دانشگاه صنعتی امیرکبیر
(پلی‌تکنیک تهران)
دانشکده برق

درس

معماری کامپیوتر و ریزپردازنده

عنوان

گزارش تمرین عملی دوم - VHDL

نگارش

پارسا محمّدی ۹۹۲۳۱۲۱

استاد درس

دکتر شریعتمدار مرتضوی

فروردین ۱۴۰۲

فهرست **صفحه**

پیاده‌سازی ماژول Full Adder و Half Adder

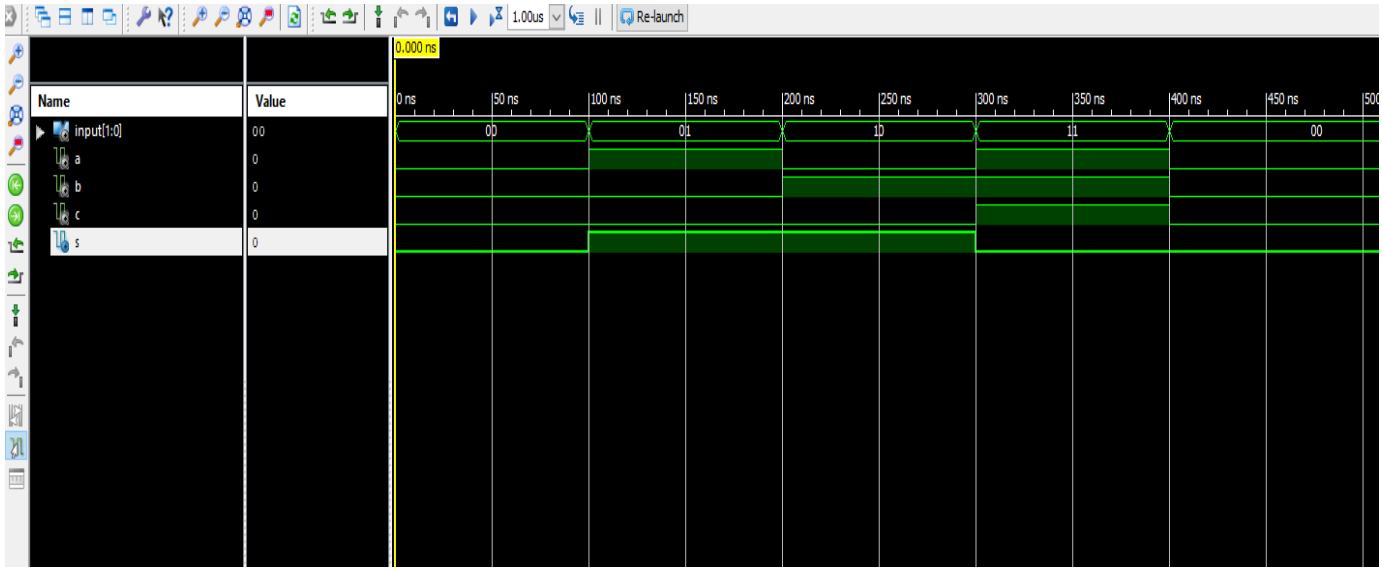
```vhdl
-- Full Adder
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Full_adder is
    Port ( x : in  STD_LOGIC;
           y : in  STD_LOGIC;
           cin : in  STD_LOGIC;
           sum : out  STD_LOGIC;
           carrout : out  STD_LOGIC);
end Full_adder;

architecture Behavioral of Full_adder is

component half_adder is port (a, b : in STD_LOGIC;
                              c, s : out STD_LOGIC);
                              end component;
signal carr1,s, carr2: std_logic;

begin

ha1 : half_adder port map (x, y , carr1 ,s);
ha2 : half_adder port map ( s, cin , carr2, sum );
carrout <= carr1 or carr2;


end Behavioral;
```

تصویر روبه‌رو کد مربوط به یک فول اددر را نمایش می دهد که با استفاده یک half adder که به صورت کامپوننت اضافه شده است.

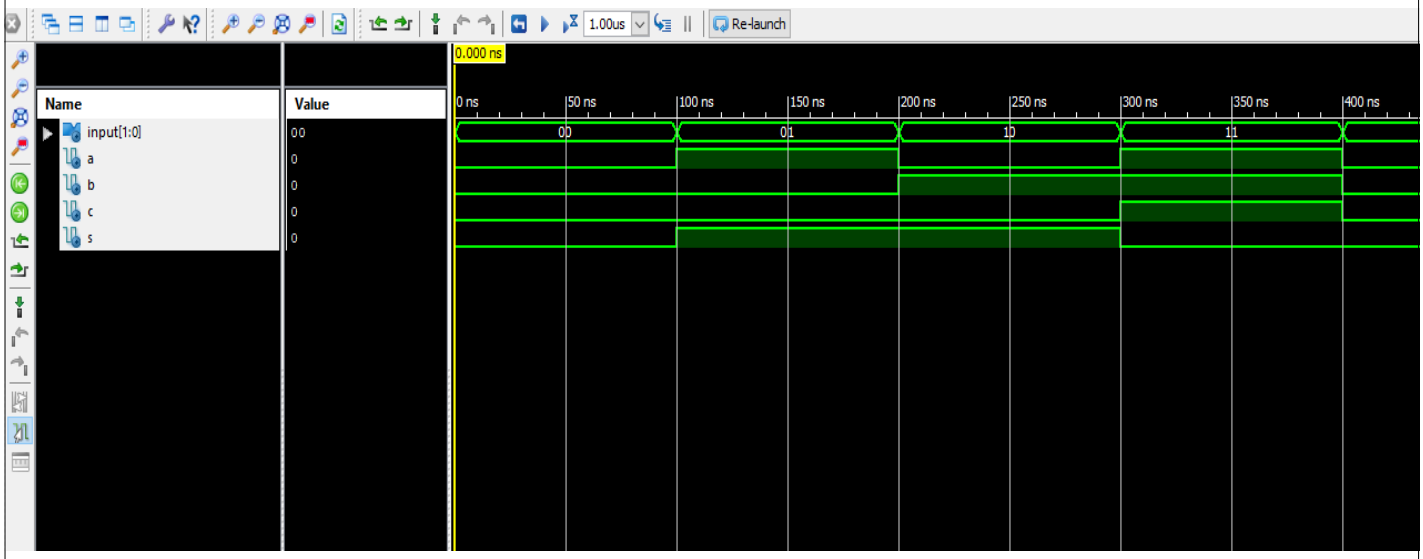
معماری و بررسی Test Bench های مربوط به full adder و half adder در تمرین عملی اول انجام شده است.

```vhdl
-- Module Name:    Half_Adder - Behavioral
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;


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

architecture Behavioral of half_adder is

begin

c <= a and b;
s <= a xor b;

end Behavioral;
```

تصویر روبه‌رو کد مربوط به half adder را نمایش می‌دهد. از این ماژول در ساخت یک full adder استفاده شده است. یک half adder بر خلاف یک full adder کری ورودی ندارد.



تست بنچ Full Adder

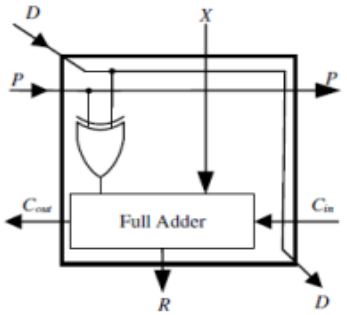


تست بنچ Half Adder

پیاده‌سازی ماژول CAS

این ماژول همان‌گونه که در صورت پروژه بیان شده است پیاده سازی شده است.

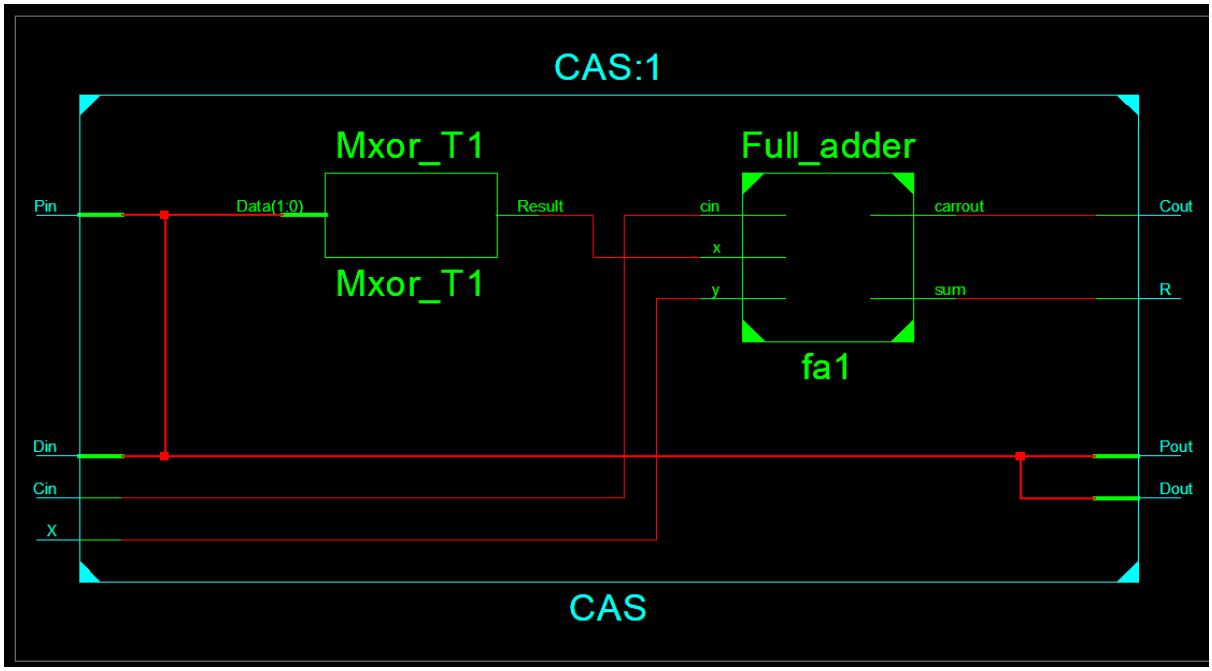
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity CAS is
    Port ( Pin : in  STD_LOGIC;
           Din : in  STD_LOGIC;
           X : in  STD_LOGIC;
           Cin : in  STD_LOGIC;
           Pout : out  STD_LOGIC;
           R : out  STD_LOGIC;
           Dout : out  STD_LOGIC;
           Cout : out  STD_LOGIC);
end CAS;

architecture Behavioral of CAS is

component full_adder
          is port ( x : in  STD_LOGIC;
                    y : in  STD_LOGIC;
                    cin : in  STD_LOGIC;
                    sum : out  STD_LOGIC;
                    carrout : out  STD_LOGIC);
end component;

signal T : STD_LOGIC;

begin

T <= Pin xor Din;
fal : full_adder port map(T,X,Cin,R,Cout);
pout <= Pin;
Dout <= Din;


end Behavioral;
```



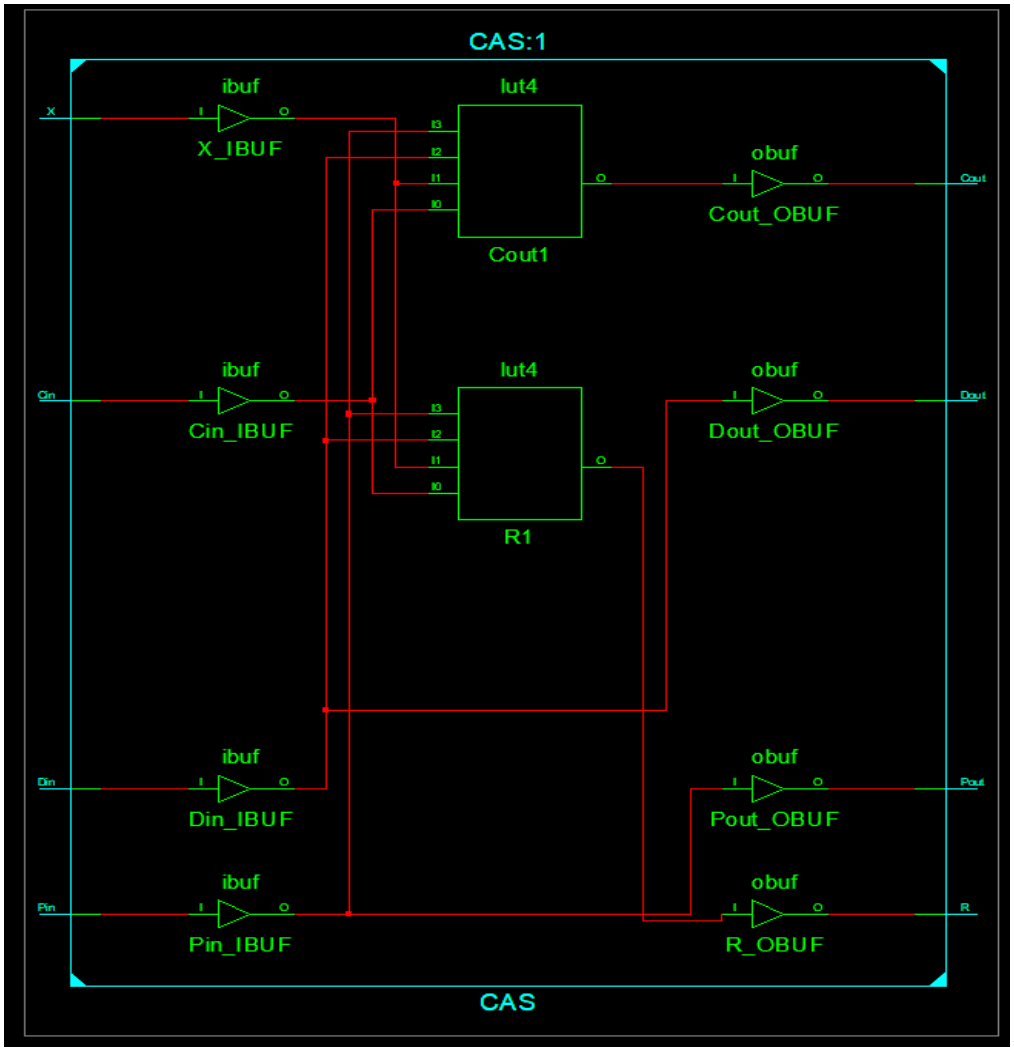
همین طور که در معماری ماژول مشخص است این ماژول دارای چهار ورودی و چهار خروجی می‌باشد. و تمام ورودی و خروجی ها منطقی اند.

برای ساخت این ماژول از یک ماژول Full Adder به عنوان کامپوننت استفاده شده است. با توجه به شکل معماری ورودی های P و D با هم XOR شده اند. خروجی در یک سیگنال منطقی به نام T ریخته شده است . این سیگنال در ادامه به Full Adder داده شده است. ورودی ها و خروجی ماژول Full Adder با دستور port map به ماژول داده شده اند. خروجی p و d به صورت مستقیم از ورودی گرفته شده اند با توجه به شکل معماری قطعه.

ماژول CAS در ادامه برای طراحی ماژول های X و Y استفاده می‌شود.



شماتیک بلوک RTL – CAS



تصویر Technology Schematic بلوک CAS

گزارش سطح مصرف CAS - FPGA

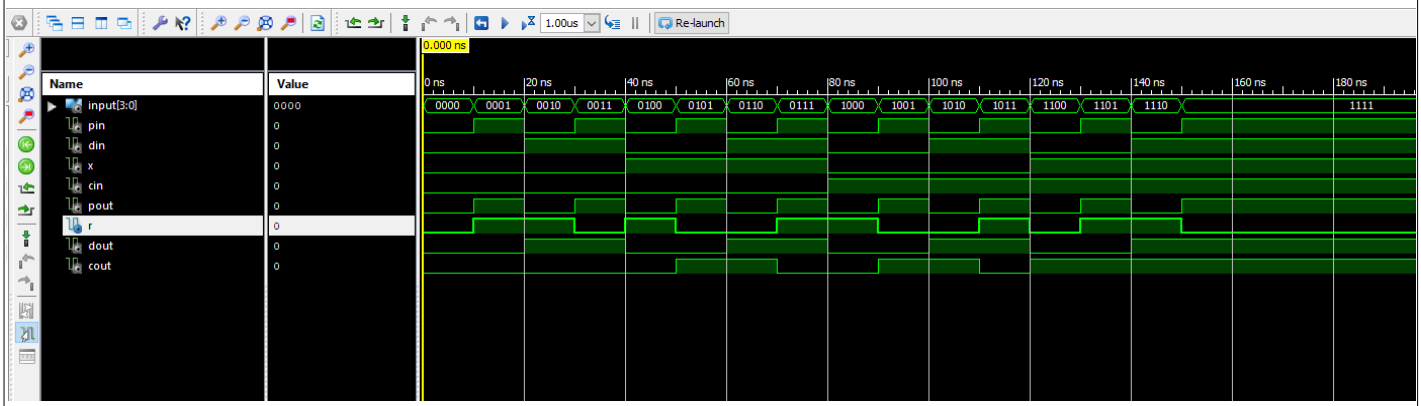
```
=========================================================================
*                      Design Summary
=========================================================================

Top Level Output File Name         : CAS.ngc

Primitive and Black Box Usage:
------------------------------
# BELS                             : 2
#      LUT4                        : 2
# IO Buffers                       : 8
#      IBUF                        : 4
#      OBUF                        : 4

Device utilization summary:
---------------------------

Selected Device : 7a100tcsg324-3


Slice Logic Utilization:
 Number of Slice LUTs:                  2  out of  63400      0%
    Number used as Logic:               2  out of  63400      0%

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:    2
   Number with an unused Flip Flop:     2  out of      2    100%
   Number with an unused LUT:           0  out of      2      0%
   Number of fully used LUT-FF pairs:   0  out of      2      0%
   Number of unique control sets:       0

IO Utilization:
 Number of IOs:                         8
 Number of bonded IOBs:                 8  out of    210      3%

Specific Feature Utilization:

---------------------------
Partition Resource Summary:
---------------------------

  No Partitions were found in this design.

---------------------------


=========================================================================
```

تصاویر روبه‌رو منابع مصرف شده را در بلوک CAS را نمایش می‌دهد.

همان طور که مشخص است ۲ عدد از LUT ها از 63400 مصرف شده است که تقریبا برار صفر درصد کل تعداد LUT ها می‌باشد.

همان طور که مشخص است ۸ عدد IO ها از ۲۱۰ تا مصرف شده است که تقریبا برابر ۳ درصد می‌باشد.

تست بنچ CAS



تصویر بالا خروجی شبیه سازی تست بنچ CAS می‌باشد. تمام نتایج همان گونه است که انتظار می‌رفت.

تصویر روبه‌رو کد مربوط به تست بنچ ماژول CAS را نشان می‌دهد.

```
12  ARCHITECTURE behavior OF CAS_TB IS
13      COMPONENT CAS
14      PORT(
15           Pin : IN  std_logic;
16           Din : IN  std_logic;
17           X : IN  std_logic;
18           Cin : IN  std_logic;
19           Pout : OUT  std_logic;
20           R : OUT  std_logic;
21           Dout : OUT  std_logic;
22           Cout : OUT  std_logic
23          );
24      END COMPONENT;
25     --Inputs
26     signal input : std_logic_vector(3 downto 0) := "0000";
27     signal Pin : std_logic := '0';
28     signal Din : std_logic := '0';
29     signal X : std_logic := '0';
30     signal Cin : std_logic := '0';
31
32     --Outputs
33     signal Pout : std_logic;
34     signal R : std_logic;
35     signal Dout : std_logic;
36     signal Cout : std_logic;
37
38  BEGIN
39
40     -- Instantiate the Unit Under Test (UUT)
41     uut: CAS PORT MAP (
42             Pin => Pin,
43             Din => Din,
44             X => X,
45             Cin => Cin,
46             Pout => Pout,
47             R => R,
48             Dout => Dout,
49             Cout => Cout
50           );
51
52     -- Stimulus process
53     stim_proc: process
54     begin
55
56     for i in 0 to 15 loop
57
58        input <= std_logic_vector(to_unsigned(i,4));
59        wait for 0.00000001 ps;
60        Pin <= input(0);
61        Din <= input(1);
62        X   <= input(2);
63        Cin <= input(3);
64        wait for 10 ns;
65
66     end loop;
67
```

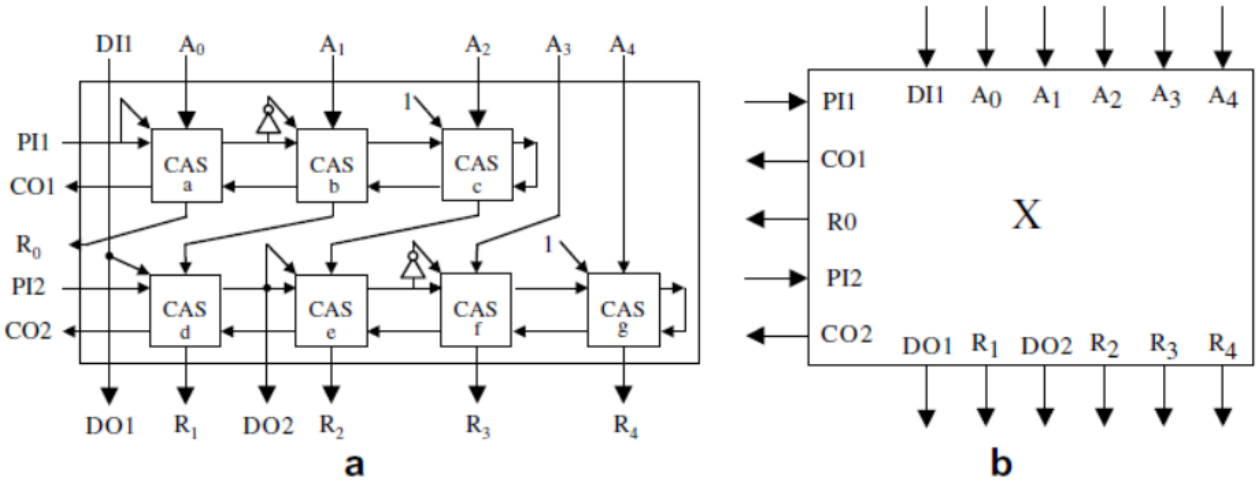
در بخش سیگنال ها سیگنالی اضافه به نام input به صورت آرایه منطقی قرار گرفته است. این متغییر در خط ۵۸ مقدار دهی می‌شود و از المان های آن برای مقدار دهی به ورودی های ماژول استفاده می‌شود.

حلقه for تعرف شده در خط ۵۶ این امکان را فراهم می‌کند تا تمام حالات ممکن برای ورودی این ماژول ایجاد شود و به ورودی ها داده شود.

نکته: در خط ۵۹ بعد از مقدار دهی به متغییر input یک زمان بسیار کوچک برای wait در نظر گرفته شده است. این تاخیر برای این است که ابتدا مقدار دهی به input داده شود و سپس به ورودی های ماژول انجام شود. اگر این تاخیر گذاشته نشود اعملیات مقدار دهی به صورت موازی انجام می‌شود و رووری ها یک سیکل عقب تر از input قرار می‌گرند. این تاخیر مانع این مشکل می‌شود.

پیاده‌سازی بلوک X

همان‌طور در صورت تمرین بیان شده است معماری بلوک X به صورت زیر می‌باشد. این بلوک از هفت عدد CAS ساخته شده است. و ۱۷ ورودی و خروجی دارد. نکته در پیاده سازی این ماژول این است که از پایه D خروجی هیچکدام از CAS استفاده نشده است آن ها به هیچ جایی وصل نیستند.




```
1   -- Module Name:    X - Behavioral
2
3   library IEEE;
4   use IEEE.STD_LOGIC_1164.ALL;
5
6   entity X is Port ( DI1 : in  STD_LOGIC;
7                      A0 : in  STD_LOGIC;
8                      A1 : in  STD_LOGIC;
9                      A2 : in  STD_LOGIC;
10                     A3 : in  STD_LOGIC;
11                     A4 : in  STD_LOGIC;
12                     PI1 : in  STD_LOGIC;
13                     PI2 : in  STD_LOGIC;
14                     R0 : out  STD_LOGIC;
15                     R1 : out  STD_LOGIC;
16                     R2 : out  STD_LOGIC;
17                     R3 : out  STD_LOGIC;
18                     R4 : out  STD_LOGIC;
19                     DO1 : out  STD_LOGIC;
20                     DO2 : out  STD_LOGIC;
21                     CO1 : out  STD_LOGIC;
22                     CO2 : out  STD_LOGIC);
23                     end X;
24
25  architecture Behavioral of X is
26
27  component CAS is port( Pin : in  STD_LOGIC;
28                         Din : in  STD_LOGIC;
29                         X : in  STD_LOGIC;
30                         Cin : in  STD_LOGIC;
31                         Pout : out  STD_LOGIC;
32                         R : out  STD_LOGIC;
33                         Dout : out  STD_LOGIC;
34                         Cout : out  STD_LOGIC);
35                         end component;
36
37  signal R_a ,R_b ,R_c ,R_d ,R_e ,R_f ,R_g : std_logic;
38  signal Pout_a ,Pout_b ,Pout_c ,Pout_d ,Pout_e ,Pout_f ,Pout_g : std_logic;
```

```
27  component CAS is port( Pin : in  STD_LOGIC;
28                         Din : in  STD_LOGIC;
29                         X : in  STD_LOGIC;
30                         Cin : in  STD_LOGIC;
31                         Pout : out  STD_LOGIC;
32                         R : out  STD_LOGIC;
33                         Dout : out  STD_LOGIC;
34                         Cout : out  STD_LOGIC);
35                         end component;
36
37  signal R_a ,R_b ,R_c ,R_d ,R_e ,R_f ,R_g : std_logic;
38  signal Pout_a ,Pout_b ,Pout_c ,Pout_d ,Pout_e ,Pout_f ,Pout_g : std_logic;
39  signal Cout_a ,Cout_b ,Cout_c ,Cout_d ,Cout_e ,Cout_f ,Cout_g : std_logic;
40  signal z : std_logic_vector(6 downto 0);--dummy variable
41
42  signal NPout_a,Npout_e :std_logic;
43
44  begin
45
46  NPout_a <= not Pout_a;
47  Npout_e <= not Pout_e;
48
49  CAS_a : CAS port map(PI1    ,PI1        ,A0  ,Cout_b ,Pout_a ,R0  ,z(0) ,CO1 );
50  CAS_b : CAS port map(Pout_a ,NPout_a    ,A1  ,Cout_c ,Pout_b ,R_b ,z(1) ,Cout_b );
51  CAS_c : CAS port map(Pout_b ,'1'        ,A2  ,Pout_c ,Pout_c ,R_c ,z(2) ,Cout_c );
52  CAS_d : CAS port map(PI2    ,DI1        ,R_b ,Cout_e ,Pout_d ,R1  ,Z(3) ,CO2 );
53  CAS_e : CAS port map(Pout_d ,Pout_d     ,R_c ,Cout_f ,Pout_e ,R2  ,Z(4) ,Cout_e );
54  CAS_f : CAS port map(Pout_e ,Npout_e    ,A3  ,Cout_g ,Pout_f ,R3  ,Z(5) ,Cout_f );
55  CAS_g : CAS port map(Pout_f ,'1'        ,A4  ,Pout_g ,Pout_g ,R4  ,Z(6) ,Cout_g );
56
57  DO1 <= DI1;
58  DO2 <= Pout_d;
59
60
61
62  end Behavioral;
```

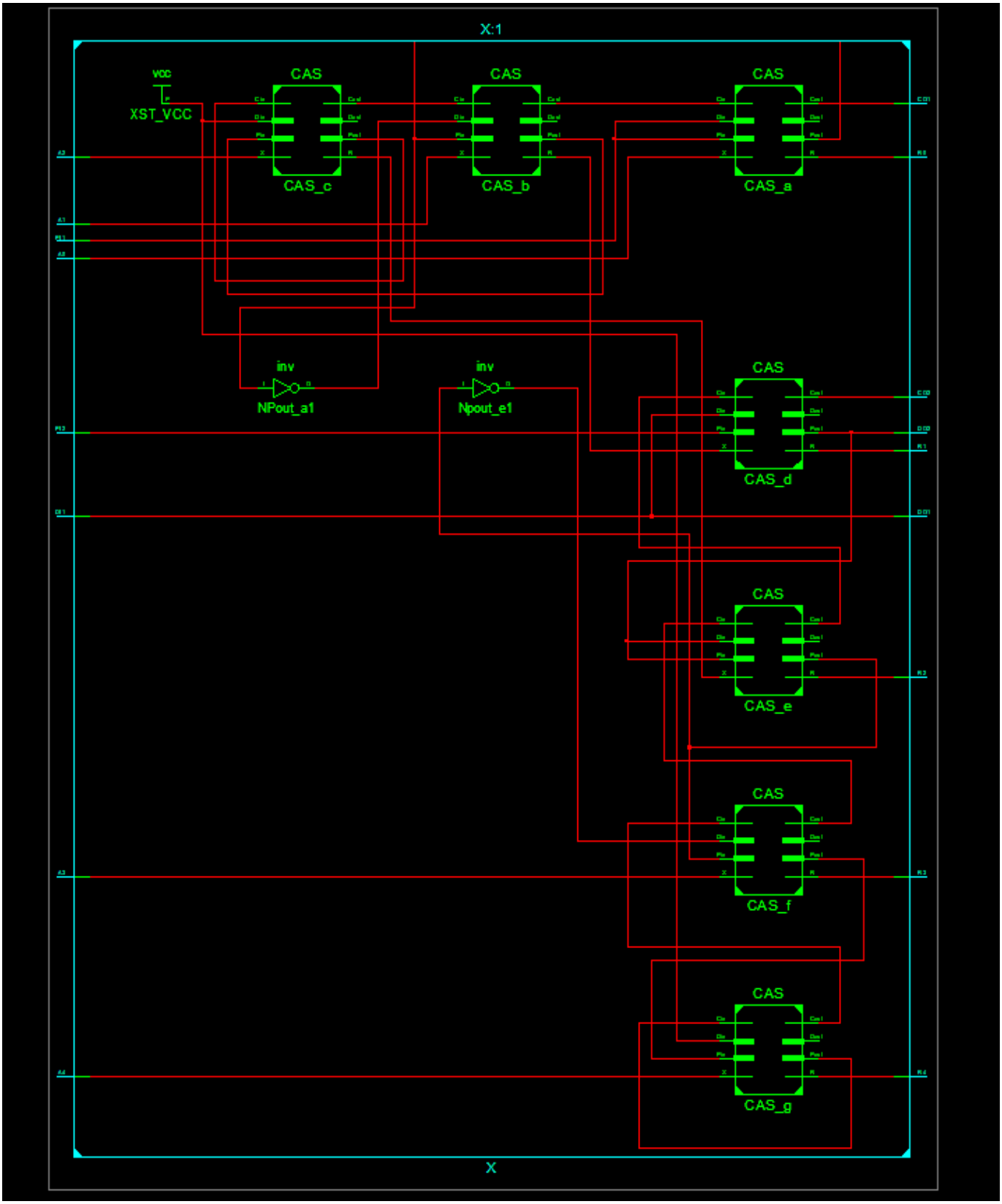
دو تصویر بالا کد مربوط به بلوک X را نمایش می‌دهند. این ماژول تمام ۱۷ ورودی و خروجی را به صورت یک بیت منطقی دریافت می‌کند. علت پیاده سازی به این صورت این است که به علت زیاد بودن ارتباط ها در کد اصلی یعنی محاسبه کنند جذر بهتر است برای راحتی نوشتن کد به این صورت پیاده سازی شود.

سیگنال های در بخش خط های ۳۷ تا ۳۹ نوشته شده اند برای برقراری ارتباط های داخلی ماژول می‌باشند.

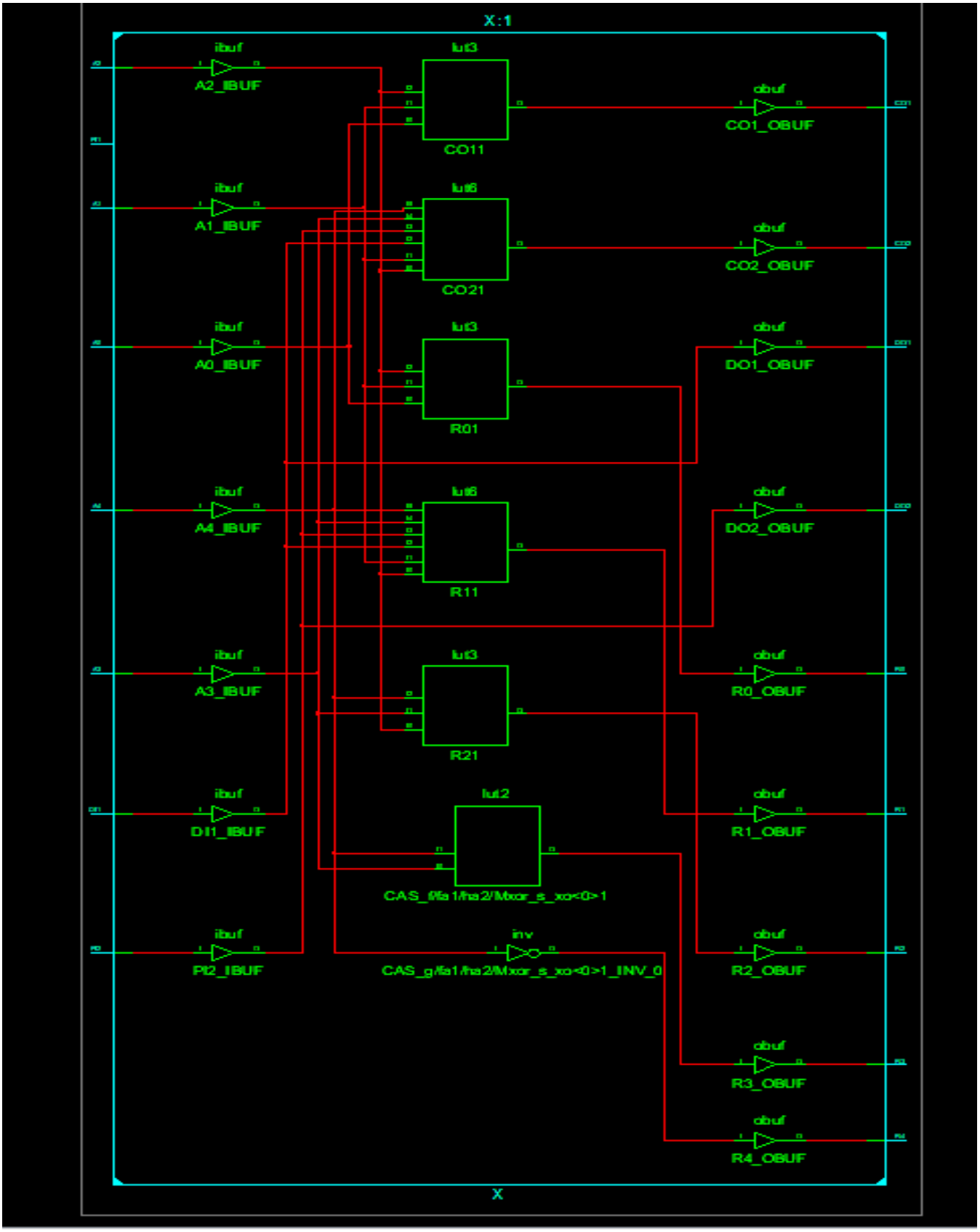
متغییر z که در خط 40 نوشته شده است یک متغییر غیر مهم است. این متغییر برای جاهایی استفاده می‌شود که پایه خروجی و ورودی قرار نیست به جایی متصل شود و مقدار آن اهمیتی ندارد.

در خط ۴۲ سیگنال هایی تعریف شده‌اند که برای ذخیره کردن مغادیر معکوس Pout_a و Pout_e استفاده می‌شوند. در خط های ۴۶ و ۴۷ این معکوس سازی انجام شده است.

بقیه کد پیاده سازی معماری و وصل کردن ورودی ها و خروجی های هر ماژول CAS با استفاد دستور Port mapاست.



تصویر شماتیک بلوک X (RTL)



تصویر Technology Schematic بلوک X

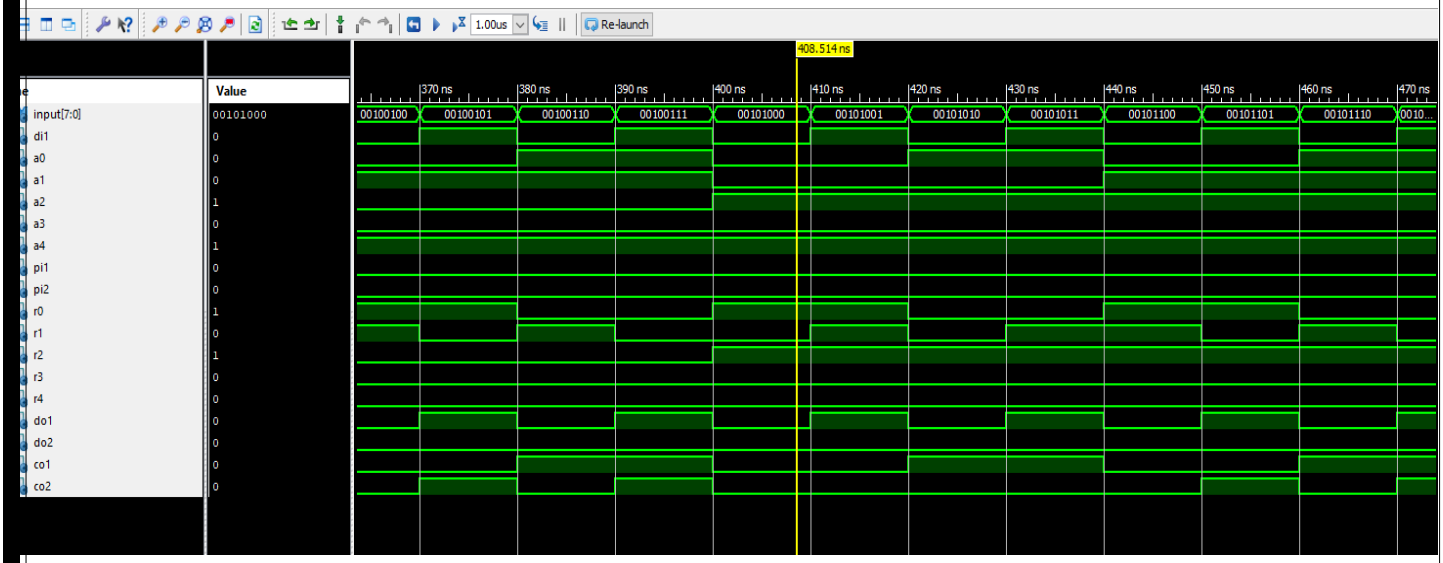
گزارش سطح مصرف FPGA

```
--------------------------------------------------------------------------------
*                            Design Summary
================================================================================

Top Level Output File Name         : X.ngc

Primitive and Black Box Usage:
------------------------------
# BELS                             : 7
#      INV                         : 1
#      LUT2                        : 1
#      LUT3                        : 3
#      LUT6                        : 2
# IO Buffers                       : 16
#      IBUF                        : 7
#      OBUF                        : 9

Device utilization summary:
---------------------------

Selected Device : 7a100tcsg324-3


Slice Logic Utilization:
 Number of Slice LUTs:                    7  out of  63400     0%
    Number used as Logic:                 7  out of  63400     0%

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:      7
   Number with an unused Flip Flop:       7  out of      7   100%
   Number with an unused LUT:             0  out of      7     0%
   Number of fully used LUT-FF pairs:     0  out of      7     0%
   Number of unique control sets:         0

IO Utilization:
 Number of IOs:                          17
 Number of bonded IOBs:                  16  out of    210     7%

Specific Feature Utilization:

---------------------------
Partition Resource Summary:
---------------------------
```

تصاویر بالا منابع مصرف شده را در بلوک X را نمایش می‌دهد.

- همان طور که مشخص است ۷ عدد از LUT ها از 63400 مصرف شده است که تقریبا برار صفر درصد کل تعداد LUT ها می‌باشد.
- همان طور که مشخص است ۱۶ عدد IO ها از ۲۱۰ تا مصرف شده است که تقریبا برابر ۷ درصد کل می‌باشد.

تست بنچ ماژول X



تصویر بالا خروجی شبیه سازی تست بنچ X می‌باشد. تمام نتایج همان گونه است که انتظار می‌رفت.

تصویر روبه‌رو کد مربوط به تست بنچ ماژول X را نشان می‌دهد.

```
60  BEGIN
61
62     -- Instantiate the Unit Under Test (UUT)
63     uut: X PORT MAP (
64            DI1 => DI1,
65            A0 => A0,
66            A1 => A1,
67            A2 => A2,
68            A3 => A3,
69            A4 => A4,
70            PI1 => PI1,
71            PI2 => PI2,
72            R0 => R0,
73            R1 => R1,
74            R2 => R2,
75            R3 => R3,
76            R4 => R4,
77            DO1 => DO1,
78            DO2 => DO2,
79            CO1 => CO1,
80            CO2 => CO2
81          );
82
83
84     -- Stimulus process
85     stim_proc: process
86     begin
87
88
89        for i in 0 to 255 loop
90          input <= std_logic_vector(to_unsigned(i, 8));
91          wait for 0.000000001 ps;
92          DI1 <= input(0);
93          A0  <= input(1);
94          A1  <= input(2);
95          A2  <= input(3);
96          A3  <= input(4);
97          A4  <= input(5);
98          PI1 <= input(6);
99          PI2 <= input(7);
100
101       wait for 10 ns;
102       end loop;
103
104       wait;
105    end process;
106
107 END;
108
```

در بخش سیگنال ها سیگنالی اضافه به نام input به صورت آرایه منطقی قرار گرفته است. این متغییر در خط ۹۰ مقدار دهی می‌شود و از المان های آن برای مقدار دهی به ورودی های ماژول استفاده می‌شود.

حلقه for تعرف شده در خط ۸۹ این امکان را فراهم می‌کند تا تمام حالات ممکن برای ورودی این ماژول ایجاد شود و به ورودی ها داده شود.

نکته: در خط ۹۱ بعد از مقدار دهی به متغییر input یک زمان بسیار کوچک برای wait در نظر گرفته شده است. این تاخیر برای این است که ابتدا مقدار دهی به input داده شود و سپس به ورودی های ماژول انجام شود. اگر این تاخیر گذاشته نشود

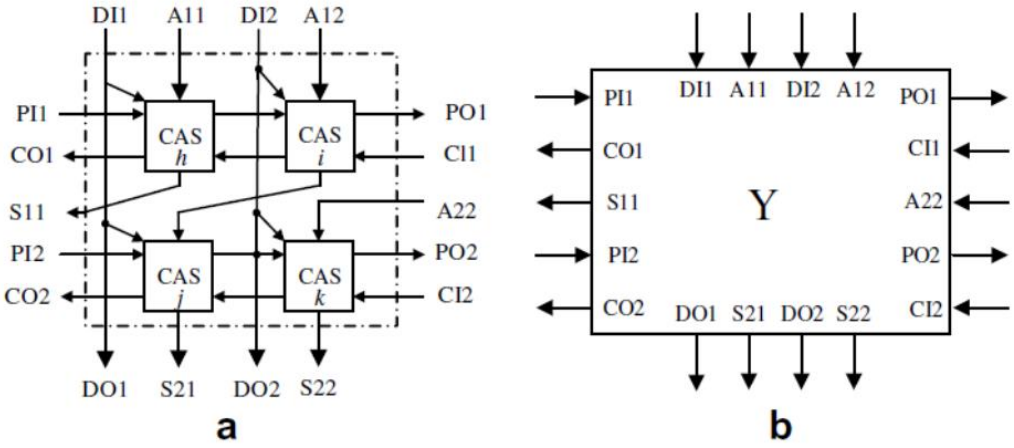
اعملیات مقدار دهی به صورت موازی انجام می‌شود و رووری ها یک سیکل عقب تر از input قرار می‌گرند.
این تاخیر مانع این مشکل می‌شود.

پیاده سازی بلوک Y

همان‌طور در صورت تمرین بیان شده است معماری بلوک Y به صورت زیر می‌باشد. این بلوک از چهار عدد CAS ساخته شده است. و ۱۸ ورودی و خروجی دارد. نکته در پیاده سازی این ماژول این است که اتصال مشخص شده بین Pout ماژول CAS j و DI2 اشتباه بوده است. و در کد پیاده‌سازی نشده است.

جهت فلش ها ورودی و یا خروجی بودن هر پایه را بیان می‌کند.

بلوک Y :



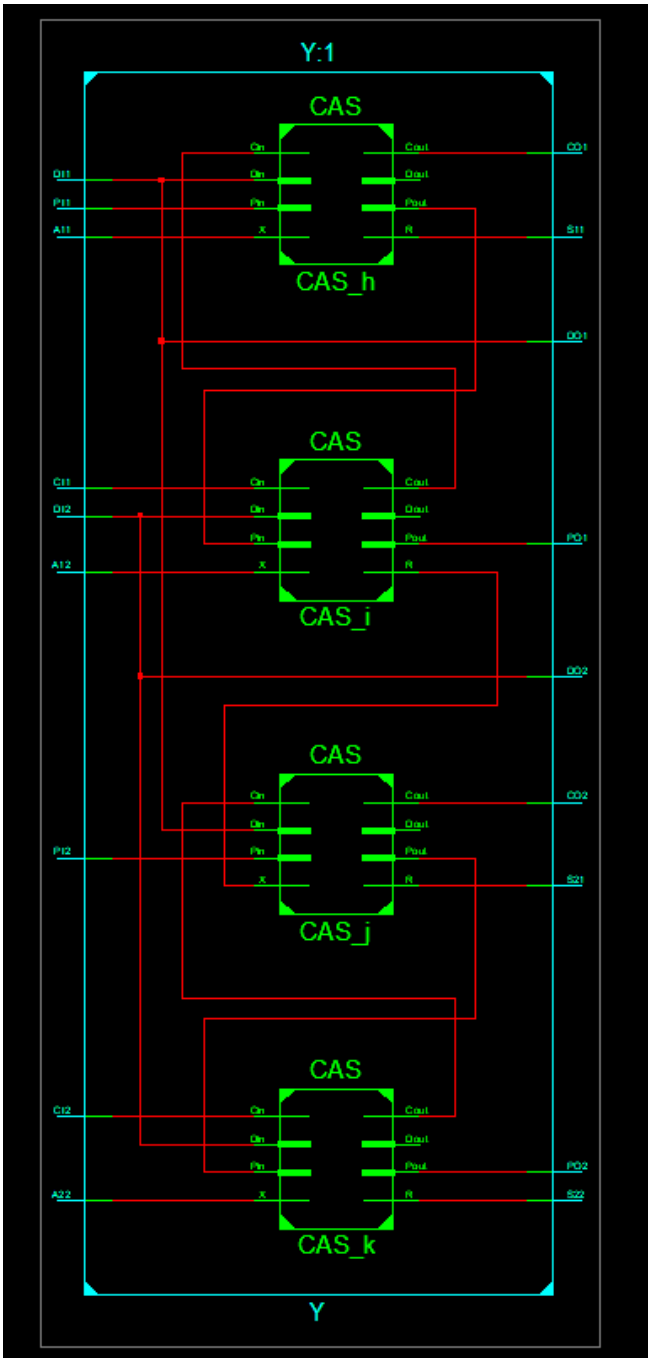

```
-- Module Name:    Y - Behavioral

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Y is
    Port ( DI1 : in  STD_LOGIC;
           DI2 : in  STD_LOGIC;
           A11 : in  STD_LOGIC;
           A12 : in  STD_LOGIC;
           A22 : in  STD_LOGIC;
           PI1 : in  STD_LOGIC;
           PI2 : in  STD_LOGIC;
           CI1 : in  STD_LOGIC;
           CI2 : in  STD_LOGIC;
           CO1 : out  STD_LOGIC;
           CO2 : out  STD_LOGIC;
           PO1 : out  STD_LOGIC;
           PO2 : out  STD_LOGIC;
           DO1 : out  STD_LOGIC;
           DO2 : out  STD_LOGIC;
           S11 : out  STD_LOGIC;
           S21 : out  STD_LOGIC;
           S22 : out  STD_LOGIC);
end Y;

architecture Behavioral of Y is

component CAS is port ( Pin : in  STD_LOGIC;
                        Din : in  STD_LOGIC;
                        X : in  STD_LOGIC;
                        Cin : in  STD_LOGIC;
                        Pout : out  STD_LOGIC;
                        R : out  STD_LOGIC;
                        Dout : out  STD_LOGIC;
                        Cout : out  STD_LOGIC);
                        end component;

architecture Behavioral of Y is

component CAS is port ( Pin : in  STD_LOGIC;
                        Din : in  STD_LOGIC;
                        X : in  STD_LOGIC;
                        Cin : in  STD_LOGIC;
                        Pout : out  STD_LOGIC;
                        R : out  STD_LOGIC;
                        Dout : out  STD_LOGIC;
                        Cout : out  STD_LOGIC);
                        end component;

signal R_h,R_i,R_j,R_k : std_logic;
signal Pout_h,Pout_i,Pout_j,Pout_k : std_logic;
signal Cout_h,Cout_i,Cout_j,Cout_k : std_logic;
signal z : std_logic_vector(3 downto 0);--dummy variable

begin

CAS_h : CAS port map(PI1 ,DI1 ,A11 ,Cout_i ,Pout_h ,S11 ,z(0) ,CO1 );
CAS_i : CAS port map(Pout_h ,DI2 ,A12 ,CI1 ,PO1 ,R_i ,z(1) ,Cout_i );
CAS_j : CAS port map(PI2 ,DI1 ,R_i ,Cout_k ,Pout_j ,S21 ,z(2) ,CO2 );
CAS_k : CAS port map(Pout_j ,DI2 ,A22 ,CI2 ,PO2 ,S22 ,Z(3) ,Cout_k );

DO1 <= DI1;
DO2 <= DI2;

end Behavioral;
```

دو تصویر بالا کد مربوط به بلوک Y را نمایش می‌دهند. این ماژول تمام ۱۸ ورودی و خروجی را به صورت یک بیت منطقی دریافت می‌کند. علت پیاده سازی به این صورت این است که به علت زیاد بودن ارتباط ها در کد اصلی یعنی محاسبه کنند جذر بهتر است برای راحتی نوشتن کد به این صورت پیاده سازی شود.

سیگنال های در بخش خط های ۳۹ تا ۴۱ نوشته شده اند برای برقراری ارتباط های داخلی ماژول می‌باشند.

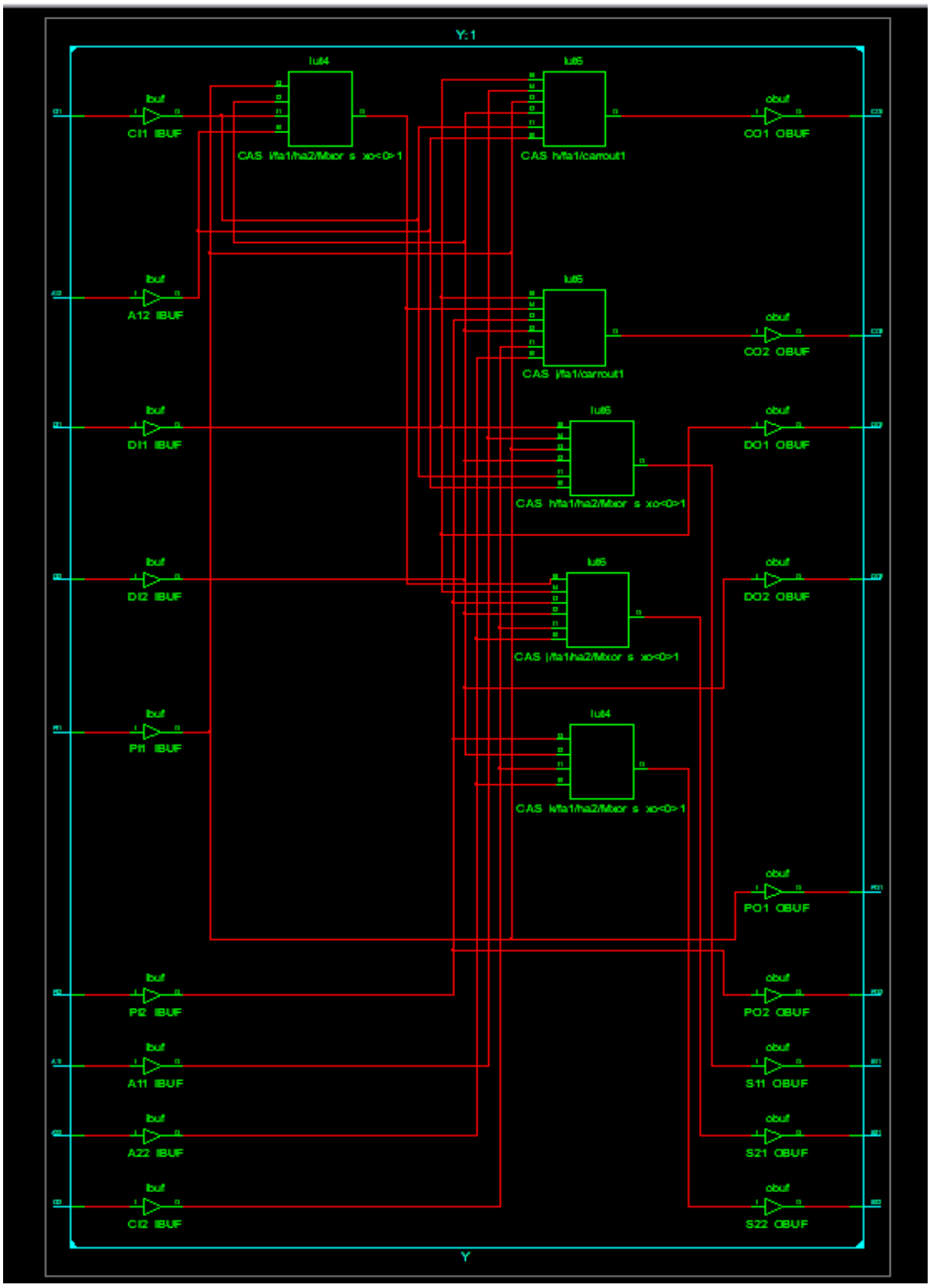
متغییر z که در خط ۴۲ نوشته شده است یک متغییر غیر مهم است. این متغییر برای جاهایی استفاده می‌شود که پایه خروجی و ورودی قرار نیست به جایی متصل شود و مقدار آن اهمیتی ندارد.

بقیه کد پیاده سازی معماری و وصل کردن ورودی ها و خروجی های هر ماژول CAS با استفاد دستور Port mapاست.



تصویر روبه‌رو شماتیک (RTL) یک ماژول Y را نمایش می‌دهد.

همین طور که مشخص است مانند معماری داده شده در پروژه است.



تصویر Technology Schematic بلوک Y

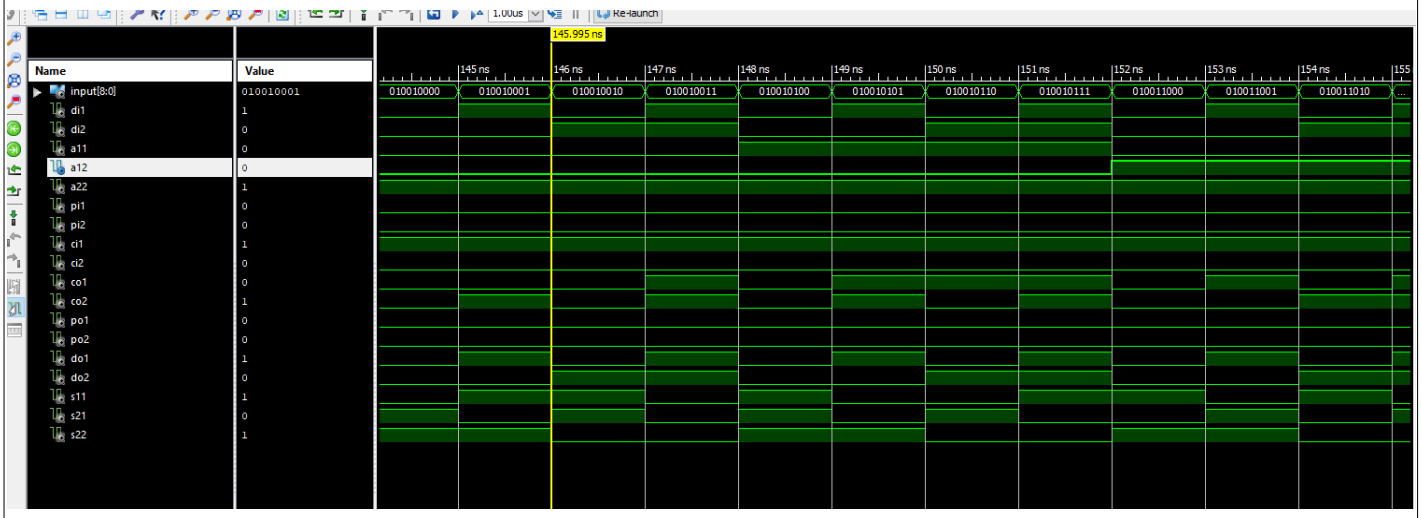
گزارش سطح مصرف FPGA – بلوک Y

```
================================================================================
*                            Design Summary
================================================================================

Top Level Output File Name         : Y.ngc

Primitive and Black Box Usage:
------------------------------
# BELS                             : 6
#      LUT4                        : 2
#      LUT6                        : 4
# IO Buffers                       : 18
#      IBUF                        : 9
#      OBUF                        : 9

Device utilization summary:
---------------------------

Selected Device : 7a100tcsg324-3


Slice Logic Utilization:
 Number of Slice LUTs:                    6  out of  63400      0%
    Number used as Logic:                 6  out of  63400      0%

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:      6
   Number with an unused Flip Flop:       6  out of      6    100%
   Number with an unused LUT:             0  out of      6      0%
   Number of fully used LUT-FF pairs:     0  out of      6      0%
   Number of unique control sets:         0

IO Utilization:
 Number of IOs:                          18
 Number of bonded IOBs:                  18  out of    210      8%

Specific Feature Utilization:

---------------------------
Partition Resource Summary:
---------------------------

  No Partitions were found in this design.

---------------------------
```

تصاویر بالا منابع مصرف شده را در بلوک Y را نمایش می‌دهد.

- همان طور که مشخص است 6 عدد از LUT ها از 63400 مصرف شده است که تقریبا برار صفر درصد کل تعداد LUT ها می‌باشد.
- همان طور که مشخص است ۱۸ عدد IO ها از ۲۱۰ تا مصرف شده است که تقریبا برابر ۸ درصد کل می‌باشد.

تست بنچ بلوکY



تصویر بالا بخشی از خروجی شبیه سازی تست بنچ Y می‌باشد. تمام نتایج همان گونه است که انتظار می‌رفت

```
64  BEGIN
65
66      -- Instantiate the Unit Under Test (UUT)
67      uut: Y PORT MAP (
68             DI1 => DI1,
69             DI2 => DI2,
70             A11 => A11,
71             A12 => A12,
72             A22 => A22,
73             PI1 => PI1,
74             PI2 => PI2,
75             CI1 => CI1,
76             CI2 => CI2,
77             CO1 => CO1,
78             CO2 => CO2,
79             PO1 => PO1,
80             PO2 => PO2,
81             DO1 => DO1,
82             DO2 => DO2,
83             S11 => S11,
84             S21 => S21,
85             S22 => S22
86           );
87
88
89      -- Stimulus process
90      stim_proc: process
91      begin
92         for i in 0 to 511 loop
93           input <= std_logic_vector(to_unsigned(i, 9));
94           wait for 0.000000001 ps;
95           DI1 <= input(0);
96           DI2 <= input(1);
97           A11 <= input(2);
98           A12 <= input(3);
99           A22 <= input(4);
100          PI1 <= input(5);
101          PI2 <= input(6);
102          CI1 <= input(7);
103          CI2 <= input(8);
104          wait for 1 ns;
105        end loop;
106
107        wait;
108     end process;
109
110 END;
111
```

تصویر روبه‌رو کد مربوط به تست بنچ ماژول Y را نشان می‌دهد.

در بخش سیگنال ها سیگنالی اضافه به نام input به صورت آرایه منطقی قرار گرفته است. این متغییر در خط ۹۳ مقدار دهی می‌شود و از المان های آن برای مقدار دهی به ورودی های ماژول استفاده می‌شود.

حلقه for تعرف شده در خط ۹۲ این امکان را فراهم می‌کند تا تمام حالات ممکن برای ورودی این ماژول ایجاد شود و به ورودی ها داده شود.

نکته: در خط ۹۴ بعد از مقدار دهی به متغییر input یک زمان بسیار کوچک برای wait در نظر گرفته شده است. این تاخیر برای این است که ابتدا مقدار دهی به input داده شود و سپس به ورودی های ماژول انجام شود. اگر این تاخیر گذاشته نشود

اعملیات مقدار دهی به صورت موازی انجام می‌شود و رووری ها یک سیکل عقب تر از input قرار می‌گرند. این تاخیر مانع این مشکل می‌شود.

پیاده‌سازی جذگیر

در این بخش پیاده سازی بخش اصلی تمرین عملی دوم یعنی جذرگیرنده شرح داده می‌شود. جذرگیرنده مانند معماری که در صورت پروژه داده شده است پیاده سازی شده است. با توجه به توضیحات تدرسیار در مورد پروژه ورودی این ماژول یک عدد ۱۶ بیتی می‌باشد. ورودی ها با A نمایش داده می‌شوند که با ارزشترین بیت، بیت A1 می‌باشد و کم ارزشترین بیت، بیت A16 می باشد. خروجی ۸ بیتی می‌باشد که با q نمایش داده می‌شود که با ارزشترین بیت، بیت q1 می‌باشد و کم ارزشترین بیت، بیت q8 می باشد.

پس با توجه به این توضیحات برای تعرف بردار منطقی(LOGIC_VETOR) A و q بجای downto از to برای مشخص کردن المان ها استفاده می‌کنیم.

```
2
3  library IEEE;
4  use IEEE.STD_LOGIC_1164.ALL;
5  entity Root is
6      Port ( A : in  STD_LOGIC_VECTOR(1 TO 16);
7             q : out  STD_LOGIC_VECTOR(1 TO 8);
8             R : out  STD_LOGIC_VECTOR(16 DOWNTO 1));
9  end Root;
10
11 architecture Behavioral of Root is
12
13 component X is port( DI1 : in  STD_LOGIC;
14                      A0  : in  STD_LOGIC;
15                      A1  : in  STD_LOGIC;
16                      A2  : in  STD_LOGIC;
17                      A3  : in  STD_LOGIC;
18                      A4  : in  STD_LOGIC;
19                      PI1 : in  STD_LOGIC;
20                      PI2 : in  STD_LOGIC;
21                      R0  : out  STD_LOGIC;
22                      R1  : out  STD_LOGIC;
23                      R2  : out  STD_LOGIC;
24                      R3  : out  STD_LOGIC;
25                      R4  : out  STD_LOGIC;
26                      DO1 : out  STD_LOGIC;
27                      DO2 : out  STD_LOGIC;
28                      CO1 : out  STD_LOGIC;
29                      CO2 : out  STD_LOGIC);
30                      end component;
31
32 component Y is port( DI1 : in  STD_LOGIC;
33                      DI2 : in  STD_LOGIC;
34                      A11 : in  STD_LOGIC;
35                      A12 : in  STD_LOGIC;
36                      A22 : in  STD_LOGIC;
37                      PI1 : in  STD_LOGIC;
38                      PI2 : in  STD_LOGIC;
39                      CI1 : in  STD_LOGIC;
40                      CI2 : in  STD_LOGIC;
41                      CO1 : out  STD_LOGIC;
42                      CO2 : out  STD_LOGIC;
43                      PO1 : out  STD_LOGIC;
44                      PO2 : out  STD_LOGIC;
45                      DO1 : out  STD_LOGIC;
46                      DO2 : out  STD_LOGIC;
47                      S11 : out  STD_LOGIC;
48                      S21 : out  STD_LOGIC;
49                      S22 : out  STD_LOGIC);
50                      end component;
```

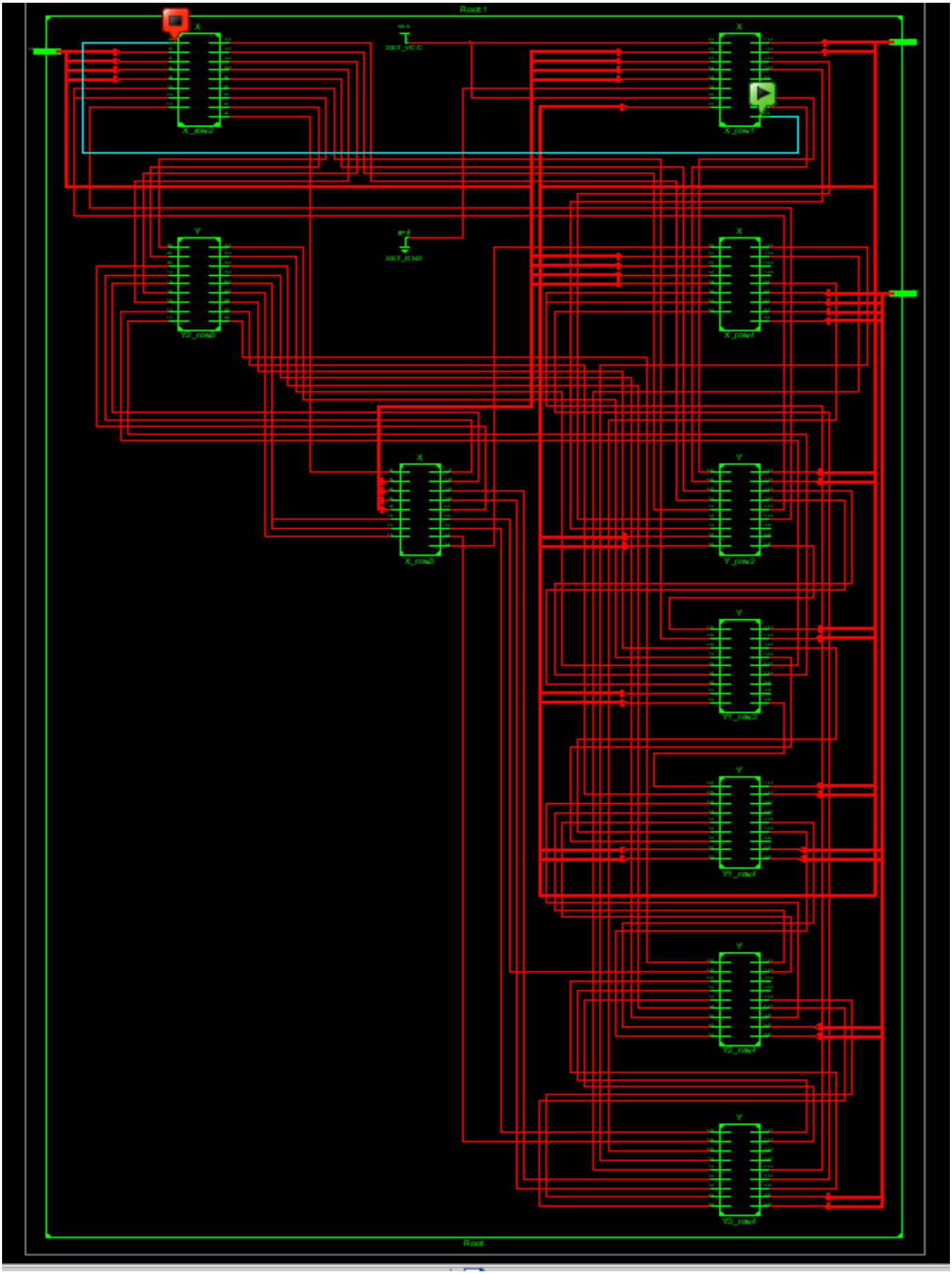
کد جذگیرنده بخش اول

```
51
52
53  signal R0_X_row ,R1_X_row ,R2_X_row ,R3_X_row ,R4_X_row ,DO1_X_row ,DO2_X_row ,CO1_X_row
54        ,CO2_X_row :STD_LOGIC_vector(4 downto 1);
55
56
57  signal  CO1_Y_row ,CO2_Y_row ,CO1_Y1_row ,CO2_Y1_row ,CO1_Y2_row ,CO2_Y2_row ,CO1_Y3_row ,CO2_Y3_row
58        ,DO1_Y_row ,DO2_Y_row ,DO1_Y1_row ,DO2_Y1_row ,DO1_Y2_row ,DO2_Y2_row ,DO1_Y3_row ,DO2_Y3_row
59        ,PO1_Y_row ,PO2_Y_row ,PO1_Y1_row ,PO2_Y1_row ,PO1_Y2_row ,PO2_Y2_row ,PO1_Y3_row ,PO2_Y3_row
60        ,S11_Y_row ,S11_Y1_row,S11_Y2_row ,S11_Y3_row ,S21_Y_row  ,S21_Y1_row ,S21_Y2_row ,S22_Y_row
61        ,S22_Y1_row,S22_Y2_row : STD_LOGIC_vector(4 downto 1);
62
63  signal z : STD_LOGIC_VECTOR(15 downto 1); --Dummy Variable
64
65  begin
66
67  --Row 1
68  X_row1 : X port map('0' ,'1' ,A(1) ,A(2) ,A(3) ,A(4) ,'1' ,CO1_X_row(1) ,Z(1) ,Z(2) ,R2_X_row(1) ,R3_X_row(1) , R4_X_row(1) , DO1_X_row(1) , DO2_X_row(1) , CO1_X_row(1) , CO2_X_row(1) );
69  q(1) <= CO1_X_row(1);
70  q(2) <= CO2_X_row(1);
71
72  --Row 2
73  Y_row2 : Y port map(DO1_X_row(1) ,DO2_X_row(1) ,R2_X_row(1) ,R3_X_row(1) ,R0_X_row(2) ,CO2_X_row(1) ,CO1_Y_row(2) ,CO1_X_row(2) ,CO2_X_row(2) ,CO1_Y_row(2) ,CO2_Y_row(2) ,PO1_Y_row(2) ,PO2_Y_row(2) ,
74  X_row2 : X port map(PO1_Y_row(2) ,R4_X_row(1) ,A(5) ,A(6) ,A(7) ,A(8) ,PO1_Y_row(2) ,PO2_Y_row(2) ,R0_X_row(2) ,R1_X_row(2) ,R2_X_row(2) ,R3_X_row(2) ,R4_X_row(2) ,DO1_X_row(2) ,DO2_X_row(2) ,CO1_X_r
75  q(3) <= CO1_Y_row(2);
76  q(4) <= CO2_Y_row(2);
77
78  --Row 3
79  Y1_row3 : Y port map(DO1_Y_row(2) ,DO2_Y_row(2) ,S22_Y_row(2) ,R1_X_row(2) ,S11_Y2_row(3) ,CO2_Y_row(2) ,CO1_Y1_row(3) ,CO1_Y2_row(3) ,CO2_Y2_row(3) ,CO1_Y1_row(3) ,CO2_Y1_row(3) ,PO1_Y1_row(3) ,PO2_
80  Y2_row3 : Y port map(DO1_X_row(2) ,DO2_X_row(2) ,R2_X_row(2) ,R3_X_row(2) ,R0_X_row(3) ,PO1_Y1_row(3) ,PO2_Y1_row(3) ,CO1_X_row(3) ,CO2_X_row(3) ,CO1_Y2_row(3) ,CO2_Y2_row(3) ,PO1_Y2_row(3) ,PO2_Y2_r
81  X_row3  : X port map(PO1_Y2_row(3) ,R4_X_row(2) ,A(9) ,A(10) ,A(11) ,A(12) ,PO1_Y2_row(3) ,PO2_Y2_row(3) ,R0_X_row(3) ,R1_X_row(3) ,R2_X_row(3) ,R3_X_row(3) ,R4_X_row(3) ,DO1_X_row(3) ,DO2_X_row(3) ,
82  q(5) <= CO1_Y1_row(3);
83  q(6) <= CO2_Y1_row(3);
84
85  --Row 4
86  Y1_row4 : Y port map(DO1_Y1_row(3) ,DO2_Y1_row(3) ,S22_Y1_row(3) ,S21_Y2_row(3) ,S11_Y2_row(4) ,CO2_Y1_row(3) ,CO1_Y1_row(4) ,CO1_Y2_row(4) ,CO2_Y2_row(4) ,CO1_Y1_row(4) ,q(8)  ,PO1_Y1_row(4) ,PO2_Y1
87  Y2_row4 : Y port map(DO1_Y2_row(3) ,DO2_Y2_row(3) ,S22_Y2_row(3) ,R1_X_row(3) ,S11_Y3_row(4) ,PO1_Y1_row(4) ,PO2_Y1_row(4) ,CO1_Y3_row(4) ,CO2_Y3_row(4) ,CO1_Y2_row(4) ,CO2_Y2_row(4) ,PO1_Y2_row(4) ,
88  Y3_row4 : Y port map(DO1_X_row(3) ,DO2_X_row(3) ,R2_X_row(3) ,R3_X_row(3) ,R0_X_row(4) ,PO1_Y2_row(4) ,PO2_Y2_row(4) ,CO1_X_row(4) ,CO2_X_row(4) ,CO1_Y3_row(4) ,CO2_Y3_row(4) ,PO1_Y3_row(4) ,PO2_Y3_r
89  X_row4  : X port map(PO1_Y3_row(4) ,R4_X_row(3) ,A(13) ,A(14) ,A(15) ,A(16) ,PO1_Y3_row(4) ,PO2_Y3_row(4) ,R0_X_row(4) ,R(13) ,R(14) ,R(15) ,R(16) ,Z(14) ,Z(15) ,CO1_X_row(4) ,CO2_X_row(4)  );
90  q(7) <= CO1_Y1_row(4);
91
92
93  end Behavioral;
94
```

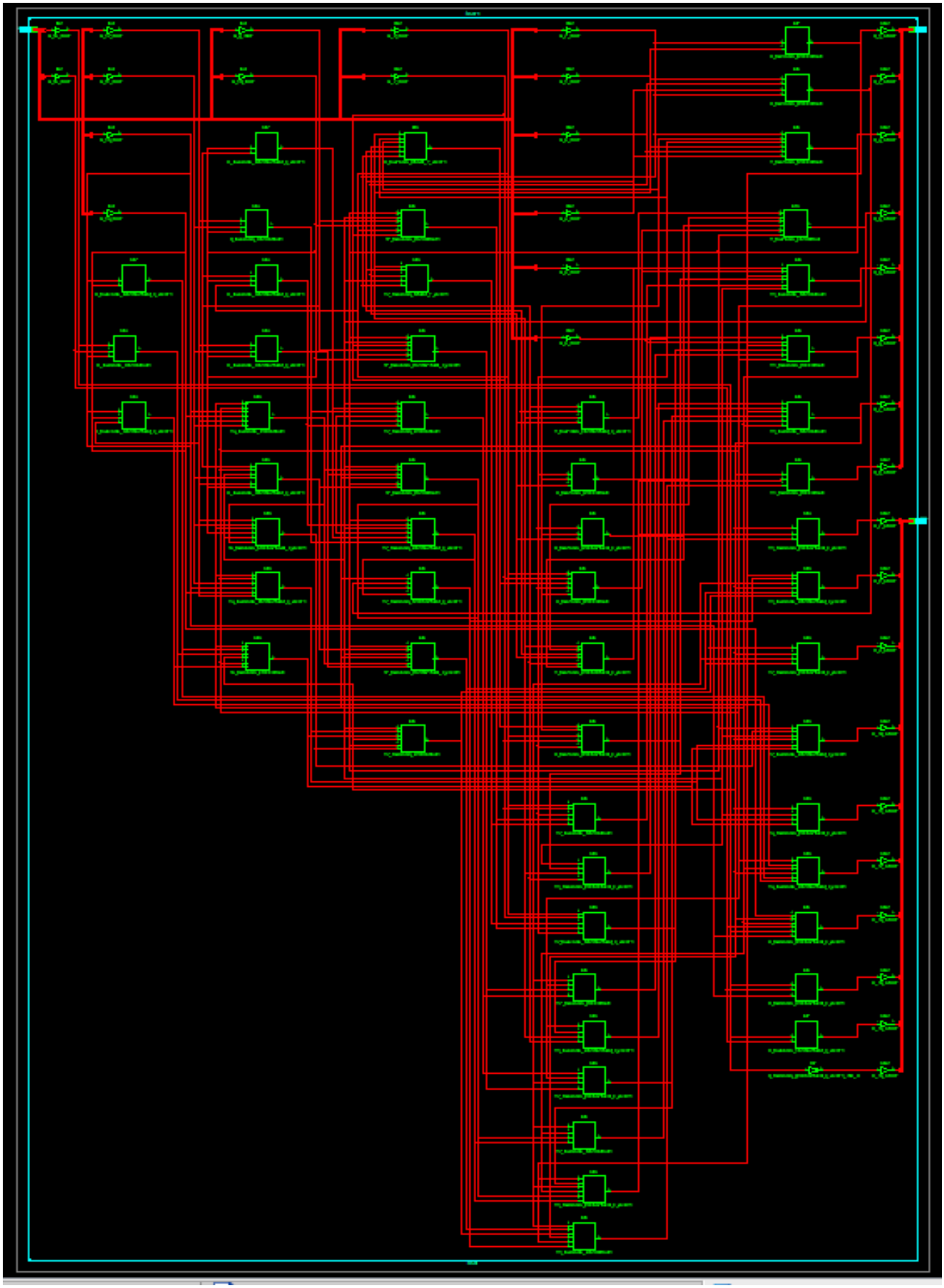
کد مربوط به جذگیرنده بخش دوم – به علت طولانی بودن کد بخشی از کد در تصویر نیفتاده است

در خط ۵۳ تا ۶۱ سیگنال های تعریف شده اند که برای مشخص کردن ارتباط های داخلی قطعه کابرد دارند. متغییر z که در خط ۶۳ نوشته شده است یک متغییر غیر مهم است. این متغییر برای جاهایی استفاده می‌شود که پایه خروجی و ورودی قرار نیست به جایی متصل شود و مقدار آن اهمیتی ندارد.

بقیه کد پیاده سازی معماری و وصل کردن ورودی ها و خروجی های هر ماژول CAS با استفاد دستور Port mapاست.



تصویر شماتیک (RTL) از جذرگیرنده



تصویر Technology Schematic از جذرگیرنده

گزارش سطح مصرف FPGA

```
=========================================================================
*                            Design Summary
=========================================================================

Top Level Output File Name         : Root.ngc

Primitive and Black Box Usage:
------------------------------
# BELS                             : 55
#       INV                        : 1
#       LUT2                       : 4
#       LUT3                       : 10
#       LUT4                       : 10
#       LUT5                       : 5
#       LUT6                       : 25
# IO Buffers                       : 34
#       IBUF                       : 16
#       OBUF                       : 18

Device utilization summary:
---------------------------

Selected Device : 7a100tcsg324-3


Slice Logic Utilization:
 Number of Slice LUTs:                  55  out of  63400     0%
    Number used as Logic:               55  out of  63400     0%

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:    55
   Number with an unused Flip Flop:     55  out of     55   100%
   Number with an unused LUT:            0  out of     55     0%
   Number of fully used LUT-FF pairs:    0  out of     55     0%
   Number of unique control sets:        0

IO Utilization:
 Number of IOs:                         40
 Number of bonded IOBs:                 34  out of    210    16%

Specific Feature Utilization:

---------------------------
Partition Resource Summary:
---------------------------

  No Partitions were found in this design.
```

تصاویر بالا منابع مصرف شده را در جذرگیرنده را نمایش می‌دهد.

- همان طور که مشخص است ۵۵ عدد از LUT ها از 63400 مصرف شده است که تقریبا برار صفر درصد کل تعداد LUT ها می‌باشد.
- همان طور که مشخص است ۳۴ عدد IO ها از ۲۱۰ تا مصرف شده است که تقریبا برابر ۱۶ درصد کل می‌باشد.

تست بنچ

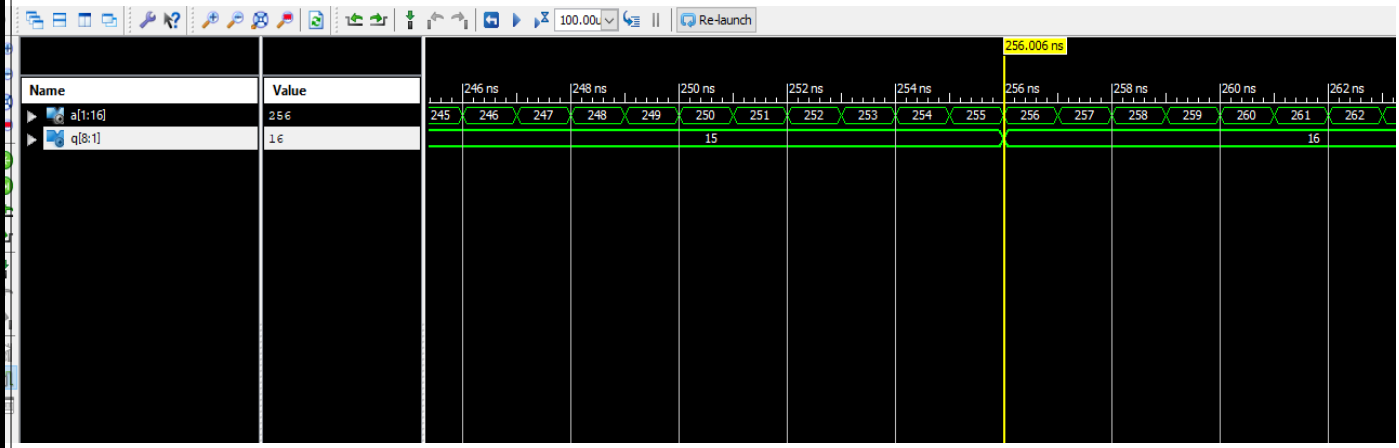
```
11
12  ENTITY Root_TB IS
13  END Root_TB;
14
15  ARCHITECTURE behavior OF Root_TB IS
16
17      -- Component Declaration for the Unit Under Test (UUT)
18
19      COMPONENT Root
20      PORT(
21           A : IN  std_logic_vector(1 to 16);
22           q : OUT  std_logic_vector(1 TO 8);
23           R : OUT  std_logic_vector(16 downto 1)
24          );
25      END COMPONENT;
26
27
28     --Inputs
29     signal A : std_logic_vector(1 to 16) := (others => '0');
30
31     --Outputs
32     signal q : std_logic_vector(8 downto 1);
33     -- appropriate port name
34
35  BEGIN
36
37     -- Instantiate the Unit Under Test (UUT)
38     uut: Root PORT MAP (
39            A => A,
40            q => q
41          );
42
43
44
45     -- Stimulus process
46     stim_proc: process
47     begin
48
49
50        for i in 0 to 65535 loop
51          A <= std_logic_vector(to_unsigned(i, 16));
52          wait for 1 ns;
53         end loop;
54
55
56        wait;
57     end process;
58
59  END;
```

کد روبه‌رو مربوط به تست بنچ جذرگیرنده می‌باشد. در این تست بنچ سیگنال R تعریف نشده است زیرا خروجی آن در این تمرین برای ما اهمیتی ندارد.

در بخش uut سیگنال ها به ورودی و خروجی های قطعه با دستور PORT MAP متصل شده‌اند.

در ادامه یم حلقه تعریف شده است که تمام حالات ممکن ورودی تولید می‌کند و جذرگیرنده می‌دهد. برای استفاده از دستور to_unsigned باید از کتابخانه ieee.numeric_std استفاده شود.قسمت اضافه کردن کتابخانه ها در تصویر نیفتاده است.

با توجه به زیاد بودن حالات ورودی اگر سیمولیشن در زمان پیشفرض خود یعنی ۱ میکرو ثانیه اجرا شود تمام حالا ممکن نمایش داده نمی‌شود زیرا زمان سیمولیشن برای نمایش تمام حالات کوتاه است. برای نمایش تمام حالات ورودی باید زمان سیمولیشن را به ۱۰۰ میکرو ثانیه افزایش دهیم و شبیه سازی را دوباره اجرا کنیم.



تصویر بالا بخشی از شبیه سازی را نمایش می‌دهد. نکته : نمایش اعداد از باینری به دسیمال تغییر داده شده است تا مقایسه درستی جواب ها راحت تر باشد. چون خروجی جرگیرنده عدد صحیح می‌باشد و بخش اعشاری ندارد خروجی جذرگیرنده به پایین تقریب زده می شود. برای مثال جذر عدد ۲۵۰ عدد ۱۵.۸ می‌باشد که خروجی جذرگیرنده عدد ۱۵ باینری را خروجی می‌دهد.