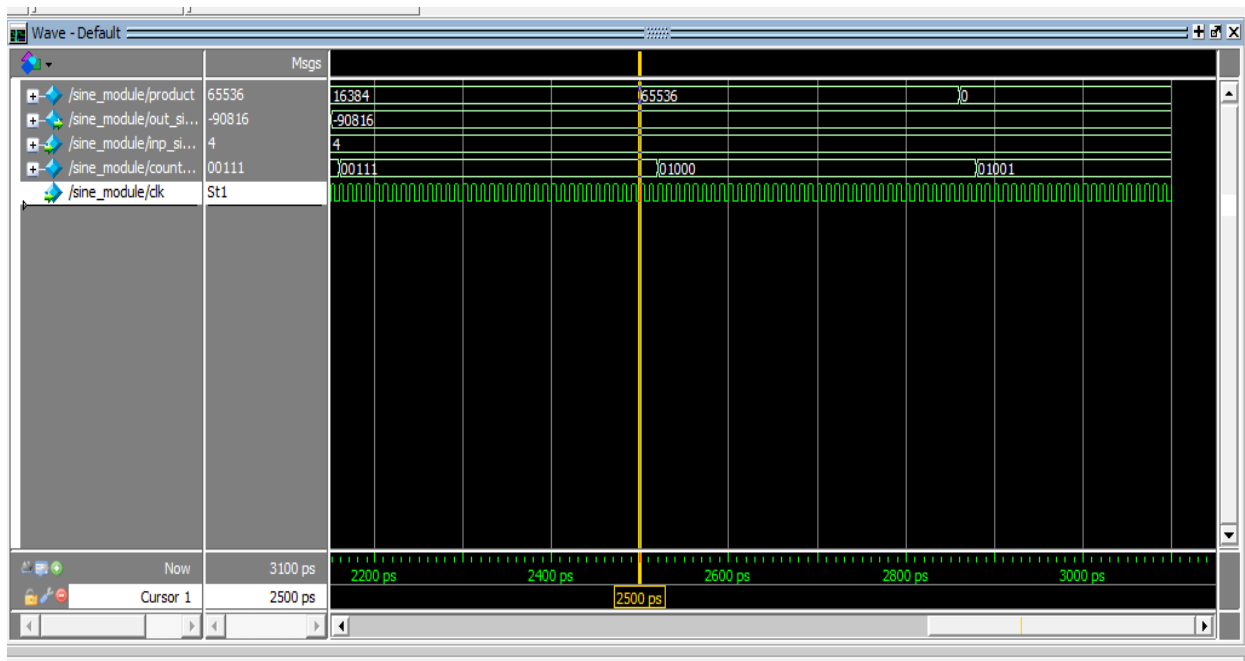


Figure 5: product result in zero due to the limited range of internal registers when $\sin(4)$

Written algorithm works for the values for $x=0,1,2,3$ but it is not working for the other values due to the 16 bit internal registers of the booth's algorithm.

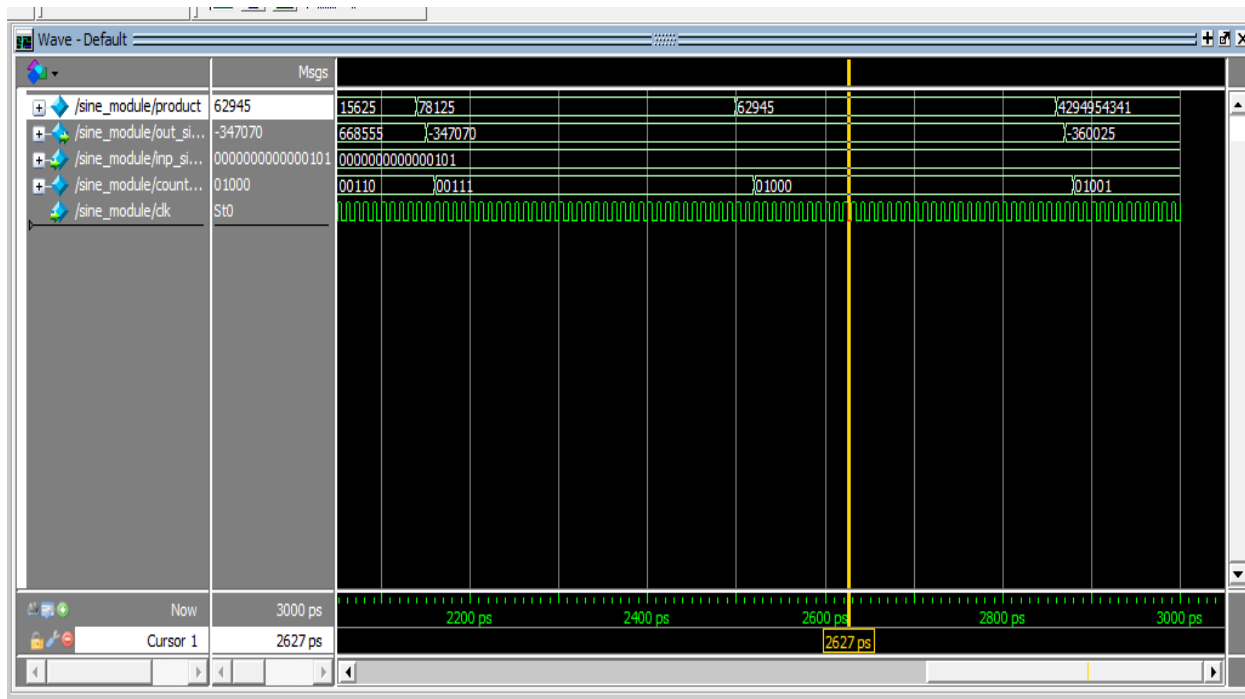


Figure 6: Another example of overflow case when $\sin(x=8)$, disruption can be observed two different product values.

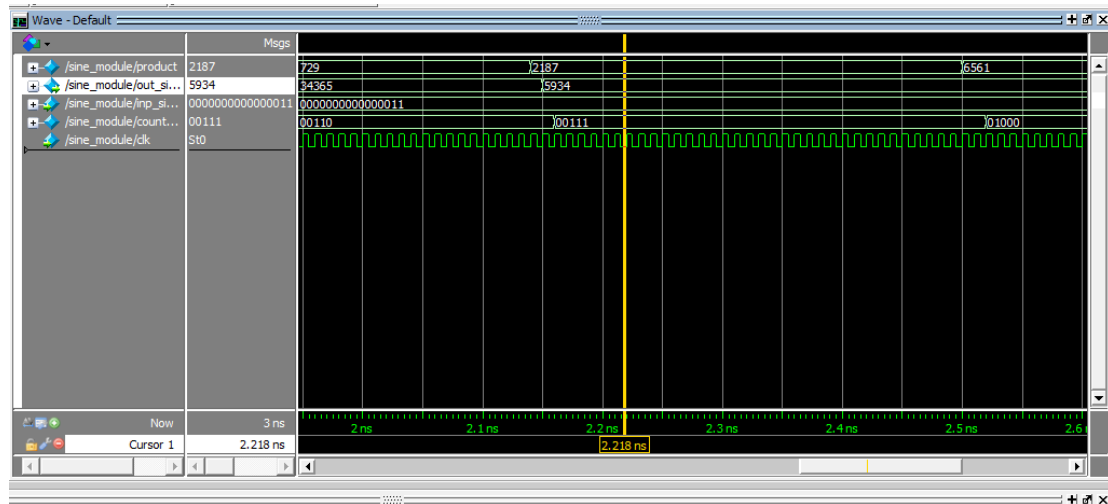


Figure 7: simulation result for sin (3), out_sin_x function shows the result.

Source number 5934	Input base 10 <small>Input numeral system base</small>	Target base 2 <small>Target numeral system base</small>
Calculation precision Digits after the decimal point: 0		
CALCULATE		
Target number 1011100101110	Conversion details undefined	Source number (decimal) 5934
Target number (decimal) 5934	Conversion error (decimal) 0	Maximum conversion error possible (decimal) 1

Figure 8: binary conversion of 5934(decimal)

Conversion of fractional numbers between numeral systems		
Source number 0.00001011100101110	Input base 2 <small>Input numeral system base</small>	Target base 10 <small>Target numeral system base</small>
Calculation precision Digits after the decimal point: 20		
CALCULATE		
Target number 0.04527282714843750668	Conversion details undefined	Source number (decimal) 0.0452728271484375
Target number (decimal) 0.04527282714843751	Conversion error (decimal) 0	Maximum conversion error possible (decimal) 0.00000000000000000001

Figure 9: result observed from the sin (3)

Due to the limited internal register (limited at 16 bit), we cannot above 3 for these functions. Since if we put 5 for example, $5^9 = 1953125$ that is above of the number 131071 that is represented by 16 bit at most. Therefore, I run my code 3.

Cosine module:

Similar overflow case is true for the cosine module, since it is also using same booth's module for multiplication.

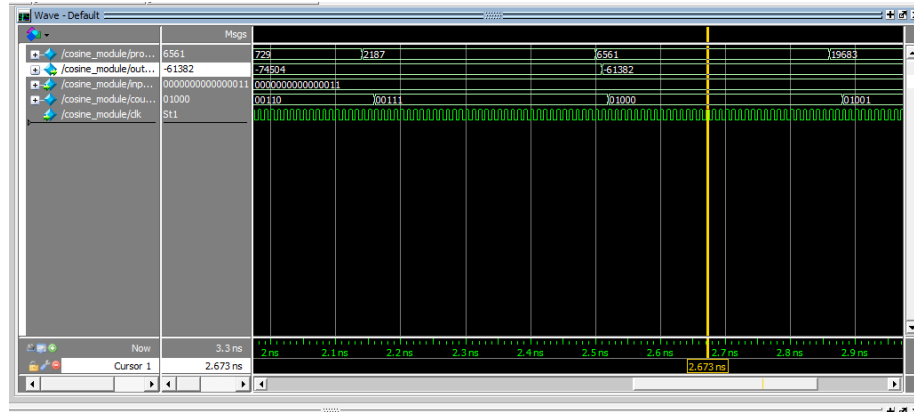


Figure 10: 61382 is the result of the algorithm.

Conversion of fractional numbers between numeral systems

Source number 61382	Input base 10 Input numeral system base	Target base 2 Target numeral system base
------------------------	---	--

Calculation precision
Digits after the decimal point: 16

CALCULATE

Target number 111011111000110.00000 00000000000	Conversion details undefined	Source number (decimal) 61382
Target number (decimal) 61382	Conversion error (decimal) 0	Maximum conversion error possible (decimal) 0.0000076293945313

Figure 11: 61382 is converted corresponding binary value

Conversion of fractional numbers between numeral systems

Source number 0.111011111000110	Input base 2 Input numeral system base	Target base 10 Target numeral system base
------------------------------------	--	---

Calculation precision
Digits after the decimal point: 20

CALCULATE

Target number 0.93661499023437488220	Conversion details undefined	Source number (decimal) 0.936614990234375
Target number (decimal) 0.9366149902343749	Conversion error (decimal) 0	Maximum conversion error possible (decimal) 0.00000000000000000001

Figure 12: The result for the above value that is founded in the simulation is the 0.936 = cos (3)

There are some discrepancies on the results due to the loss of some fractional parts while processing both sine and cosine algorithms. Algorithm results are not exactly the same of the theoretical ones but very close them.