گزارش تکلیف سری یک vlsi معماری کامپیوتر

آنوشا شریعتی 9923041

قسمت الف:

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:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity fulladder is
    Port ( a : in  STD_LOGIC;
           b : in  STD_LOGIC;
           cin : in  STD_LOGIC;
           s : out  STD_LOGIC;
           cout : out  STD_LOGIC);
end fulladder;

architecture Behavioral of fulladder is

begin

s <= a xor b xor cin ;
cout <= (a or b) and (a or cin) and (b or cin) ;

end Behavioral;
```

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

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity sixbitadder is
    Port ( a : in  STD_LOGIC_VECTOR (5 downto 0);
           b : in  STD_LOGIC_VECTOR (5 downto 0);
           s : out  STD_LOGIC_VECTOR (5 downto 0);
           caout : out  STD_LOGIC;
           cain : in  STD_LOGIC);
end sixbitadder;

architecture Behavioral of sixbitadder is

component fulladder is
    Port ( a : in  STD_LOGIC;
           b : in  STD_LOGIC;
           cin : in  STD_LOGIC;
           s : out  STD_LOGIC;
           cout : out  STD_LOGIC);
end component;

signal c : STD_LOGIC_VECTOR (4 downto 0);

begin

u0 : fulladder port map (
   a=>a(0),
   b=>b(0),
   cin=>cain,
   s=>s(0),
   cout=>c(0)
);

u1 : fulladder port map (
   a=>a(1),
   b=>b(1),
   cin=>c(0),
   s=>s(1),
   cout=>c(1)
);

u2 : fulladder port map (
   a=>a(2),
   b=>b(2),
   cin=>c(1),
   s=>s(2),
   cout=>c(2)
);
```

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 میدهیم.

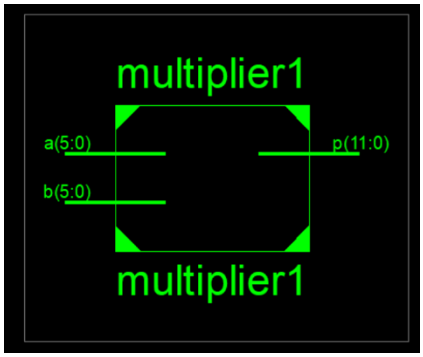
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity multiplierl is
    Port ( a : in  STD_LOGIC_VECTOR (5 downto 0);
           b : in  STD_LOGIC_VECTOR (5 downto 0);
           p : out  STD_LOGIC_VECTOR (11 downto 0));
end multiplierl;

architecture Behavioral of multiplierl is

component sixbitadder is
    Port ( a : in  STD_LOGIC_VECTOR (5 downto 0);
           b : in  STD_LOGIC_VECTOR (5 downto 0);
           s : out  STD_LOGIC_VECTOR (5 downto 0);
           caout : out  STD_LOGIC;
           cain : in  STD_LOGIC);
end component;

signal x1 : STD_LOGIC_VECTOR (5 downto 0);
signal x2 : STD_LOGIC_VECTOR (5 downto 0);
signal x3 : STD_LOGIC_VECTOR (5 downto 0);
signal x4 : STD_LOGIC_VECTOR (5 downto 0);
signal x5 : STD_LOGIC_VECTOR (5 downto 0);
```

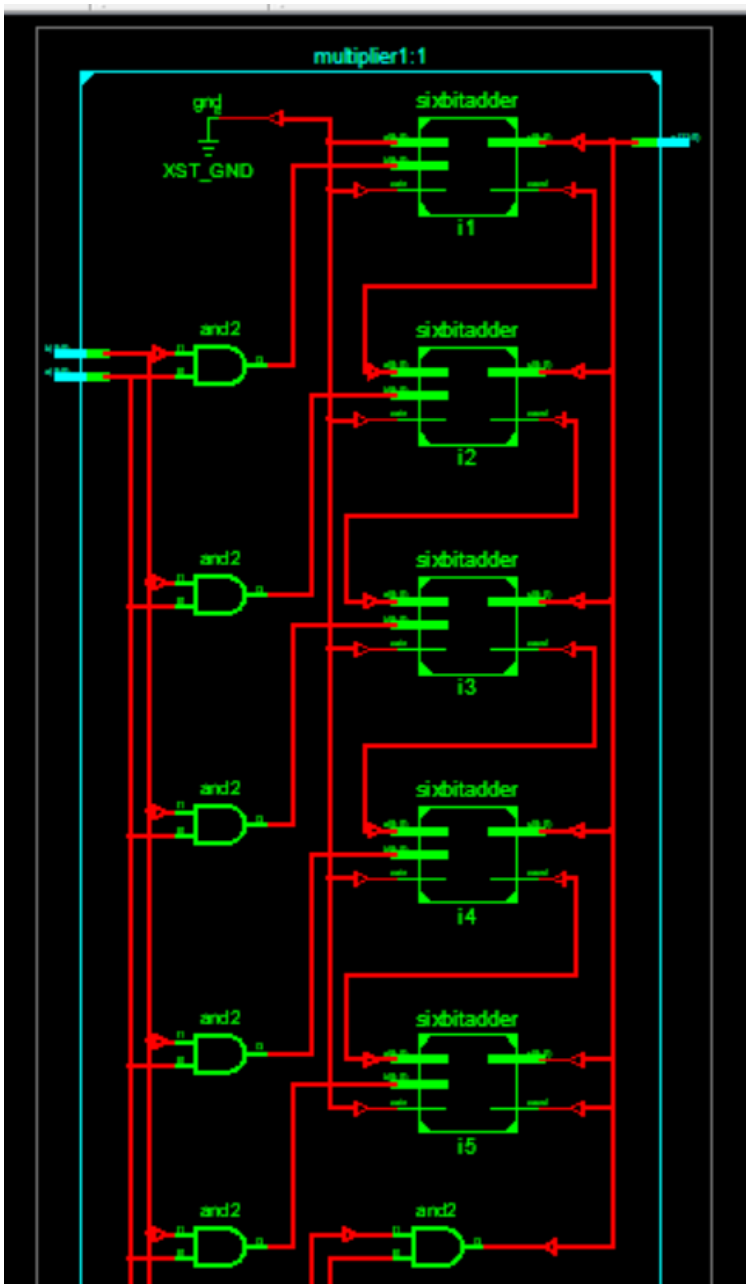
```
signal y1 : STD_LOGIC_VECTOR (5 downto 0);
signal y2 : STD_LOGIC_VECTOR (5 downto 0);
signal y3 : STD_LOGIC_VECTOR (5 downto 0);
signal y4 : STD_LOGIC_VECTOR (5 downto 0);
signal y5 : STD_LOGIC_VECTOR (5 downto 0);

signal s1 : STD_LOGIC_VECTOR (5 downto 0);
signal s2 : STD_LOGIC_VECTOR (5 downto 0);
signal s3 : STD_LOGIC_VECTOR (5 downto 0);
signal s4 : STD_LOGIC_VECTOR (5 downto 0);
signal s5 : STD_LOGIC_VECTOR (5 downto 0);

signal cout : STD_LOGIC_VECTOR (5 downto 1);


begin

--first digit
 p(0) <= a(0) and b(0);
 x1(0) <= a(1) and b(0);
 x1(1) <= a(2) and b(0);
 x1(2) <= a(3) and b(0);
 x1(3) <= a(4) and b(0);
 x1(4) <= a(5) and b(0);
 x1(5) <= '0';
```

```

 y1(0) <= a(0) and b(1);
 y1(1) <= a(1) and b(1);
 y1(2) <= a(2) and b(1);
 y1(3) <= a(3) and b(1);
 y1(4) <= a(4) and b(1);
 y1(5) <= a(5) and b(1);

 i1 : sixbitadder port map (
   a=>x1,
   b=>y1,
   cain=> '0',
   s=>s1,
   caout=>cout(1)
);

--second digit

 p(1) <= s1(0);
 x2(0) <= s1(1);
 x2(1) <= s1(2);
 x2(2) <= s1(3);
 x2(3) <= s1(4);
 x2(4) <= s1(5);
 x2(5) <= cout(1);

 y2(0) <= a(0) and b(2);
 y2(1) <= a(1) and b(2);
 y2(2) <= a(2) and b(2);
 y2(3) <= a(3) and b(2);
 y2(4) <= a(4) and b(2);
 y2(5) <= a(5) and b(2);

 i2 : sixbitadder port map (
   a=>x2,
   b=>y2,
   cain=> '0',
   s=>s2,
   caout=>cout(2)
);
```

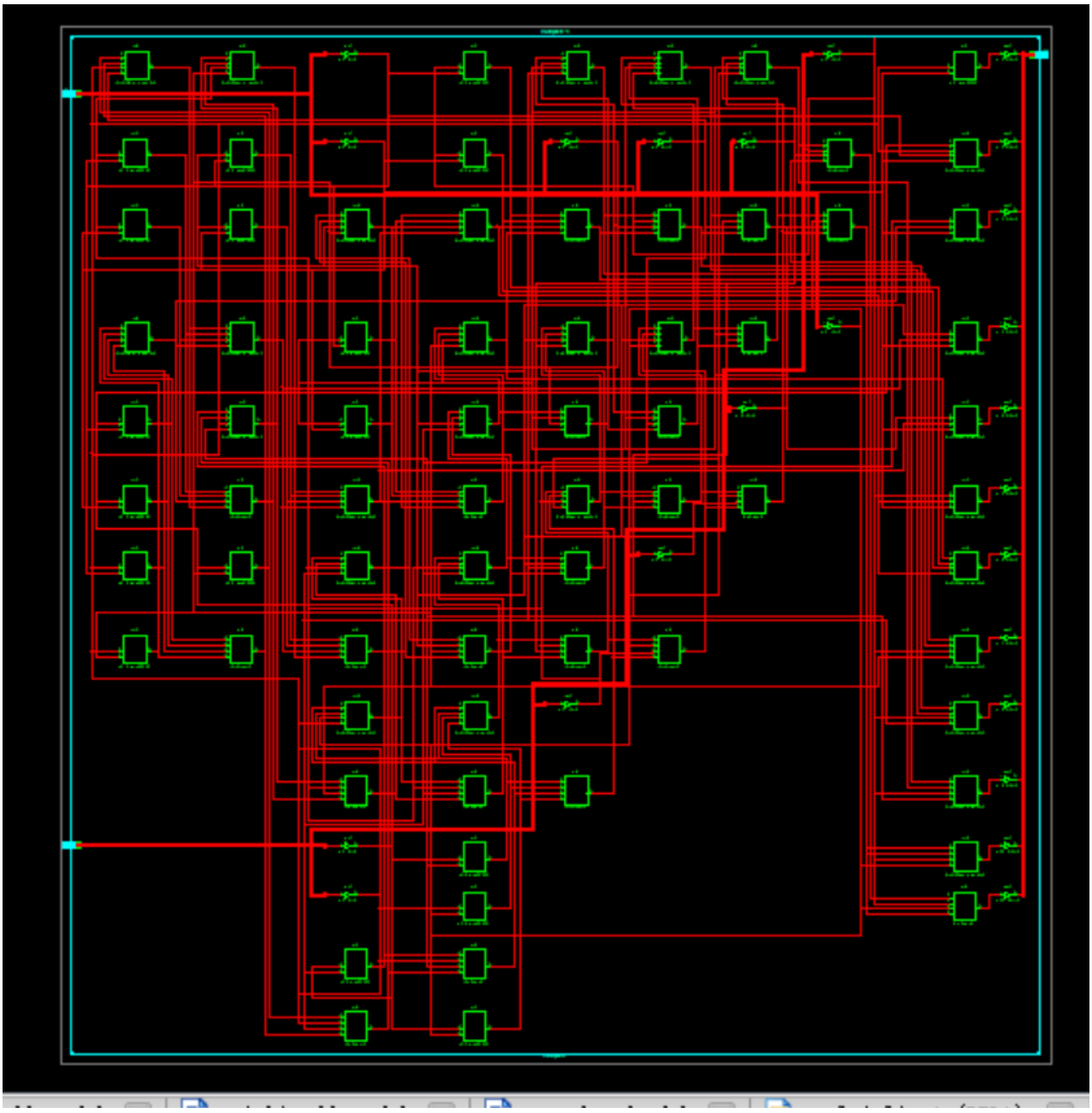
```

 p(5) <= s5(0);
 p(6) <= s5(1);
 p(7) <= s5(2);
 p(8) <= s5(3);
 p(9) <= s5(4);
 p(10) <= s5(5);
 p(11) <= cout(5);


end Behavioral;
```

آر تی ال شماتیک:



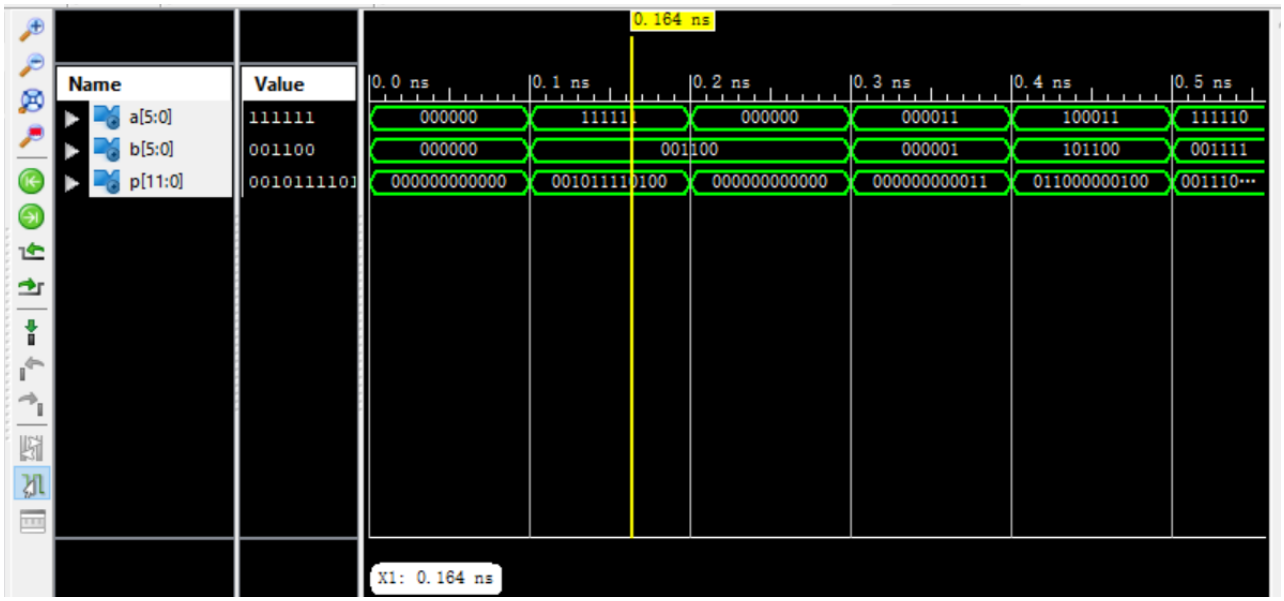


تکنولوژی شماتیک :



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:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;


entity ha is
    Port ( a : in  STD_LOGIC;
           b : in  STD_LOGIC;
           s : out  STD_LOGIC;
           c : out  STD_LOGIC);
end ha;

architecture Behavioral of ha is

begin
c <= a and b;
s <= a xor b;
end Behavioral;
```

کد وی اچ دی ال ماژول فول ادر:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;



entity fa is
    Port ( x : in  STD_LOGIC;
           y : in  STD_LOGIC;
           z : in  STD_LOGIC;
           s : out  STD_LOGIC;
           c : out  STD_LOGIC);
end fa;

architecture Behavioral of fa is

begin

s <= x xor y xor z;
c <= (x or y) and (x or z) and (y or z) ;

end Behavioral;

```

کد اصلی (خلاصه شده):

با اضافه کردن ماژول 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 اخر در اینجا آورده شده است.

```
entity multiplier2 is
    Port ( a : in  STD_LOGIC_VECTOR (5 downto 0);
           b : in  STD_LOGIC_VECTOR (5 downto 0);
           p : out  STD_LOGIC_VECTOR (11 downto 0));
end multiplier2;

architecture Behavioral of multiplier2 is

signal x1 : STD_LOGIC_VECTOR (5 downto 1);
signal x2 : STD_LOGIC_VECTOR (6 downto 1);
signal x3 : STD_LOGIC_VECTOR (6 downto 1);
signal x4 : STD_LOGIC_VECTOR (6 downto 1);
signal x5 : STD_LOGIC_VECTOR (6 downto 1);
signal x6 : STD_LOGIC_VECTOR (5 downto 2);

signal y1 : STD_LOGIC_VECTOR (6 downto 1);
signal y2 : STD_LOGIC_VECTOR (5 downto 1);
signal y3 : STD_LOGIC_VECTOR (5 downto 1);
signal y4 : STD_LOGIC_VECTOR (5 downto 1);
signal y5 : STD_LOGIC_VECTOR (5 downto 1);
signal y6 : STD_LOGIC_VECTOR (5 downto 1);

signal z2 : STD_LOGIC_VECTOR (5 downto 2);
signal z3 : STD_LOGIC_VECTOR (5 downto 2);
signal z4 : STD_LOGIC_VECTOR (5 downto 2);
signal z5 : STD_LOGIC_VECTOR (5 downto 2);
signal z6 : STD_LOGIC_VECTOR (5 downto 2);
```

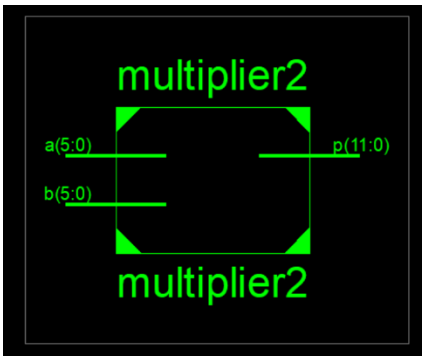
```
component ha is
    Port ( a : in  STD_LOGIC;
           b : in  STD_LOGIC;
           s : out  STD_LOGIC;
           c : out  STD_LOGIC);
end component;

component fa is
    Port ( x : in  STD_LOGIC;
           y : in  STD_LOGIC;
           z : in  STD_LOGIC;
           s : out  STD_LOGIC;
           c : out  STD_LOGIC);
end component;


begin

--first stage

p(0) <= a(0) and b(0);
x1(1) <= a(1) and b(0);
x1(2) <= a(2) and b(0);
x1(3) <= a(3) and b(0);
x1(4) <= a(4) and b(0);
x1(5) <= a(5) and b(0);
```

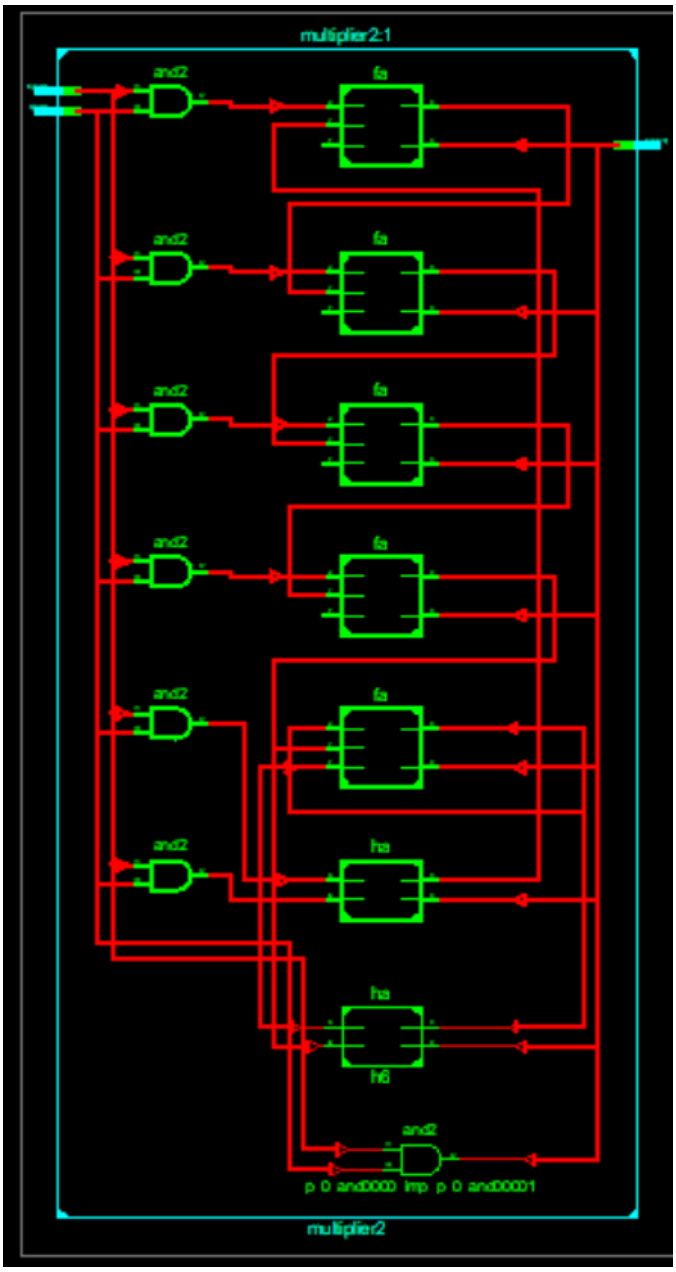
```
y1(1) <= a(0) and b(1);
y1(2) <= a(1) and b(1);
y1(3) <= a(2) and b(1);
y1(4) <= a(3) and b(1);
y1(5) <= a(4) and b(1);

y1(6) <= a(5) and b(1);

h1 : ha port map (
   a => y1(1),
   b => x1(1),
   c => y2(1),
   s => p(1)
);

h2 : ha port map (
   a => y1(2),
   b => x1(2),
   c => y2(2),
   s => z2(2)
);

h3 : ha port map (
   a => y1(3),
   b => x1(3),
   c => y2(3),
   s => z2(3)
);

h4 : ha port map (
   a => y1(4),
   b => x1(4),
   c => y2(4),
   s => z2(4)
);

h5 : ha port map (
   a => y1(5),
   b => x1(5),
   c => y2(5),
   s => z2(5)
);
```

```
--sixth stage

h6 : ha port map (
   a => z6(2),
   b => y6(1),
   c => x6(2),
   s => p(6)
);

f21 : fa port map (
   x => x6(2),
   y => y6(2),
   z => z6(3),
   s => p(7),
   c => x6(3)
);

f22 : fa port map (
   x => x6(3),
   y => y6(3),
   z => z6(4),
   s => p(8),
   c => x6(4)
);
```

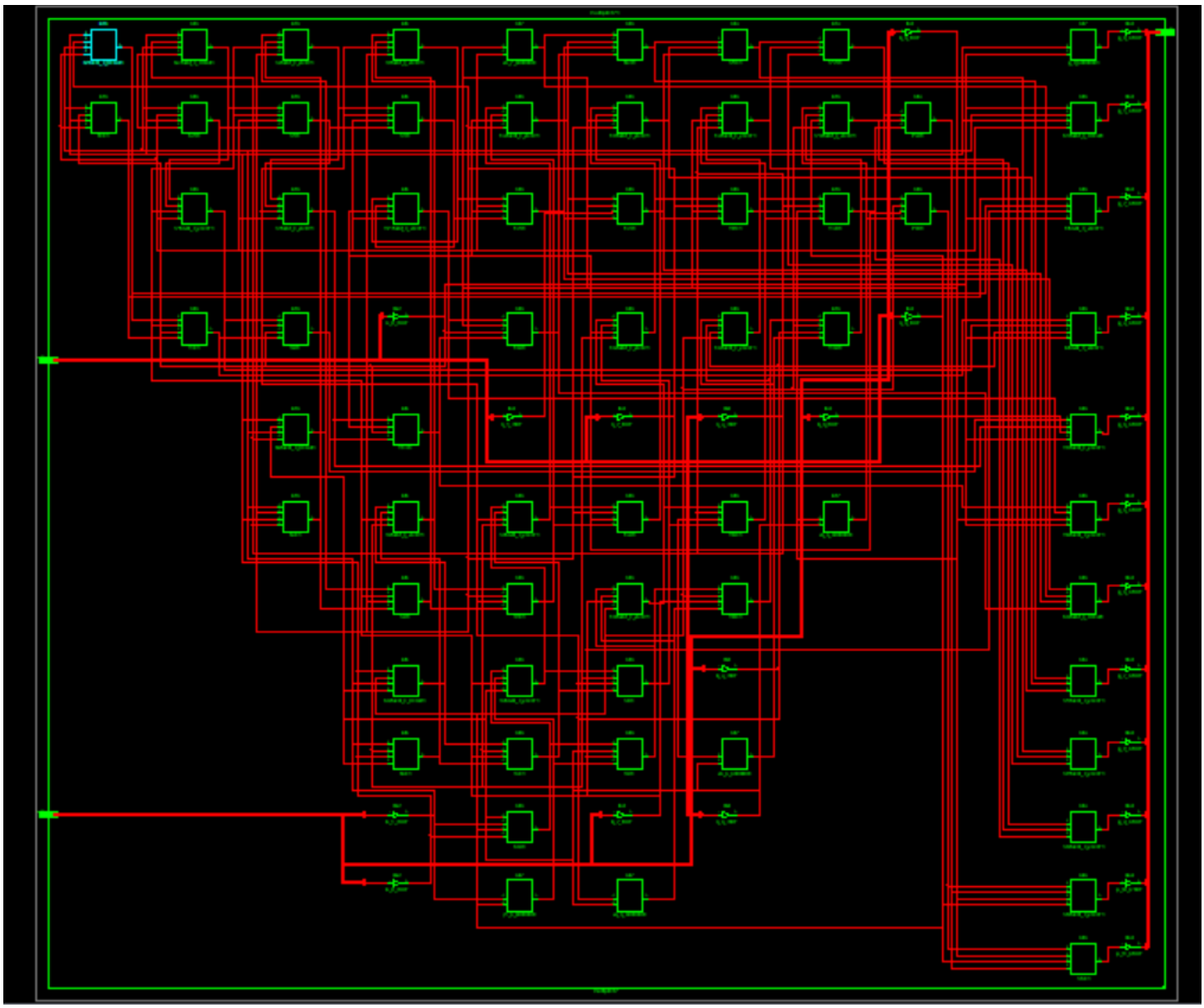
```
f22 : fa port map (
   x => x6(3),
   y => y6(3),
   z => z6(4),
   s => p(8),
   c => x6(4)
);

f23 : fa port map (
   x => x6(4),
   y => y6(4),
   z => z6(5),
   s => p(9),
   c => x6(5)
);

f24 : fa port map (
   x => x6(5),
   y => y6(5),
   z => x5(6),
   s => p(10),
   c => p(11)
);


end Behavioral;
```

آر تی ال شماتیک:



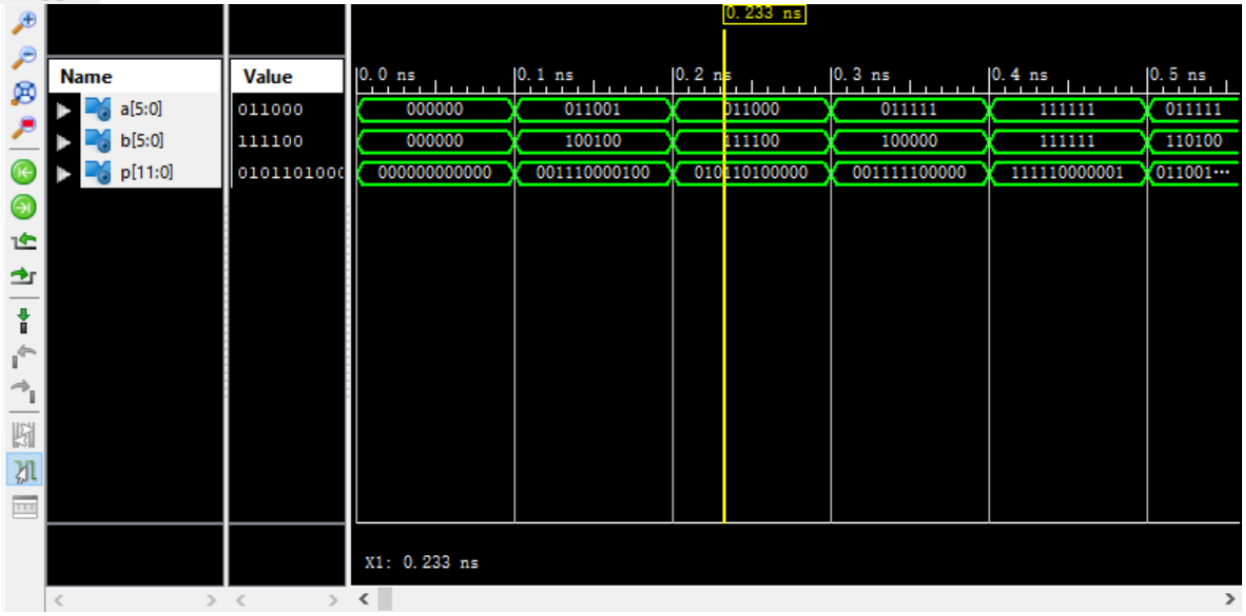


تکنولوژی مپ:



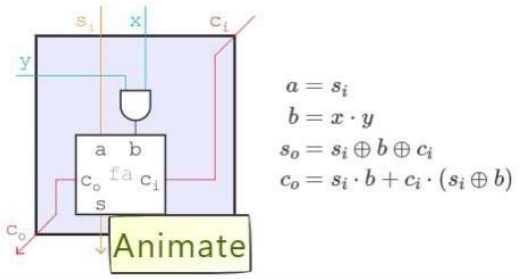
کد تست بنچ و شبیه سازی:

```
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 مالتیپلاییینگ ادر نیاز داشتیم. بلوک مالتیپلاییینگ ادر با توجه به روابط زیر و با استفاده از فول ادر طراحی شد.



$$a = s_i$$
$$b = x \cdot y$$
$$s_o = s_i \oplus b \oplus c_i$$
$$c_o = s_i \cdot b + c_i \cdot (s_i \oplus b)$$

1-bit multiplying-adder

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
```

```
24
25 begin
26 z <= x and y;
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
```

کد اصلی (خلاصه شده طبقه اول و آخر) :

```
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);
```

```
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 );
```

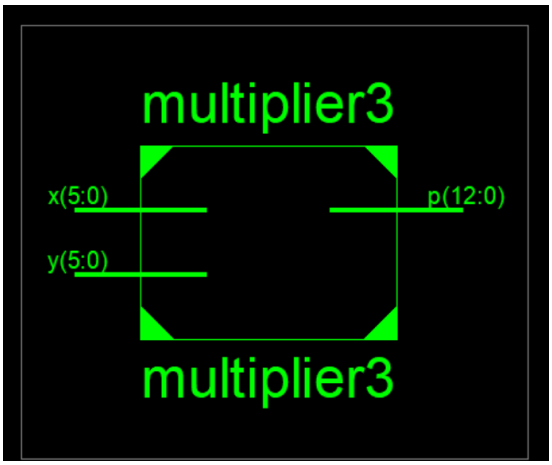
```
49 ma01 : ma port map(
50    x => x(1),
51    y => y(0),
52    si => '0',
53    ci => '0',
54    co => c0(1),
55    so => s0(1)
56 );
57 ma02 : ma port map(
58    x => x(2),
59    y => y(0),
60    si => '0',
61    ci => '0',
62    co => c0(2),
63    so => s0(2)
64 );
65 ma03 : ma port map(
66    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
```

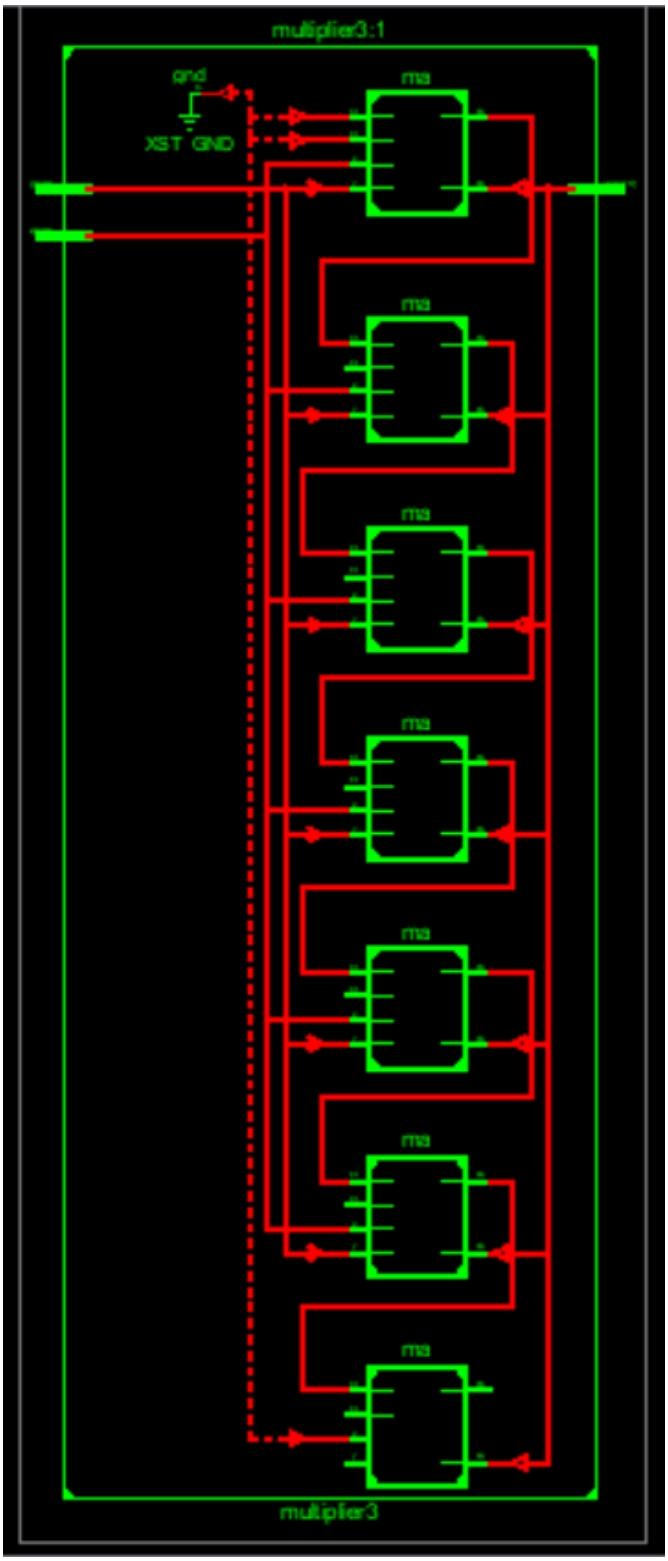
```
350 --seventh stage
351
352 ma60 : ma port map(
353    x => '0',
354    y => '1',
355    si => s5(1),
356    ci => c5(0),
357    co => c6(0),
358    so => p(6)
359 );
360
361 ma61 : ma port map(
362    x => c6(0),
363    y => '1',
364    si => s5(2),
365    ci => c5(1),
366    co => c6(1),
367    so => p(7)
368 );
```

```
369 ma62 : ma port map(
370    x => c6(1),
371    y => '1',
372    si => s5(3),
373    ci => c5(2),
374    co => c6(2),
375    so => p(8)
376 );
377 ma63 : ma port map(
378    x => c6(2),
379    y => '1',
380    si => s5(4),
381    ci => c5(3),
382    co => c6(3),
383    so => p(9)
384 );
385 ma64 : ma port map(
386    x => c6(3),
387    y => '1',
388    si => s5(5),
389    ci => c5(4),
390    co => c6(4),
391    so => p(10)
392 );
```

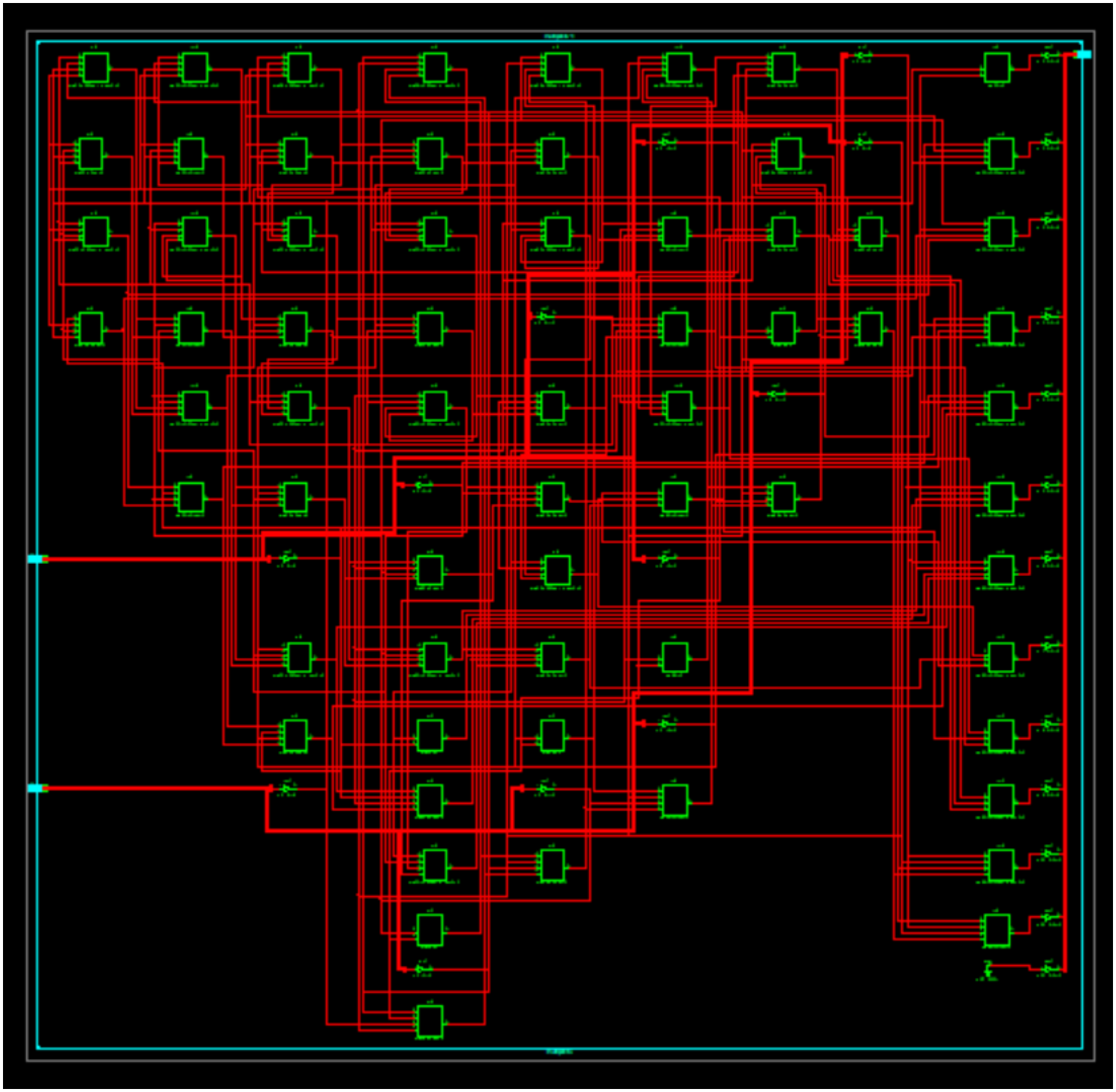
```
385 ma64 : ma port map(
386    x => c6(3),
387    y => '1',
388    si => s5(5),
389    ci => c5(4),
390    co => c6(4),
391    so => p(10)
392 );
393 ma65 : ma port map(
394    x => c6(4),
395    y => '1',
396    si => '0',
397    ci => c5(5),
398    co => p(12),
399    so => p(11)
400 );
401
402
403 end Behavioral;
404
```

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
```