EE224: Assignment-1 Report

Multiplication Of 8-Bit Signed Numbers
Using Booth Encoding

Jayesh Choudhary, 170070038

1 Introduction

The usual shift-and-add method of multiplication is a bit lengthy and requires a longer computation time. We can decrease this time by decreasing the number of partial sums by using 2-bits of the number at once while multiplying to the multiplicand. Booth encoding is one such method. There is further modification in this method in which we use only have multiplier as +1, -1, +2, -2 or 0. But in this case we take 3-bits of the number to be multiplied at once and the next set have one overlapping bit.

2 Algorithm

We first take the last two-bits of the number along with 0 at the end to find the multiplier of this iteration from the table shown below.According to the control pins, our carry and control pins are defined.
Using these control pins we obtain a 9-bit number which we use to obtain a 16-bit number considering the sign of the number and shift.
Then we again take 3-bits having last bit same as in the previous 3-bits and perform the same steps to obtain another 16-bit number and finally add this partial term to the partial term obtained in the step above.
We repeat this step 2 more times to finally obtain the output.

3 VHDL files

- Gates
- Full_Adder
- mux
- decoder
- sixteenbit
- booth
- Testbench
- DUT

3.1 Gates, Full_Adder

These contains the entity description of basic logic gates and some other function that we are going to use as a component in our project.

3.2 decoder

This file contains an entity which takes the 3-bit of the number and gives out a 3-bit output which is then used to find out the corresponding multiplier. The first two bits of output are the control pins for mux and the last bit is the initial carry for the corresponding partial sum (The carry is given as 1 if the multiplier is negative). The logic was developed for the same using K-maps.

3-bits	Multiplier
000	0
001	+1
010	+1
011	+2
100	-2
101	-1
110	-1
111	0

Figure 1: Multiplier

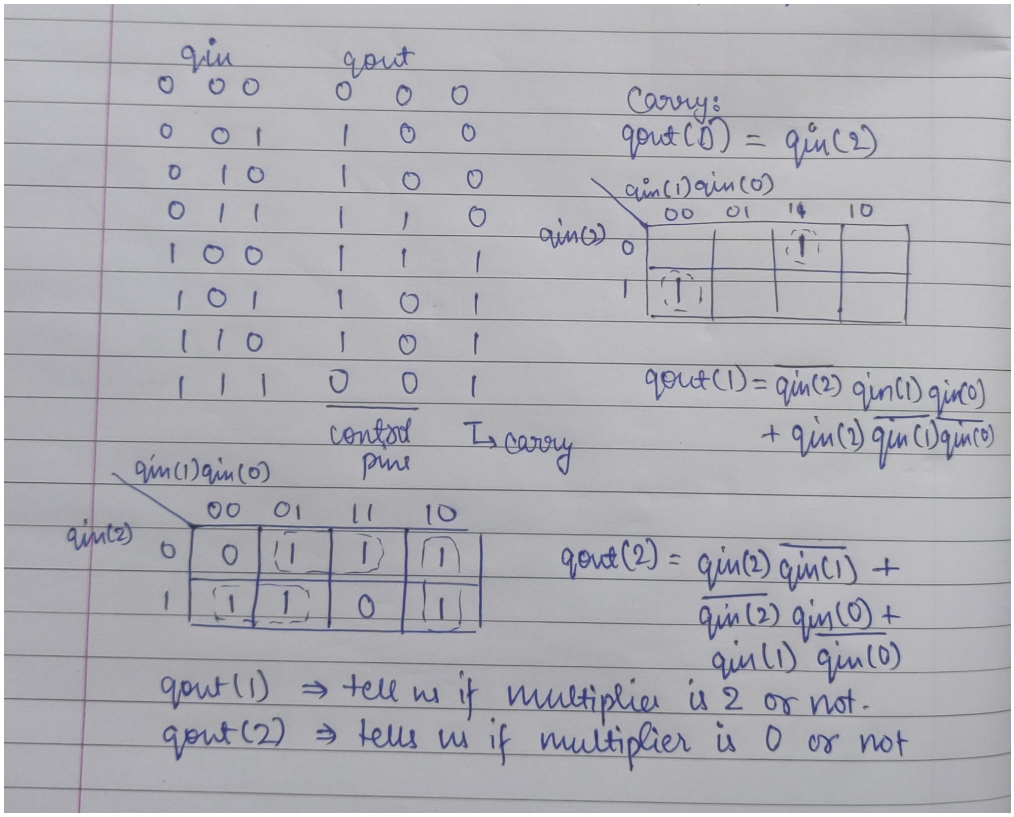

Figure 2: Decoder Logic

3.3 mux

This vhdl file contains an entity mux that we are using to get the partial term for addition according to the output of the decoder. Its input are multiplicand and the control pins which we get from the decoder. For the control bits, first one is the main control pin which tells us if the multiplier is zero or not. The second control pin tells us whether the multiplier is 2 or 1.

3.4 sixteenbit

This file have an entity which performs the function of adding the partial sums. Its input are 16-bit sequence and the initial carry (that we get from decoder). If we get a carry as 0 then it is the usual addition but if the carry is 1 then we add the complement of that bit-string so as to get the subtraction.

3.5 booth

This file contains the main driver of the program which takes the input as 2 8-bit numbers and then perform the multiplication of the two numbers using booth encoding by using the above mentioned components.

3.6 DUT

This file is for the wrapping-up of the whole code.

3.7 Testbench

This file is for the interaction of the code with the test cases so that we can apply various inputs and check the output.

4 RTL View

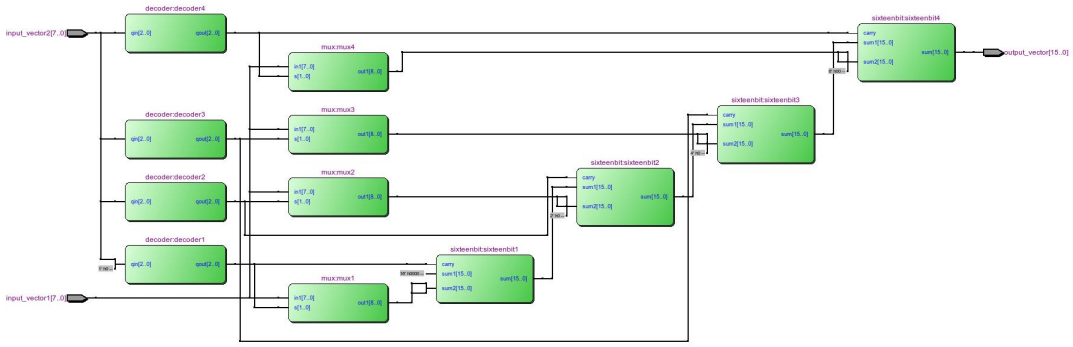

Figure 3: Design

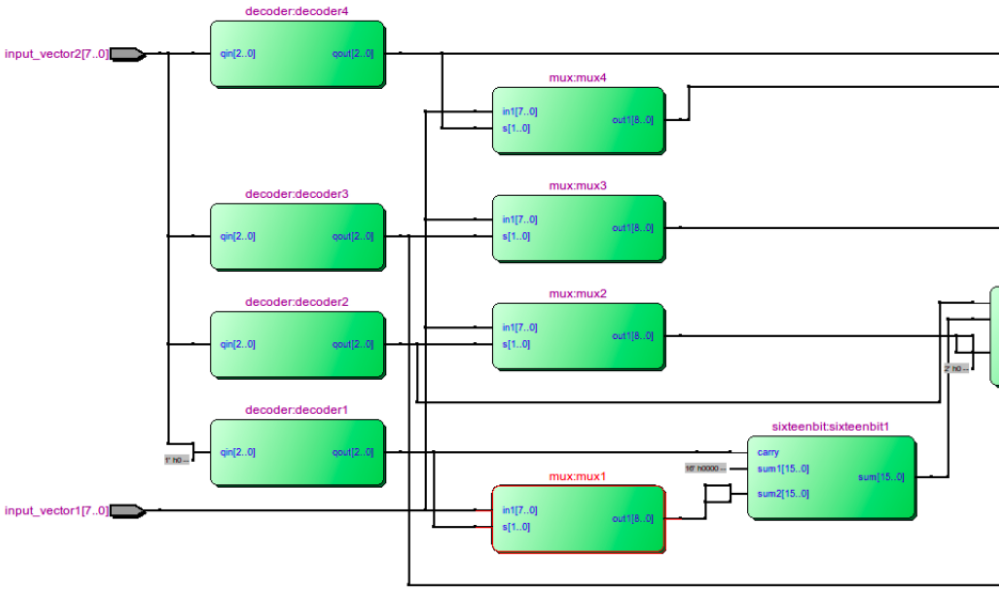

Figure 4: decoder and mux units to produce partial terms for addition and initial carry

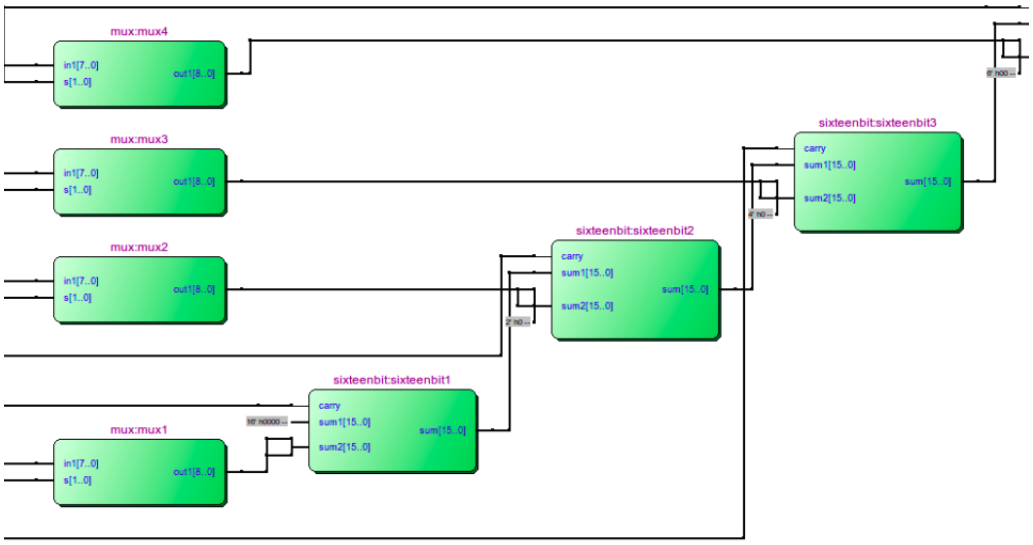

Figure 5: sixteenbit adder units with initial carry and partial terms

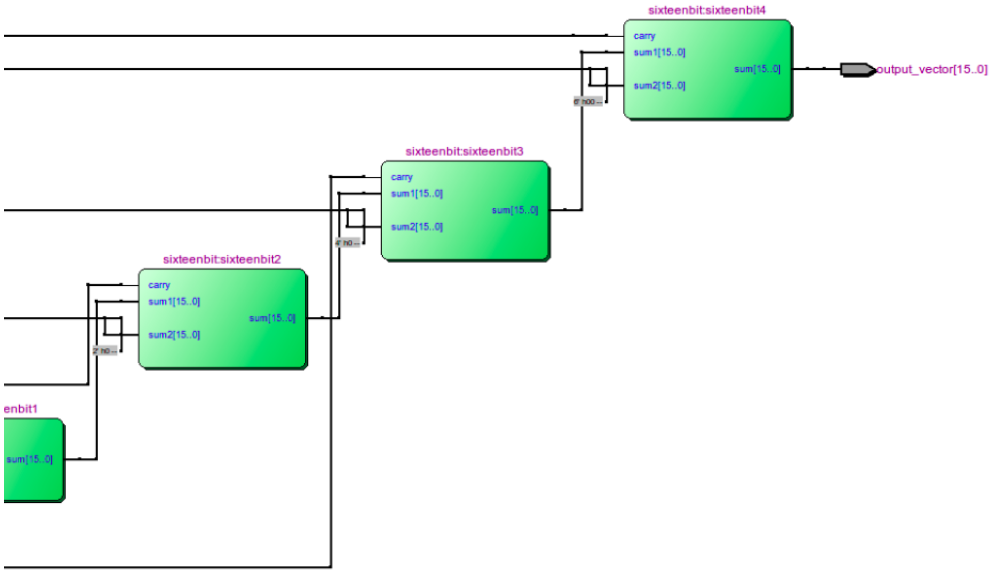

Figure 6: cascaded sixteen bit adder units to generate the final output

5 Compilation and Success full Test-passing

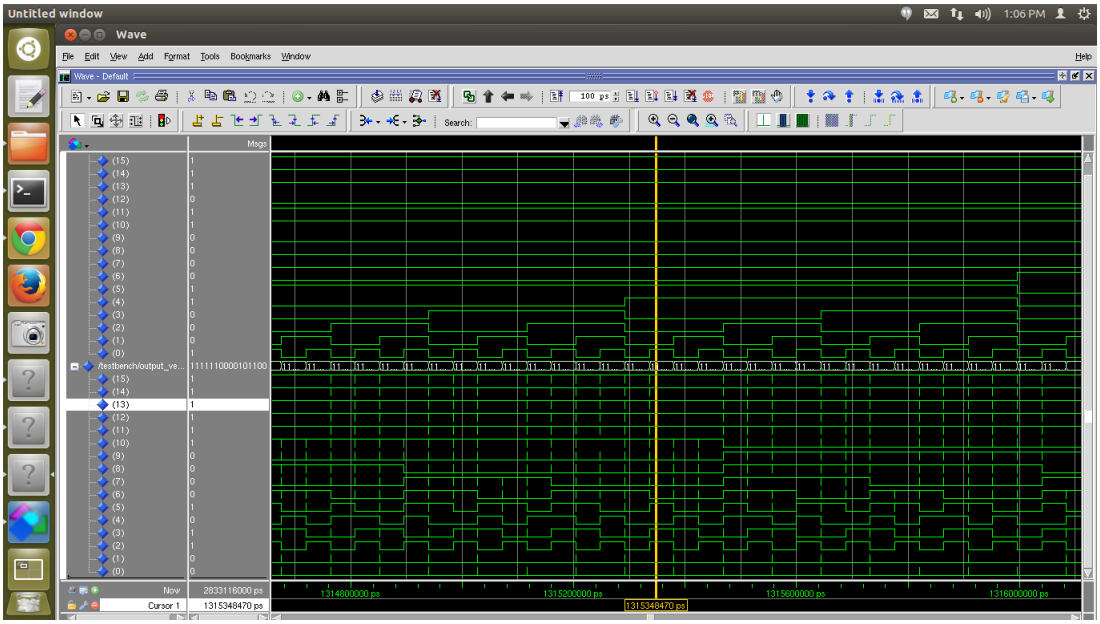

Figure 7: Output waveform for the given testcases

Figure 8: Transcript for passing all the test cases