

گزارش تکلیف سری یک vlsi معماری کامپیوتر

آنوشا شریعتی 9923041

قسمت الف:

```
1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3
4 entity fulladder is
5     Port ( a : in  STD_LOGIC;
6           b : in  STD_LOGIC;
7           cin : in  STD_LOGIC;
8           s : out  STD_LOGIC;
9           cout : out  STD_LOGIC);
10 end fulladder;
11
12 architecture Behavioral of fulladder is
13
14 begin
15
16 s <= a xor b xor cin ;
17 cout <= (a or b) and (a or cin) and (b or cin) ;
18
19 end Behavioral;
20
21
```

برای طراحی این قسمت به دو ماژول فول ادر و جمع کننده 6 بیتی باینری نیاز داشتیم که با استفاده از گیت های منطقی آنها به صورت زیر طراحی شد.

کد وی اچ دی ال ماژول فول ادر:

کد وی اچ دی ال ماژول ادر 6 بیتی(به طور خلاصه):

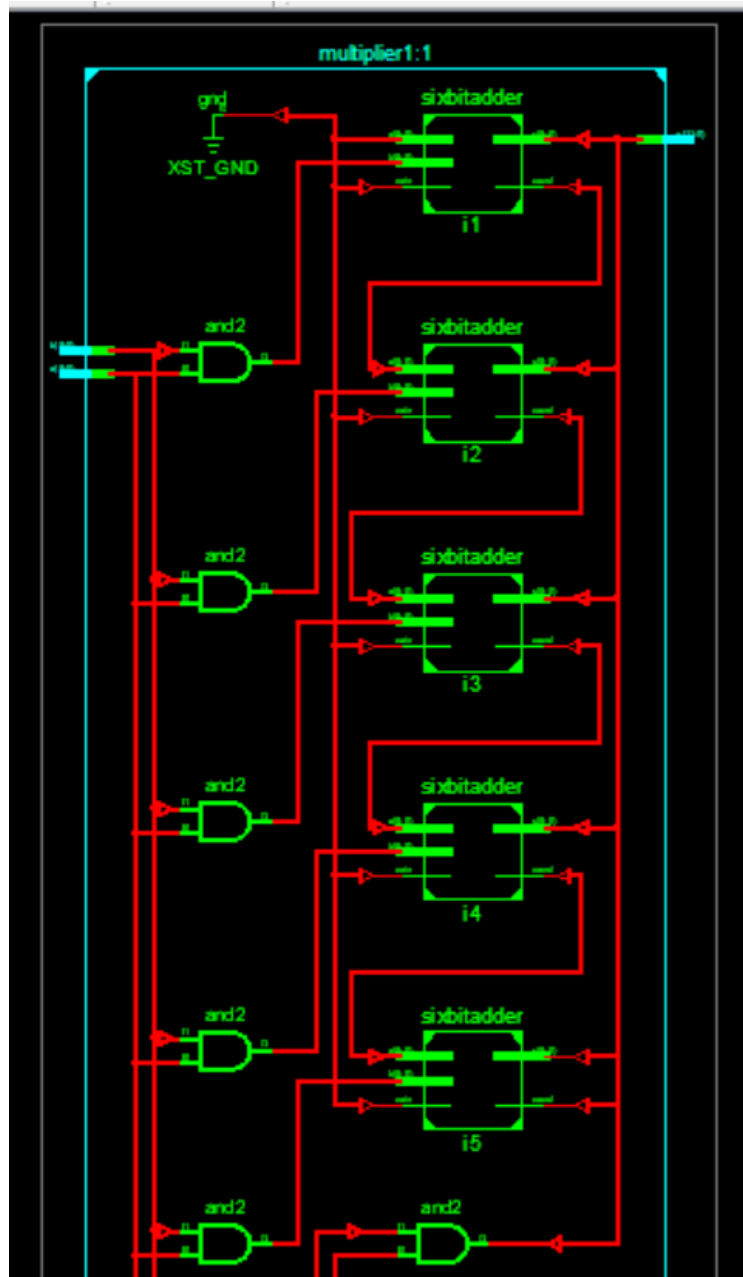
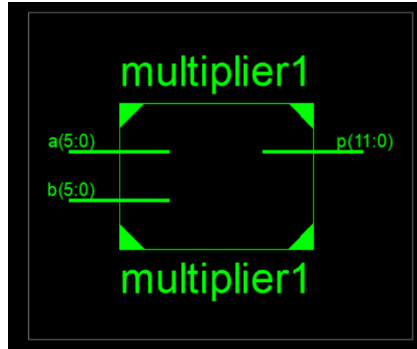
```
1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3
4 entity sixbitadder is
5     Port ( a : in  STD_LOGIC_VECTOR (5 downto 0);
6           b : in  STD_LOGIC_VECTOR (5 downto 0);
7           s : out  STD_LOGIC_VECTOR (5 downto 0);
8           caout : out  STD_LOGIC;
9           cain : in  STD_LOGIC);
10 end sixbitadder;
11
12 architecture Behavioral of sixbitadder is
13
14 component fulladder is
15     Port ( a : in  STD_LOGIC;
16           b : in  STD_LOGIC;
17           cin : in  STD_LOGIC;
18           s : out  STD_LOGIC;
19           cout : out  STD_LOGIC);
20 end component;
21
22 signal c : STD_LOGIC_VECTOR (4 downto 0);
23
24 begin
25
26 u0 : fulladder port map (
27     a=>a(0),
28     b=>b(0),
29     cin=>cain,
30     s=>s(0),
31     cout=>c(0)
32 );
33
34 u1 : fulladder port map (
35     a=>a(1),
36     b=>b(1),
37     cin=>c(0),
38     s=>s(1),
39     cout=>c(1)
40 );
41
42 u2 : fulladder port map (
43     a=>a(2),
44     b=>b(2),
45     cin=>c(1),
46     s=>s(2),
47     cout=>c(2)
48 );
```

کد اصلی (به طور خلاصه) :

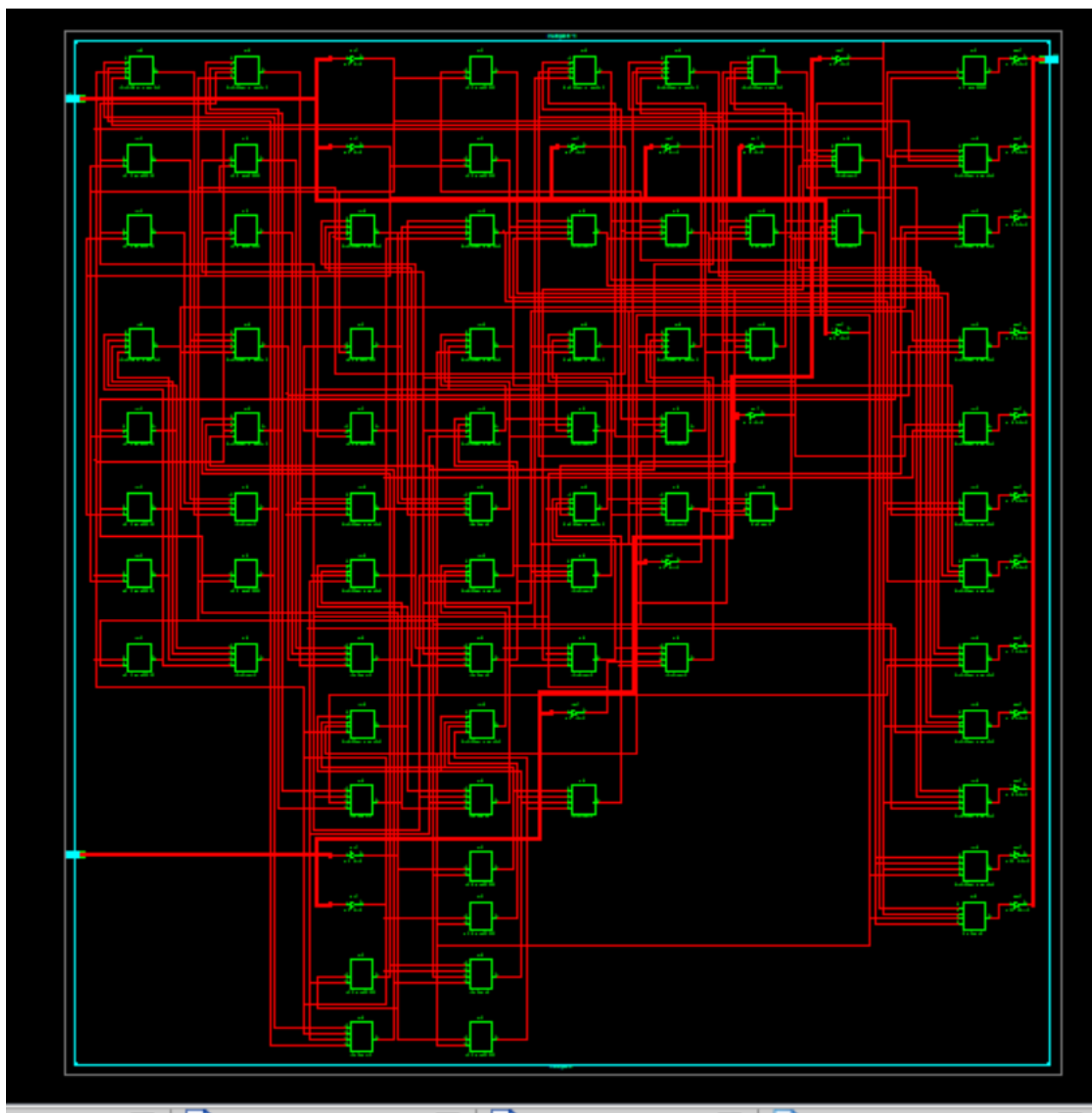
با اضافه کردن ماژول های طراحی شده در قسمت قبل و دستور پورت مپ طبق شکل ورودی ها و خروجی ها را اختصاص می دهیم.

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity multiplier1 is
5      Port ( a : in  STD_LOGIC_VECTOR (5 downto 0);
6            b : in  STD_LOGIC_VECTOR (5 downto 0);
7            p : out STD_LOGIC_VECTOR (11 downto 0));
8  end multiplier1;
9
10 architecture Behavioral of multiplier1 is
11
12     component sixbitadder is
13         Port ( a : in  STD_LOGIC_VECTOR (5 downto 0);
14               b : in  STD_LOGIC_VECTOR (5 downto 0);
15               s : out  STD_LOGIC_VECTOR (5 downto 0);
16               caout : out  STD_LOGIC;
17               cain : in  STD_LOGIC);
18     end component;
19
20     signal x1 : STD_LOGIC_VECTOR (5 downto 0);
21     signal x2 : STD_LOGIC_VECTOR (5 downto 0);
22     signal x3 : STD_LOGIC_VECTOR (5 downto 0);
23     signal x4 : STD_LOGIC_VECTOR (5 downto 0);
24     signal x5 : STD_LOGIC_VECTOR (5 downto 0);
25
26     signal y1 : STD_LOGIC_VECTOR (5 downto 0);
27     signal y2 : STD_LOGIC_VECTOR (5 downto 0);
28     signal y3 : STD_LOGIC_VECTOR (5 downto 0);
29     signal y4 : STD_LOGIC_VECTOR (5 downto 0);
30     signal y5 : STD_LOGIC_VECTOR (5 downto 0);
31
32     signal s1 : STD_LOGIC_VECTOR (5 downto 0);
33     signal s2 : STD_LOGIC_VECTOR (5 downto 0);
34     signal s3 : STD_LOGIC_VECTOR (5 downto 0);
35     signal s4 : STD_LOGIC_VECTOR (5 downto 0);
36     signal s5 : STD_LOGIC_VECTOR (5 downto 0);
37
38     signal cout : STD_LOGIC_VECTOR (5 downto 1);
39
40 begin
41
42     --first digit
43     p(0) <= a(0) and b(0);
44     x1(0) <= a(1) and b(0);
45     x1(1) <= a(2) and b(0);
46     x1(2) <= a(3) and b(0);
47     x1(3) <= a(4) and b(0);
48     x1(4) <= a(5) and b(0);
49     x1(5) <= '0';
50
51     y1(0) <= a(0) and b(1);
52     y1(1) <= a(1) and b(1);
53     y1(2) <= a(2) and b(1);
54     y1(3) <= a(3) and b(1);
55     y1(4) <= a(4) and b(1);
56     y1(5) <= a(5) and b(1);
57
58     i1 : sixbitadder port map (
59         a=>x1,
60         b=>y1,
61         cain=>'0',
62         s=>s1,
63         caout=>cout(1)
64     );
65
66     --second digit
67     p(1) <= s1(0);
68     x2(0) <= s1(1);
69     x2(1) <= s1(2);
70     x2(2) <= s1(3);
71     x2(3) <= s1(4);
72     x2(4) <= s1(5);
73     x2(5) <= cout(1);
74
75     y2(0) <= a(0) and b(2);
76     y2(1) <= a(1) and b(2);
77     y2(2) <= a(2) and b(2);
78     y2(3) <= a(3) and b(2);
79     y2(4) <= a(4) and b(2);
80     y2(5) <= a(5) and b(2);
81
82     i2 : sixbitadder port map (
83         a=>x2,
84         b=>y2,
85         cain=>'0',
86         s=>s2,
87         caout=>cout(2)
88     );
89
90     y3(0) <= a(0) and b(3);
91     y3(1) <= a(1) and b(3);
92     y3(2) <= a(2) and b(3);
93     y3(3) <= a(3) and b(3);
94     y3(4) <= a(4) and b(3);
95     y3(5) <= a(5) and b(3);
96
97     i3 : sixbitadder port map (
98         a=>x3,
99         b=>y3,
100        cain=>'0',
101        s=>s3,
102        caout=>cout(3)
103    );
104
105     y4(0) <= a(0) and b(4);
106     y4(1) <= a(1) and b(4);
107     y4(2) <= a(2) and b(4);
108     y4(3) <= a(3) and b(4);
109     y4(4) <= a(4) and b(4);
110     y4(5) <= a(5) and b(4);
111
112     i4 : sixbitadder port map (
113         a=>x4,
114         b=>y4,
115         cain=>'0',
116         s=>s4,
117         caout=>cout(4)
118     );
119
120     y5(0) <= a(0) and b(5);
121     y5(1) <= a(1) and b(5);
122     y5(2) <= a(2) and b(5);
123     y5(3) <= a(3) and b(5);
124     y5(4) <= a(4) and b(5);
125     y5(5) <= a(5) and b(5);
126
127     i5 : sixbitadder port map (
128         a=>x5,
129         b=>y5,
130         cain=>'0',
131         s=>s5,
132         caout=>cout(5)
133     );
134
135     p(5) <= s5(0);
136     p(6) <= s5(1);
137     p(7) <= s5(2);
138     p(8) <= s5(3);
139     p(9) <= s5(4);
140     p(10) <= s5(5);
141     p(11) <= cout(5);
142
143 end Behavioral;
```

آر تی ال شماتیک:



تکنولوژی شماتیک :

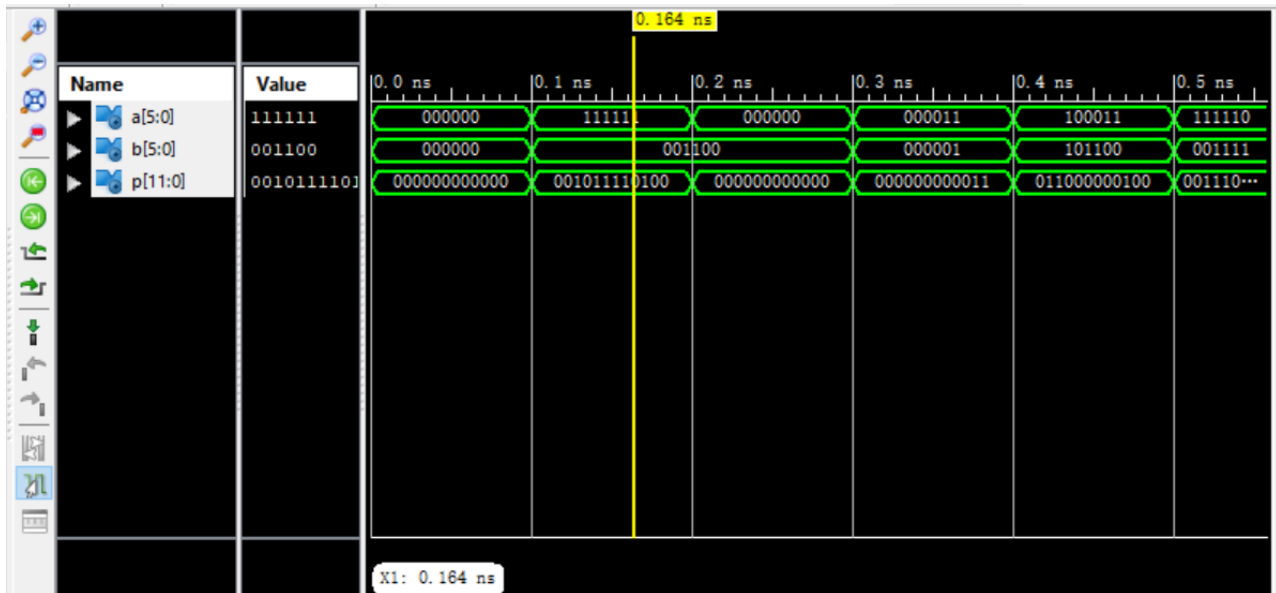


کد و شبیه سازی تست بنچ:

```

39  -- Stimulus process
40  stim_proc: process
41  begin
42      wait for 100 ps;
43      a <= "111111";
44      b <= "001100";
45      wait for 100 ps;
46      a <= "000000";
47      b <= "001100";
48      wait for 100 ps;
49      a <= "000011";
50      b <= "000001";
51      wait for 100 ps;
52      a <= "100011";
53      b <= "101100";
54      wait for 100 ps;
55      a <= "111110";
56      b <= "001111";
57      wait for 100 ps;
58      a <= "111111";
59      b <= "111111";
60      wait for 100 ps;
61      a <= "110101";
62      b <= "001101";
63      wait for 100 ps;

```



قسمت ب:

برای طراحی قسمت ب به دو ماژول فول ادر و هف ادر نیاز داشتیم. کد آن ها با استفاده از گیت های منطقی در زیر آمده است.

کد وی اچ دی ال ماژول هف ادر:

```
1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3
4
5 entity ha is
6     Port ( a : in  STD_LOGIC;
7           b : in  STD_LOGIC;
8           s : out  STD_LOGIC;
9           c : out  STD_LOGIC);
10 end ha;
11
12 architecture Behavioral of ha is
13
14 begin
15     c <= a and b;
16     s <= a xor b;
17 end Behavioral;
```

کد وی اچ دی ال ماژول فول ادر:

```
1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3
4
5
6 entity fa is
7     Port ( x : in  STD_LOGIC;
8           y : in  STD_LOGIC;
9           z : in  STD_LOGIC;
10          s : out  STD_LOGIC;
11          c : out  STD_LOGIC);
12 end fa;
13
14 architecture Behavioral of fa is
15
16 begin
17
18     s <= x xor y xor z;
19     c <= (x or y) and (x or z) and (y or z) ;
20
21 end Behavioral;
```

کد اصلی (خلاصه شده):

با اضافه کردن مازول های تعریف شده در قسمت قبل توسط دستور پورت مپ و تعریف کردن سیگنال های x, y, z برای ربط دادن خروجی هر طبقه به ورودی طبقه بعد میتوان مدار داده شده در صورت سوال را طراحی

کرد. به عنوان نمونه کد طبقه اول و آخر در اینجا آورده شده است.

```

5  entity multiplier2 is
6      Port ( a : in  STD_LOGIC_VECTOR (5 downto 0);
7            b : in  STD_LOGIC_VECTOR (5 downto 0);
8            p : out STD_LOGIC_VECTOR (11 downto 0));
9  end multiplier2;
10
11 architecture Behavioral of multiplier2 is
12
13  signal x1 : STD_LOGIC_VECTOR (5 downto 1);
14  signal x2 : STD_LOGIC_VECTOR (6 downto 1);
15  signal x3 : STD_LOGIC_VECTOR (6 downto 1);
16  signal x4 : STD_LOGIC_VECTOR (6 downto 1);
17  signal x5 : STD_LOGIC_VECTOR (6 downto 1);
18  signal x6 : STD_LOGIC_VECTOR (5 downto 2);
19
20  signal y1 : STD_LOGIC_VECTOR (6 downto 1);
21  signal y2 : STD_LOGIC_VECTOR (5 downto 1);
22  signal y3 : STD_LOGIC_VECTOR (5 downto 1);
23  signal y4 : STD_LOGIC_VECTOR (5 downto 1);
24  signal y5 : STD_LOGIC_VECTOR (5 downto 1);
25  signal y6 : STD_LOGIC_VECTOR (5 downto 1);
26
27  signal z2 : STD_LOGIC_VECTOR (5 downto 2);
28  signal z3 : STD_LOGIC_VECTOR (5 downto 2);
29  signal z4 : STD_LOGIC_VECTOR (5 downto 2);
30  signal z5 : STD_LOGIC_VECTOR (5 downto 2);
31  signal z6 : STD_LOGIC_VECTOR (5 downto 2);
32
33  component ha is
34      Port ( a : in  STD_LOGIC;
35            b : in  STD_LOGIC;
36            s : out STD_LOGIC;
37            c : out STD_LOGIC);
38  end component;
39
40  component fa is
41      Port ( x : in  STD_LOGIC;
42            y : in  STD_LOGIC;
43            z : in  STD_LOGIC;
44            s : out STD_LOGIC;
45            c : out STD_LOGIC);
46  end component;
47
48  begin
49
50  --first stage
51
52  p(0) <= a(0) and b(0);
53  x1(1) <= a(1) and b(0);
54  x1(2) <= a(2) and b(0);
55  x1(3) <= a(3) and b(0);
56  x1(4) <= a(4) and b(0);
57  x1(5) <= a(5) and b(0);

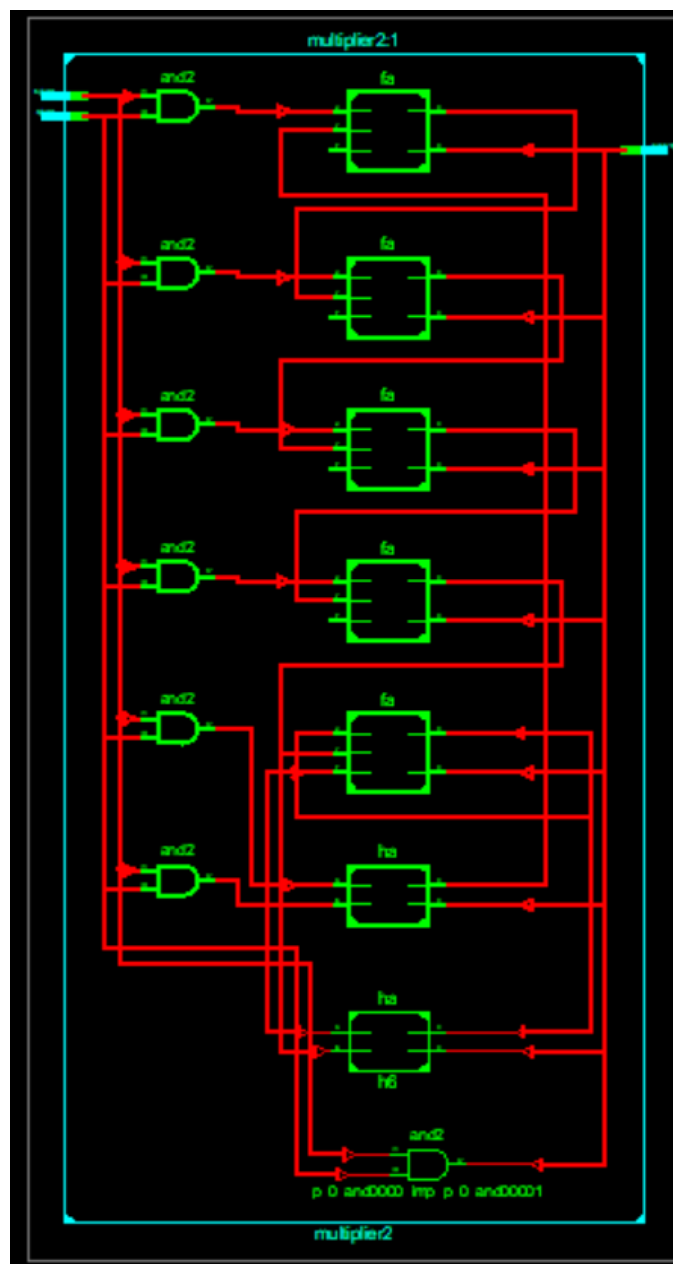
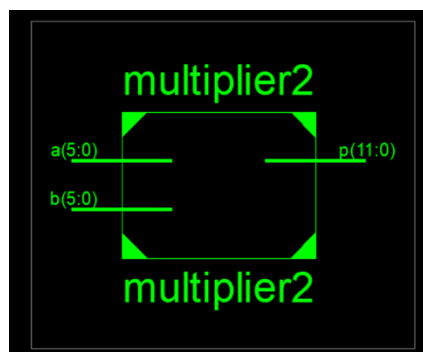
```

```

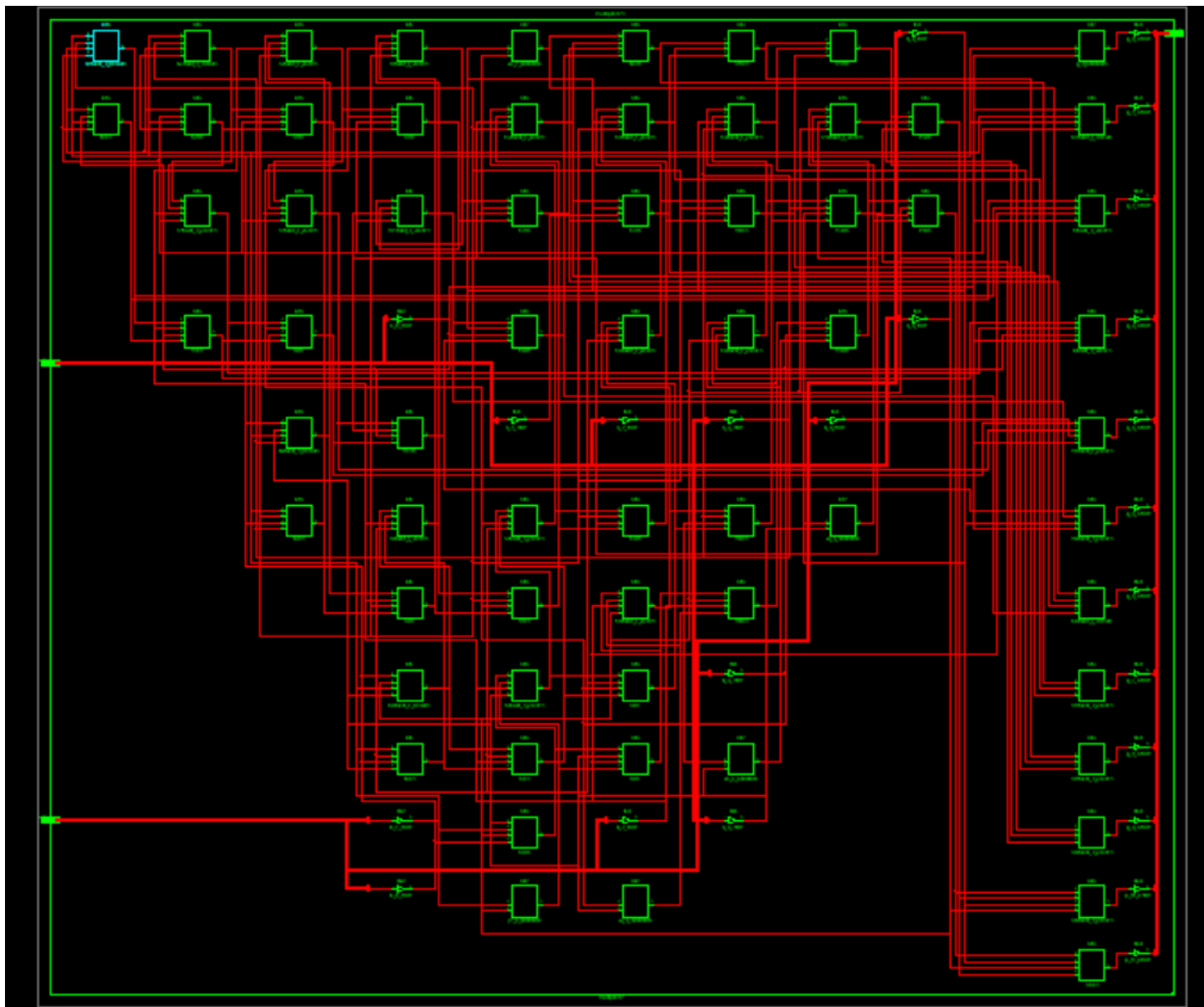
60
61  y1(1) <= a(0) and b(1);
62  y1(2) <= a(1) and b(1);
63  y1(3) <= a(2) and b(1);
64  y1(4) <= a(3) and b(1);
65  y1(5) <= a(4) and b(1);
66
67  y1(6) <= a(5) and b(1);
68
69  h1 : ha port map (
70      a => y1(1),
71      b => x1(1),
72      c => y2(1),
73      s => p(1)
74  );
75
76  h2 : ha port map (
77      a => y1(2),
78      b => x1(2),
79      c => y2(2),
80      s => z2(2)
81  );
82
83  h3 : ha port map (
84      a => y1(3),
85      b => x1(3),
86      c => y2(3),
87      s => z2(3)
88  );
89
90  h4 : ha port map (
91      a => y1(4),
92      b => x1(4),
93      c => y2(4),
94      s => z2(4)
95  );
96
97  h5 : ha port map (
98      a => y1(5),
99      b => x1(5),
100     c => y2(5),
101     s => z2(5)
102 );
103
104 --sixth stage
105
106 h6 : ha port map (
107     a => z6(2),
108     b => y6(1),
109     c => x6(2),
110     s => p(6)
111 );
112
113 f21 : fa port map (
114     x => x6(2),
115     y => y6(2),
116     z => z6(3),
117     s => p(7),
118     c => x6(3)
119 );
120
121 f22 : fa port map (
122     x => x6(3),
123     y => y6(3),
124     z => z6(4),
125     s => p(8),
126     c => x6(4)
127 );
128
129 f23 : fa port map (
130     x => x6(4),
131     y => y6(4),
132     z => z6(5),
133     s => p(9),
134     c => x6(5)
135 );
136
137 f24 : fa port map (
138     x => x6(5),
139     y => y6(5),
140     z => x5(6),
141     s => p(10),
142     c => p(11)
143 );
144
145 end Behavioral;

```

آر تی ال شماتیک:

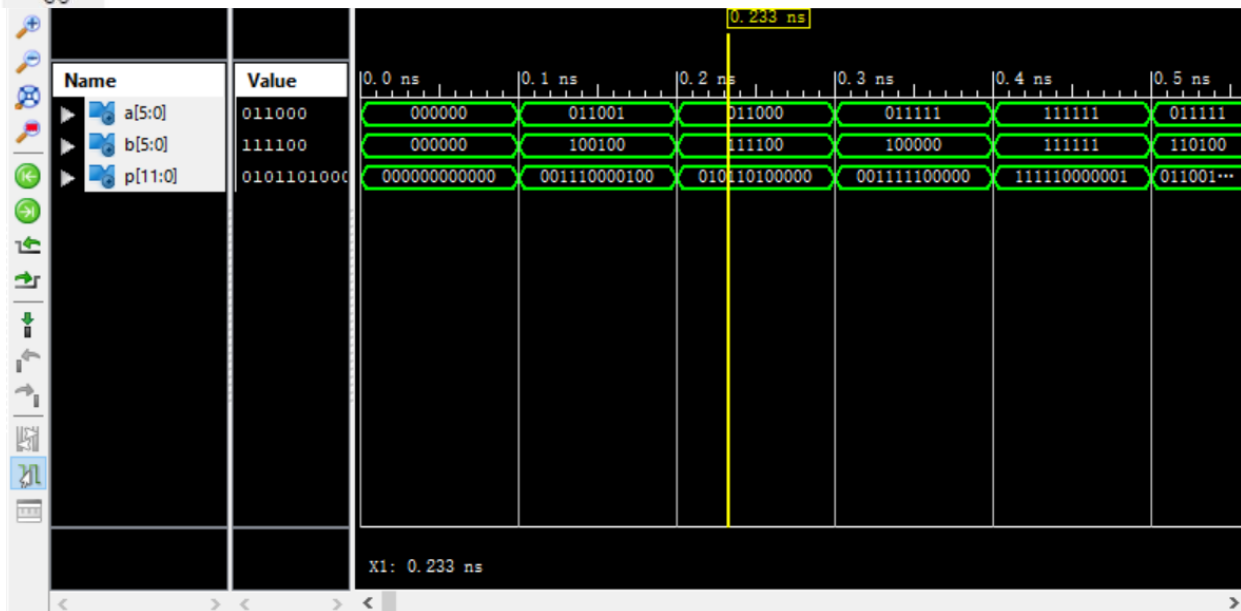


تکنولوژی مپ:



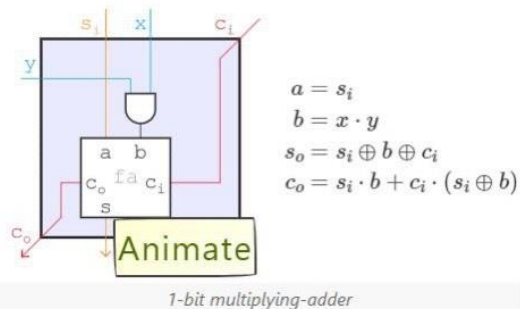
کد تست بنچ و شبیه سازی:

```
35
36 stim_proc: process
37 begin
38
39     wait for 100 ps;
40     a <= "011001";
41     b <= "100100";
42     wait for 100 ps;
43     a <= "011000";
44     b <= "111100";
45     wait for 100 ps;
46     a <= "011111";
47     b <= "100000";
48     wait for 100 ps;
49     a <= "111111";
50     b <= "111111";
51     wait for 100 ps;
52     a <= "011111";
53     b <= "110100";
54     wait for 100 ps;
55
56     wait;
57 end process;
58
59 END;
60
```



قسمت ج :

برای طراحی قسمت ج به دو ماژول فول ادر و مالتیپلایینگ ادر نیاز داشتیم. بلوک مالتیپلایینگ ادر با توجه به روابط زیر و با استفاده از فول ادر طراحی شد.



ماژول فول ادر:

```

1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity fa is
5      Port ( a : in  STD_LOGIC;
6            b : in  STD_LOGIC;
7            cin : in  STD_LOGIC;
8            s : out  STD_LOGIC;
9            cout : out  STD_LOGIC);
10 end fa;
11
12 architecture Behavioral of fa is
13
14 begin
15
16 s <= a xor b xor cin;
17 cout <= (a or b) and (a or cin) and (b or cin);
18
19
20 end Behavioral;
21

```

ماژول مالتیپلایر ادر:

```

1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity ma is
5      Port ( x : in  STD_LOGIC;
6            y : in  STD_LOGIC;
7            si : in  STD_LOGIC;
8            ci : in  STD_LOGIC;
9            co : out  STD_LOGIC;
10           so : out  STD_LOGIC);
11 end ma;
12
13 architecture Behavioral of ma is
14
15 component fa is
16     Port ( a : in  STD_LOGIC;
17           b : in  STD_LOGIC;
18           cin : in  STD_LOGIC;
19           s : out  STD_LOGIC;
20           cout : out  STD_LOGIC);
21 end component;
22
23 signal z : STD_LOGIC ;
24
25
26
27
28 ul: fa port map(
29     a => si ,
30     b => z ,
31     cin => ci,
32     s => so ,
33     cout => co
34 );
35
36
37
38
39 end Behavioral;
40

```

```

1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity multiplier3 is
5      Port ( x : in  STD_LOGIC_VECTOR (5 downto 0);
6            y : in  STD_LOGIC_VECTOR (5 downto 0);
7            p : out STD_LOGIC_VECTOR (12 downto 0));
8  end multiplier3;
9
10 architecture Behavioral of multiplier3 is
11
12 signal s0 : STD_LOGIC_VECTOR (5 downto 1);
13 signal s1 : STD_LOGIC_VECTOR (5 downto 1);
14 signal s2 : STD_LOGIC_VECTOR (5 downto 1);
15 signal s3 : STD_LOGIC_VECTOR (5 downto 1);
16 signal s4 : STD_LOGIC_VECTOR (5 downto 1);
17 signal s5 : STD_LOGIC_VECTOR (5 downto 1);
18
19 signal c0 : STD_LOGIC_VECTOR (5 downto 0);
20 signal c1 : STD_LOGIC_VECTOR (5 downto 0);
21 signal c2 : STD_LOGIC_VECTOR (5 downto 0);
22 signal c3 : STD_LOGIC_VECTOR (5 downto 0);
23 signal c4 : STD_LOGIC_VECTOR (5 downto 0);
24 signal c5 : STD_LOGIC_VECTOR (5 downto 0);
25 signal c6 : STD_LOGIC_VECTOR (4 downto 0);
26
27 component ma is
28     Port ( x : in  STD_LOGIC;
29           y : in  STD_LOGIC;
30           si : in  STD_LOGIC;
31           ci : in  STD_LOGIC;
32           co : out STD_LOGIC;
33           so : out STD_LOGIC);
34 end component;
35
36 begin
37
38 --first stage
39
40 ma00 : ma port map(
41     x => x(0),
42     y => y(0),
43     si => '0',
44     ci => '0',
45     co => c0(0),
46     so => p(0)
47 );
48
49 ma01 : ma port map(
50     x => x(1),
51     y => y(0),
52     si => '0',
53     ci => '0',
54     co => c0(1),
55     so => s0(1)
56 );
57 ma02 : ma port map(
58     x => x(2),
59     y => y(0),
60     si => '0',
61     ci => '0',
62     co => c0(2),
63     so => s0(2)
64 );
65 ma03 : ma port map(
66     x => x(3),
67     y => y(0),
68     si => '0',
69     ci => '0',
70     co => c0(3),
71     so => s0(3)
72 );
73 ma04 : ma port map(
74     x => x(4),
75     y => y(0),
76     si => '0',
77     ci => '0',
78     co => c0(4),
79     so => s0(4)
80 );
81 ma05 : ma port map(
82     x => x(5),
83     y => y(0),
84     si => '0',
85     ci => '0',
86     co => c0(5),
87     so => s0(5)
88 );
89

```

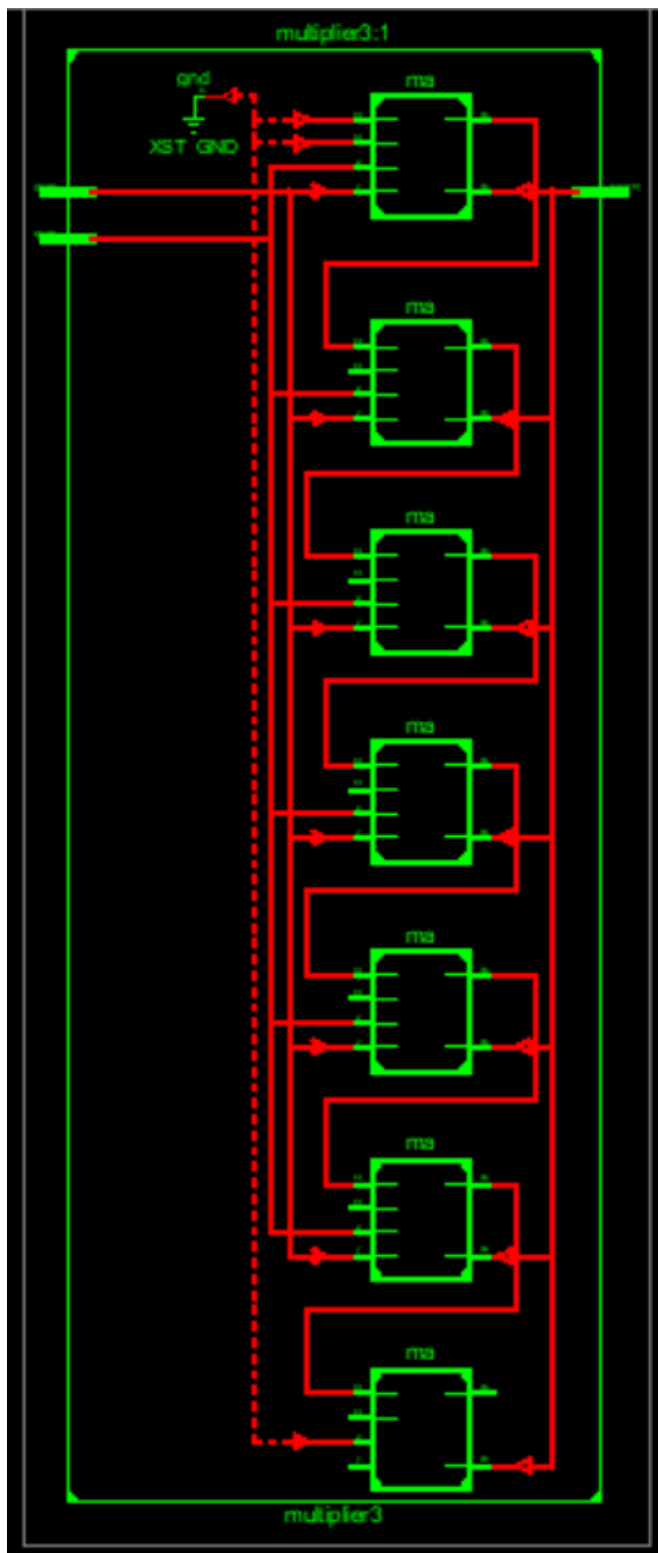
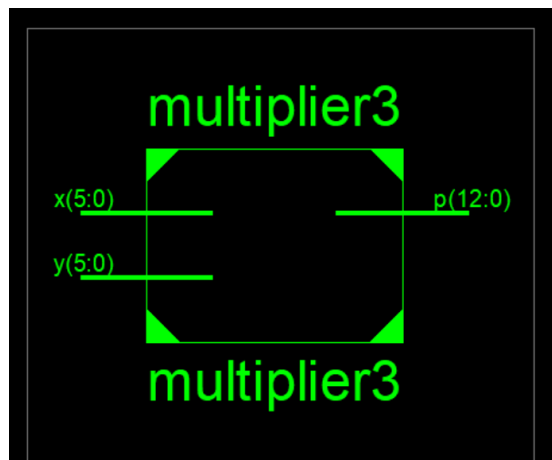
كد اصلی (خلاصه شده طبقه اول و آخر) :

```

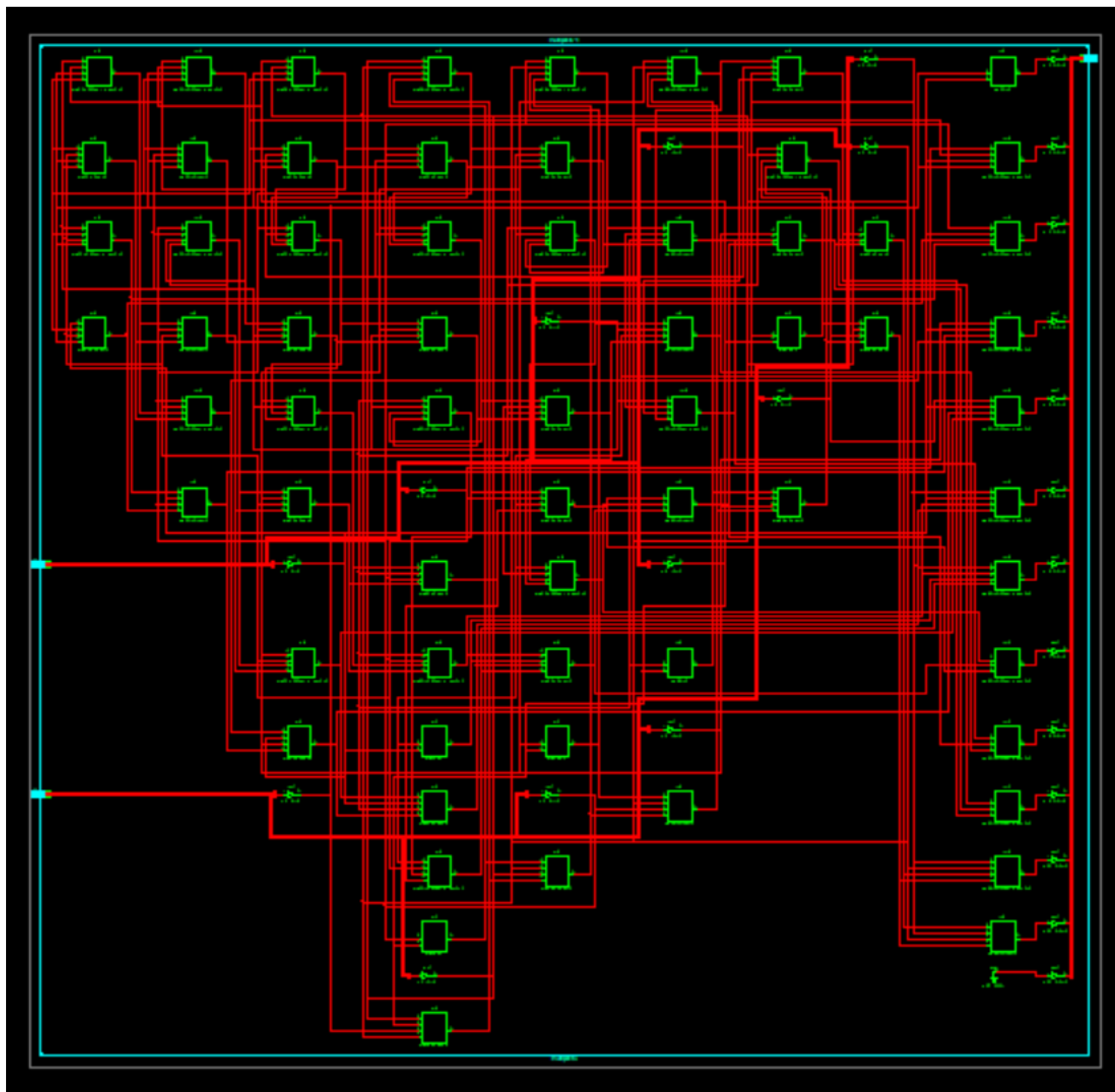
350 --seventh stage
351
352 ma60 : ma port map(
353     x => '0',
354     y => '1',
355     si => s5(1),
356     ci => c5(0),
357     co => c6(0),
358     so => p(6)
359 );
360
361 ma61 : ma port map(
362     x => c6(0),
363     y => '1',
364     si => s5(2),
365     ci => c5(1),
366     co => c6(1),
367     so => p(7)
368 );
369
370 ma62 : ma port map(
371     x => c6(1),
372     y => '1',
373     si => s5(3),
374     ci => c5(2),
375     co => c6(2),
376     so => p(8)
377 );
378
379 ma63 : ma port map(
380     x => c6(2),
381     y => '1',
382     si => s5(4),
383     ci => c5(3),
384     co => c6(3),
385     so => p(9)
386 );
387
388 ma64 : ma port map(
389     x => c6(3),
390     y => '1',
391     si => s5(5),
392     ci => c5(4),
393     co => c6(4),
394     so => p(10)
395 );
396
397 ma64 : ma port map(
398     x => c6(3),
399     y => '1',
400     si => s5(5),
401     ci => c5(4),
402     co => c6(4),
403     so => p(10)
404 );
405
406 ma65 : ma port map(
407     x => c6(4),
408     y => '1',
409     si => '0',
410     ci => c5(5),
411     co => p(12),
412     so => p(11)
413 );
414
415 end Behavioral;
416

```

آر تی ال شماتیک:



تکنولوژی مپ:



کد تست بنچ و شبیه سازی:

```

36
37 -- Stimulus process
38 stim_proc: process
39 begin
40     -- hold reset state for 100 ns.
41     wait for 100 ps;
42     x <= "110011";
43     y <= "100010";
44     wait for 100 ps;
45     x <= "111111";
46     y <= "100110";
47     wait for 100 ps;
48     x <= "000011";
49     y <= "101010";
50     wait for 100 ps;
51     x <= "111111";
52     y <= "111111";
53     wait for 100 ps;
54     x <= "100100";
55     y <= "101110";
56     wait for 100 ps;
57     x <= "111101";
58     y <= "100000";
59     wait for 100 ps;
60
61
62     wait;
63 end process;
64
65 END;
66

```

