Assignment-5

16-bit MAC

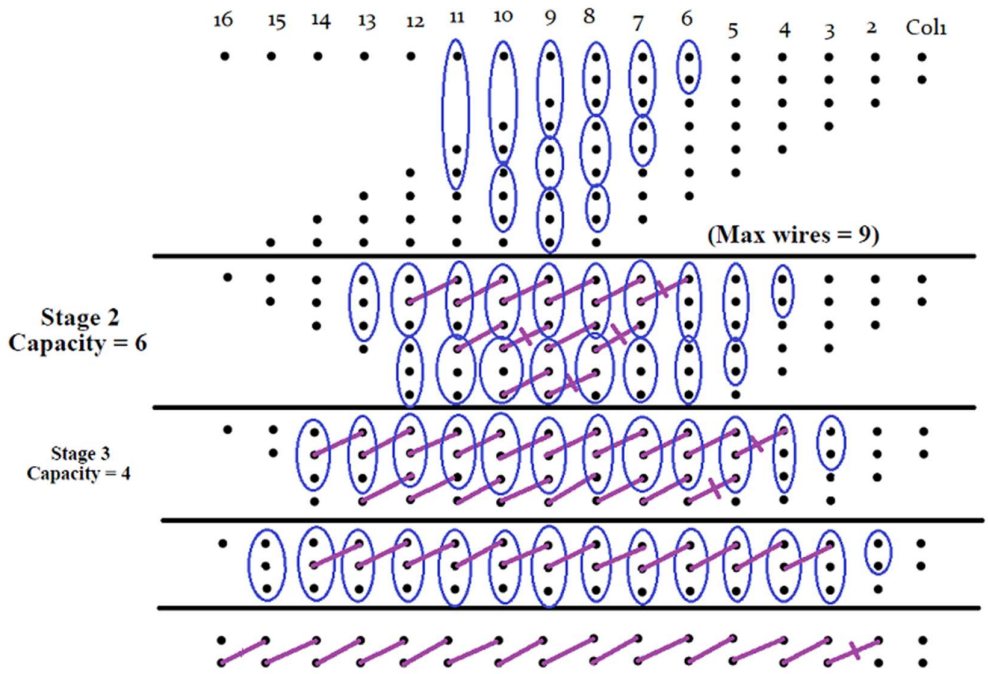
Vagadia Yash Dinesh

Roll Number:20D070086

Course:EE671 VLSI Design

Q-1

Designing a 16-bit MAC with two 8-bit numbers as a input and a 16-bit number C as a input, in VHDL and simulating it using a test bench.



Dot diagram of MAC, wire reduction using DADDA scheme. From the 1st stage wires are named Colj(i), where j column number and i the index for element in that column.

From stage-2 output are named Wj(i) and same way from stage-3,4,5 as X,Z,Y. These are wires are defined as signals at the start of architecture.

Full adders, Half adder and logarithmic adder entities are defined in *Gates.vhdl* file using the given basic gates.

a.

An 8 cross 8 array of and gates is defined to generate partial product bits.

```
begin
--8*8 array of and gates==stage1     a7, a6, a5,.....a0
---                                  b7, b6, b5, ....b0 index of b will in row and that of in column
--right most column is col1
--bellow group is for row1
arr00: andgate port map (A=>A(0) , B=>B(0) , prod=>Col1(1));
arr01: andgate port map (A=>A(1) , B=>B(0) , prod=>Col2(1));
arr02: andgate port map (A=>A(2) , B=>B(0) , prod=>Col3(1));
arr03: andgate port map (A=>A(3) , B=>B(0) , prod=>Col4(1));
arr04: andgate port map (A=>A(4) , B=>B(0) , prod=>Col5(1));
arr05: andgate port map (A=>A(5) , B=>B(0) , prod=>Col6(1));
arr06: andgate port map (A=>A(6) , B=>B(0) , prod=>Col7(1));
arr07: andgate port map (A=>A(7) , B=>B(0) , prod=>Col8(1));

arr10: andgate port map (A=>A(0) , B=>B(1) , prod=>Col2(2));
arr11: andgate port map (A=>A(1) , B=>B(1) , prod=>Col3(2));
arr12: andgate port map (A=>A(2) , B=>B(1) , prod=>Col4(2));
arr13: andgate port map (A=>A(3) , B=>B(1) , prod=>Col5(2));
arr14: andgate port map (A=>A(4) , B=>B(1) , prod=>Col6(2));
arr15: andgate port map (A=>A(5) , B=>B(1) , prod=>Col7(2));
arr16: andgate port map (A=>A(6) , B=>B(1) , prod=>Col8(2));
arr17: andgate port map (A=>A(7) , B=>B(1) , prod=>Col9(1));

arr20: andgate port map (A=>A(0) , B=>B(2) , prod=>Col3(3));
arr21: andgate port map (A=>A(1) , B=>B(2) , prod=>Col4(3));
arr22: andgate port map (A=>A(2) , B=>B(2) , prod=>Col5(3));
arr23: andgate port map (A=>A(3) , B=>B(2) , prod=>Col6(3));
arr24: andgate port map (A=>A(4) , B=>B(2) , prod=>Col7(3));
arr25: andgate port map (A=>A(5) , B=>B(2) , prod=>Col8(3));
arr26: andgate port map (A=>A(6) , B=>B(2) , prod=>Col9(2));
arr27: andgate port map (A=>A(7) , B=>B(2) , prod=>Col10(1));
```

```

  arr30: andgate port map (A=>A(0) , B=>B(3) , prod=>Col4(4));
  arr31: andgate port map (A=>A(1) , B=>B(3) , prod=>Col5(4));
  arr32: andgate port map (A=>A(2) , B=>B(3) , prod=>Col6(4));
  arr33: andgate port map (A=>A(3) , B=>B(3) , prod=>Col7(4));
  arr34: andgate port map (A=>A(4) , B=>B(3) , prod=>Col8(4));
  arr35: andgate port map (A=>A(5) , B=>B(3) , prod=>Col9(3));
  arr36: andgate port map (A=>A(6) , B=>B(3) , prod=>Col10(2));
  arr37: andgate port map (A=>A(7) , B=>B(3) , prod=>Col11(1));

  arr40: andgate port map (A=>A(0) , B=>B(4) , prod=>Col5(5));
  arr41: andgate port map (A=>A(1) , B=>B(4) , prod=>Col6(5));
  arr42: andgate port map (A=>A(2) , B=>B(4) , prod=>Col7(5));
  arr43: andgate port map (A=>A(3) , B=>B(4) , prod=>Col8(5));
  arr44: andgate port map (A=>A(4) , B=>B(4) , prod=>Col9(4));
  arr45: andgate port map (A=>A(5) , B=>B(4) , prod=>Col10(3));
  arr46: andgate port map (A=>A(6) , B=>B(4) , prod=>Col11(2));
  arr47: andgate port map (A=>A(7) , B=>B(4) , prod=>Col12(1));

  arr50: andgate port map (A=>A(0) , B=>B(5) , prod=>Col6(6));
  arr51: andgate port map (A=>A(1) , B=>B(5) , prod=>Col7(6));
  arr52: andgate port map (A=>A(2) , B=>B(5) , prod=>Col8(6));
  arr53: andgate port map (A=>A(3) , B=>B(5) , prod=>Col9(5));
  arr54: andgate port map (A=>A(4) , B=>B(5) , prod=>Col10(4));
  arr55: andgate port map (A=>A(5) , B=>B(5) , prod=>Col11(3));
  arr56: andgate port map (A=>A(6) , B=>B(5) , prod=>Col12(2));
  arr57: andgate port map (A=>A(7) , B=>B(5) , prod=>Col13(1));
```

```
arr60: andgate port map (A=>A(0) , B=>B(6) , prod=>Col7(7));
arr61: andgate port map (A=>A(1) , B=>B(6) , prod=>Col8(7));
arr62: andgate port map (A=>A(2) , B=>B(6) , prod=>Col9(6));
arr63: andgate port map (A=>A(3) , B=>B(6) , prod=>Col10(5));
arr64: andgate port map (A=>A(4) , B=>B(6) , prod=>Col11(4));
arr65: andgate port map (A=>A(5) , B=>B(6) , prod=>Col12(3));
arr66: andgate port map (A=>A(6) , B=>B(6) , prod=>Col13(2));
arr67: andgate port map (A=>A(7) , B=>B(6) , prod=>Col14(1));

arr70: andgate port map (A=>A(0) , B=>B(7) , prod=>Col8(8));
arr71: andgate port map (A=>A(1) , B=>B(7) , prod=>Col9(7));
arr72: andgate port map (A=>A(2) , B=>B(7) , prod=>Col10(6));
arr73: andgate port map (A=>A(3) , B=>B(7) , prod=>Col11(5));
arr74: andgate port map (A=>A(4) , B=>B(7) , prod=>Col12(4));
arr75: andgate port map (A=>A(5) , B=>B(7) , prod=>Col13(3));
arr76: andgate port map (A=>A(6) , B=>B(7) , prod=>Col14(2));
arr77: andgate port map (A=>A(7) , B=>B(7) , prod=>Col15(1));
```

b.

Then Half adders and full adders are used for wire reduction as shown in dot diagram.

```
--stage2
ha16: HALF_ADDER port map (A=>C(5), B=>Col6(1), S=>W6(1), C=>W7(2)); --haij i is stage and j is column

fa17: Full_Adder port map (A=>C(6), B=>Col7(1), Cin=>Col7(2),  S=>W7(1), Cout=>W8(2));
ha17: HALF_ADDER port map (A=>Col7(3), B=>Col7(4), S=>W7(3), C=>W8(4));

fa18: Full_Adder port map (A=>C(7), B=>Col8(1), Cin=>Col8(2),  S=>W8(1), Cout=>W9(2));
fa18ii: Full_Adder port map (A=>Col8(3), B=>Col8(4), Cin=>Col8(5),  S=>W8(3), Cout=>W9(4));
ha18: HALF_ADDER port map (A=>Col8(6), B=>Col8(7), S=>W8(5), C=>W9(6));

fa19: Full_Adder port map (A=>C(8), B=>Col9(1), Cin=>Col9(2),  S=>W9(1), Cout=>W10(2));
ha19: HALF_ADDER port map (A=>Col9(3), B=>Col9(4), S=>W9(3), C=>W10(4));
fa19ii: Full_Adder port map (A=>Col9(5), B=>Col9(6), Cin=>Col9(7),  S=>W9(5), Cout=>W10(5));

fa110: Full_Adder port map (A=>C(9), B=>Col10(1), Cin=>Col10(2),  S=>W10(1), Cout=>W11(2));
fa110ii: Full_Adder port map (A=>Col10(3), B=>Col10(4), Cin=>Col10(5),  S=>W10(3), Cout=>W11(3));

fa111: Full_Adder port map (A=>C(10), B=>Col11(1), Cin=>Col11(2),  S=>W11(1), Cout=>W12(1));
```

```
--stage3
ha24: HALF_ADDER port map (A=>C(3), B=>Col4(1), S=>X4(1), C=>X5(2));

fa25: Full_Adder port map (A=>C(4), B=>Col5(1), Cin=>Col5(2),  S=>X5(1), Cout=>X6(2));
ha25: HALF_ADDER port map (A=>Col5(3), B=>Col5(4), S=>X5(3), C=>X6(4));

fa26: Full_Adder port map (A=>W6(1), B=>Col6(2), Cin=>Col6(3),  S=>X6(1), Cout=>X7(2));
fa26ii: Full_Adder port map (A=>Col6(4), B=>Col6(5), Cin=>Col6(6),  S=>X6(3), Cout=>X7(4));

fa27: Full_Adder port map (A=>W7(1), B=>W7(2), Cin=>W7(3), S=>X7(1), Cout=>X8(2));
fa27ii: Full_Adder port map (A=>Col7(5), B=>Col7(6), Cin=>Col7(7),  S=>X7(3), Cout=>X8(4));

fa28: Full_Adder port map (A=>W8(1), B=>W8(2), Cin=>W8(3), S=>X8(1), Cout=>X9(2));
fa28ii: Full_Adder port map (A=>W8(4), B=>W8(5), Cin=>Col8(8), S=>X8(3), Cout=>X9(4));

fa29: Full_Adder port map (A=>W9(1), B=>W9(2), Cin=>W9(3), S=>X9(1), Cout=>X10(2));
fa29ii: Full_Adder port map (A=>W9(4), B=>W9(5), Cin=>W9(6), S=>X9(3), Cout=>X10(4));

fa210: Full_Adder port map (A=>W10(1), B=>W10(2), Cin=>W10(3), S=>X10(1), Cout=>X11(2));
fa210ii: Full_Adder port map (A=>W10(4), B=>W10(5), Cin=>Col10(6), S=>X10(3), Cout=>X11(4));

fa211: Full_Adder port map (A=>W11(1), B=>W11(2), Cin=>W11(3), S=>X11(1), Cout=>X12(2));
fa211ii: Full_Adder port map (A=>Col11(3), B=>Col11(4), Cin=>Col11(5=), S=>X11(3), Cout=>X12(4));

fa212: Full_Adder port map (A=>C(11), B=>Col12(1), Cin=>Col12(2), S=>X12(1), Cout=>X13(2));
fa212ii: Full_Adder port map (A=>Col12(3), B=>Col12(4), Cin=>W12(1), S=>X12(3), Cout=>X13(3));

fa213: Full_Adder port map (A=>C(12), B=>Col13(1), Cin=>Col13(2), S=>X13(1), Cout=>X14(1));
```

```
213     --stage4
214     ha33: HALF_ADDER port map (A=>C(2), B=>Col3(1), S=>Z3(1), C=>Z4(2));
215
216     fa34: Full_Adder port map (A=>X4(1), B=>Col4(2), Cin=>Col4(3), S=>Z4(1), Cout=>Z5(2));
217     fa35: Full_Adder port map (A=>X5(1), B=>X5(2), Cin=>X5(3), S=>Z5(1), Cout=>Z6(2));
218     fa36: Full_Adder port map (A=>X6(1), B=>X6(2), Cin=>X6(3), S=>Z6(1), Cout=>Z7(2));
219     fa37: Full_Adder port map (A=>X7(1), B=>X7(2), Cin=>X7(3), S=>Z7(1), Cout=>Z8(2));
220     fa38: Full_Adder port map (A=>X8(1), B=>X8(2), Cin=>X8(3), S=>Z8(1), Cout=>Z9(2));
221     fa39: Full_Adder port map (A=>X9(1), B=>X9(2), Cin=>X9(3), S=>Z9(1), Cout=>Z10(2));
222     fa310: Full_Adder port map (A=>X10(1), B=>X10(2), Cin=>X10(3), S=>Z10(1), Cout=>Z11(2));
223     fa311: Full_Adder port map (A=>X11(1), B=>X11(2), Cin=>X11(3), S=>Z11(1), Cout=>Z12(2));
224     fa312: Full_Adder port map (A=>X12(1), B=>X12(2), Cin=>X12(3), S=>Z12(1), Cout=>Z13(2));
225     fa313: Full_Adder port map (A=>X13(1), B=>X13(2), Cin=>X13(3), S=>Z13(1), Cout=>Z14(2));
226     fa314: Full_Adder port map (A=>X14(1), B=>C(13), Cin=>Col14(1), S=>Z14(1), Cout=>Z15(1));
227
228     --stage5
229     ha42: HALF_ADDER port map (A=>C(1), B=>Col2(1), S=>Y2(1), C=>Y3(2));
230
231     fa43: Full_Adder port map (A=>Z3(1), B=>Col3(2), Cin=>Col3(3), S=>Y3(1), Cout=>Y4(2));
232     fa44: Full_Adder port map (A=>Z4(1), B=>Z4(2), Cin=>Col4(4), S=>Y4(1), Cout=>Y5(2));
233     fa45: Full_Adder port map (A=>Z5(1), B=>Z5(2), Cin=>Col5(5), S=>Y5(1), Cout=>Y6(2));
234     fa46: Full_Adder port map (A=>Z6(1), B=>Z6(2), Cin=>X6(4), S=>Y6(1), Cout=>Y7(2));
235     fa47: Full_Adder port map (A=>Z7(1), B=>Z7(2), Cin=>X7(4), S=>Y7(1), Cout=>Y8(2));
236     fa48: Full_Adder port map (A=>Z8(1), B=>Z8(2), Cin=>X8(4), S=>Y8(1), Cout=>Y9(2));
237     fa49: Full_Adder port map (A=>Z9(1), B=>Z9(2), Cin=>X9(4), S=>Y9(1), Cout=>Y10(2));
238     fa410: Full_Adder port map (A=>Z10(1), B=>Z10(2), Cin=>X10(4), S=>Y10(1), Cout=>Y11(2));
239     fa411: Full_Adder port map (A=>Z11(1), B=>Z11(2), Cin=>X11(4), S=>Y11(1), Cout=>Y12(2));
240     fa412: Full_Adder port map (A=>Z12(1), B=>Z12(2), Cin=>X12(4), S=>Y12(1), Cout=>Y13(2));
241     fa413: Full_Adder port map (A=>Z13(1), B=>Z13(2), Cin=>Col13(3), S=>Y13(1), Cout=>Y14(2));
242     fa414: Full_Adder port map (A=>Z14(1), B=>Z14(2), Cin=>Col14(2), S=>Y14(1), Cout=>Y15(2));
243     fa415: Full_Adder port map (A=>Z15(1), B=>Col15(1), Cin=>C(14), S=>Y15(1), Cout=>Y16(1));
```

c. At finally 16-bit logarithmic adder is used from assignment-4 to final answer. From DUT final product and Carry is output.

```
246     --Full adder of 16 bit
247     FA16: Full_Adder16 port map (
248             in_A(0)=>C(0),   in_A(1)=>Y2(1),  in_A(2)=>Y3(1), in_A(3)=>Y4(1), in_A(4)=>Y5(1), in_A(5)=>Y6(1), in_A(6)=>Y
249             in_B(0)=>Col1(1),in_B(1)=>Col2(2),in_B(2)=>Y3(2), in_B(3)=>Y4(2), in_B(4)=>Y5(2), in_B(5)=>Y6(2), in_B(6)=>Y
250             Cin=>'0', Sum=>Product, Cout=>Cout);
251
252
253     end Struct;
254
```

There are 4 files in the project made in Quartus-MAC, DUT, Testbench and Gates
In MAC, structural code is written, DUT is for test the device.
Tracefile.txt (in which input combinations are written) is generated using python *tracefile.py*
I am using 20 random combinations of inputs for verification and a max value.

For an input—10101100000001111010001110011110 and output-01010100001010010
In simulation it starts at **247 ns** and settles at **249.8 ns**, so **2.8 ns** delay to generate final output.

```
 5  library ieee;
 6  use ieee.std_logic_1164.all;
 7  entity DUT is --ABC , (8,8,16)
 8     port(input_vector: in std_logic_vector(31 downto 0);
 9          output_vector: out std_logic_vector(16 downto 0)); --prod
10  end entity;
11
12  architecture DutWrap of DUT is
13     component MAC is
14      port (A, B: in std_logic_vector(7 downto 0); C: in std_logic_vector(15 downto 0);
15          Product: out std_logic_vector(15 downto 0); Cout: out std_logic);
16     end component;
17  begin
18
19     -- input/output vector element ordering is critical,
20     -- and must match the ordering in the trace file!
21     add_instance: MAC
22          port map (
23
24                A => input_vector(31 downto 24),
25                B => input_vector(23 downto 16),
26                C  => input_vector(15 downto 0),
27                Cout =>  output_vector(16),
28                Product => output_vector(15 downto 0)
29
30                );  -- order of inputs ABC
31
```

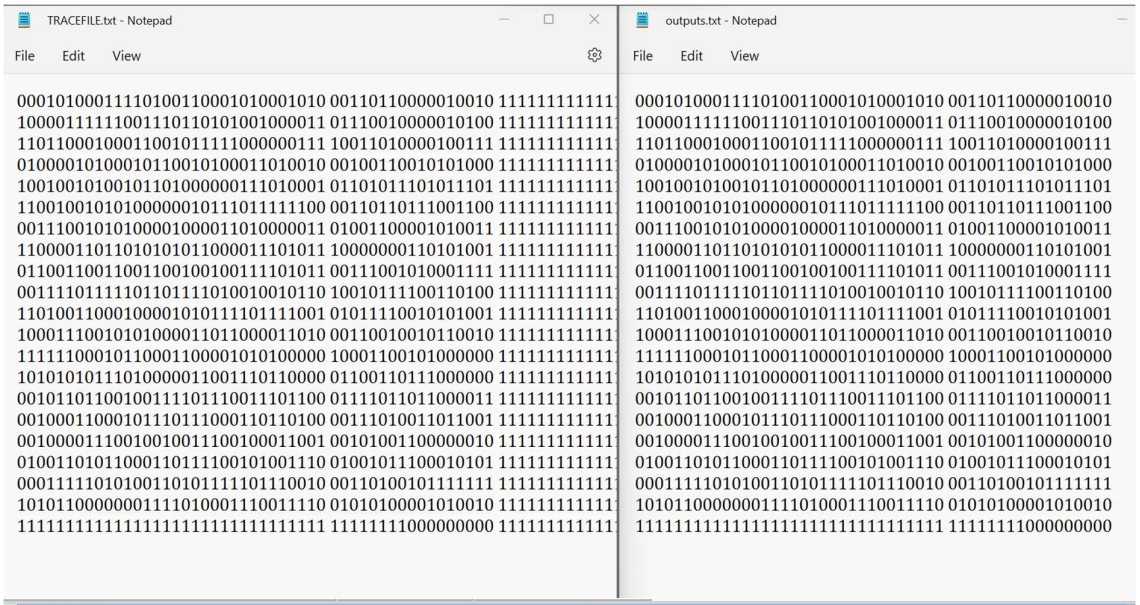
DUT code

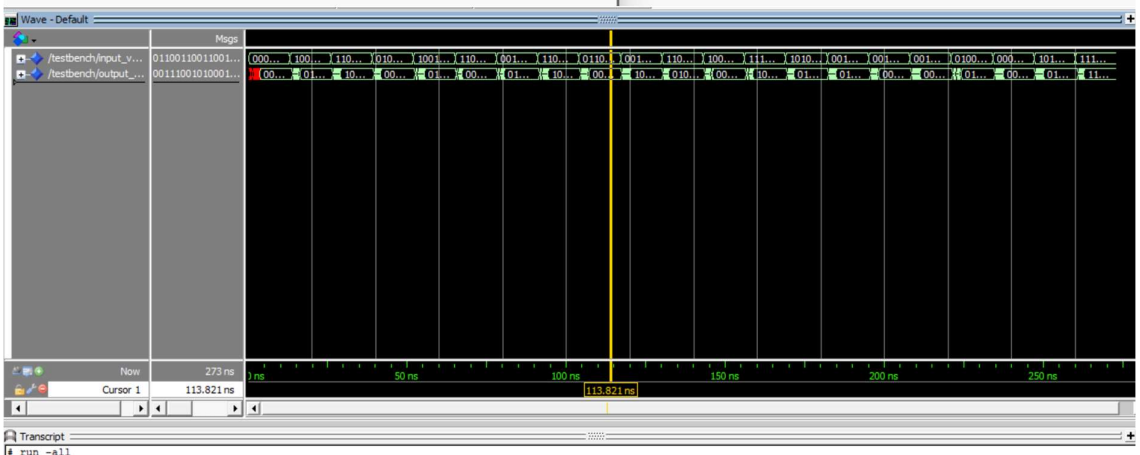
```
112        -- wait for the circuit to settle
113        wait for 9 ns;
114
115        -- check output.
116             output_comp_var := (to_std_logic_vector(output_mask_var) and
117                  (output_vector xor to_std_logic_vector(output_vector_var)));
118        if (output_comp_var  /= ZZZZ) then
119                write(OUTPUT_LINE,to_string("ERROR: line "));
120                write(OUTPUT_LINE, LINE_COUNT);
121                writeline(OUTFILE, OUTPUT_LINE);
122                err_flag := true;
123             end if;
124
125             write(OUTPUT_LINE, to_bit_vector(input_vector));
126             write(OUTPUT_LINE, to_string(" "));
127             write(OUTPUT_LINE, to_bit_vector(output_vector));
128             writeline(OUTFILE, OUTPUT_LINE);
129
130        -- advance time by 4 ns.
131        wait for 4 ns;
132       end loop;
133
134       assert (err_flag) report "SUCCESS, all tests passed." severity note;
135       assert (not err_flag) report "FAILURE, some tests failed." severity error;
136
137       wait;
138     end process;
139
140     dut_instance: DUT
141         port map(input_vector => input_vector, output_vector => output_vector);
```

Testbench code

Output file is generated at simulation/modelsim/output.txt

All the cases passed for the given random combination of input with no error.





RTL simulation