VHDL

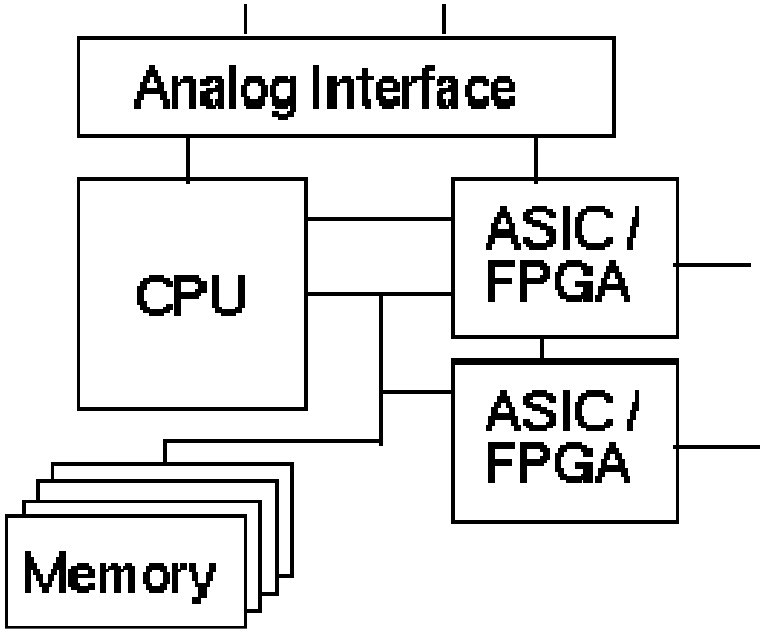
Ece Olcay Güneş, Berna Örs Yalçın

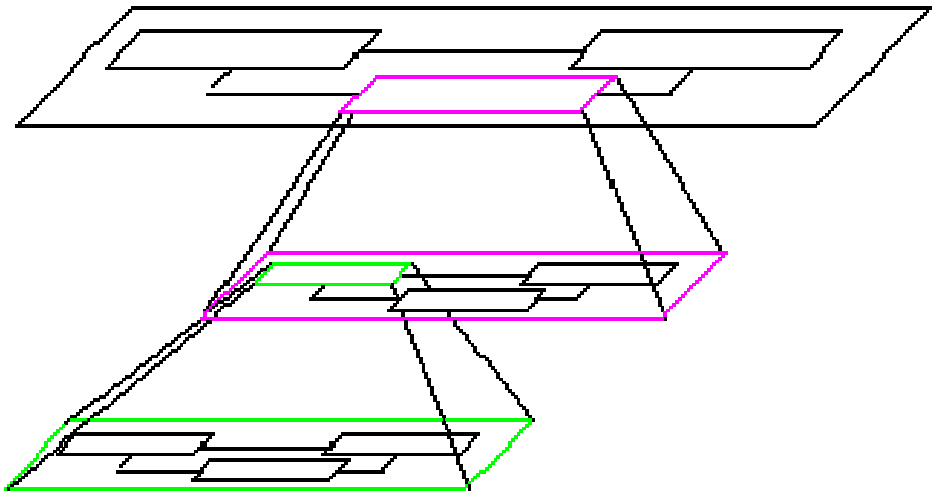
Giriş

- VHDL “VHSIC Hardware Description Language” in kısaltmasıdır.
- VHSIC “Very High Speed Integrated Circuits” in kısaltmasıdır.
- Devre çizimi yapmak yerine bir programlama dili ile devreyi tanımlamada kullanılır.
- Büyük sayısal devre tasarımında pek çok kullanım şekli vardır.
  - Açıklama (Documentation)
  - Doğrulama (Verification)
  - Gerçekleme (Synthesis)
- Üç farklı devre tanımlama yöntemi kullanılır:
  - Yapısal (Structural)
  - Veri Akışı (Data flow)
  - Davranışsal (Behavioral)
- Çoğu zaman bu üç yöntemin karışımı kullanılır.
- VHDL 1976 yılında IEEE tarafından geliştirilmiş bir standartdır.
- Dil pek çok yenilemeden geçmiştir. Şu anda 1993 yılı üretimi en çok kullanılan halidir.

Alt Bloklar

- Bir tasarımı daha anlaşılabilir ve değişikliklere açık hale getirebilmek için devreler alt bloklara ayrılır.
- Daha sonra bu bloklar bütün devreyi oluşturmak üzere uygun şekilde bağlanır.





Entity

VHDL'de her blok kendi başına bir devre olarak düşünülür ve **entity** olarak adlandırılır.

Entity: Verilen bir lojik fonksiyon için bütün giriş ve çıkışları tanımlar. Yani lojik fonksiyonun dış dünyayla bağlantısını tanımlar. Her VHDL tasarım mutlaka en az bir entity içerir.

entity entity tanımlayıcı **is**
port (işaret tanımlama);
end entity tanımlayıcı;

Port: Port giriş ve çıkış işaretidir. Port ifadesi, her işaret için, port tanımlayıcı, port yönü ve port veri tipini belirlemelidir. Port da 3 yön kullanılır. Giriş için **in**, çıkış için **out**, çift yönlü portlar için **inout** kullanılır. Pek çok değişik veri tipi kullanılabilir.

Veri tipleri:

bit: 0 ve 1 değerini alabilir
bit-vector: aynı isim altında bir dizi 0 ve 1'I göstermek için kullanılır.
integer: pozitif ya da negatif tamsayılara karşı düşer
natural: 0'dan başlayıp istenen bir limit değere kadar tamsayılar için kullanılır.
positive: 1'den başlayıp istenen bir limit değere kadar tamsayılar için kullanılır.
Boolean: Doğru ve yanlış olmak üzere 2 değerli bir veri tipidir.

Entity Örneği

```
entity Ornek is
    port (A,B: in bit; Z: out naturel range 0 to 31);
end Ornek;
```

- İlk satır yeni tanımlanacak devrenin ismini bildirir: Ornek
- Son satır tanımlamanın bittiğini gösterir.
- Aradaki satırlar devrenin giriş cıkışlarını tanımlar.
- Port tanımında her satır bir giriş-çıkış listesi, modunu ve veri tipini gösterir.
- Son satırın dışındaki her satır ; ile bitirilir. Port tanımının tamamı da ; ile bitirilir.

Architecture

Lojik fonksiyonun işlevi **architecture** tarafından belirlenir. Her **entity** için bir **architecture** tanımlanır.

architecture mimari ismi **of** tanımlayıcı ismi **is**

begin

lojik fonksiyonun yapacağı iş burada tanımlanır;

end mimari ismi;

Architecture Örneği

```
entity AND_Gate is
   port(A, B: in bit; X:out bit);
end entity AND_Gate;
Architecture dataflow of AND_Gate is
begin
  X <= A and B;
end dataflow;
```

- İlk satır dataflow isimli mimarinin AND_Gate isimli devreye ait olduğunu gösteriyor.
- **begin** ve **end** arasındaki satırlar AND_Gate in nasıl bir işlem gerçekleyeceğini belirtir.

Yapısal Tanımlamalar

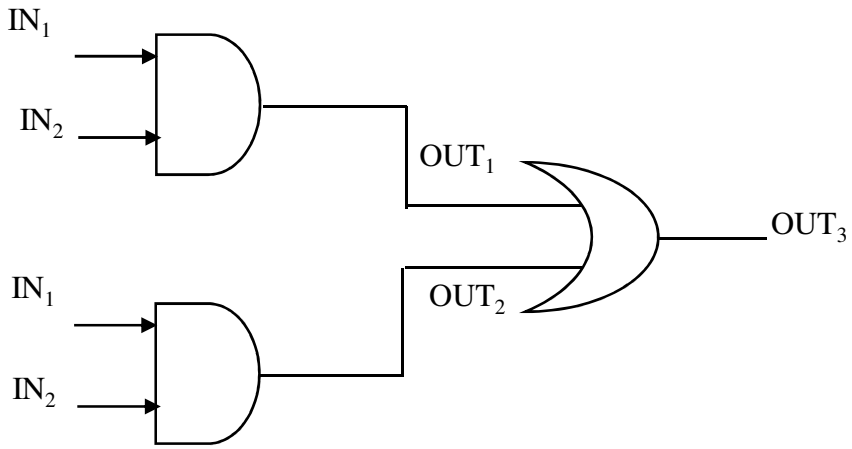
- Tasarımımızdaki temel bloklar **entity** ve karşılık gelen **architecture** kullanılarak tanımlandıktan sonra diğer tasarımlarda kullanılmak üzere birleştirilebilirler.
- Bu işlem yapılırken alt bloklar üst bloğun içinde **component** olarak tanımlanırlar.

component elemanın-adı **is**

port (uç tanımları)

end component;

Component Örneği



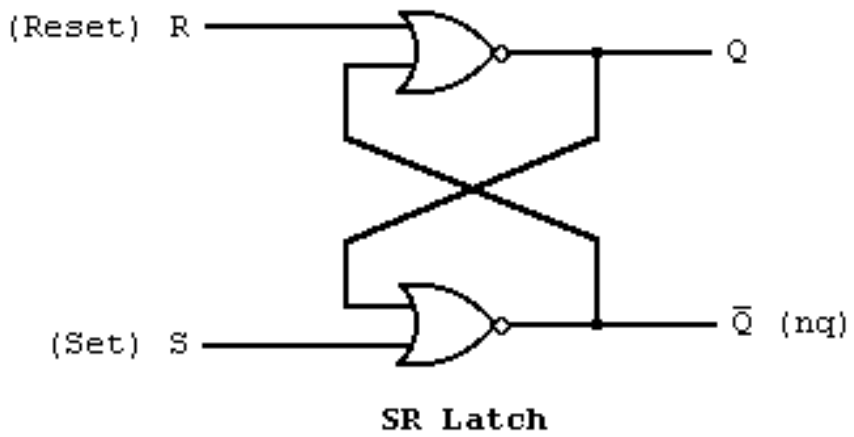
Ara bağlantılara karşı düşer. Entegre içinde dış bağlantısı olmayan elemanlar arası bağlantılardır.

```
entity AND-OR-logic is
    port(IN1, IN2, IN3, IN4: in bit; OUT3: out bit);
end AND-OR-logic;
architecture LogicOperation of AND-OR-logic is
 component AND_Gate is
    port(A, B:in bit; X:out bit);
 end component;
 component OR_Gate is
    port(A,B:in bit; X:out bit);
end component;
signal OUT1, OUT2:bit;
begin
 G1: AND_Gate
       port map (A⇒IN1, B⇒IN2,
                 X⇒OUT1);
 G2: AND_Gate
       port map (A⇒IN3, B⇒IN4,
                 X⇒OUT2);
 G3: OR_Gate
       port map (A⇒OUT1, B⇒OUT2,
                 X⇒OUT3);
end LogicOperation;
```

Veri Akışı Tanımlama

- Veri akışı tanımlamada temel blokların (örneğin *and* kapısı) girişlerinin ve çıkışlarının devre içinde nasıl bağlanacağı tanımlanır.
- İşaretlerin devre içinde nasıl akacağı tanımlanır.

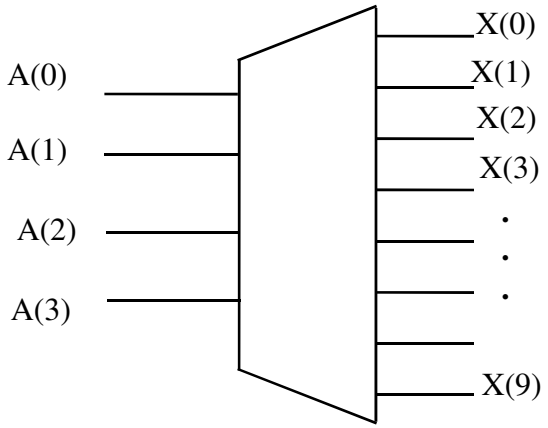
Veri Akışı Tanımlama Örneği 1



SR Latch

```
entity latch is
  port (s,r : in bit; q,nq :
out bit);
end latch;
architecture dataflow of
latch is
begin
  q<=r nor nq;
  nq<=s nor q;
end dataflow;
```

Sayıları BCD'den Decimal'e çeviren decoder tasarımı




```
 entity decoder is
     port (A: in bit-vector(0 to3); X:out bit-vector (0 to 9));
end decoder;

architecture BCD-to-decimal of decoder is
begin
    X(0)⇐not A(3) and not A(2) and not A(1) and not A(0);
    X(1)⇐not A(3) and not A(2) and not A(1) and A(0);
    X(2)⇐not A(3) and not A(2) and A(1) and not A(0);
    X(3)⇐not A(3) and not A(2) and A(1) and A(0);
    X(4)⇐not A(3) and A(2) and not A(1) and not A(0);
    X(5)⇐not A(3) and A(2) and not A(1) and A(0);
    X(6)⇐not A(3) and A(2) and A(1) and not A(0);
    X(7)⇐not A(3) and A(2) and A(1) and A(0);
    X(8)⇐A(3) and not A(2) and not A(1) and not A(0);
    X(9)⇐A(3) and not A(2) and not A(1) and A(0);
end BCD-to-decimal;
```

Not

- Bu örnekte birden fazla giriş çıkış söz konusu olduğu için bunlar VHDL'de bir dizi (bit-vector) veri tipi ile gösterilebilir.
- Genel olarak bit dizilerinin büyüklüğü 0'dan n'ye seçilir.
- 0 her zaman dizinin ilk elemanına karşı düşer.
- Bu nedenle n dizideki eleman sayısından 1 eksik olarak seçilmelidir.
- n+1 giriş veya çıkış dizi olarak bit-vector (0 to n) olarak gösterilir.
- Örnekte giriş bit-vector (0 to 3)
- Cıkış bit-vector (0 to 9) biçimindedir.

Kombinezonsal Devre Tasarımı Örneği

Elemanları iki bitlik olan iki matrisin çarpımını bulan devre tasarlanacaktır.

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$$

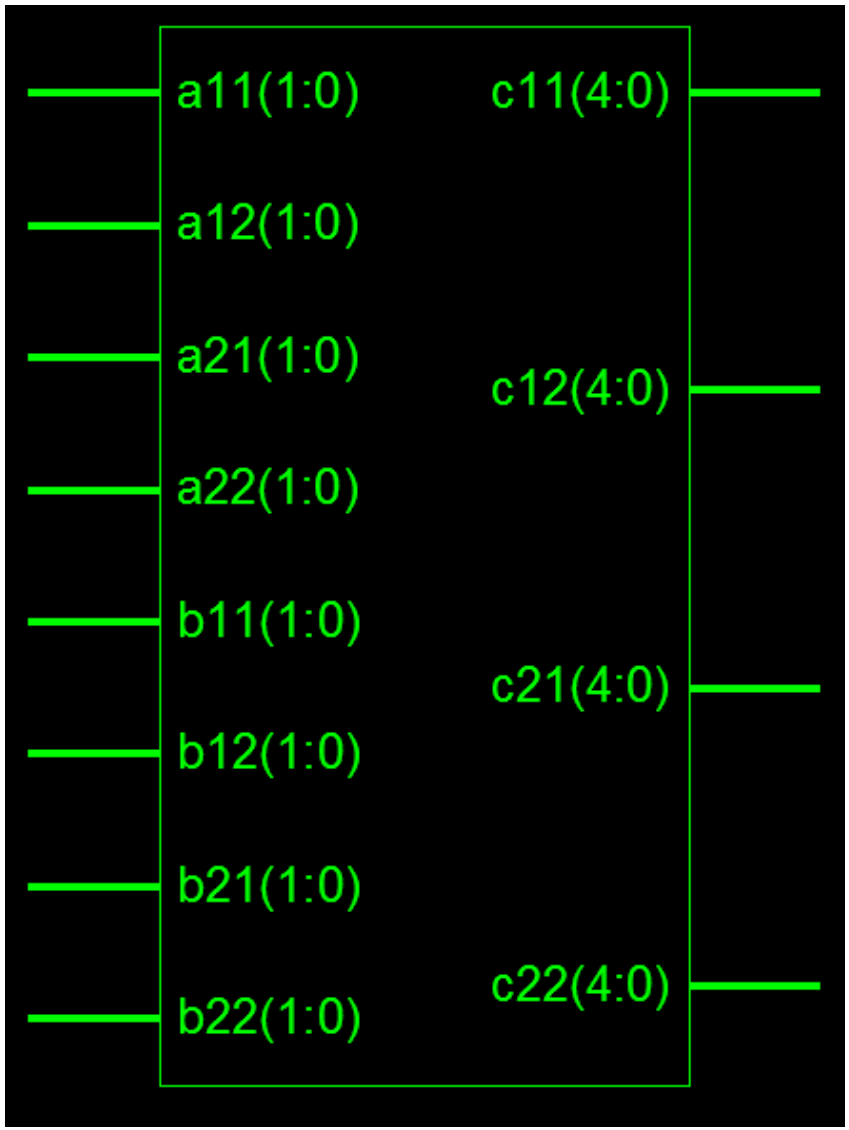
$$c_{11} = a_{11}b_{11} + a_{12}b_{21} \qquad c_{12} = a_{11}b_{12} + a_{12}b_{22}$$

$$c_{21} = a_{21}b_{12} + a_{22}b_{22} \qquad c_{22} = a_{21}b_{12} + a_{22}b_{22}$$

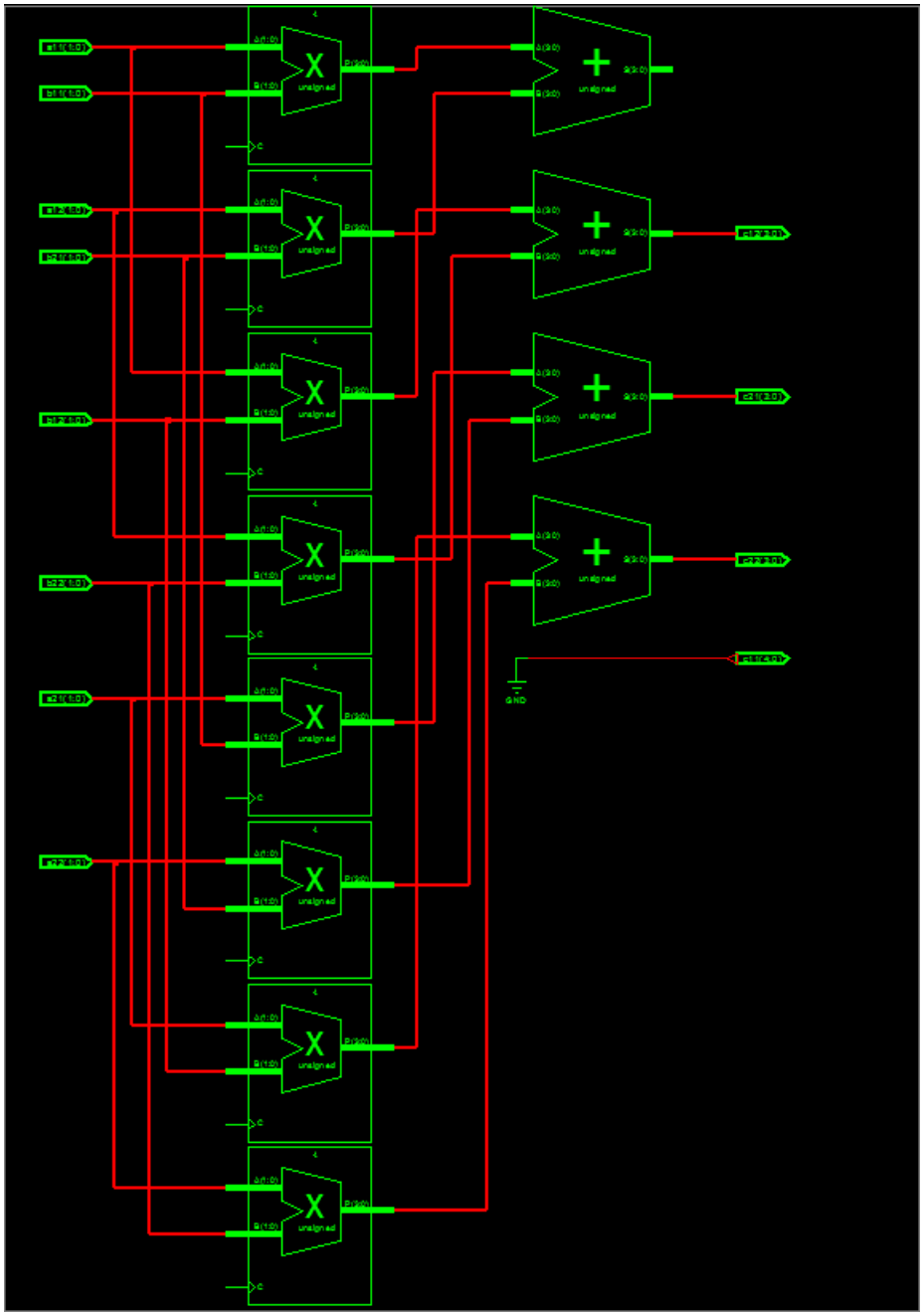
$a_{ij}b_{kl}$ Çarpımı 4-bitliktir.

c_{ij} 5-bitlik olmalıdır.

```
entity Matris_Carpici is
  port(a11 : in std_logic_vector(1 downto 0);
       a12 : in std_logic_vector(1 downto 0);
       a21 : in std_logic_vector(1 downto 0);
       a22 : in std_logic_vector(1 downto 0);
       b11 : in std_logic_vector(1 downto 0);
       b12 : in std_logic_vector(1 downto 0);
       b21 : in std_logic_vector(1 downto 0);
       b22 : in std_logic_vector(1 downto 0);
       c11 : out std_logic_vector(4 downto 0);
       c12 : out std_logic_vector(4 downto 0);
       c21 : out std_logic_vector(4 downto 0);
       c22 : out std_logic_vector(4 downto 0));
  end Matris_Carpici;
```

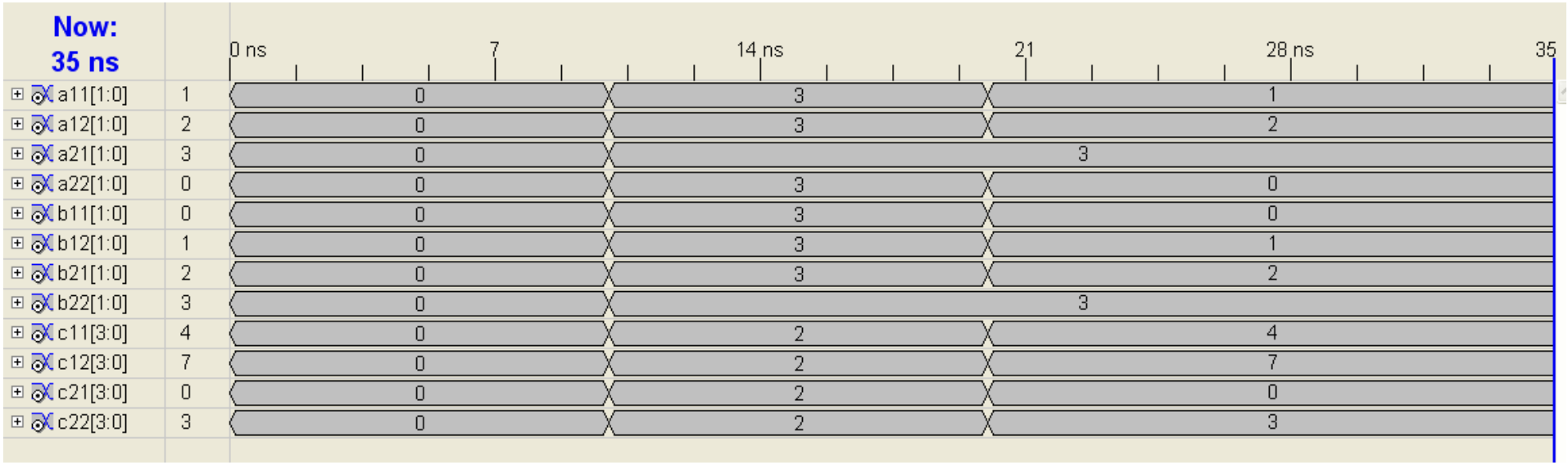

a11(1:0)
a12(1:0)
a21(1:0)
a22(1:0)
b11(1:0)
b12(1:0)
b21(1:0)
b22(1:0)
c11(4:0)
c12(4:0)
c21(4:0)
c22(4:0)

```
architecture dataflow of
    Matris_Carpici is
  begin
    c11 <= (a11 * b11) + (a12 * b21);
    c12 <= (a11 * b12) + (a12 * b22);
    c21 <= (a21 * b11) + (a22 * b21);
    c22 <= (a21 * b12) + (a22 * b22);
  end dataflow;
```

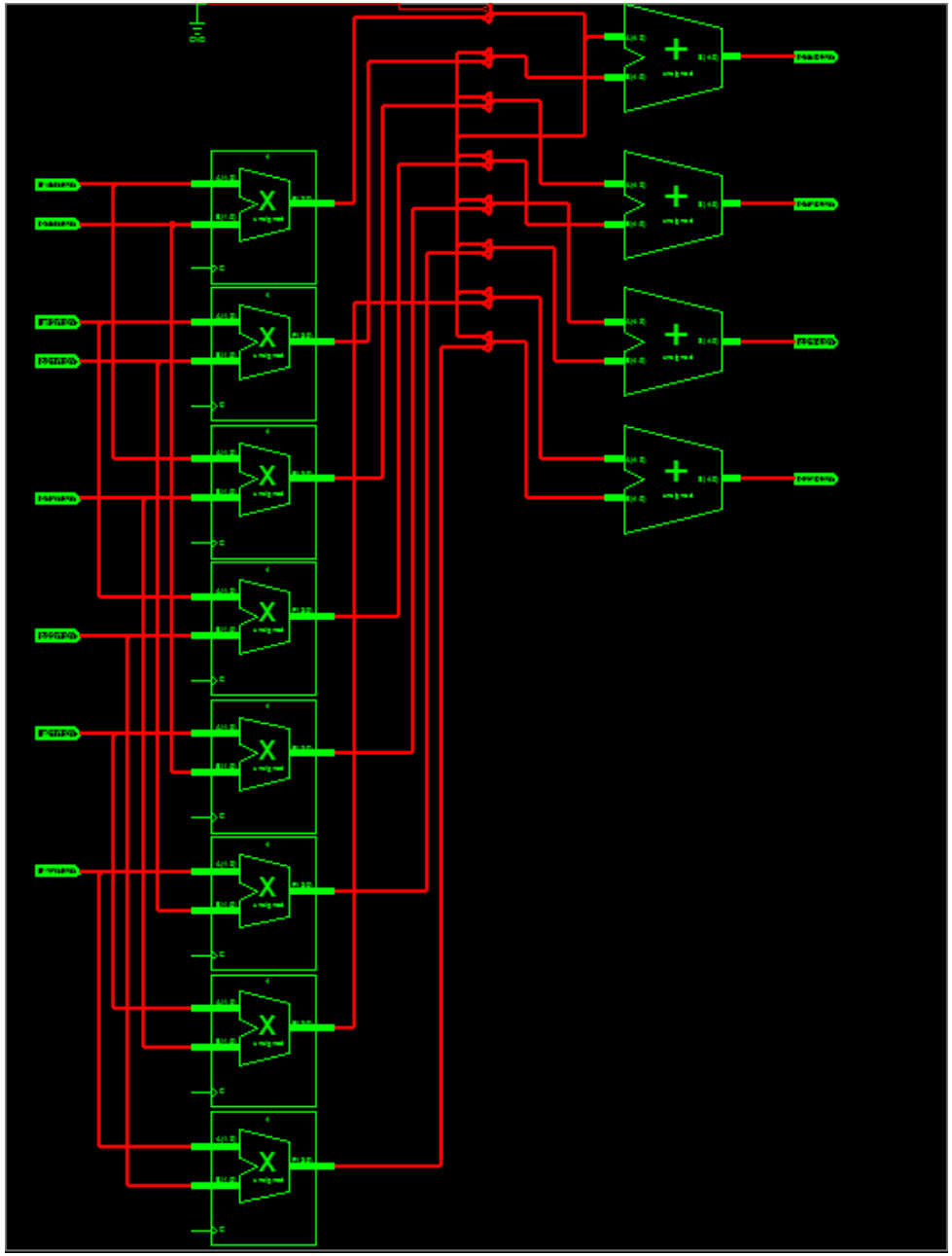


```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_unsigned.all;
USE ieee.numeric_std.ALL;
ENTITY matris_5bitlik_tb_vhd IS
END matris_5bitlik_tb_vhd;
ARCHITECTURE behavior OF matris_5bitlik_tb_vhd
IS
  COMPONENT Matris_Carpici
    PORT(a11,a12,a21,a22,b11,b12,b21,
              b22 : IN std_logic_vector(1 downto 0);
              c11,c12,c21,
              c22 : OUT std_logic_vector(3 downto 0));
  END COMPONENT;
  SIGNAL a11,a12,a21,a22,b11,b12,b21,b22:
        std_logic_vector(1 downto 0) := (others=>'0');
  SIGNAL c11,c12,c21,
        c22:std_logic_vector(3 downto 0);
```

```
BEGIN
uut: Matris_Carpici PORT MAP(
    a11 => a11, a12 => a12, a21 => a21, a22 => a22,
    b11 => b11, b12 => b12, b21 => b21, b22 => b22,
    c11 => c11, c12 => c12, c21 => c21, c22 => c22);
  tb : PROCESS
    BEGIN
      wait for 10 ns;
      a11 <= "11"; a12 <= "11"; a21 <= "11"; a22 <= "11";
      b11 <= "11"; b12 <= "11"; b21 <= "11"; b22 <= "11";
      wait for 10 ns;
      a11 <= "01"; a12 <= "10"; a21 <= "11"; a22 <= "00";
      b11 <= "00"; b12 <= "01"; b21 <= "10 "; b22 <= "11";
      wait;
  END PROCESS;
END;
```

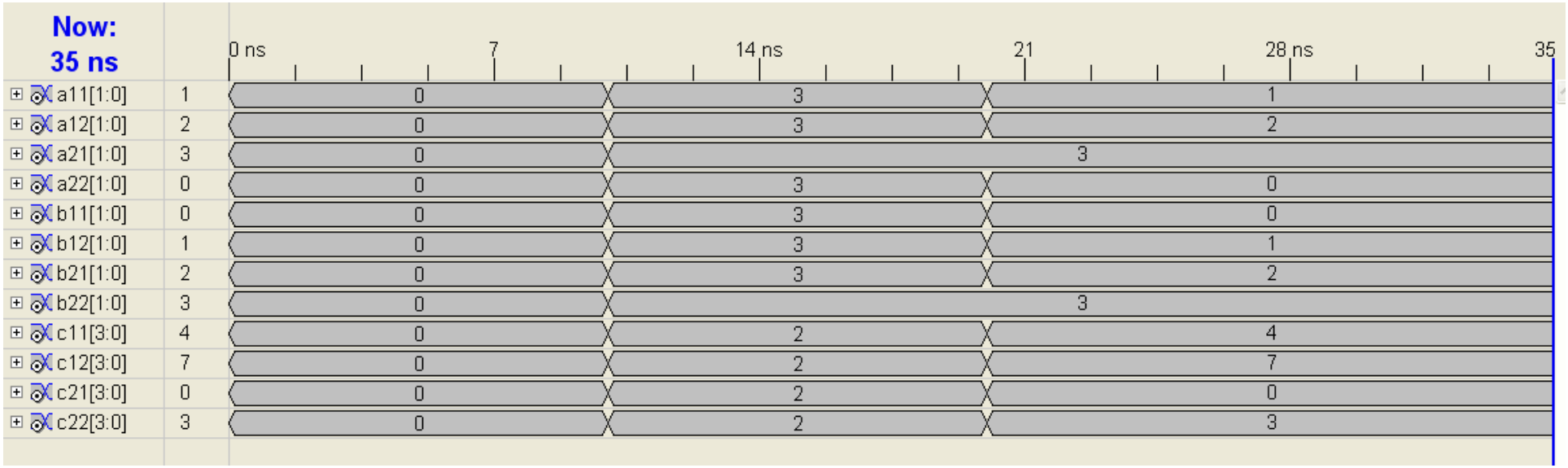


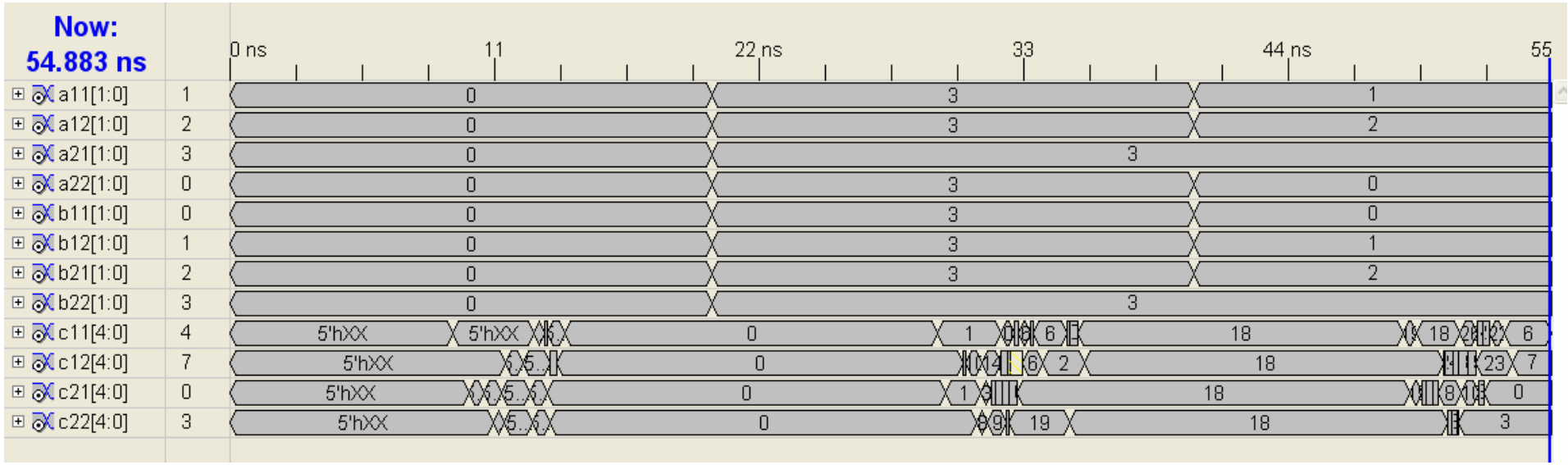
```
architecture dataflow of Matris_Carpici is
    signal ara1, ara2, ara3, ara4, ara5, ara6,
ara7, ara8:  std_logic_vector(4 downto 0);
  begin
    ara1 <= (a11 * b11);
    ara2 <= (a12 * b21);
    ara3 <= (a11 * b12);
    ara4 <= (a12 * b22);
    ara5 <= (a21 * b11);
    ara6 <= (a22 * b21);
    ara7 <= (a21 * b12);
    ara8 <= (a22 * b22);
    c11 <= ara1 + ara2;
    c12 <= ara3 + ara4;
    c21 <= ara5 + ara6;
    c22 <= ara7 + ara8;
  end dataflow;
```

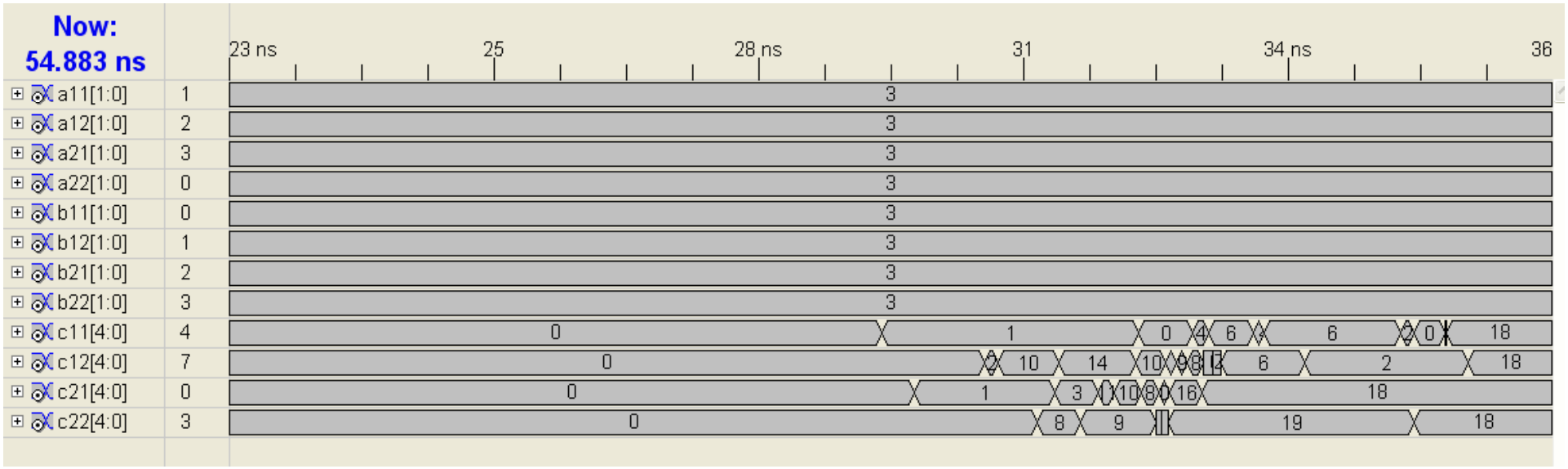


```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_unsigned.all;
USE ieee.numeric_std.ALL;
ENTITY matris_5bitlik_aracarpim_tb_vhd IS
END matris_5bitlik_aracarpim_tb_vhd;
ARCHITECTURE behavior OF
matris_5bitlik_aracarpim_tb_vhd IS
  COMPONENT Matris_Carpici
    PORT(a11,a12,a21,a22,b11,b12,b21,
              b22 : IN std_logic_vector(1 downto 0);
              c11,c12,c21,
              c22 : OUT std_logic_vector(3 downto 0));
  END COMPONENT;
  SIGNAL a11,a12,a21,a22,b11,b12,b21,b22:
        std_logic_vector(1 downto 0) := (others=>'0');
  SIGNAL c11,c12,c21,
        c22:std_logic_vector(3 downto 0);
```

```
BEGIN
uut: Matris_Carpici PORT MAP(
    a11 => a11, a12 => a12, a21 => a21, a22 => a22,
    b11 => b11, b12 => b12, b21 => b21, b22 => b22,
    c11 => c11, c12 => c12, c21 => c21, c22 => c22);
  tb : PROCESS
    BEGIN
      wait for 10 ns;
      a11 <= "11"; a12 <= "11"; a21 <= "11"; a22 <= "11";
      b11 <= "11"; b12 <= "11"; b21 <= "11"; b22 <= "11";
      wait for 10 ns;
      a11 <= "01"; a12 <= "10"; a21 <= "11"; a22 <= "00";
      b11 <= "00"; b12 <= "01"; b21 <= "10 "; b22 <= "11";
      wait;
  END PROCESS;
END;
```




Now:
54.883 ns
0 ns
11
22 ns
33
44 ns
55
a11[1:0]
a12[1:0]
a21[1:0]
a22[1:0]
b11[1:0]
b12[1:0]
b21[1:0]
b22[1:0]
c11[4:0]
c12[4:0]
c21[4:0]
c22[4:0]


Now:
54.883 ns
23 ns
25
28 ns
31
34 ns
36
a11[1:0]
a12[1:0]
a21[1:0]
a22[1:0]
b11[1:0]
b12[1:0]
b21[1:0]
b22[1:0]
c11[4:0]
c12[4:0]
c21[4:0]
c22[4:0]

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

package matris_tanimi is
  type satir_giris is array (integer range 1 to 2) of
                              std_logic_vector(1 downto 0);
  type satir_cikis is array (integer range 1 to 2) of
                              std_logic_vector(4 downto 0);
  type matris_giris is array (integer range 1 to 2) of satir_giris;
  type matris_cikis is array (integer range 1 to 2) of satir_cikis;
  type ara_carpim is array (integer range 1 to 8) of
std_logic_vector(4 downto 0);
end matris_tanimi;
```

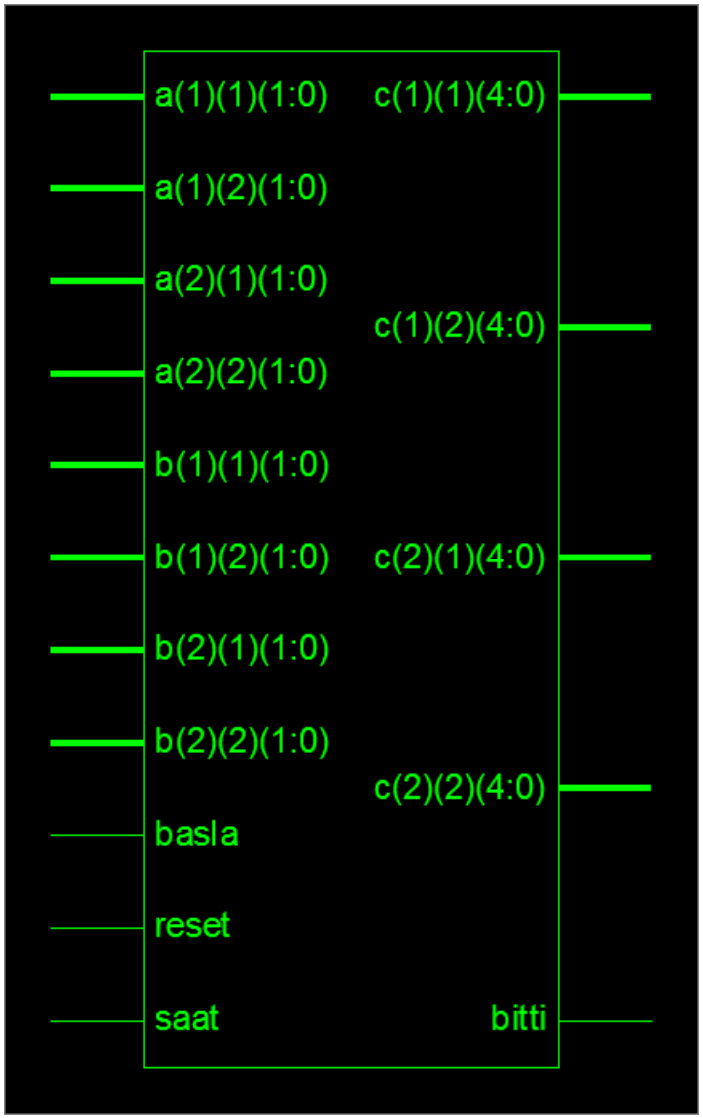

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use matris_tanimi.all;

entity Matris_Carpici is
  port(a : in matris_giris;
       b : in matris_giris;
       c : out matris_cikis);
  end Matris_Carpici;

  architecture dataflow of Matris_Carpici is
  begin
    process(a,b)
      variable ara: satir_cikis;
    begin
      for i in 1 to 2 loop
        for j in 1 to 2 loop
          for k in 1 to 2 loop
            ara(k) := a(i)(k) * b(k)(j);
          end loop;
          c(i)(j) <= ara(1) + ara(2);
        end loop;
    end loop;
  end process;
end dataflow;
```

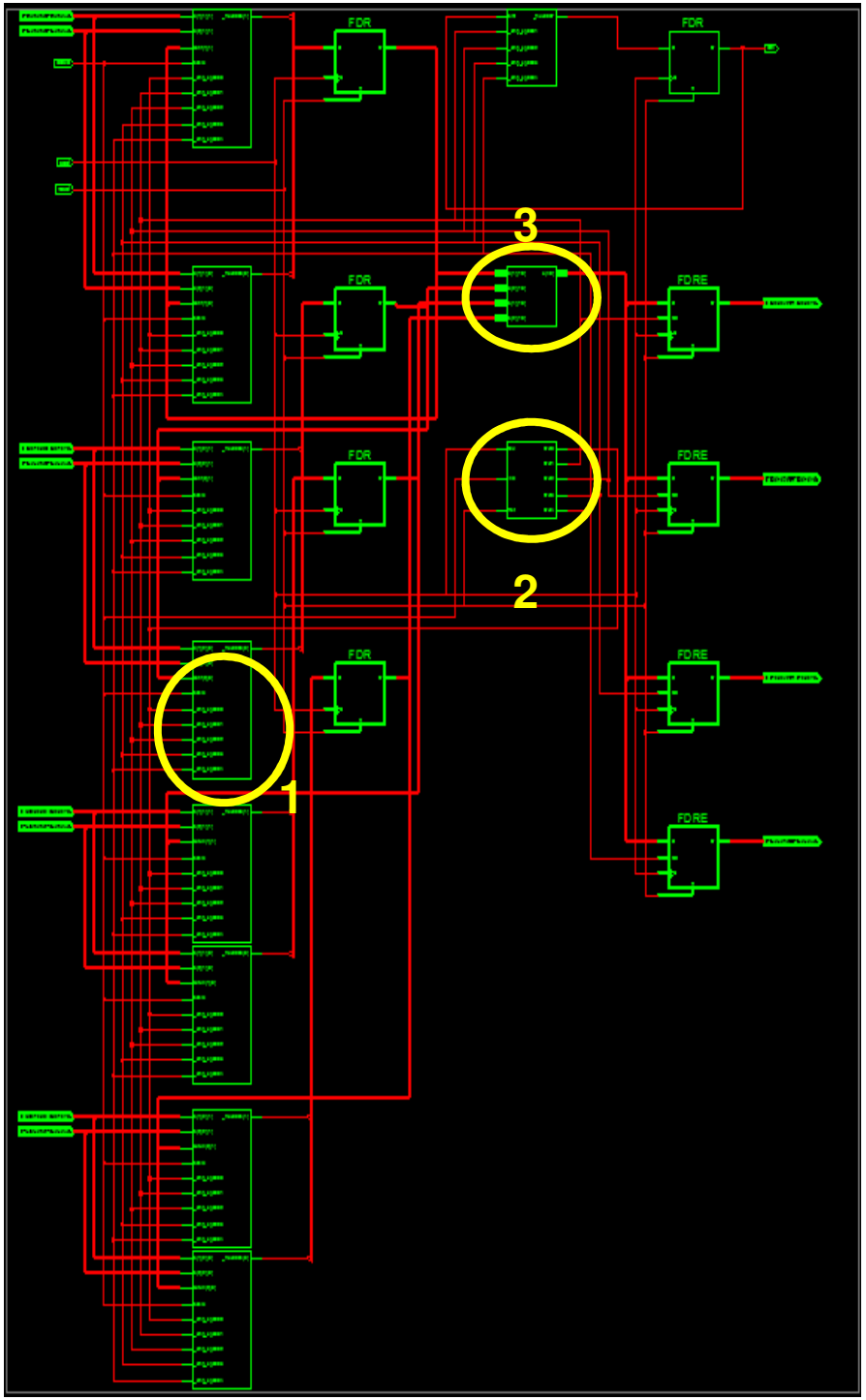
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use matris_tanimi.all;

entity Satir_Sutun is
  port(a : in satir_giris;
       b : in satir_giris;
       c : out std_logic_vector(4 downto 0));
  end Satir_Sutun;

  architecture dataflow of Satir_Sutun is
  begin
    process(a,b)
      variable ara: satir_cikis;
    begin
      for k in 1 to 2 loop
        ara(k) := a(k) * b(k);
      end loop;
      c <= ara(1) + ara(2);
    end process;
  end dataflow;
```

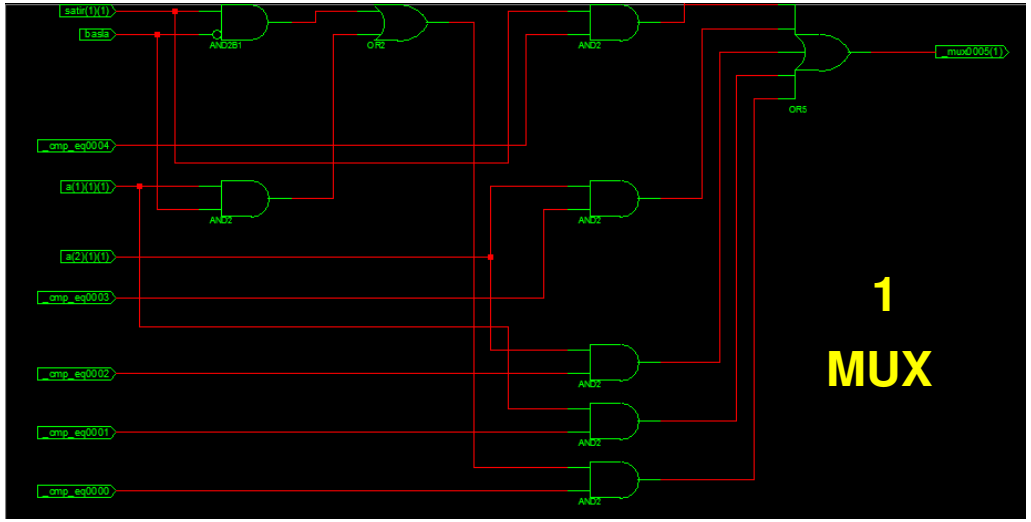
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use matris_tanimi.all;

entity Matris_Carpici is
  port(a : in matris_giris;
       b : in matris_giris;
       c : out matris_cikis);
  end Matris_Carpici;

  architecture structure of Matris_Carpici is
  component Satir_Sutun is
  port(a : in satir_giris;
       b : in satir_giris;
       c : out std_logic_vector(4 downto 0));
  end component;
  begin

  SS1: Satir_Sutun port map(a => a(1), b => b(1), c => c(1)(1));
  SS2: Satir_Sutun port map(a => a(1), b => b(2), c => c(1)(2));
  SS3: Satir_Sutun port map(a => a(2), b => b(1), c => c(2)(1));
  SS4: Satir_Sutun port map(a => a(2), b => b(2), c => c(2)(2));

end structure;
```

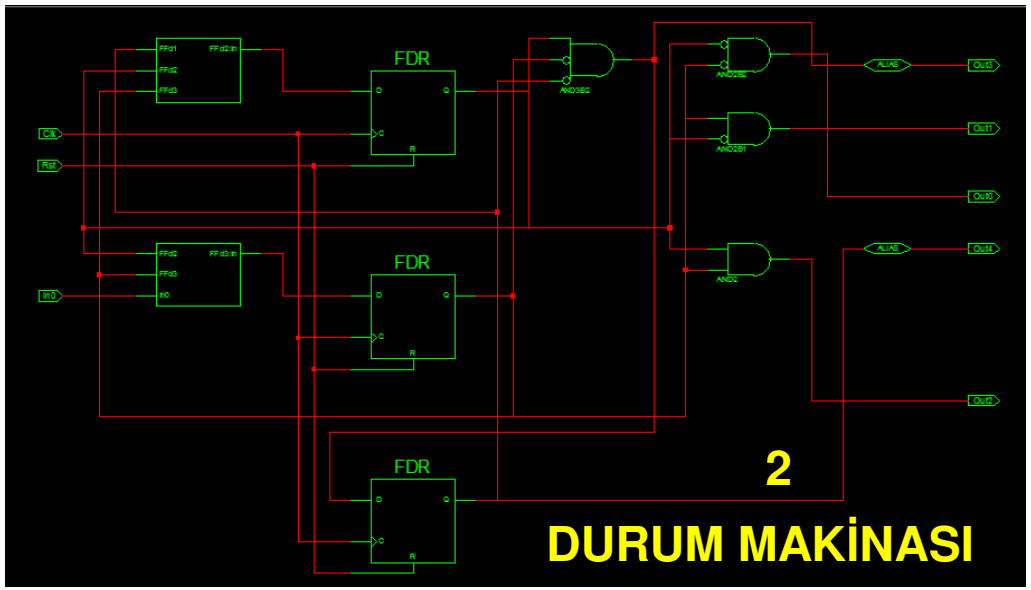
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use matris_tanimi.all;

entity Matris_Carpici is
  port(a : in matris_giris;
       b : in matris_giris;
       c : out matris_cikis);
  end Matris_Carpici;

  architecture structure of Matris_Carpici is
  component Satir_Sutun is
  port(a : in satir_giris;
       b : in satir_giris;
       c : out std_logic_vector(4 downto 0));
  end component;
  begin
  satir: for i in 1 to 2 generate
    sutun: for j in 1 to 2 generate
      SS: Satir_Sutun port map(a => a(i), b => b(j),
                               c => c(i)(j));
    end generate sutun;
  end generate satir;
end structure;
```

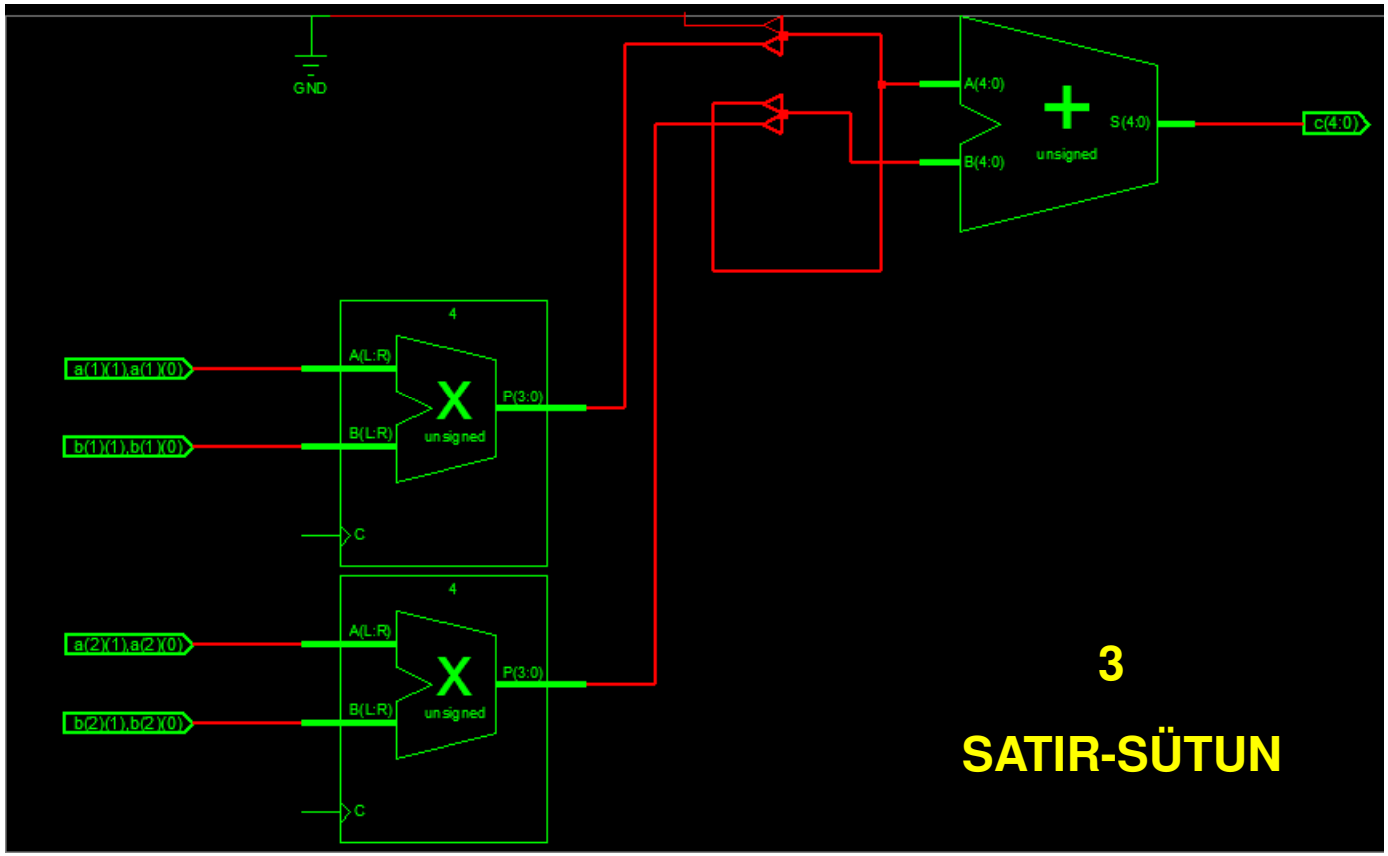
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use matris_tanimi.all;

entity Matris_Carpici is
  port(a : in matris_giris;
    b : in matris_giris;
    c : out matris_cikis;
    saat, reset, basla : in std_logic;
    bitti : out std_logic);
  end Matris_Carpici;

  architecture structure of Matris_Carpici is
    component Satir_Sutun is
    port(a,b : in satir_giris;
      c : out std_logic_vector(4 downto 0));
    end component;
    type state_type is (baslangic, durum1, durum2,
                        durum3, durum4);
    signal durum: state_type;
    signal satir, sutun : satir_giris;
    signal carpim : std_logic_vector(4 downto 0);
  begin
    SS: Satir_Sutun port map(a => satir, b => sutun,
                             c => carpim);
process(saat)
    begin
if rising_edge(saat) then
       if reset='1' then
      for i in 1 to 2 loop
        satir(i) <= (others => '0'); sutun(i) <= (others => '0');
         for j in 1 to 2 loop
            c(i)(j) <= (others => '0');
         end loop;
      end loop;
      bitti <= '0'; durum <= baslangic;
    else
      case durum is
        when baslangic =>
            bitti <= '0';
            if basla='1' then
              durum <= durum1; satir <= a(1); sutun <= b(1);
            else
              durum <= baslangic;
            end if;
        when durum1 =>
          durum <= durum2; satir <= a(1); sutun <= b(2);
          c(1)(1) <= carpim;
        when durum2 =>
          durum <= durum3; satir <= a(2); sutun <= b(1);
          c(1)(2) <= carpim;
        when durum3 =>
          durum <= durum4; satir <= a(2); sutun <= b(2);
          c(2)(1) <= carpim;
        when durum4 =>
          durum <= baslangic; c(2)(2) <= carpim; bitti <= '1';
       end case;
      end if;
    end if;
 end process;
end structure;
```


a(1)(1)(1:0)
a(1)(2)(1:0)
a(2)(1)(1:0)
a(2)(2)(1:0)
b(1)(1)(1:0)
b(1)(2)(1:0)
b(2)(1)(1:0)
b(2)(2)(1:0)
basla
reset
saat
c(1)(1)(4:0)
c(1)(2)(4:0)
c(2)(1)(4:0)
c(2)(2)(4:0)
bitti


FDR
FDRE
3
2
1

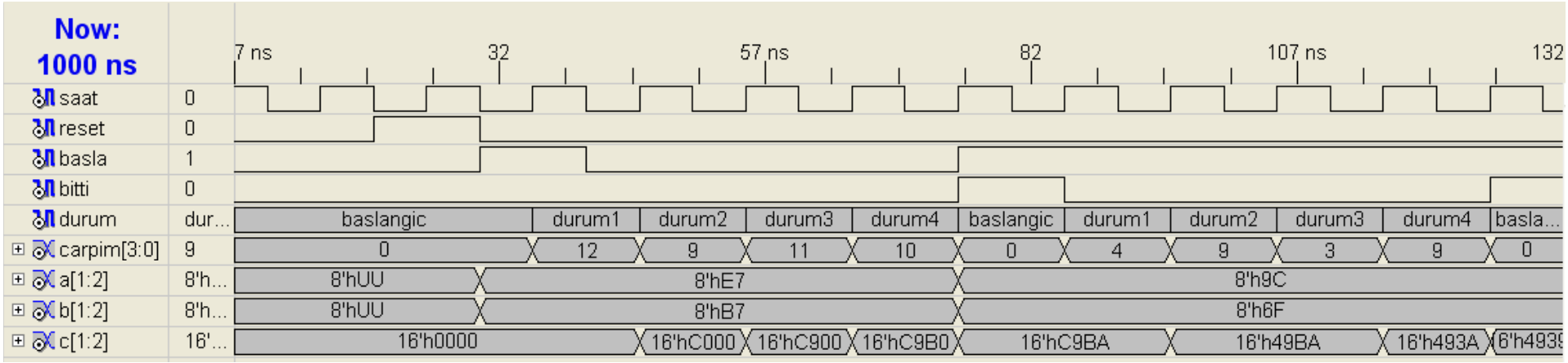

satir(1)(1)
basla
cmp_eq0004
ai(1)(1)(1)
ai(2)(1)(1)
cmp_eq0003
cmp_eq0002
cmp_eq0001
cmp_eq0000
mux0005(1)
AND2B1
OR2
AND2
OR5
1
MUX


FDR
Clk
Rst
In0
AND2B1
2
DURUM MAKİNASI


GND
A(4:0)
B(4:0)
unsigned
S(4:0)
c(4:0)
4
A(L:R)
B(L:R)
unsigned
P(3:0)
C
a(1)(1),a(1)(0)
b(1)(1),b(1)(0)
4
A(L:R)
B(L:R)
unsigned
P(3:0)
C
a(2)(1),a(2)(0)
b(2)(1),b(2)(0)
3
SATIR-SÜTUN

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_unsigned.all;
USE ieee.numeric_std.ALL;
use matris_tanimi.all;

ENTITY matris_state_tb_vhd IS
END matris_state_tb_vhd;

ARCHITECTURE behavior OF matris_state_tb_vhd IS
  COMPONENT Matris_Carpici
    PORT(a,b : IN matris_giris;
         saat,reset,basla : IN std_logic;
         c : OUT matris_cikis;
         bitti : OUT std_logic);
  END COMPONENT;

  SIGNAL saat :  std_logic := '0';
  SIGNAL reset :  std_logic := '0';
  SIGNAL basla :  std_logic := '0';
  SIGNAL a :  matris_giris;
  SIGNAL b :  matris_giris;

  SIGNAL c :  matris_cikis;
  SIGNAL bitti :  std_logic;

  constant cycle : time := 10 ns;

BEGIN
```

```
 uut: Matris_Carpici PORT MAP(a => a,b => b,c => c,
        saat => saat,reset => reset,basla => basla,bitti =>
bitti);

   saat <= not saat after cycle/2;

   tb : PROCESS
     BEGIN
       wait for 20 ns;
       reset <= '1';
       wait for cycle;
       reset <= '0';
       basla <= '1';

       a(1)(1) <= "11"; a(1)(2) <= "10";
       a(2)(1) <= "01"; a(2)(2) <= "11";
       b(1)(1) <= "10"; b(1)(2) <= "11";
       b(2)(1) <= "01"; b(2)(2) <= "11";
       wait for cycle;
       basla <= '0';
       wait until bitti='1';
       basla <= '1';
       a(1)(1) <= "10"; a(1)(2) <= "01";
       a(2)(1) <= "11"; a(2)(2) <= "00";
       b(1)(1) <= "01"; b(1)(2) <= "10";
       b(2)(1) <= "11"; b(2)(2) <= "11";
       wait until bitti='1';
       wait;
   END PROCESS;
END;
```


Now:
1000 ns
7 ns
32
57 ns
82
107 ns
132
saat
0
reset
0
basla
1
bitti
0
durum
dur...
baslangic
durum1
durum2
durum3
durum4
baslangic
durum1
durum2
durum3
durum4
basla...
carpim[3:0]
9
0
12
9
11
10
0
4
9
3
9
0
a[1:2]
8'h...
8'hUU
8'hE7
8'h9C
b[1:2]
8'h...
8'hUU
8'hB7
8'h6F
c[1:2]
16'...
16'h0000
16'hC000
16'hC900
16'hC9B0
16'hC9BA
16'h49BA
16'h493A

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_unsigned.all;
USE ieee.numeric_std.ALL;
use matris_tanimi.all;

entity Satir_Sutun is
  port(a : in satir_giris;
       b : in satir_giris;
       c : out std_logic_vector(4 downto 0));
end entity Satir_Sutun;

architecture structure of Satir_Sutun is
component Toplama is
  port (a : in std_logic_vector(3 downto 0);
        b : in std_logic_vector(3 downto 0);
        toplam: out std_logic_vector(4 downto 0));
end component;
component Carpma is
  port (a : in std_logic_vector(1 downto 0);
        b : in std_logic_vector(1 downto 0);
        carpim : out std_logic_vector(3 downto 0));
end component;
signal ara : ara_carpim;
  begin
  Carp: for i in 1 to 2 generate
    C1: Carpma port map (a => a(i),b => b(i),carpim =>
ara(i));
  end generate;
  T1: Toplama port map (a => ara(1),b => ara(2),toplam
=> c);
end architecture structure;
```

$$\begin{array}{cccc} & a_1 & a_0 & \\ \times & b_1 & b_0 & \\ \hline c_3 & c_2 & c_1 & c_0 \end{array}$$

$$\begin{array}{ccccc} a_3 & a_2 & a_1 & a_0 & \\ + \; b_3 & b_2 & b_1 & b_0 & \\ \hline c_4 & c_3 & c_2 & c_1 & c_0 \end{array}$$

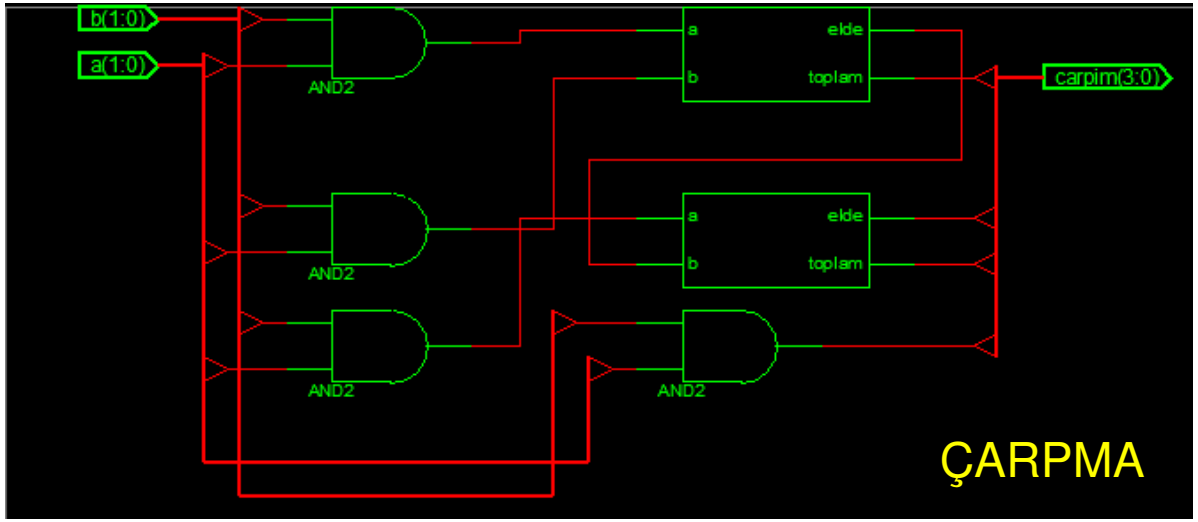
```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_unsigned.all;
USE ieee.numeric_std.ALL;
use matris_tanimi.all;

entity Satir_Sutun is
  port(a,b : in satir_giris;
         c : out std_logic_vector(4 downto 0);
         saat,reset,basla : in std_logic;
         bitti : out std_logic);
end entity Satir_Sutun;

architecture structure of Satir_Sutun is
component Toplama is
  port (a,b : in std_logic_vector(3 downto 0);
          toplam: out std_logic_vector(4 downto 0));
end component;
component Carpma is
  port (a,b : in std_logic_vector(1 downto 0);
         carpim : out std_logic_vector(3 downto 0));
end component;

type state_type is (baslangic, durum1, durum2);
signal durum: state_type;

signal carpim_giris1,
          carpim_giris2 : std_logic_vector(1 downto 0);
signal carpim, ara1,ara2 : std_logic_vector(3 downto 0);
begin
  process(saat)
  begin
    if rising_edge(saat) then
      if reset='1' then
        durum <= baslangic; carpim_giris1  <= (others=>'0');
        carpim_giris2  <= (others=>'0'); bitti <= '0';
      else
        case durum is
          when baslangic =>
            if basla='1' then
              durum <= durum1; carpim_giris1 <= a(1);
              carpim_giris2 <= b(1);
            else
              durum <= baslangic;
            end if;
          when durum1 =>
            durum <= durum2; carpim_giris1 <= a(2);
            carpim_giris2 <= b(2); ara1 <= carpim;
          when durum2 =>
            durum <= baslangic; ara2 <= carpim;
            bitti <= '1';
        end case;
    end if;
  end if;
end process;
  Carp: Carpma port map (a => carpim_giris1,b =>
carpim_giris2,carpim => carpim);
  Topla: Toplama port map (a => ara1,b => ara2,
                                          toplam => c);
end architecture structure;
```

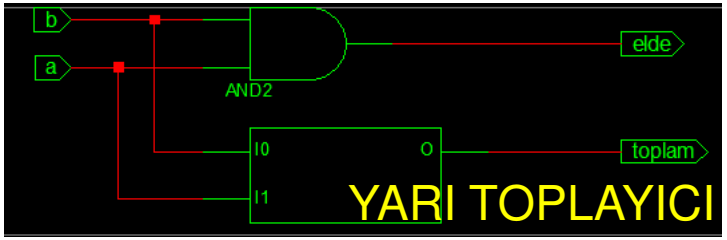
```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

entity Carpma is
  port (a : in std_logic_vector(1 downto 0);
        b : in std_logic_vector(1 downto 0);
        carpim : out std_logic_vector(3 downto 0));
end Carpma;

architecture structure of Carpma is
component Yari_Toplayici is
  port(a,b : in std_logic;
         toplam, elde : out std_logic);
end component;
signal ara_carpim : std_logic_vector(3 downto 0);
signal elde : std_logic;
  begin
    carpim(0) <= a(0) and b(0);
    ara_carpim(0) <= a(1) and b(0);
    ara_carpim(1) <= a(0) and b(1);
    ara_carpim(2) <= a(1) and b(1);
    YT1: Yari_Toplayici
        port map (a => ara_carpim(0),b => ara_carpim(1),
                  toplam => carpim(1),elde => elde);
    YT2: Yari_Toplayici
        port map (a => ara_carpim(2),b => elde,
                  toplam => carpim(2),elde => carpim(3));
end architecture structure;
```

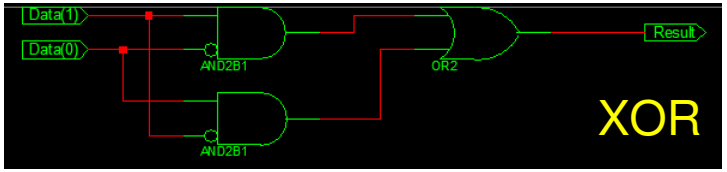
```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

entity Toplama is
  port (a : in std_logic_vector(3 downto 0);
        b : in std_logic_vector(3 downto 0);
        toplam: out std_logic_vector(4 downto 0));
end Toplama;

architecture structure of Toplama is
component Yari_Toplayici is
  port(a,b : in std_logic;
       toplam, elde : out std_logic);
end component;
component Tam_Toplayici is
  port(a,b, elde_giris : in std_logic;
       toplam, elde_cikis : out std_logic);
end component;
signal elde : std_logic_vector(2 downto 0);
begin
YT: Yari_Toplayici
        port map (a => a(0),b => b(0),
                  toplam => toplam(0),elde => elde(0));
TT: for i in 1 to 2 generate
    TT1: Tam_Toplayici
         port map (a => a(i),b => b(i),elde_giris => elde(i-1),
                   toplam => toplam(i),elde_cikis => elde(i));
end generate;
TT3: Tam_Toplayici
        port map (a => a(3),b => b(3),elde_giris => elde(2),
                  toplam => toplam(3),elde_cikis => toplam(4));
end architecture structure;
```

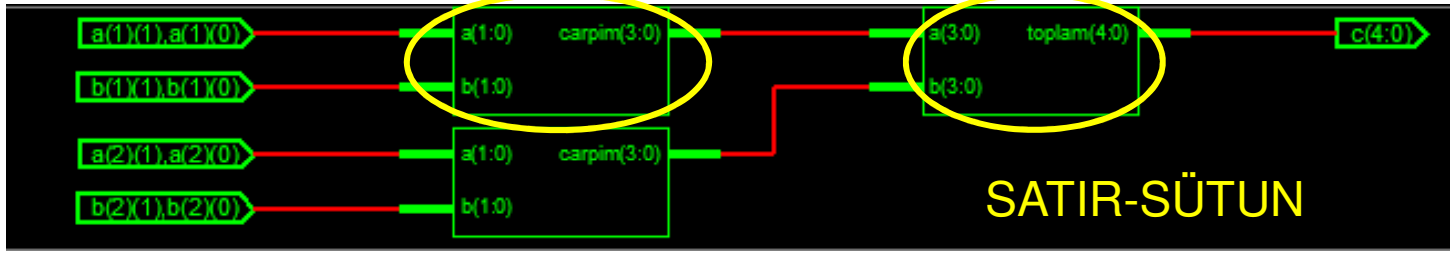
```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

entity Yari_Toplayici is
  port(a : in std_logic;
       b : in std_logic;
       toplam : out std_logic;
                          elde : out std_logic);
end Yari_Toplayici;

architecture dataflow of Yari_Toplayici is
  begin
  toplam <= a xor b;
  elde <= a and b;
end architecture dataflow;
```

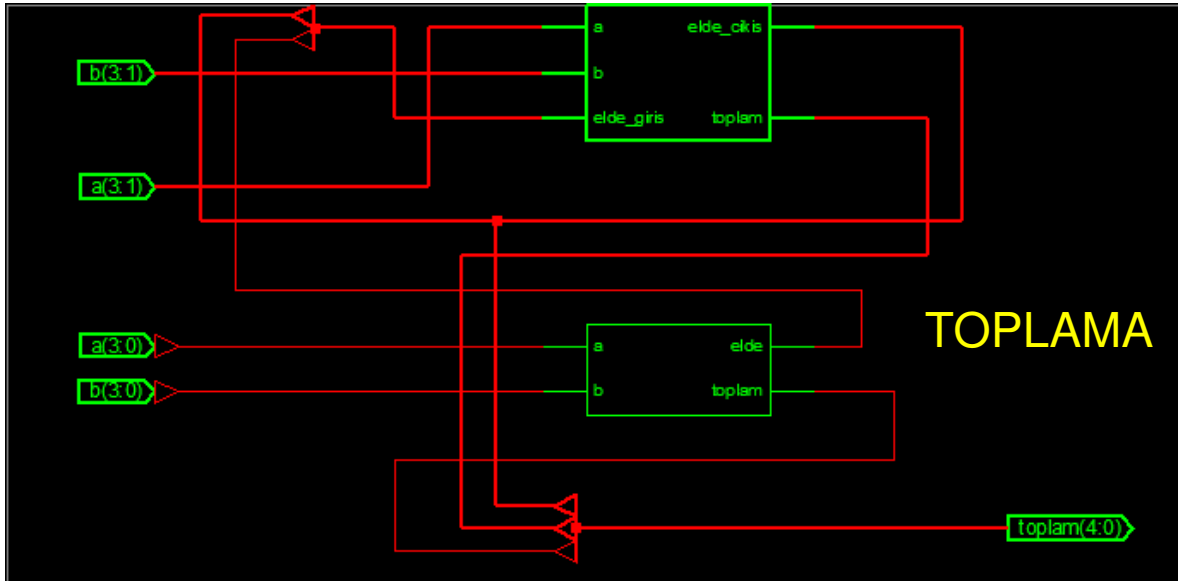
```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

entity Tam_Toplayici is
  port(a : in std_logic;
       b : in std_logic;
       elde_giris : in std_logic;
       toplam : out std_logic;
                          elde_cikis : out
std_logic);
end Tam_Toplayici;

architecture dataflow of Tam_Toplayici is
  begin
  toplam <= a xor b xor elde_giris;
  elde_cikis <= (a and b) or (a and
elde_giris) or (b and elde_giris);
end architecture dataflow;
```

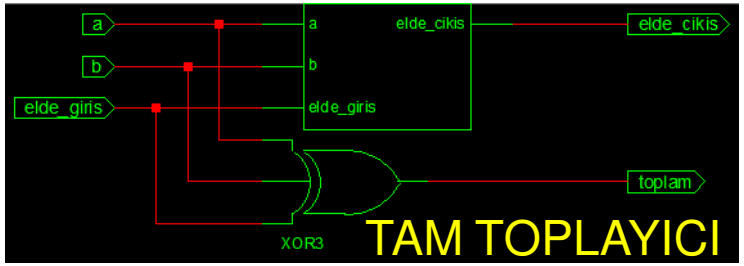

b(1:0)
a(1:0)
AND2
AND2
AND2
AND2
a
b
elde
toplam
a
b
elde
toplam
carpim(3:0)
ÇARPMA

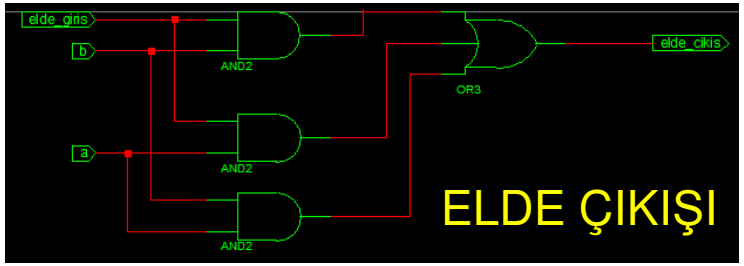

b
a
AND2
elde
I0
I1
O
toplam
YARI TOPLAYICI

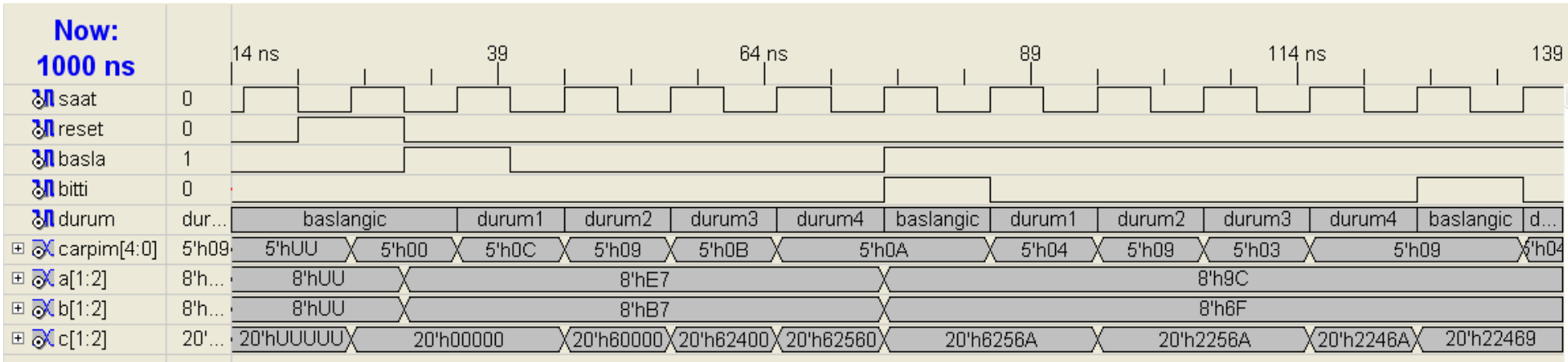

Data(1)
Data(0)
AND2B1
AND2B1
OR2
Result
XOR


a(1)(1),a(1)(0)
b(1)(1),b(1)(0)
a(2)(1),a(2)(0)
b(2)(1),b(2)(0)
a(1:0)
b(1:0)
carpim(3:0)
a(1:0)
b(1:0)
carpim(3:0)
a(3:0)
b(3:0)
toplam(4:0)
c(4:0)
SATIR-SÜTUN


b(3:1)
a(3:1)
a(3:0)
b(3:0)
a
b
elde_giris
elde_cikis
toplam
a
elde
b
toplam
toplam(4:0)
TOPLAMA

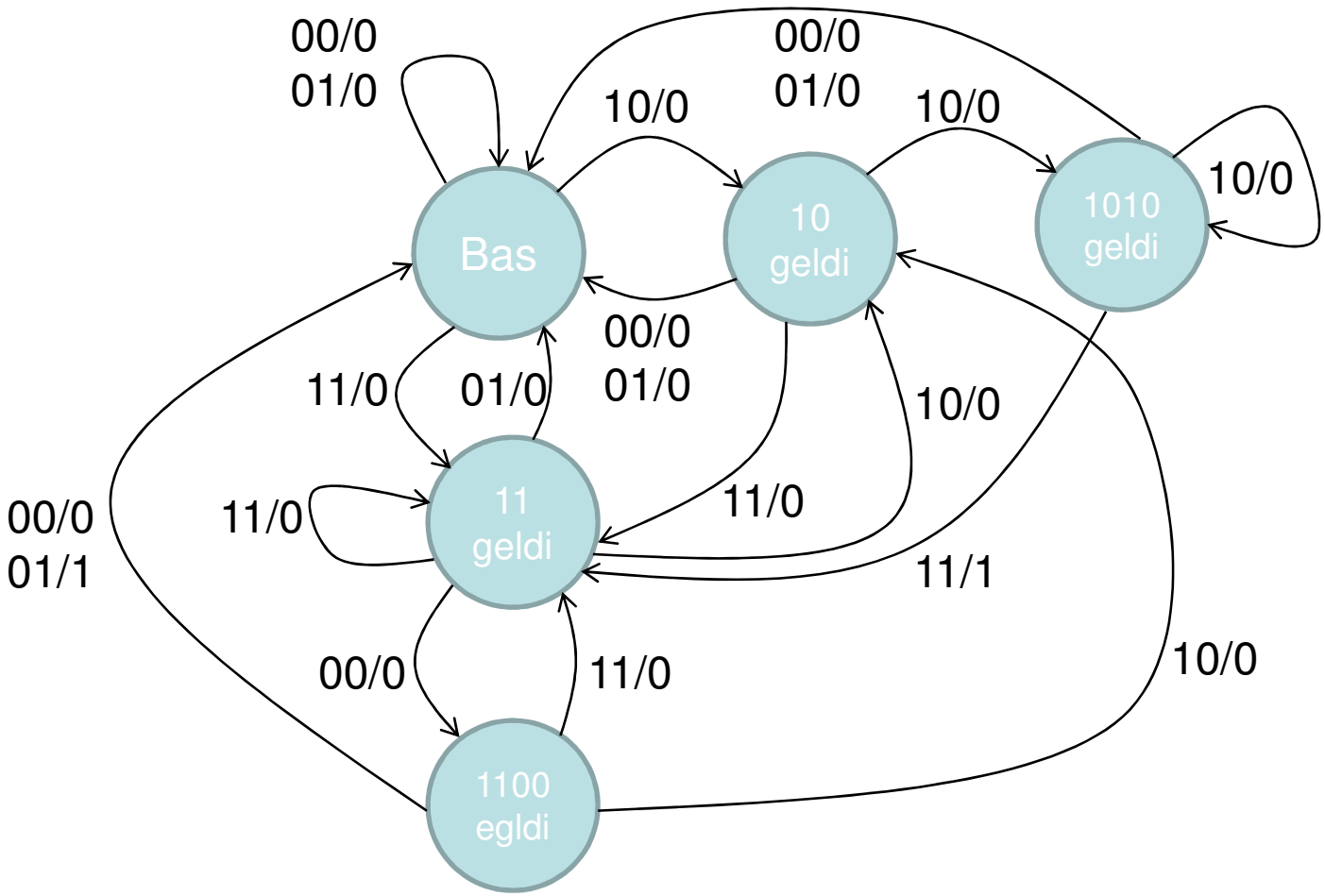

a
b
elde_giris
a
b
elde_giris
elde_cikis
elde_cikis
toplam
XOR3
TAM TOPLAYICI


elde_giris
b
a
AND2
AND2
AND2
OR3
elde_cikis
ELDE ÇIKIŞI


Now:
1000 ns
14 ns
39
64 ns
89
114 ns
139
saat
0
reset
0
basla
1
bitti
0
durum
dur...
baslangic
durum1
durum2
durum3
durum4
baslangic
durum1
durum2
durum3
durum4
baslangic
d...
carpim[4:0]
5'h09
5'hUU
5'h00
5'h0C
5'h09
5'h0B
5'h0A
5'h04
5'h09
5'h03
5'h09
5'h04
a[1:2]
8'h...
8'hUU
8'hE7
8'h9C
b[1:2]
8'h...
8'hUU
8'hB7
8'h6F
c[1:2]
20'...
20'hUUUUU
20'h00000
20'h60000
20'h62400
20'h62560
20'h6256A
20'h2256A
20'h2246A
20'h22469

Dizi Sezici Tasarımı

- Girişlerinden 10-10-11 veya 11-00-01 geldiğinde çıkışı 1 olan devre



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

entity DiziSezici is
    Port (x : in  STD_LOGIC_VECTOR (1 downto 0);
          z : out  STD_LOGIC;
          clk , rst: in  STD_LOGIC);
end DiziSezici;

architecture Behavioral of DiziSezici is
type state_type is (baslangic, geldi10, geldi1010,
      geldi11, geldi1100);
signal durum: state_type;
begin
  process(clk)
  begin
    if(clk'event and clk='1') then
      if(rst='1') then
        durum <= baslangic;
        z <= '0';
      else
        case durum is
          when baslangic =>
            if(x="10") then
              durum <= geldi10;
            elsif(x="11") then
              durum <= geldi11;
            else
              durum <= baslangic;
            end if;
            z <= '0';
          when geldi10 =>
            if(x="00" or x="01") then
              durum <= baslangic;
            elsif(x="10") then
              durum <= geldi1010;
            else
              durum <= geldi11;
             end if;
             z <= '0';
          when geldi1010 =>
            if(x="00" or x="01") then
              durum <= baslangic;
              z <= '0';
            elsif(x="10") then
              durum <= geldi1010;
              z <= '0';
            else
              durum <= geldi11;
              z <= '1';
            end if;
          when geldi11 =>
            if(x="00") then
              durum <= geldi1100;
            elsif(x="01") then
              durum <= baslangic;
             elsif(x="10") then
                durum <= geldi10;
              else
                durum <= geldi11;
              end if;
              z <= '0';
            when geldi1100 =>
              if(x="00") then
                durum <= baslangic;
                z <= '0';
              elsif(x="01") then
                durum <= baslangic;
                z <= '1';
               elsif(x="10") then
                 durum <= geldi10;
                 z <= '0';
               else
                 durum <= geldi11;
                 z <= '0';
               end if;
            end case;
          end if;
        end if;
    end process;
end Behavioral;
```

---- Source Options
Top Module Name : DiziSezici
Automatic FSM Extraction : YES
FSM Encoding Algorithm : Auto
Safe Implementation : No
FSM Style : lut
RAM Extraction : Yes
RAM Style : Auto
ROM Extraction : Yes
Mux Style : Auto
Decoder Extraction : YES
Priority Encoder Extraction : YES
Shift Register Extraction : YES
Logical Shifter Extraction : YES
XOR Collapsing : YES
ROM Style : Auto
Mux Extraction : YES
Resource Sharing : YES
Asynchronous To Synchronous : NO
Multiplier Style : auto
Automatic Register Balancing : No

---- Target Options
Add IO Buffers : YES
Global Maximum Fanout : 500
Add Generic Clock Buffer(BUFG) : 24
Register Duplication : YES
Slice Packing : YES
Optimize Instantiated Primitives : NO
Use Clock Enable : Yes
Use Synchronous Set : Yes
Use Synchronous Reset : Yes
Pack IO Registers into IOBs : auto
Equivalent register Removal : YES

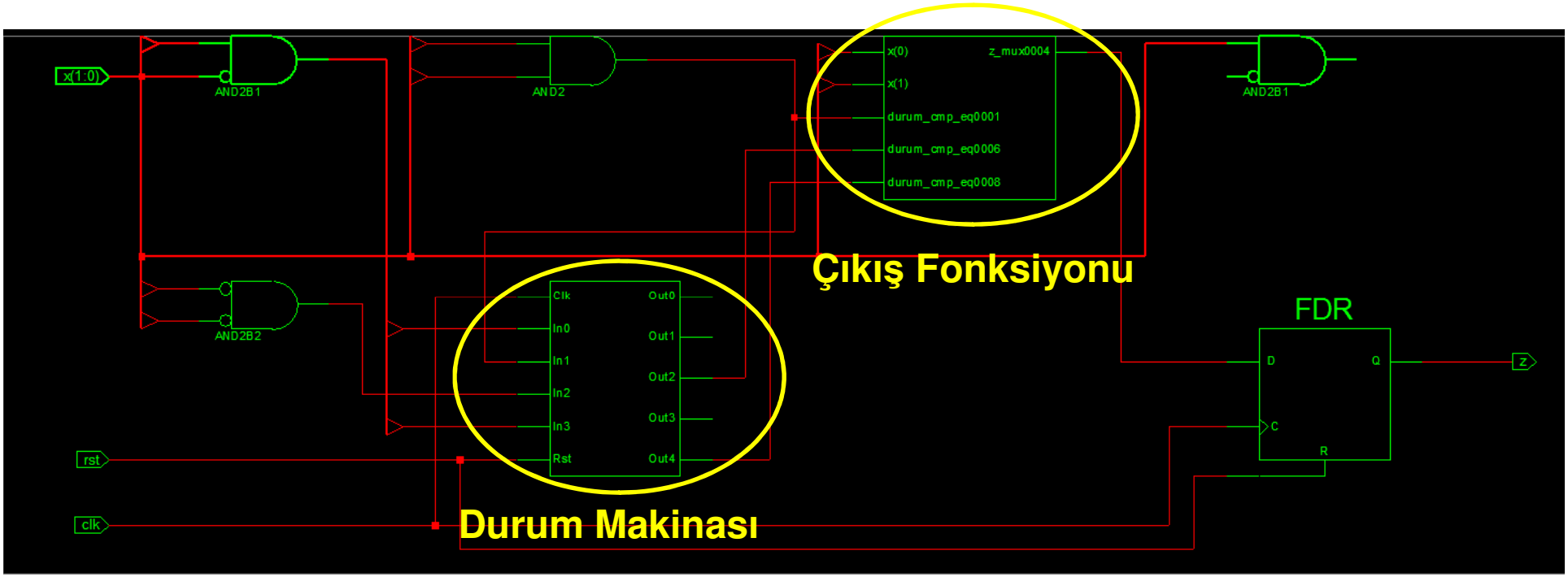
---- General Options
Optimization Goal : Speed
Optimization Effort : 1
Library Search Order : DiziSezici.lso
Keep Hierarchy : NO
RTL Output : Yes
Global Optimization : AllClockNets
Read Cores : YES
Write Timing Constraints : NO
Cross Clock Analysis : NO
Hierarchy Separator : /
Bus Delimiter : <>
Case Specifier : maintain
Slice Utilization Ratio : 100
BRAM Utilization Ratio : 100
Verilog 2001 : YES
Auto BRAM Packing : NO
Slice Utilization Ratio Delta : 5

```
HDL Synthesis
============
Performing bidirectional port resolution...

Synthesizing Unit <DiziSezici>.
    Related source file is
      "D:/Berna/Dersler/Lisans/LD/SayisalDevrelerDiziSezici/Di
      ziSezici.vhd".
    Found finite state machine <FSM_0> for signal <durum>.
    -----------------------------------------------------------------
    | States             | 5                                        |
    | Transitions        | 19                                       |
    | Inputs             | 4                                        |
    | Outputs            | 5                                        |
    | Clock              | clk (rising_edge)                        |
    | Reset              | rst (positive)                           |
    | Reset type         | synchronous                              |
    | Reset State        | baslangic                                |
    | Power Up State     | baslangic                                |
    | Encoding           | automatic                                |
    | Implementation     | LUT                                      |
    -----------------------------------------------------------------
    Found 1-bit register for signal <z>.
    Summary:
      inferred   1 Finite State Machine(s).
      inferred   1 D-type flip-flop(s).
Unit <DiziSezici> synthesized.

Analyzing FSM <FSM_0> for best encoding.
Optimizing FSM <durum> on signal
<durum[1:3]> with sequential encoding.
----------------------
 State     | Encoding
----------------------
 baslangic | 000
 geldi10   | 001
 geldi1010 | 011
 geldi11   | 010
 geldi1100 | 100
----------------------
Macro Statistics
# FSMs                                 : 1
# Registers                            : 4
 Flip-Flops                           : 4
Design Statistics
# IOs                        : 5
Cell Usage :
# BELS                       : 6
#      LUT2                  : 2
#      LUT3                  : 2
#      LUT4                  : 1
#      MUXF5                 : 1
# FlipFlops/Latches               : 4
#      FDR                   : 4
# Clock Buffers                  : 1
#      BUFGP                 : 1
# IO Buffers                 : 4
#      IBUF                  : 3
#      OBUF                  : 1
```

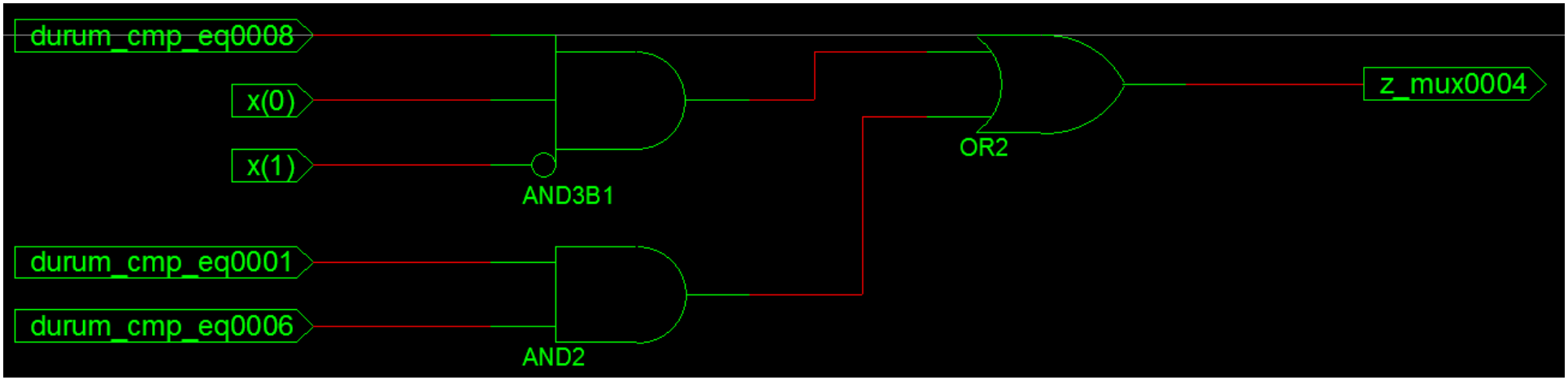
```
Device utilization summary:
---------------------------
Selected Device : 3s100evq100-5
 Number of Slices:                  3  out of    960     0%
 Number of Slice Flip Flops:        4  out of   1920     0%
 Number of 4 input LUTs:            5  out of   1920     0%
 Number of IOs:                     5
 Number of bonded IOBs:             5  out of     66     7%
 Number of GCLKs:                   1  out of     24     4%
-----------------------------------+------------------------+-------+
Clock Signal                       | Clock buffer(FF name)  | Load  |
-----------------------------------+------------------------+-------+
clk                                | BUFGP                  | 4     |
-----------------------------------+------------------------+-------+
   Minimum period: 2.192ns (Maximum Frequency: 456.132MHz)
   Minimum input arrival time before clock: 2.954ns
   Maximum output required time after clock: 4.040ns
   Maximum combinational path delay: No path found
Timing Detail:
--------------
All values displayed in nanoseconds (ns)
Timing constraint: Default period analysis for Clock 'clk'
  Clock period: 2.192ns (frequency: 456.132MHz)
  Total number of paths / destination ports: 6 / 3
Delay:               2.192ns (Levels of Logic = 2)
  Source:            durum_FFd3 (FF)
  Destination:       z (FF)
  Source Clock:      clk rising
  Destination Clock: clk rising

  Data Path: durum_FFd3 to z
                                Gate     Net
    Cell:in->out      fanout   Delay   Delay  Logical
Name (Net Name)
    ----------------------------------------  ------------
     FDR:C->Q             3   0.514   0.520
durum_FFd3 (durum_FFd3)
     LUT3:I1->O           1   0.612   0.000
z_mux00041 (N9)
     MUXF5:I1->O          1   0.278   0.000
z_mux0004_f5 (z_mux0004)
     FDR:D                    0.268          z
    ----------------------------------------
    Total                    2.192ns (1.672ns logic,
0.520ns route)
                                     (76.3% logic, 23.7%
route)

Timing constraint: Default OFFSET IN BEFORE
for Clock 'clk'
  Total number of paths / destination ports: 13 / 8
Offset:              2.954ns (Levels of Logic = 3)
  Source:            x<0> (PAD)
  Destination:       z (FF)
  Destination Clock: clk rising
```

Entity

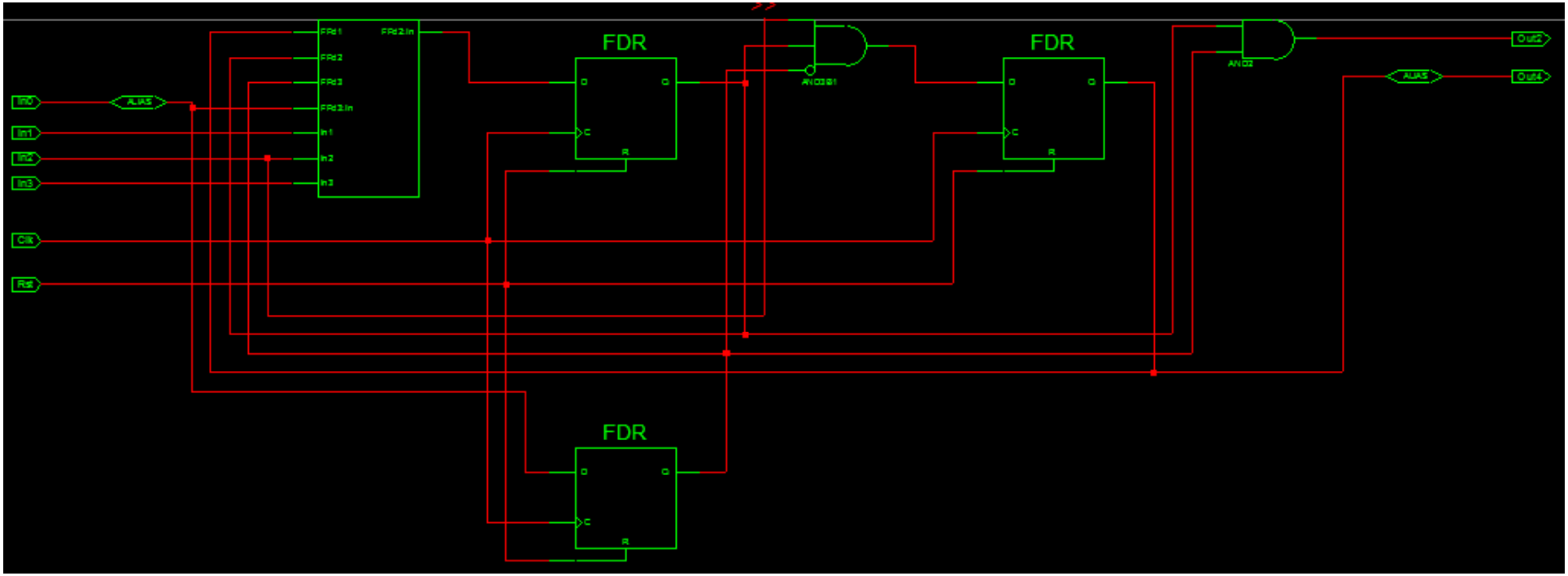
Architecture



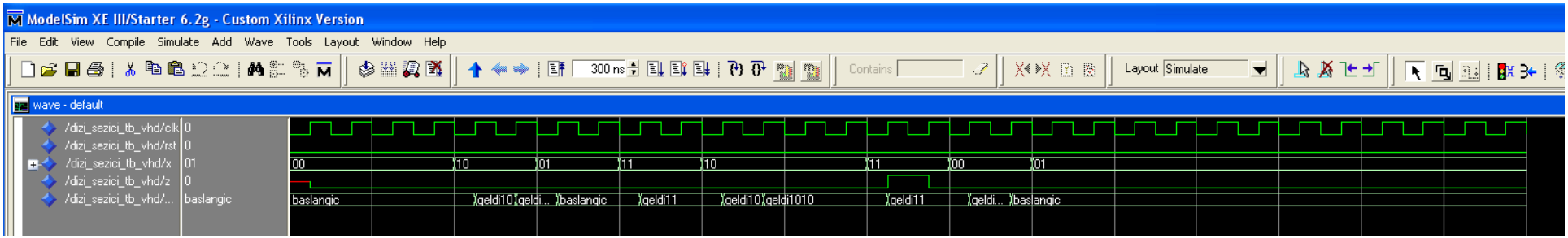
Çıkış Fonksiyonu



Durum Makinası



Simülasyon


ModelSim XE III/Starter 6.2g - Custom Xilinx Version
File Edit View Compile Simulate Add Wave Tools Layout Window Help
300 ns
Contains
Layout Simulate
wave - default
/dizi_sezici_tb_vhd/clk 0
/dizi_sezici_tb_vhd/rst 0
/dizi_sezici_tb_vhd/x 01
/dizi_sezici_tb_vhd/z 0
/dizi_sezici_tb_vhd/... baslangic
00 10 01 11 10 11 00 01
baslangic geldi10 geldi... baslangic geldi11 geldi10 geldi1010 geldi11 geldi... baslangic