

Solution:

Multiply-Accumulate circuit (MAC) with Dadda reduction scheme to multiply two 16-bit unsigned integer operands and add it to a 32-bit unsigned accumulator is implemented in VHDL and simulated using **Quartus Prime lite 18.1** edition.

Brent Kung adder implemented in Assignment-3 is used for final addition in MAC reduction

MAC with Dadda reduction dot diagram is given in the following Figure 1 to Figure 2

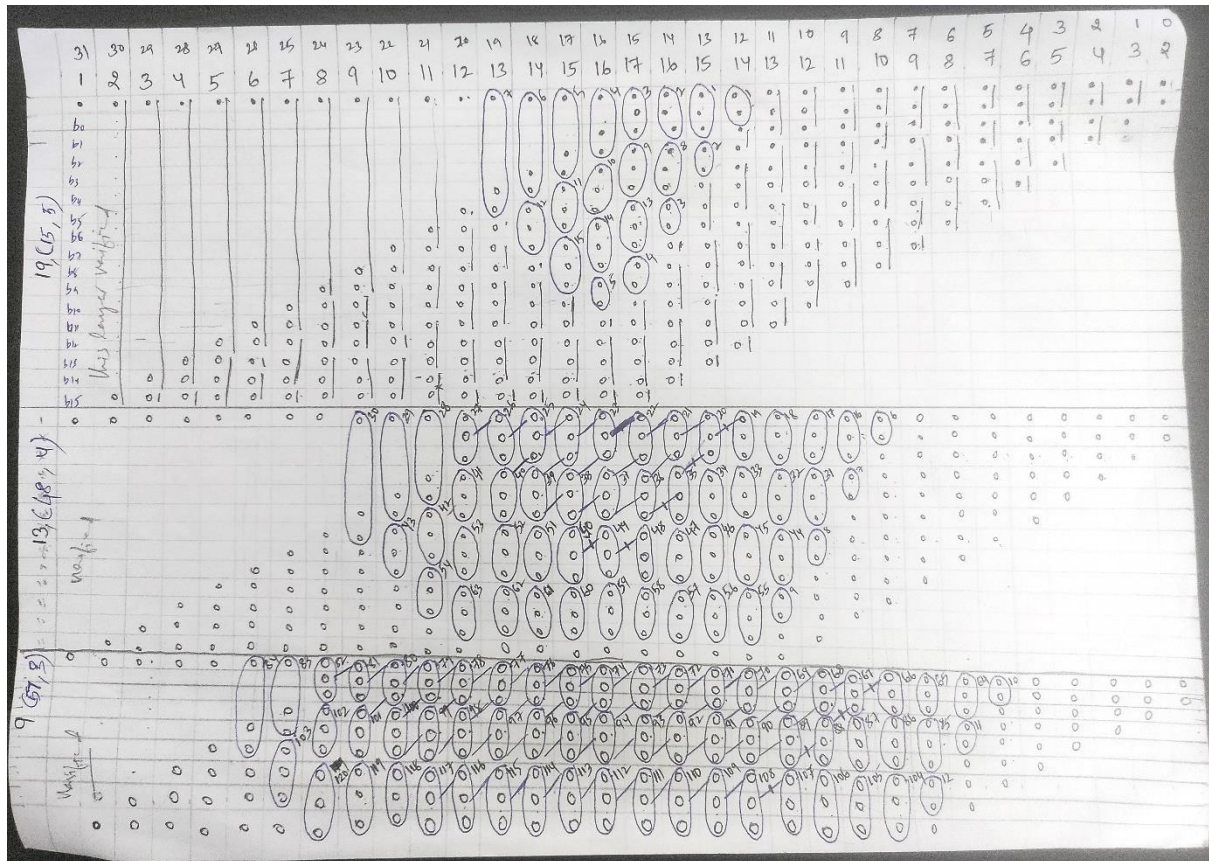


Figure 1 MAC Dadda reduction dot diagram (first 3 layers)

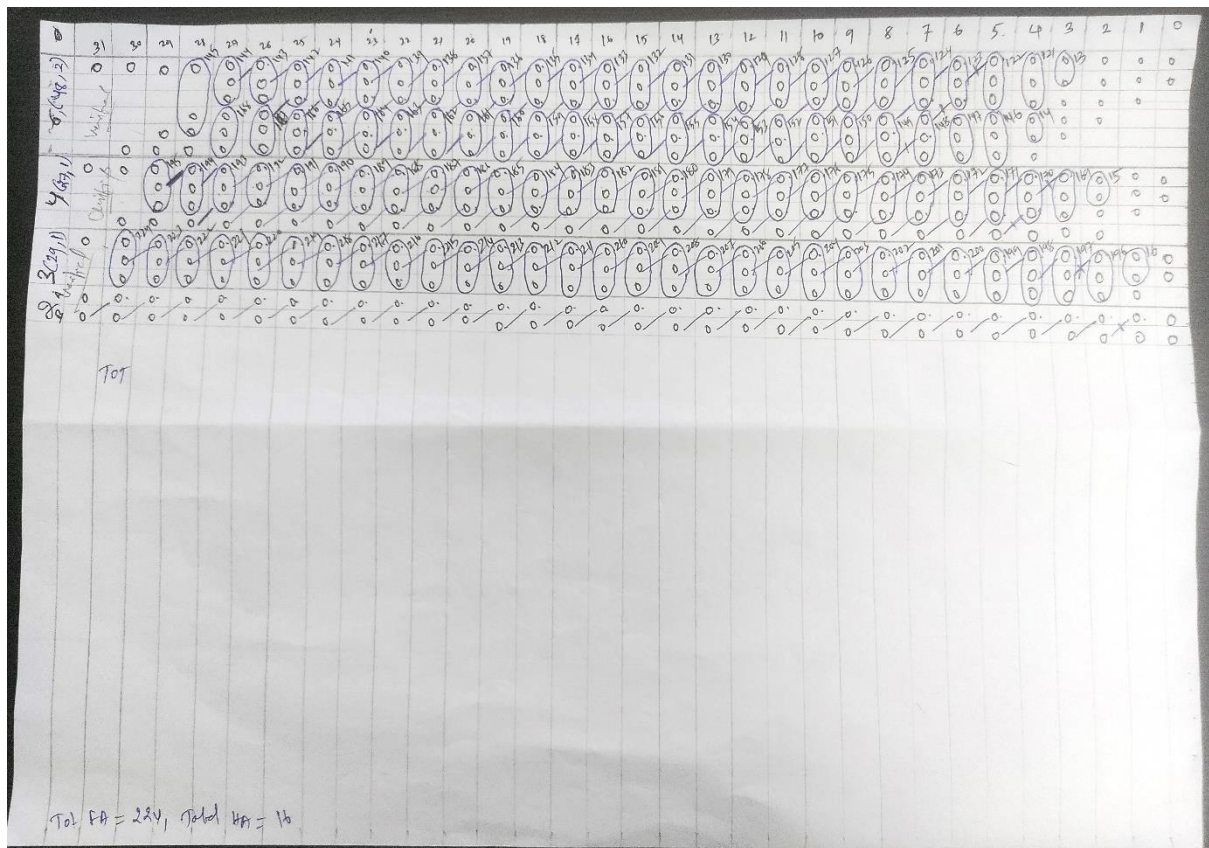


Figure 2 MAC Dadda reduction dot diagram (last 4 layers)

Input File

Input file contents are shown in theFigure 3.

```

F:\M-tech\1st Sem EE5\EE 671_VLSI Design\Assignments\Assignment-4\Assignment_4_HDL\Modelsim_user\test_MAC_Dadda_23m1200_inputs.txt - Notepad ++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
test_MAC_Dadda_23m1200_inputs.txt test_MAC_Dadda_23m1200_outputs.txt
1 0000000000001010 0000000000001111 000000000000000000000000000010100 000000000000000000000000000010101010 0
2 1111111111111111 1111111111111111 11111111111111111111111111111111111111 11111111111111000000000000000000 1
3 00000000011110000 00000000011110000 0000000000000000000000000000000000000000 00000000000000000000000000000000 0
4 00000000000010000 00000000000010000 0000000000000000000000000000000000000000 00000000000000000000000000000000 0
5 00000000000000000 00000000000000000 0000000000000000000000000000000000000000 00000000000000000000000000000000 0
6 00000000000000100 00000000000000100 0000000000000000000000000000000000000000 00000000000000000000000000000000 0
7 00000000000001000 00000000000001000 0000000000000000000000000000000000000000 00000000000000000000000000000000 0
8 00000000000010000 00000000000010000 0000000000000000000000000000000000000000 00000000000000000000000000000000 0
9 1000101010100111 1101110001000000 10110011010101001001011111111111 00101010100111011000101101111111 1
10 1010001110010011 1010101010011111 10101100101001101001011100000010 00011001101010111100110101001111 1
11 00000000000000000 00000000000000000 0000000000000000000000000000000000000000 00000000000000000000000000000000 0
12 00000000000000000 00000000000000000 0000000000000000000000000000000000000000 00000000000000000000000000000000 0
13 00000000000000000 00000000000000000 0000000000000000000000000000000000000000 00000000000000000000000000000000 0
14 10000000000000000 10000000000000000 0000000000000000000000000000000000000000 01000000000000000000000000000000 0
15 1010001110010011 1010101010011111 0000000000000000000000000000000000000000 01101101000001010011011001001101 0
16 00000000000000000 00000000000000000 11111111111111111111111111111111111111 11111111111111111111111111111111 0
17 00000000000000000 1111111111111111 11111111111111111111111111111111111111 11111111111111111111111111111111 0
18 1111111111111111 1111111111111111 0000000000000000000000000000000000000000 11111111111111000000000000000001 0
19 1111111111111111 00000000000000000 000000000000000000000000000000000000000 00000000000000000000000000000000 0
20 0000000000000000 1111111111111111 0000000000000000000000000000000000000000 00000000000000000000000000000000 0

```

Input A

Input B

Input C

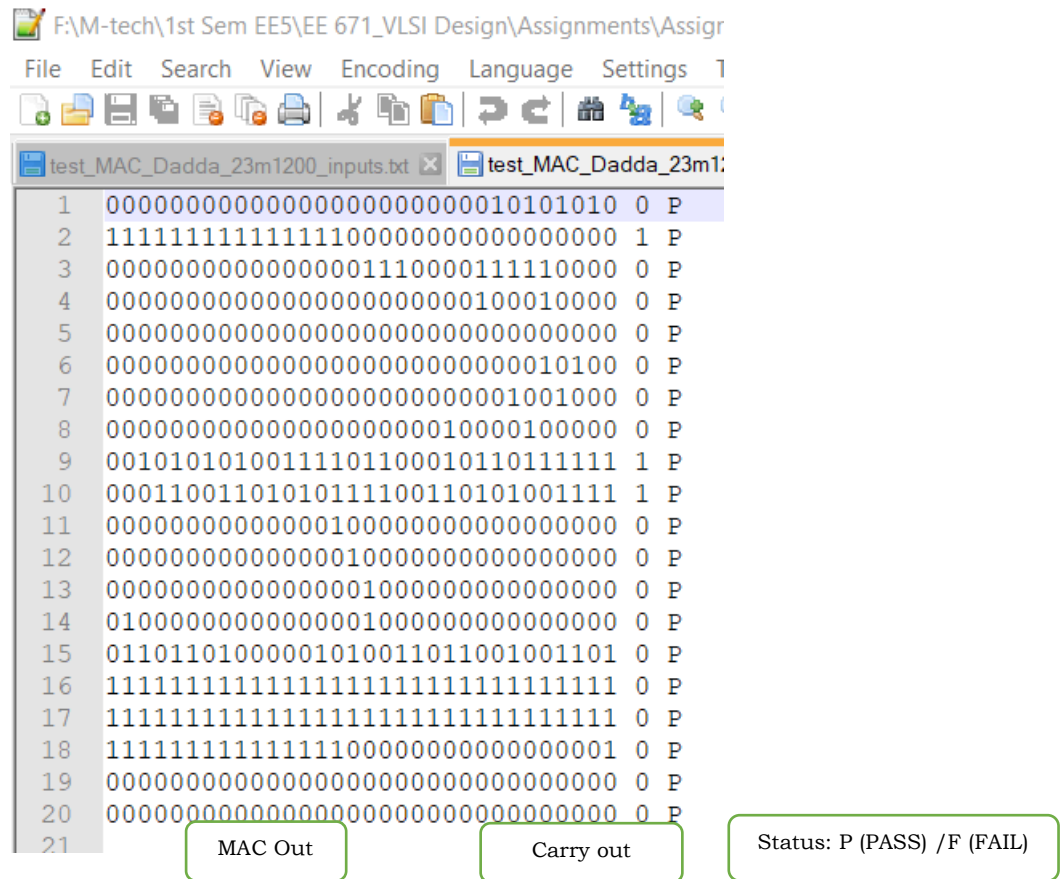
Output

Output Carry

Figure 3 Input file used in testbench

Output File

Output file contents are shown in theFigure 4.



Simulated result in waveform is shown in the Figure 5

Figure 5 Simulated waveform of stimuli