Gömülü sistemler

```
begin
 process(clock, Read)
 begin
  if (clock'event and clock = "1") then
  if Enable = '1' then
   if Read = '1' then
  Data_out <= tmp_ram (conv_integer(Read Addr));
 else
 Data_out <= (Data_out'range => 'Z');
   end if;
   end if;
   end if;
   end process;
   process (clock, write)
 begin
 if (clock'event and Clock = '1') then
  if Enable = '1' then
     if write = '1' then
    tmp_ram(conv_integer (write_Addr)) <= Data_in;
 end if;
  end if;
 end if;
end process;
end behav;
```

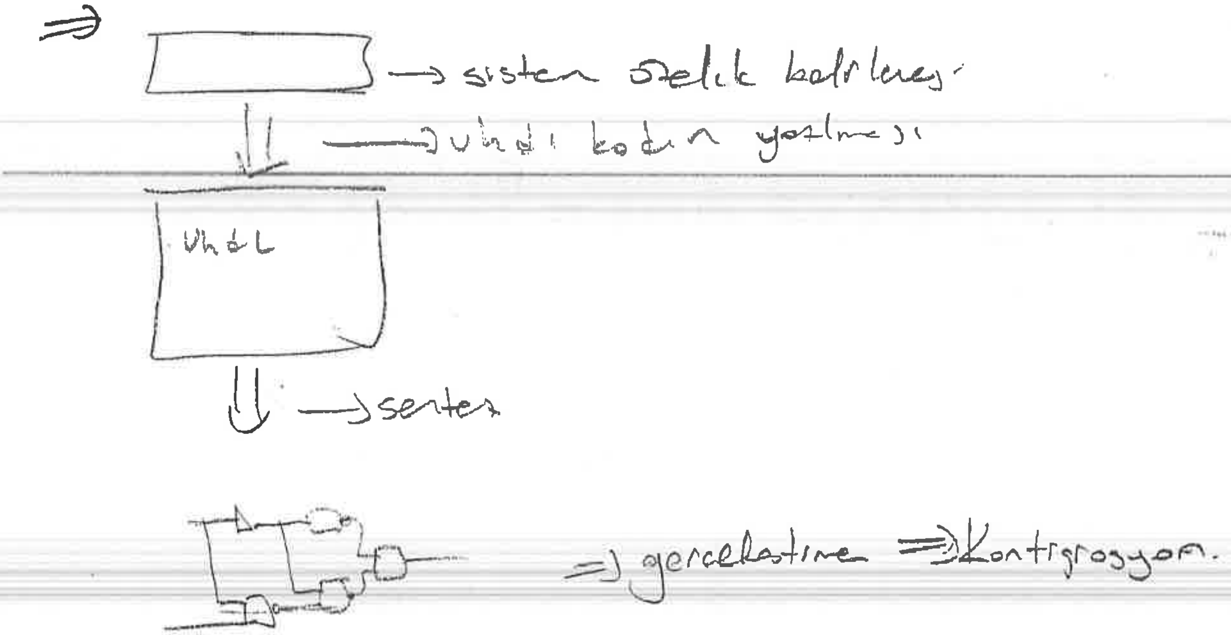
Yeni adrese 5 yazdı.

write(addre) <= 5

tmp_ram(42) = 5

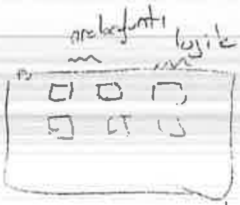
tmp_ram

FPGA Modeli



⇒ FPGA Mimarisi

⇒ fpga programlanabilir mantık blokları ve bu bloklar arasında arabağlantıdan oluşan geniş uygulama alanına sahip olan sayısal tümleşik devrelerdir.



10 milyondan fazla kapı.

Üretim teknolojisi

⇒ SRAM tabanlı mimari

⇒ tekrar tekrar programlanabilir

⇒ Fpga'lar her sistem açıldığında tekrar yapılandırılmak zorundadır.

⇒ depolama biriminin içeriğine göre transistör açık veya kapalı olacak

⇒ hızlı olması avantajdır.

⇒ 4-6 transistör içermesi dezavantajdır.

⇒ silinme yapılsa bilgi değiştirilene veya güç kesilene kadar yok olmaz.

Karşıt sigorta tabanlı Mimari

devre dışında özel olarak ile programlanırlar.

Yapılandırma kalıcıdır.

SRAM Antifuse dan [illegible] hızlı büyüktür.

⇒ İslemci gomulu sistemin kalbidir.

⇒ Gomul tasarımı icin microislemciler veya mikrocontroller sertir

⇒ 2 temel birime sahiptir. kontrol ve yurutme birimi

⇒ bu birimler herbir komutun fetch ve execute islemlerini gerceklestirir

Microislemci

⇒ Bir tek yarı iletken chiptir.

⇒ ALU, PC, stack pointer, cesitli registerlardan olusur.

Gomulu İşlemci

⇒ Gomulu bir sistemin gerekirdigini belirli bir sayıda gorevi yerine getirmek icin programlanan micro islemciler, microdenetleyiciler sayısal sinyal islemciler olabilir.

⇒ ARM7, AMD 29050 islemcilerdir.

kucukte bu
Tasarımı icin ARM, MIPS mimarisi bulunmaktadır.
gercek zamanlı isletim sistemi kullanılmaktadır DOS, Linux, QNX gibi.

⇒ Buyuk hacimde ise FPGA kullanılır

Dijital sinyal islemci

⇒ Unit islemci yetenegini icerir

⇒ Özellikle yuksek islem yuku gerektirende kullanılır

⇒ imge, ses, video isleme sistemlerinde.

⇒ Sinyal isleme fonk kullanıldıgında sistem hızlıdır.

Gomulu sistem urene ne katar

⇒ Maliyeti dusurur

⇒ daha az guc tuketimi

⇒ urune katma deger getirir.

⇒ urunler arası cesitlilik saglar.

⇒ Sistemin kolay yuklenmesine kolaylık saglar

İlk uretim 1961 yeplusu 1000$ 3$ dusmustur.

%100 guvenilir olduğu sokar.

- GÖMÜLÜ SİSTEMLER -

⇒ Gömülü sistem bir plan program veya kurallar kümesine göre bir veya birden fazla görevi organize eden giriş verilerini kullanarak sonuçlar üreten bir yapıdır.

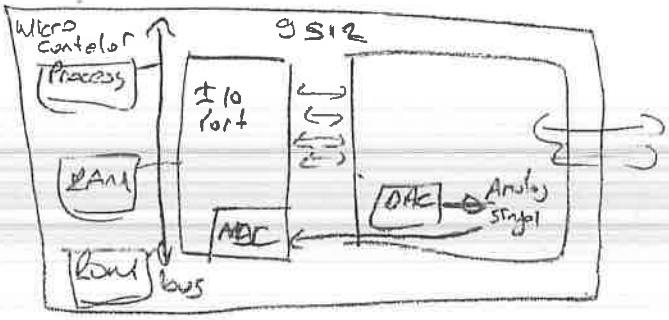
- * Digital bir sistemdir
- * Genellikle bir microişlemci kullanır
- * Kontrolör gibi kullanılır çoğu zaman

⇒ Gömülü sistem Belirli bir fonksiyonu yerine getirmek tasarlanmış yazılım ve donanım kombinasyonudur.

- * reel zamanlı çalışırlar.

⇒ Gömülü sistem Gerçek zamanda bir giriş alıp ona çıkış üretir.

Gömülü yapısı



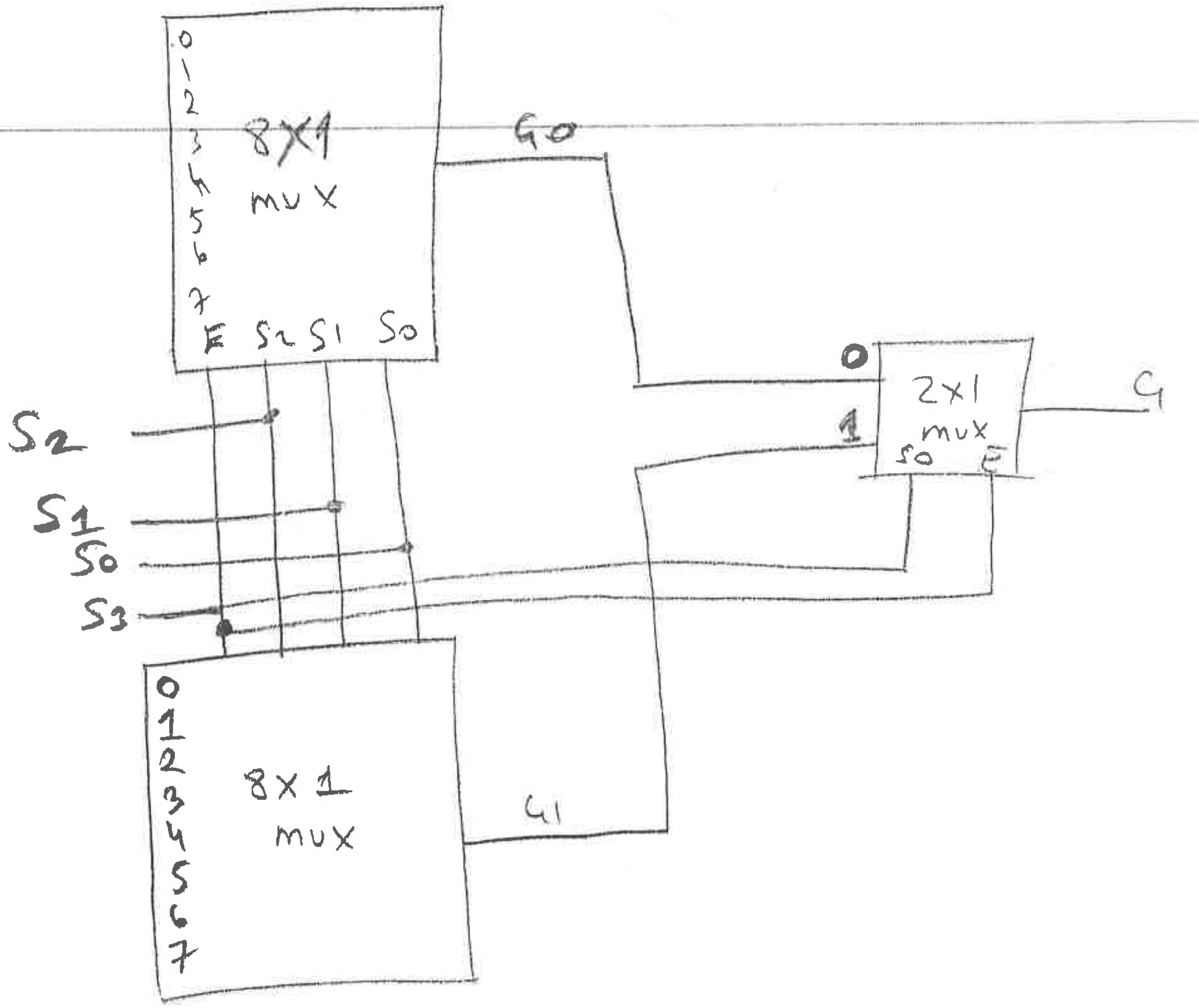
Gömülü sistem sınıflandırma

- * Küçük ölçekli gömülü sistem
- * orta " " "
- * karmaşık büyük " " "

⇒ küçük ölçekli
Bir tek 8 veya 16 bit mikrokontroler.
Az donanım karmaşık yazılım
bu sistemlerin C programı dılır
sürekli çalıştığında güç tüketimi

⇒ orta ölçekli
Bir veya birkaç 16 veya 32 bit mikrokontr.
hem donanım hem yazılım karmaşık

⇒ büyük ölçekli
Büyük donanım ve yazılım karmaşıklığı.
programlanabilir mantıksal diziler ile konfigüre edilebilir.
Donanım birimlerinin işlem hızı sistemi sınırlar.


0
1
2
3
4
5
6
7
8X1
mux
E
S2
S1
S0
G0
S2
S1
S0
S3
0
1
2x1
mux
S0
E
G
0
1
2
3
4
5
6
7
8X1
mux
G1

Cel

ndl

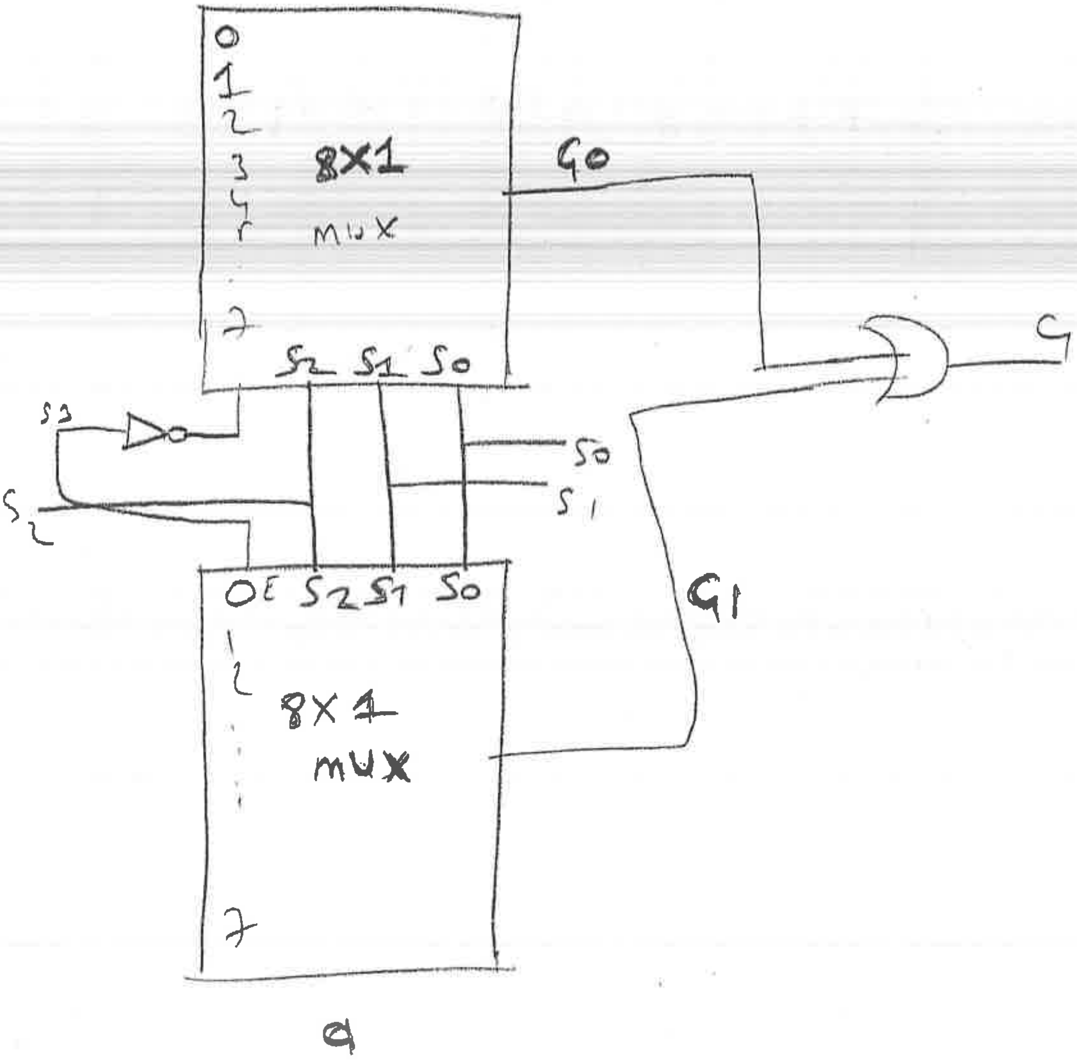
soru (1)

```
Cikis <= Cikis+1;
```
→ Azalan deneydi azalırdı,

```
    end if
    end if
    end process
    end mimari
```

* signaller architecture ile architecture in begin'i arasında olur.

16x1 mux'un 8x1 muxlar yardımıyla gerçekleştirilmesi



Sorular (4)

ÖR = 100 mhertz 2sn aralıklarla 8 bit sayan sayıcı
ega

$T = \frac{1}{f} = \frac{1}{100} = 10^{-2}$

GHz → nsn

MHz ~~mikrosn~~ egahertz → Msn → mikro sn

kHz → ms

hertz → sn

10^-2 mikrosn   1000 x1000 hep

10 nsn 1 clock geliyor.

2 snyede kaç clock geleceğini bulalım

10nsn   1 clock

2×10^9 X

$X = 2 \times 10^8$ clock ta sayıcı 0 ken 1 olacak

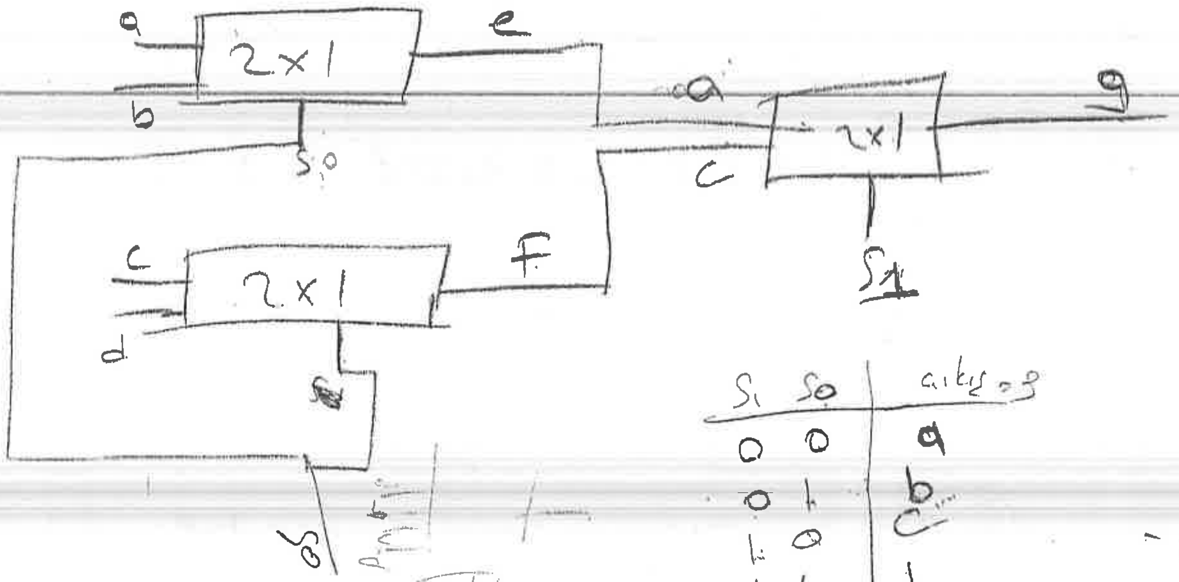
4×10^8 clockta 1 ken 2 olacak

giriş clk
çıkış 8 bit sayıcı

```
entity sayıcı is
    port( clk: in std_logic;
          cikis: out std_logic vector( 7downto 0)
    end sayıcı;
architecture model of sayıcı is
  signal sayac: integer;
begin
    process (clk)
  if ( rissing_edge)
      sayac := sayac+1;
      if sayac: 200000000 then
                sayac := 0;
```

düşen veya yükselen kenarda sayıcı artırılır

Örnek= 4x1 mux'u 2x1 muxla gerçekleştirin..
2 tane



S1	S0	çıkış=g
0	0	a
0	1	b
1	0	c
1	1	d

```
entity soru3 is
  port( a,b,c,d, : in std_logic;
        g        : out std_logic;
        S : in std_logic_vector(1 downto 0));
end soru3;
architecture cevap3 Of soru3 is
 signal e,f : std_logic;
  begin
   proces ( s)
   begin
    Case s is
         when "00" =>
         e<= a ;
          f<= c ;
          g<=e ;
         when "01."
          e<=b
          f<=d
          g<=e
```

Sayısal tasarım
mux şekline
bak
Bu soruda

Nıos işletim sistemi nedir?

Net Integrator Operating System ("Nıos")
otonom bir Linux ~~tabanlı işletim sistemi~~
olup tüm ağ altyapısını kolaylaştıran →
zeki bir işletim sistemidir.

→ 16 MB büyüklüğünde, bir flash bellek üzerinde
çalışan, standart Linux çekirdeğini kullanan
bir sistemdir.
self-aware, self configuring, self-tuning
özelliği sayesinde kendini yöneten bir sistemdir.

Nıos sunucuların faydası
Nıos yüklü sunucular diğerlerinden farklı

— Çokfazla,
— bakım ve yönetim maliyeti Düşük
— Bir ~~defa~~ kurulduktan sonra, Nıos sunucular
minimum bakım gerektirir.
— Sistem kendini bilir etrafında oluşan değişim
lere karşı kendini adapte eder.
— Nıos sunucular kendilerine korumada uzmandırlar
— Sistem güvenliği otonom çalışır. kurulum ya da
ayar gerektirmez. sunucular bunları hizmetini
— Nıosu olan yüklü

sunar
→ www servisi;
→ kurulum gerektirmeen firewall
→ Router
→ web de içerik filtreleme

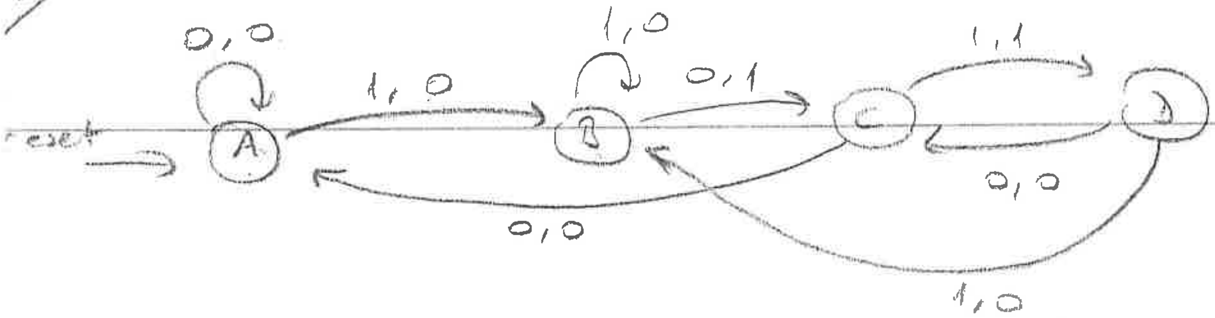
Nand kapısı

```
LIBRARY ieee;
use ieee.std_logic_1164.all;
ENTITY nand_gate IS
 PORT (
      a: IN STD_LOGIC;
      b: IN STD_LOGIC;
      z: OUT STD_LOGIC);
 END nand_gate;
ARHITECTURE model of nand_gate IS
  BEGIN
   z<= a NAND b;
    END model;
```

ÖR:

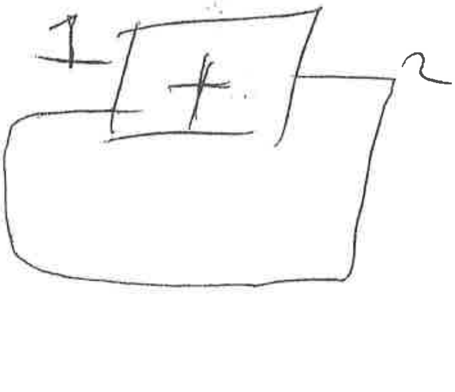
```
SIGNAL a: STD_LOGIC;
SIGNAL b: STD_LOGIC_VECTOR(3 DOWNTO 0)
SIGNAL c: STD_LOGIC_VECTOR(3 DOWNTO 0)
SIGNAL d: STD_LOGIC_VECTOR(7 DOWNTO 0)
SIGNAL e: STD_LOGIC_VECTOR(15 DOWNTO 0)
SIGNAL f: STD_LOGIC_VECTOR(8 DOWNTO 0)

  a<='1';
  b<="0000";   default olarak ikilik kabul edilir.
 c<=B"0000";   Binary belirtmek için B kullanılır.
d<="0110_0111";   okunabilirliği artırmak için _ kullanılır
e<=X"AF67";   X Hexadecimal tabanı temsil eder.
F<= O"723";   O octal tabanı temsil eder,
```

5%



```
case anlikdurum is

when A => if input = '0' then anlikdurum <= A;
                              cikis <= 0;

              else       anlikdurum <= B;
                              cikis <= 0;
                              endif;

when B => if input = '0' then anlikdurum <= C;
                              cikis <= 0;
```

```
Architecture Behavior of Fibir
   BeGiN;
                                                    Semetik
      fib! PRocesr (clock)
       BEGIN:
       IF (clock'EVENT ANC clock = '1') THEN
       IF Clear = '1' THEN

          Q<= "00000000 ";
          A<= "00.....00";
          B<= "00.... 1
          EISE
             Q ( 7 DOWNTO 0) <= B( 7 DOWNTO 0) + A(7DOWNTO 0)
             A (7 DOWN TO 0)<= B(7 DOWN O)
             B ( 7 DOWN TOO) <= Q(7 DOWNTO);

      END IF;
       END IF;
      END PROCESS;
      END Behavior;
```



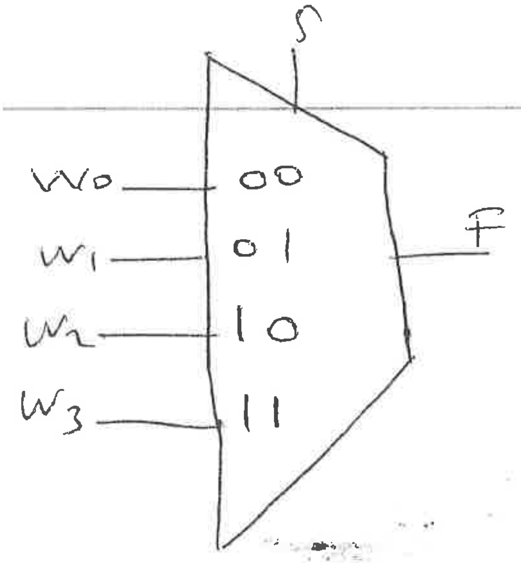
```
architecture dataflow of alu is
 Signal arith, logic: std_logic_vector (7 downto 0);
   Begin
   with sel(2 downto 0) select

    arith <= a when "000";
             b when "001";
             ⋮
             a+b+cin when others;

    logic <= Not a when "000"
              ⋮

          a+b+cin when others;
 with sel(3) select
  y <= arith when "0",
       logic when others;

    end dataflow.
```

Aynısının benzeri

Soru: Fibonacci Sayılarını hesaplayan vhdl kodu yazınız?

```
   entity fib Is
     PORT ( clear, clock : IN  STD_LOGIC;
            A        : BUFFER  STD_LOGIC_VECTOR(7 Downto 0);
            B        :   ''      ''   ''  ''
            Q        :   ''     STD_LOGIC_VECTOR(7 DOWN TO 0)
  end fib;
```

OR= 4X1 mux

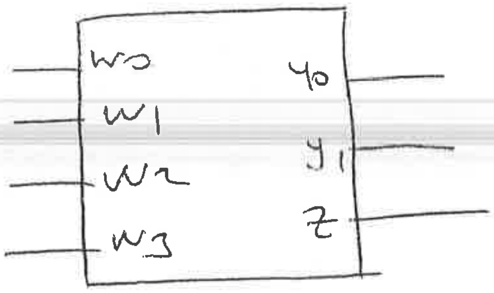


S_1	S_0	f
0	0	w0
0	1	w1
1	0	w2
1	1	w3

```
ENTITY mux4to1 IS
   port ( wo, w1,w2,w3 : IN STD_LOGIC;
    s                  : IN STD_LOGIC_VECTOR(1 Down To 0)
    f                  : OUT STD_LOGIC);

ARCHITECTURE Behavior of mux4to1 IS
     BEGIN
    WITH s SELECT
      f <= wo WHEN "00"
           w1 WHEN "01"
           w2 WHEN "10"
           w3 WHEN OTHERS;
       END Behavior.
```

```
ARCHITECTURE Behavior OF compar[illegible]
 BEGIN
  AeqB <= '1' WHEN A = B ELSE '0';
  AgtB <= '1' WHEN A>B ELSE '0';
  AltB <= '1' WHEN A<B ELSE '0';
   END Behavior;
```

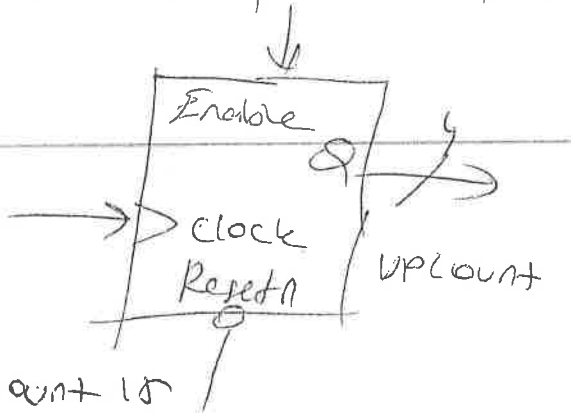
Örnek=



w3	w2	w1	w0	y1	y0	z
0	0	0	0	d	d	0
0	0	0	1	0	0	1
0	0	1	x	0	1	1
0	1	x	x	1	0	1
1	x	x	x	1	1	1

```
LIBRARY ieee;
use ieee.std_logic_1164.all;
ENTITY Priority is
port( w : IN STD_Logic vecto
                (3 DownTo 0);
      y : out  "   "  (1 Down
                          To 0
      z ; OUT  STD_logic;
END priority;

Architecture Behavior of
                   prio is

y <= "11" WHEN w(3)= '1' ELSE
     "10" WHEN w(2)= '1' ELSE
     "01" WHEN w(1)='1' ELSE
     "00"
z <= '0' WHEN w = "0000" ELSE '
END Behavior;
```

Asenkron reset ile 4-bit yukarı-sayacı



```
Architecture Behavior of upcount is
  SIGNAL Count : STD_LOGIC_VECTOR (3 DOWNTO 0);
   BEGIN
      PROCESS ( Clock, Resetn)
      BEGIN
         IF Resetn='0' THEN
                count <="0000";
      Elsif (Clock'EVENT AND Clock='1') THEN
       IF Enable = '1' THEN
            count <=Count+1;
        ENDIF;
      END IF;
     END PROCESS;
      Q <= Count;
  END Behavior;
```

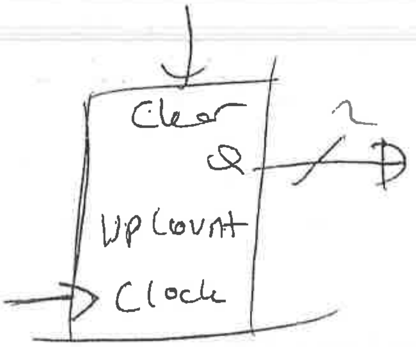
Enable ile

∅ IF Enable='1' THEN   Ehle

#Sayıcılar#

Senkron reset ile yukarı Sayacı

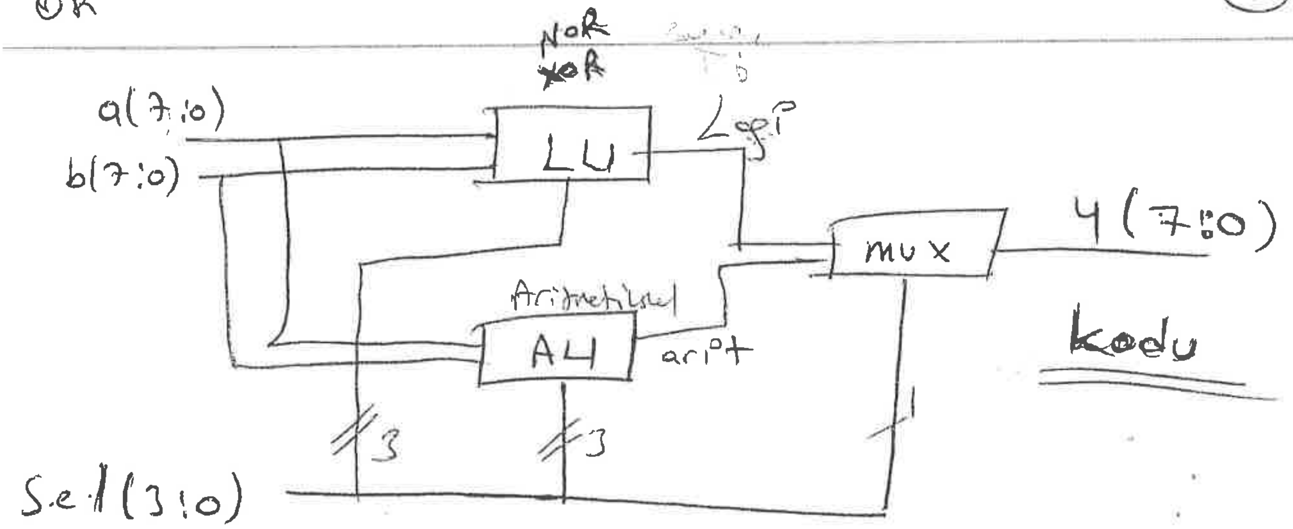
```
ENTITY UpCount IS
   PORT ( Clear, Clock : IN    STD_logic;
          Q            : Buffer STD_LOGIC_VECTOR(1 Down TO 0)
END upCount;
ARCHITECTURE Behavior Of UpCount IS
     BEGIN
   UpCount : PROCESS (clock)
     BEGIN
     IF (clock'EVENT AND Clock='1')THEN
      IF Clear='1' THEN
            Q <= "00";
         Else
         Q <= Q+"01";
         END IF;
        END IF;
      END PROCESS;
     END Behavior;
```



s'LAST_VALUE son olaydan önce s'in değerini döndürür

ÖR

②



kodu

with select yapısını kullan
sel (2 downto 0)

```
0000 → y <= a
0001 → y <= a+1
0010 → y <= a-1
0011 → y <= b
0100 → y <= b+1
0101 y<=b
0110 y<=a+b
0111 y<=a+b+cin
```

AU

```
1000 y<=not a
1001 y<=not b
1010 y<=a and b
1011 y<=a or b → y<=a nand b
1100 y<=a nor b
1101 y<=a xor b
1110
1111 y<=a xnor b
```

LU

```
Port( a,b ; in std_Logic_Vector(7 downto 0)
 sel : in std_logic_vector(3 downto 0)
 cin ; in std_Logic;
 y ; out std_logic_vector(7 downto 0)
 );
end alu
```

Gömülü Sistem

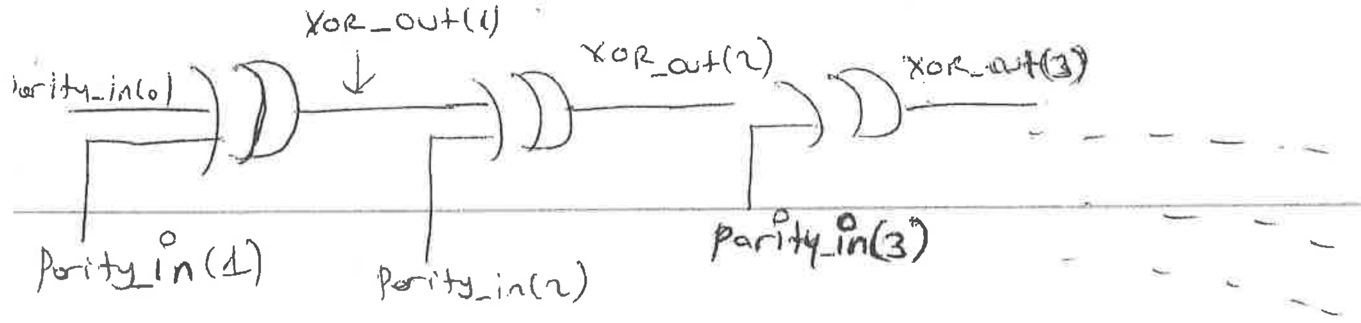
```
Signal a: BIT:='1';
Signal b: BIT_VECTOR (3 DOWNTO 0):='1100';
signal c: BIT_VECTOR (3 DOWNTO 0):='0010';
Signal d: BIT_VECTOR (7 DOWTO 0);
Signal e: INTEGER RANGE 0 TO 255;
Signal f: INTEGER RANGE -128 TO 127;
    X1 <= a & c;
    X2 <= c & b;      X1 <= ?
    X3 <= b XOR c
    X4 <= a NOR b(3)
    X5 <= b sll 2;
    X6 <= a and b(0) and NOT c(1);

  c+d ?
  e*3 ?
  d<=c ?
 d (6 downto 3) = b ?
 if(e>d) ?
F=100 ?
```

S'Event = S' de bir olay meydana geldiğinde return True
S'STABLE = S' " " " " gelmediğinde " "

S'ACTIVE = Eğer S='1' ise Return True

S'LAST_Event = Son olaydan beri geçen zaman durumunda True

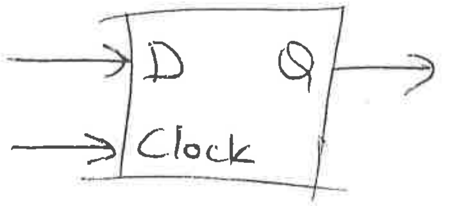
S'LAST_ACTIVE = S = '1'



```
ARCHITELTURE parity_dataflow of parity is
 SIGNAL xor_out ; std_logic_vector(6 downto 1);
   BEGIN
  xor_out(1) <= parity_in(0) XOR parity_in(1)
 G2 ; For i IN 1 TO 5 GENERATE
  xor_out(i+1) <= xor_out(i) XOR parity_in(i+1);
    end genorede G2;
 parity_out <= xor_out(6) XOR parity_in(7);
 END parity_dataflow;
```

Flip Flop

D latch



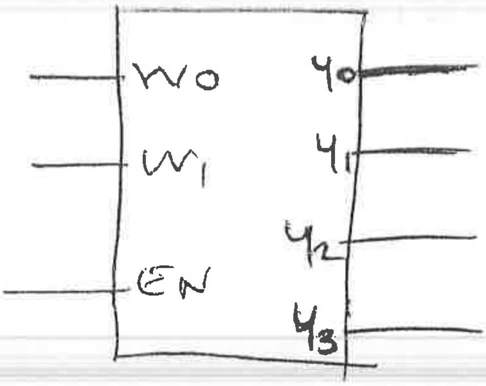
Doğruluk Tablosu

Clock	D	Q(t+1)
0	—	Q(t)
1	0	0
1	1	1

Clk	D	Q(t+1)
↑	0	0
↑	1	1
0	—	Q(t)
1	—	Q(t)

Örnek 2 TO 4 Decoder:

En	w1	w0	y0	y1	y2	y3
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1
0	X	X	0	0	0	0

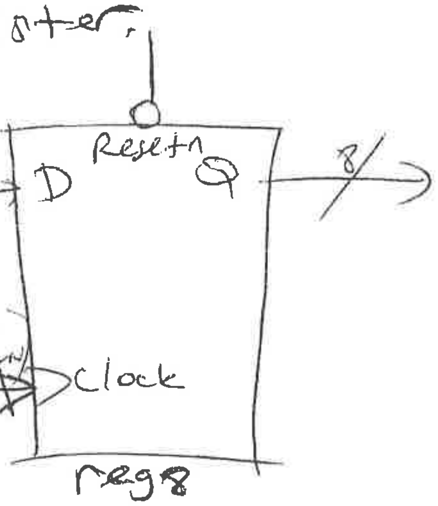


```
ENTITY dec2to4 IS
  PORT(  w: IN  STD_LOGIC_VECTOR(1 DOWNTO 0);
         En: IN  STD_LOGIC;
         y: OUT  STD_LOGIC_VECTOR(0 TO 3));
  End dec2to4;
ARCHITECTURE Behavior of dec2to4 IS
 SIGNAL Enw: STD_LOGIC_VECTOR(2 DOWNTO 0);
BEGIN
    Enw <= En & W
    WITH Enw SELECT
         y <= "1000" WHEN  "100"
              "0100" WHEN  "101"
              "0010" WHEN  "110"
              "0001" WHEN  "111"
              "0000" WHEN OTHERS
```

Senkron olursa değişiklik

BEGIN
~~WAIT UNTIL clock 'Event AND clock = '1';~~
IF Resetn = '0' THEN

Asenkron reset ile 8-bit register.

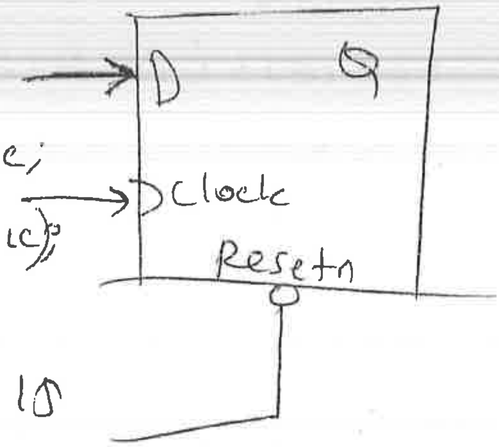


```
ENTITY reg8 IS
PORT ( D      : IN  STD_LOGIC_VECTOR(7 DOWNTO 0)
   Resetn, clock : IN  STD_LOGIC
   Q      : OUT  STD_LOGIC_VECTOR(7 DOWNTO 0)
 END reg8;

 ARCHITELTURE Behavior OF reg8 IS
    BEGIN
  PROCESS (Resetn, clock)
    BEGIN
      IF Resetn = '0' THEN
        Q <= "00000000";
   ELSIF Clock'EVENT AND Clock = '1' THEN
          Q <= D;
      END IF
    END PROCESS;
    END Behavior;
```

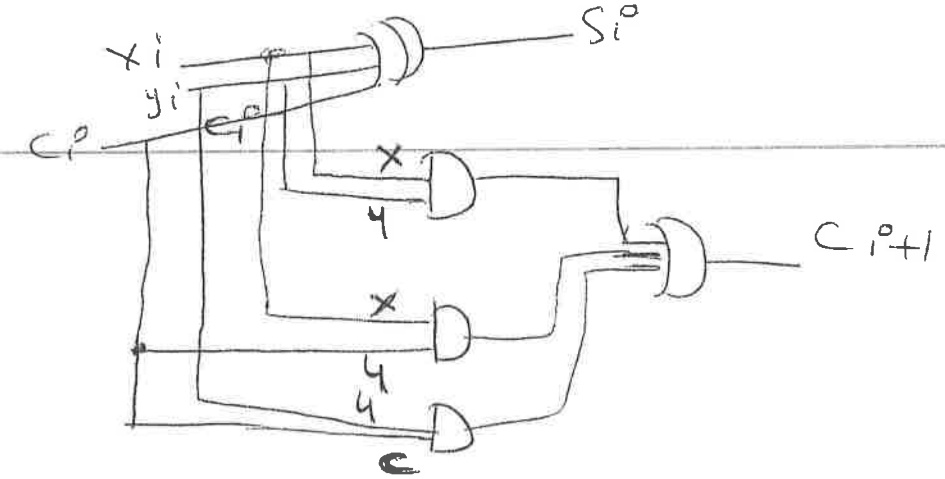
```
Entity latch is
    port ( D, clock : IN STD_LOGIC;
           Q        : OUT STD_LOGIC);
      END latch;
 Architecture Behavior of latch is
      BEGIN
        PROCESS (D, clock)
          BEGIN
        IF Clock = '1' THEN
             Q <= D;
          END IF;
        END PROCESS;
         END Behavior;
```

```
process(clock)
IF Clock'EVENT AN
   clock = '1' THEN
   Q <= D;
```

Asenkron reset ile D flip-flop



```
ENTITY flipflop is
  PORT ( D, Resetn, clock : IN STD_LOGIC;
         Q                : OUT STD_LOGIC);
   END flipflop;
Architecture Behavior of flipflop is
   BEGIN
    process (Resetn, clock)
      BEGIN
     IF Resetn = '0' THEN
        Q <= '0';
    ElsIF Clock'Event AND Clock='1' THEN
        Q <= D;
      END IF;
      ENDPROCESS;          END Behavior;
```

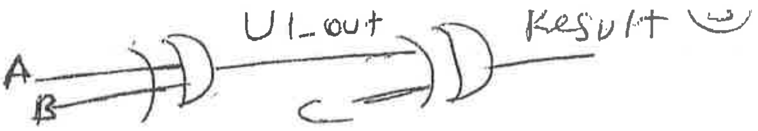


Data-flow

```
LIBRARY ieee;
Use ieee.std_logic_1164.all;
ENTITY fulladd IS
   PORT(  x : IN   STD_LOGIC;
          y : IN   STD_LOGIC;
          Cin : IN   STD_LOGIC;
          S : OUT  STD_LOGIC;
          Cout : OUT  STD_LOGIC);
  END fulladd;
ARCHITECTURE fulladd_dataflow Of Fulladd IS
  BEGIN
    S<= x XoR y XoR cin;
    Cout<=(x AND y) OR (cin AND x) OR (cin AND y);
    END fulladd_dataflow;
```

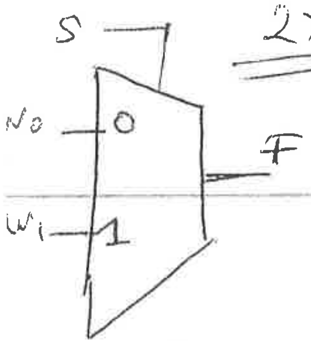
Doğru

y<=(a and b) or (c and d); → Doğru ifadeler



```
begin
 U1! xoR2 port map ( I1 ⇒ A
                     I2 ⇒ B
                     Y ⇒ U1_OUT);

U2; xoR2 portmap ( I1 ⇒ U1_OUT
                   I2 ⇒ C
                   Y ⇒ RESULT

  end XOR3_ STRUCTURAL;
       #DAvranis mimorisi#
                                          of XOR3 is
orchitecture XOR3_ BEHAVIORAL
 begin
 XOR3_BEHAVE; process(A,B,C)
 begin
 if ((A xor B xor C) = '1') then
    RESULT <= '1';
 else
  RESULT <= '0';
   end if;
  end process XOR3_ BEHAVE;
  end XOR3 BEHAVIORAL;
```

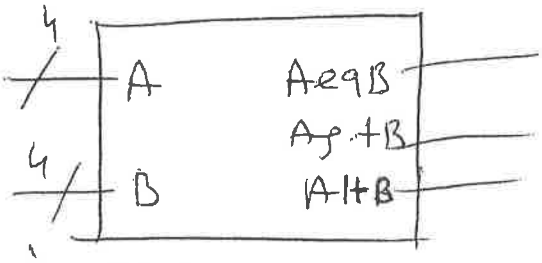
2X1 MUX



S	F
0	W0
1	W1

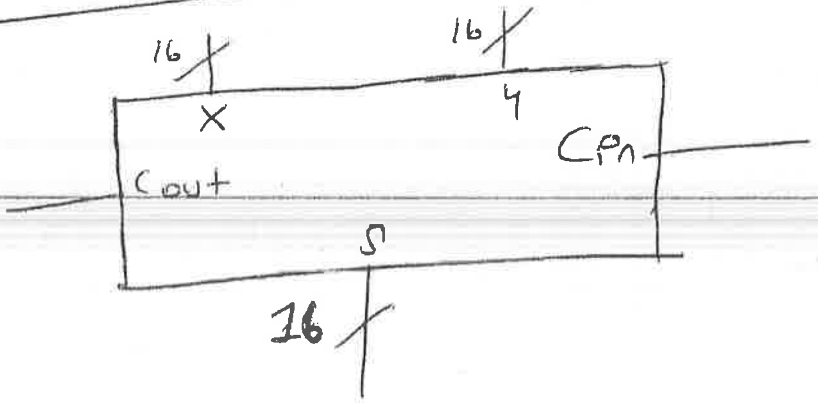
```
Library ieee;
Use ieee.std_logic_1164.all;
ENTITY mux2to1 IS
 PORT( w0,w1,S, : IN STD_LOGIC;
          F; OUT STD_LOGIC);
END mux2to1;
ARCHITECTURE Behavior of mux2to1 IS

BEGIN
 f<= w0 WHEN S='0' Else w1;
  END Behavior;
```

4-bit sayı karsilastirici



```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
Use ieee.std_logic_unsigned.all
  ENTITY Compare IS
    PORT( A,B     : IN  STD_LOGIC_VECTOR(3 Down To 0)
          AeqB, AgtB, AltB : OUT STD_LOGIC);
  END compare;
```

16-bit [illegible]



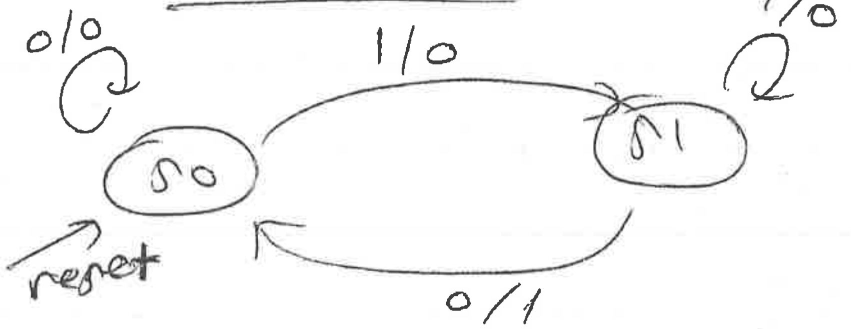
```
LIBRARY ieee;
Use ieee.std_logic_1164.all;
Use ieee.std_logic_Unsigned.all;

ENTITY adder16IS
   PORT(   Cin : IN STD_LOGIC;
           X,Y : IN STD_LOGIC_VECTOR(15 DOWNTO0
            S  : OUT STD_LOGIC_VECTOR(15 DOWNTO 0),
          Cout : OUT STD_LOGIC);
  END adder16;
  ARCHITECTURE Behavior Of adder16 IS
   SIGNAL sum : STD_LOGIC_VECTOR(16 DOWNTO 0);
  BEGIN
   Sum <= ('0' & X)+Y+Cin;
   S <= Sum(15 DOWNTO 0);   > ??
   Cout <= Sum(16);
    END Behavior;
```

Mealy

```
 Type state is (s0, s1);
 SIGNAL Mealy_state : state;

   U_Mealy : Process (clock, reset)
     BEGIN
      IF (reset = '1') THEN
       Mealy_state <= s0;
     ELSIF (clock = '1' AND clock'event) THEN
      CASE Mealy_state IS
        WHEN s0 =>
        IF input = '1' THEN
            mealy_state <= s1;
           ELSE
          Mealy_state <= s0;
           END IF;

    WHEN s1 =>
  IF input = '0' THEN
     Mealy_state <= s0;
      ELSE
       mealy_state <= s1;
       END IF;
       END CASE;
        END IF;
      END PROCESS;
  output <= '1' WHEN (Mealy_state = s1 AND
 input = '0') ELSE '0';
```

```
  WHEN S0 =>
IF input = '1' THEN
   Moore_state <= s1
   ELSE
   Moore_state <= s0;
   END IF;
  WHEN S1 =>
   IF input = '0' THEN
      Moore_State <= S2;
      Else
        Moore_State <= S1;
   END IF;
   WHEN S2 =>
 IF input = '0' THEN
     Moore_State <= s0;
     ElsE
       MOORE_State <= s1;
    END IF;
   END CASE;
  END IF;
 END PROCESS;
OutPut <= '1' WHEN Moore_State = S2 ElSE '0';
```

Mealy



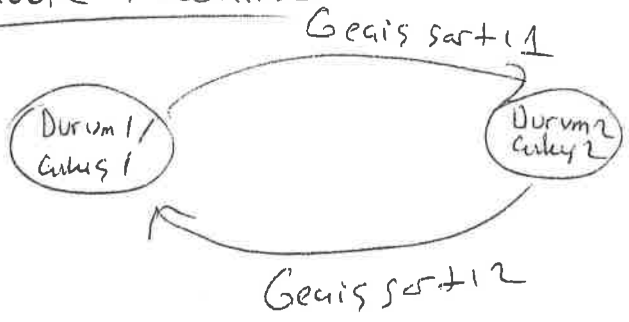
Moore

Çıkış sadece şimdiki durumun fonksiyonudur.

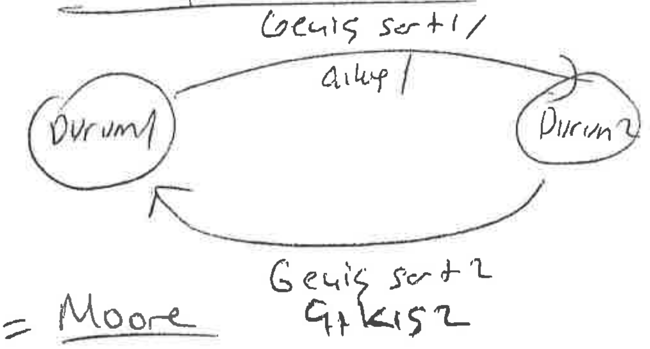
Mealy

Çıkış girişler ve şimdiki durumun fonksiyonudur.

Moore Machine

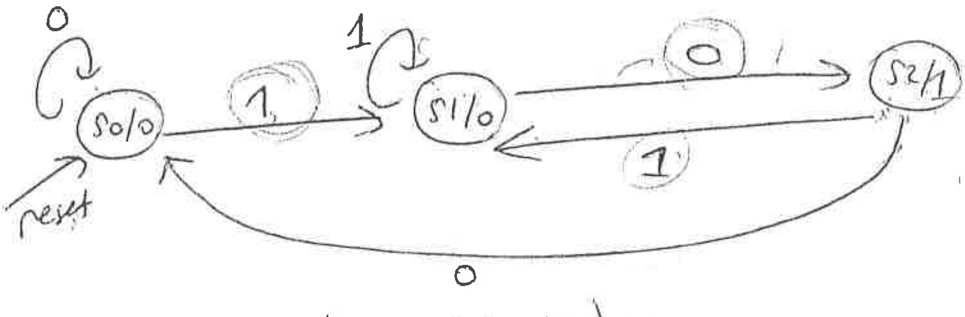


Mealy Machine



Durum

B = Moore

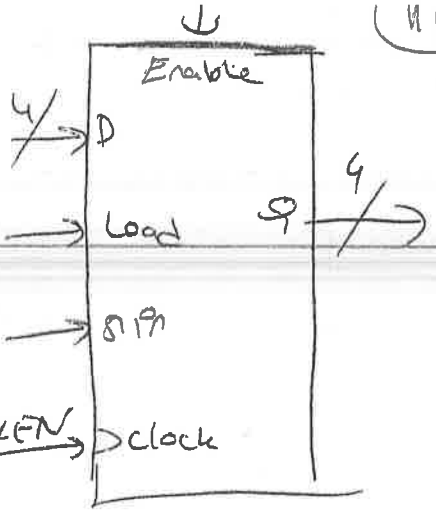


10

entity
[illegible] in [illegible]
Clock,

```
Type state IS (S0, S1, S2);
SIGNAL Moore_State: state;
 U_Moore: Process (Clock, reset)
   BEGIN
    IF (reset='1') THEN
     Moore_state <= S0;
    ELSIF (Clock='1' AND clock'event) THEN
     CASE Moore_State IS
```

Paralel yükleme ile 4-bit shift register.



```
Architecture Behavior_1 OF shift4 is
 process (clock)
 BEGIN
  IF Clock'Event AND Clock = '1' THEN
   IF Load = '1' THEN
      Q <= D;
 ELSIF Enable = '1' THEN
        Q(0) <= Q(1)
        Q(1) <= Q(2)
        Q(2) <= Q(3)
        Q(3) <= Sin;
    END IF;
  END IF;
  END PROCESS;
  END Behavior_1;
```

Nios

```
  int buton
  void (main) {
     while (1) {
 Buton = IORD_Altera_Avalon_PIO.data(Buton.base);
    if (buton='0')
     for (int i=0; i<=255; i++)
 IOWR_Altera_Avalon_PIO.data(led.base, i);
     else
      for (int i=255; i>=0; i--)
 IOWR_Altera_Avalon_PIO.data(led.base, i);
      }
        }
          }
```

→ Niosta kod yazmak icin Quartus'da islemci olusturuyoruz.

→ Quartus'ta islemciyi SOPC builder'dan olusturuyoruz

→ Elemanlar processor, ram, jtag (mutlaka)

→ Istege bagli giris-cikis birimleri

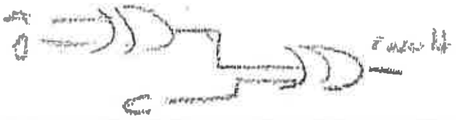
→ Kaydet

→ Niosta bu islemciyi sec

→ C dilinde kodunu yaz.

Component Yapısı

```
architecture ornek of ~~[illegible]~~ is
signal u1_out : std_logic;

component xor2 is
      port ( i1 : in std_logic;
             i2 : in std_logic;
             y : out std_logic );

      end component;

begin
```



```
u1 : xor2 port map ( i1 => A
                     i2 => B
                     y => u1out );

u2 : xor2 port map ( i1 => u1out
                     i2 => c
                     y => result );
```

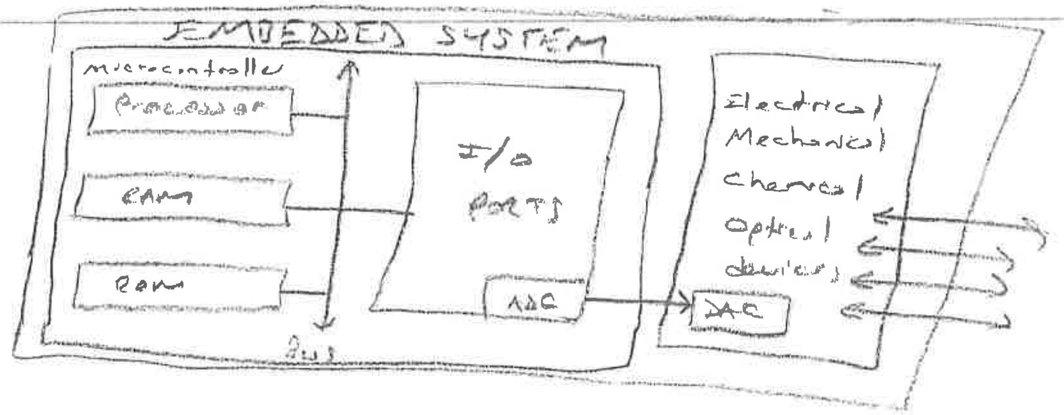
fibonacci

0 1 1 2 3 5 8 13

```
entity fibonacci is
port( birinci, ikinci : integer;
        sonuc : integer;
        clk : std_logic ); end fibonacci;
architecture ornek of fib is
      birinci := 0;  ikinci := 1;
signal sayac := integer := 0;
    begin
    process (clk)
        begin  if (sayac < 10) then
            sonuc := birinci + ikinci;
            birinci := ikinci;
            ikinci := sonuc;  end if;
             sayac = sayac + 1 ); end process end architecture
```

SÖZEL

Gömülü Sistemin Yapısı



Mikroişlemci	Mikrocontroller
⇒ Fonksiyonel bloklar: ALU, registerlar, timing, control unit	⇒ Ek olarak timer, I/O, RAM, EEPROM, ADC, DAC
⇒ Bit işleme komutları azdır	⇒ Çoktur
, Genel amaçlı sayısal bilgisayar sistemlerinin tasarımında	⇒ Uygulamaya özel kontrol sistemlerin tasarımında kullanılır

* Mikroişlemcide veri ve program mikroişlemcinin dışında tutulur. Yavaştır.

* Microkontrolerde veri ve program mikrokontroler dahilinde tutulur. Daha hızlıdır.

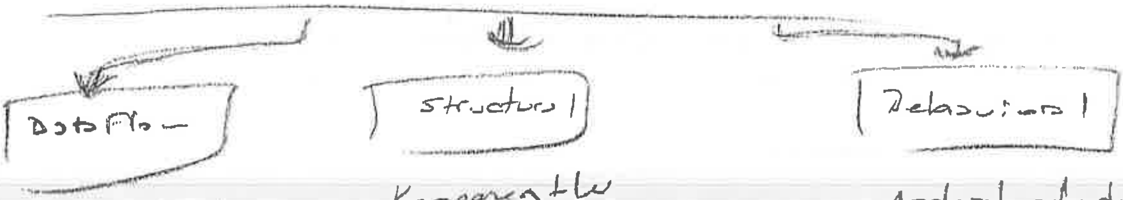
TASARIM SÜRECİNIN AŞAMALARI

top down
Requirement = Tasarım için gereksinimler
= Minimum donanımla çalışacak sistem tasarlama
Specification = Sistem fonksiyonlarının nasıl tasarlanacağını ve [illegible]
Architecture = Mimarinin ihtiyacı olduğu komponentler belirlenir.
Components
bottom up
System Integration = Çalışan sistem bütünleştirilir. Donanım yazılım birleştirilir

Tasarımın 3 ANA amacı

1. Üretim maliyeti düşüklüğü
2. İstenen performansın sağlanması
3. Güç tüketimi kontrolü

VHDL TASARIM STİLLERİ



eş zamanlı çalışan kodlar.

Komponentler ve bağlantılar kullanılır

Ardışıl modellere olanak sağlayan processler kullanılır

Gömülü Sistem Nedir?

→ En genel tanımıyla belirli bir işi yapması için tasarlanmış, mikroişlemci ve ya mikrodenetleyici tabanlı sistemlerdir.

→ Bilgisayardan farklı bilgisayar kartlarla işi yapması için.

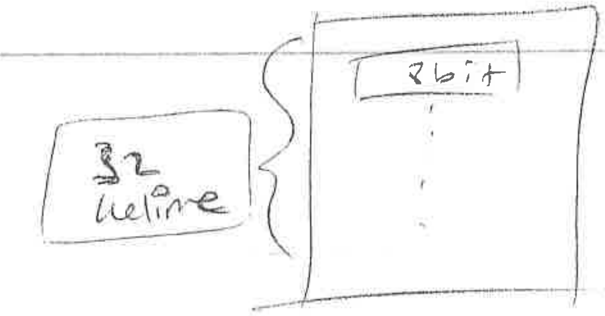
→ buzdolabı, çamaşır makinası.

#Fibonacci#

```
entity fibonacci is
  port ( birinci, ikinci : integer;
             Sonuc : integer;
             clk : std_logic );
 end fibonacci
 architecture behaviour of fibonacci is
    birinci := 0;
    ikinci := 0;
 signal Sayac : integer := 0;
        begin
    process (clk)
   begin
 if ( sayac<10) then
                       2        3
   Sonuc := birinci + ikinci;
              3
   birinci := ikinci;
            5       5
   ikinci := Sonuc;
   end if
   sayac := Sayac+1;
 end process;
end behaviour;
```

Rom (1)

32 tane 8 bitlik



??? ↓ Boşluk denek

```
entity Rom is
  port(  clock: in std_logic;
         Reset: in std_logic;
         Enable; in std_logic;
          Read : in std_logic;
         Address: in std_logic_vector(4 downto 0)
         Data_out: out std_logic_vector
                                   (7 downto 0).
       );
   end Rom
```

cputime =

kütüphaneler.

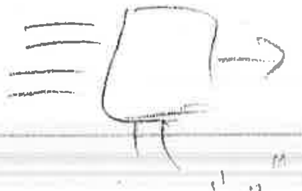
```
  Use ieee.std_logic_1164.all;
  Use ieee.std_logic_arith.all;
  Use ieee.std_logic_Unsigned.all;
  architecture Behav of Rom is
     type Rom_Array is array (0 to 31)
      of std_logic_vector(7 downto 0);
```

2

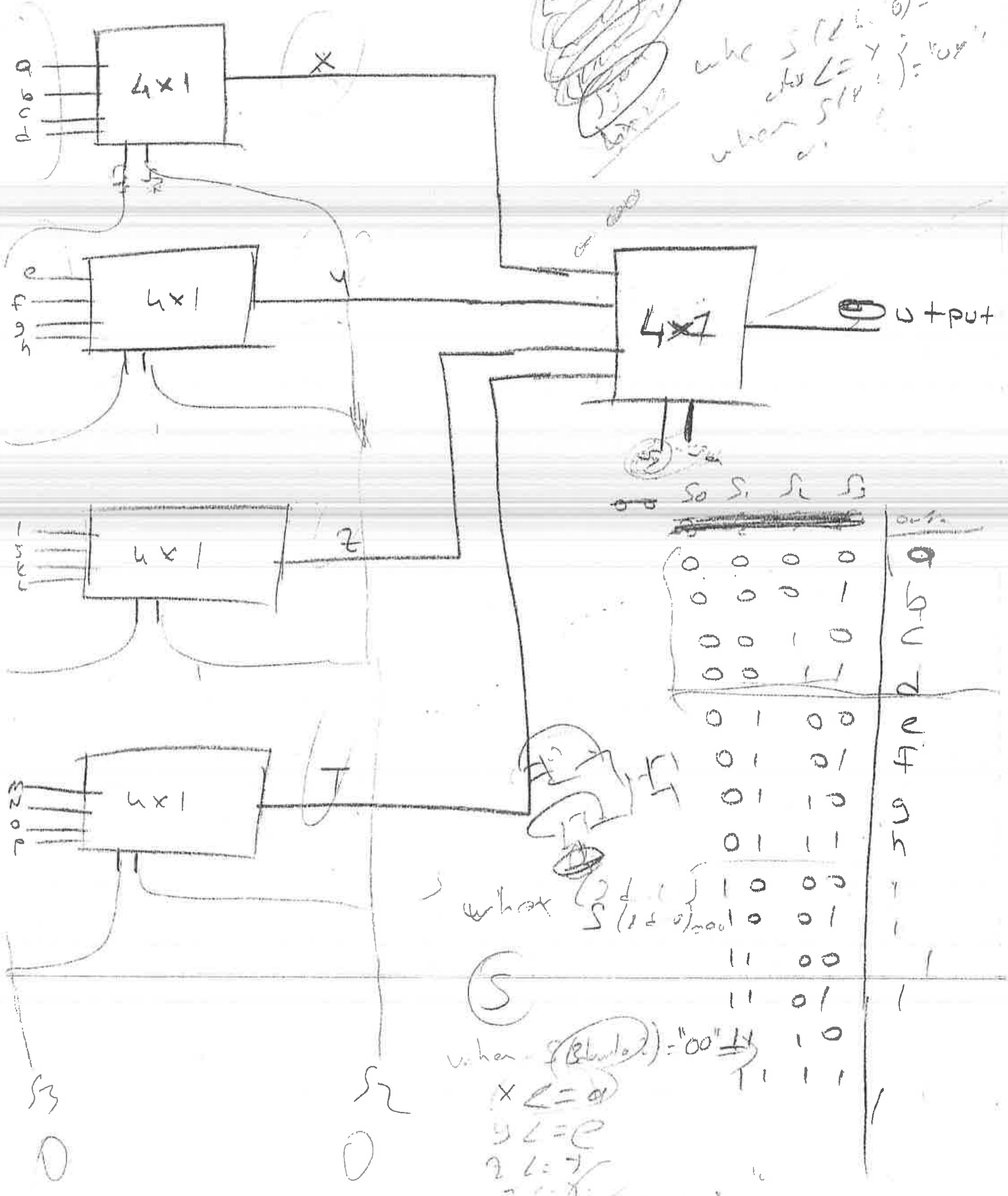
ROM

```
Constant Content : Rom_Array := (
        0 => "00000001"
        1 => "00000010"
           .
        9 => "00001010"
           .
        14 => "00001111"
        OTHERS => "11111111"
    );
begin
    process (clock, Reset, Read, Address)
     begin
      if (reset = '1') then
        Data_out <= "ZZZZZZZZ";
    elsif (clock'event and clock = '1') then
        if Enable = '1' then
           if (Read = '1') then
           Data_out <= Content (conv_integer (Address))
                                                       ;
      else
        Data_out <= "ZZZZZZZZ";
      end if;
        end if;
     end if;
    end process;
    end Behav;
```

16X1 mux 4X1 mux



whe S(1 downto 0)="00"
els <= Y)="01"
when S(1
a



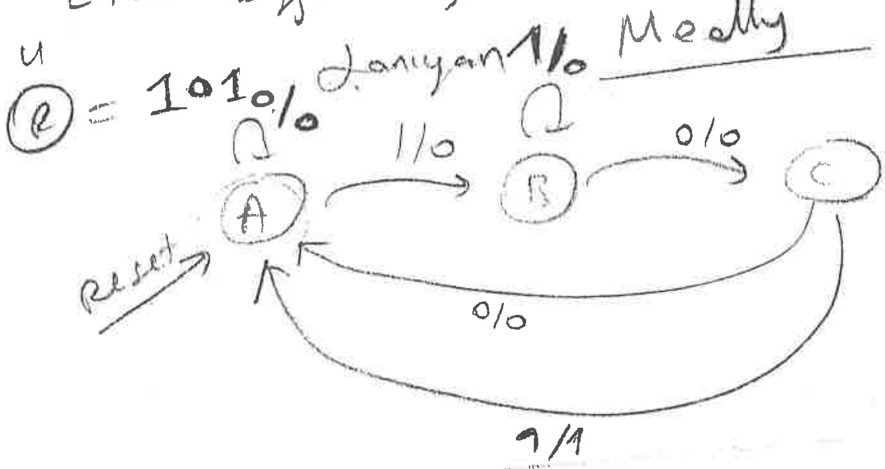
S0	S1	S2	S3	out
0	0	0	0	a
0	0	0	1	b
0	0	1	0	c
0	0	1	1	d
0	1	0	0	e
0	1	0	1	f
0	1	1	0	g
0	1	1	1	h
1	0	0	0	ı
1	0	0	1	ı
1	1	0	0	/
1	1	0	1	/
		1	0	
1	1	1	1	/

when S(1 downto 0) mod

when S(3 downto 2)="00" H

X <= a
y <= e
z <= i
T <= m
when S(3 downto 2)="01"

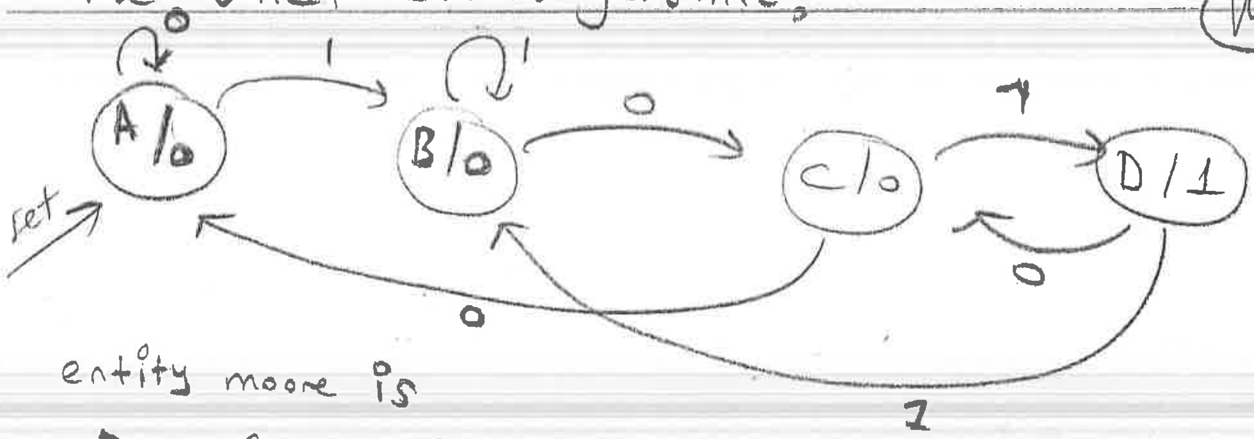
```
M_durum <= B ;
 end if;
  when C =>
 if input = '0' then
 M_durum <= A;
  else
  M_durum <=
    end if;
  when D =>
  if input = '0' then
   M_durum <= C;
    else
   m_durum <= B;
    end if;
   end case;
  end if;
 end process;
 if m_durum = D then
      cikis <= 1;
   else
      cikis <= 0;
    end if
  End uygulama;
```



```
when A =>
if durum => C and
input = '1' then
 cikis <= '1'
 end if;
```

ÖR= "101" tanıyan Fsm'i moore makine dili ile vhdl kodunu yazınız?

M1



```
entity moore is
  Port (clk, input, reset : in std_logic;
     Cikis : out std-logic);
     end moore;
 architecture Uygulama of moore is
   type State is (A,B,C,D);
   signal durum : State;
      begin
  Process (reset, clk)
     begin
   if reset = '0' then
        durum <= A;
    elsif rissing-edge (clk) then
         case durum is
     when A =>
     if input = '0' then
          durum <= A;
         else
        durum <= B;
        end if
     when B =>
      if input = '0' then
       durum <= C;
        else
```

```
int buton
  main(void){
     While(1){
    buton=IORD_Altera_Avolon_PIO_Data(Buton_Bare)
       If(buton==0){
     for(int i=0;i<=255;i++){
     IOWR_Altera_Avalon_PIO.Data(Led_Bare,i);
    else{
      for(int i=255;i>=0;i--){
       IOWR_Altera_Avalon_PIO_Data(Led_Bare,i);
      }
      }
     }
```

Nios isletim Sisdemini acıklaynız

Gömülü isletim Sisdemleri nelerdir birini acıklaynız?

Apollo Guidance · computer → ilk gömülü sisdem.

Sorular (1)

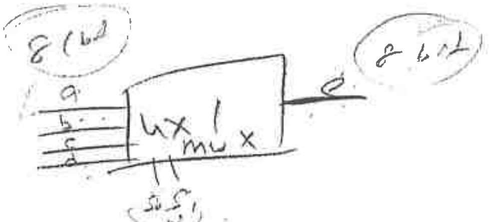
Gömülü sistemin iç yapısını çiziniz?

if clk = '1' and clk'event } her yükselen kenarda altına sartılıyor.

ya da if. rissing ~~-~~ edge } yükselen kenar

if. falling - edge } düşen kenarda

ÖR= 2 Bitlik Seçme ucu olan bir devrede a and b or, xor, NAND küçükten büyüğe doğru Not



Signalleri Begin ile Architecture arasında tanımla.

```
entity mux is
   port ( a, b, c, d : in std_Logic vector( 7 downto 0);
         s : in std_Logic vector( 1 downto 0);
         e ; Out std_logic vector( 7 downto 0));

   end mux
architecture Soru1 of mux is
begin
      process( s)
     begin
    case s is
       when "0,0" =>
            e<= a ;
          when "01" =>
            e<= b;
```

Case s is
when

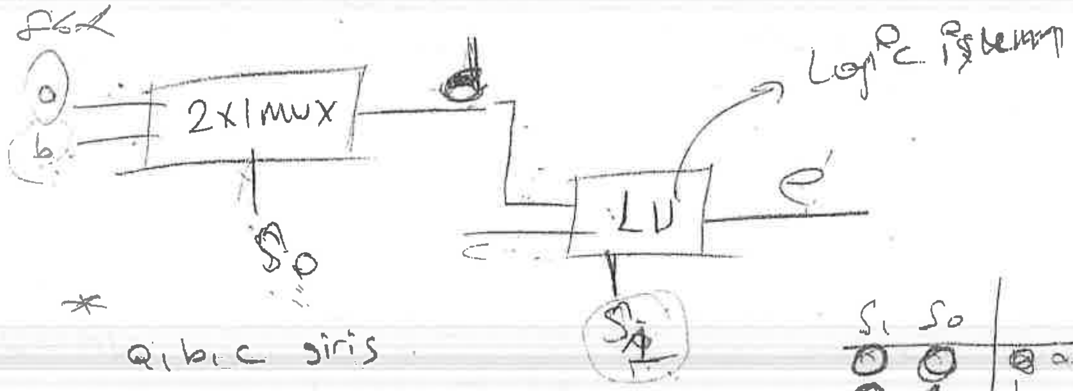
Birden fazla BSX

end Case

end case;
end Process;
ent ~~of~~ Soru1;

Sr= 2 girişden 1 ini çıkısa veren bu çıkışı 3. bir girişle and iken veya orlayan devreyi ~~tasarlayınız~~ vhdl kodunu yazınız?

2



*

a,b,c giriş

e → çıkış

d → ara signal

s → giriş

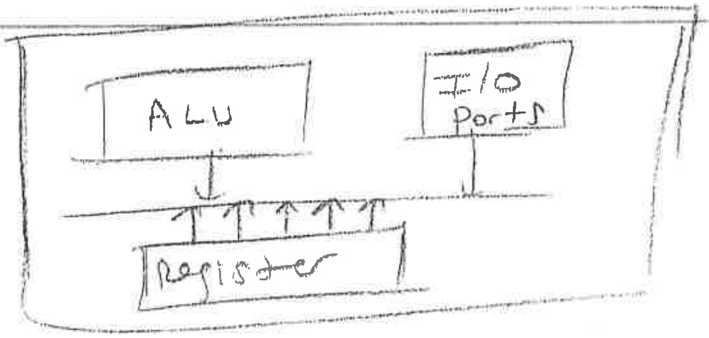
S1	S0	
0	0	a and c
0	1	b and c
1	0	a or c
1	1	b or c

```
entity soru2 is
port (   a,b,c : in std_logic_vector(7downto0);
         S     : in std_logic_vector(1downto0);
         e     : out std_logic_vector(7downto0));
end soru2;
architecture uygulama of soru2 is
   signal d : std_logic vector(7downto0);

   begin
   process ( S )
   begin
   Case S is
      when "00" =>
      d <= a;
   e<=d and c;
      when "01" =>
      d <= b;
      e <= d and c;   end case
                       end process
                       end architecture
```

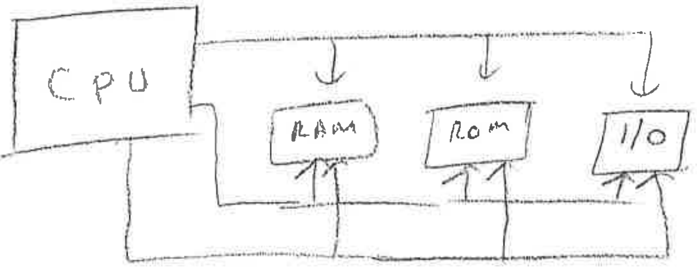
with select) data flow daha kolay

Gömülü sistem yazılım ve donanımın bir [illegible] (6)

mikroişlemcinin yapısı



Mikro denetleyicide, mikro işlemci bulunan, sayıcılar, iletişim birlikte, RAM, ROM, program, belleği, ~~ADC~~ Analog modulu, PWM sinyal üretici, I/O portları, digital dönüştürücüleri vardır.



Mikro denetleyicinin maliyeti daha az

nelerdir (Gömülü)
işletim sistemleri, JavaOS, LynxOS,
ECOS, FreeRTOS, Gömülü Linux, palmOS, prex, VxWorks
mobilinux, NucleusRTOS,

Requirement →
Specification →
Architecture →
Component →
System Integration →

—Daha kolay tasarım yapmayı sağlar, Bütün Sürecin tanımlanması ile tasarımcılara kolay anlaşılır tasarım yapmayı sağlar.

Gereksinimler

—Tasarım için gereksinimler belirlenir

— Tasarım sürecinin en üst tabakası Top down olarak adlandırılır.
Daha sonraki aşamalar Buttom up Design olarak bilinir.

— Bu yapıda katmanlar birbirleriyle ilişkilidir.

*

—isim
— Amaç
— Girişler
— Çıkışlar
—Fonksiyonlar
—performans

—imalat maliyeti
— Fiziksel Boyut ve Ağırlık
— Güç Tüketim
— Gerekli mimari bilgi

Tasarım süreci 5 adımdan oluşur

- Requirements
- Specifications
- Architecture
- Components
- System Integration

Tasarım Ana Amaçları

—Üretim maliyeti ne kadardır

— istenenen performans sağlanmalıdır.

— Güç tüketimi göz önünde bulundurulmalıdır.

* ilk adım komponent ve mimari oluşturmada gerekli bilgi edinilmelidir.

Büyük ölçekli b. Sistemler

— Büyük donanım ve yazılım karmaşıklığı

— programlanabilir mantık dizileri veya konfigüre edilebil işlemci veya ölçeklenebilir işlemci ihtiyacı

işlemci

— Gömülü sistemin kalbidir

— Gömülü sistem tasarımcıları için mikroişlemci veya mikrokontroller şarttır.

— İki ~~temel~~ birime sahiptir Bunlar kontrol birimi ve yürütme birimidir.

— Bu birimler her bir komutun fetch ve execute işlemlerini gerçekleştirir

Mikrokontroller

— tek bir silikon yonga üstünde birleştirilmiş bir mikroişlemci, veri ve program belleği, sayısal giriş ve çıkışlar (I/o) analog girişler ve daha fazla güç veren ve işler katan öteki çevre birimleri (zamanlayıcılar, sayaçlar, kesiciler, analogtan sayısala çeviriciler)

microprocessor ve MICROCONTROLLER arasındaki Farklar.

Microprocessor

— ~~1 ya da 2 bit işlemleri gerçekleştirir~~

— Genel amaçlı bilgi sistemlerin tasarımı için

— Mikroişlemci ise geniş kapsamlı ve duyarlı işlemler yapmak için seçilen bir sistemdir.

microcontroller ~~boyutu~~

— 2 den fazla ✗

— Uygulamalara özel adanmış programları için kullanılır.

— mikrodenetleyici program değişikliği olmayan sabit bir programın sürekli çalışması gereken durumlarda kullanılır.

mikrodenetleyicide bütün birimlerin tek yonga üzerinde bulunması ve mikro işlemcilere göre daha az yer kaplayarak dolayısıyla daha az maliyetlerde çalışabilirler. 4

Uygulama alanı olarak mikroişlemci Mikrodenetleyiciye göre daha kapsamlıdır.

~~BP~~ Gömülü Sistemlerde kullanılan yazılım birimlerine firmware adı verilir. Bu yazılım Rom bellek üzerine kayıtlı bir biçimde kullanılır. Geliştirilen donanım ihtiyaçlarına ve farklı işlevlerin her birini yerine getirme amaçlıdır.

— Sadece bilgisayar bileşenlerinde değil bazı elektronik eşyalarda da bulunan çeşitli donanımların veya cihazın işlevlerini nasıl yerine getireceklerini bildiren ve genellikle tekrar yazılabilir olan ufak kodlardır.

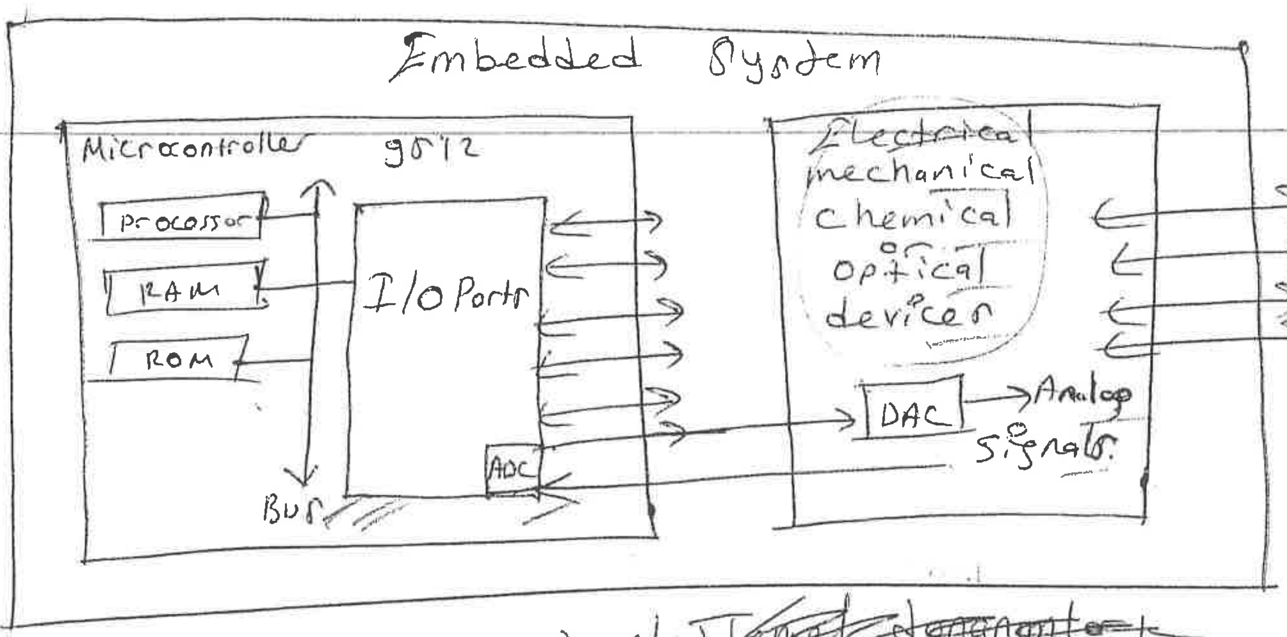
- Fimware salt okunurdur. Okunabilir fakat yazılamaz

Gömülü sistemler Bir ürüne ne katar?

- maliyeti düşürür
- Daha az bileşen, daha küçük hacim, daha az güç tüketimi, az hata kaynağı
- Ürüne katma değer getirir.
- Sistemin kolay güncellemesine olanak tanır.

Gömülü Sistem Tasarım Süreci

- Tasarım Sürecini bilmek 3 avantaj Sağlar
- Sistemin fonksiyonel testlerini icra etmek ve performansı optimize etmemize olanak sağlar.
- Bilgisayar destekli tasarım araçlarını geliştirmemize yardım eder



Gömülü sistemlerin ~~Temel donanımlar~~

Sınıflandırılması

— küçük ölçekli gömülü sistem

— orta ölçekli gömülü ″ Sistem,

— karmasık. ve büyük ölçekli gömülü sistem

küçük ölçekli gömülü

— Birtek 8 veya 16 bit mikrokontroller

— Az donanım, karmasık yazılım

— Batarya ile işletilebilirlik

— Bu sistemleri geliştirmek için C programlama dili

— sürekli olarak çalıştığı zaman güç tüketimi problem.

orta ölçekli

— hem donanım hem yazılım karmasıklığı

— Bir veya birkaç 16 veya 32 bitlik mikrokontrol veya Digital Signal Processor

— |

Gömülü Sistemler

– Sistem, sabit bir plan, program veya kurallar kümesine göre bir veya birden fazla görevi organize eden giriş verilerini kullanarak sonuçlar üreten bir yapıdır.

Bir gömülü sistem

– Bir dijital sistemdir.

– Genellikle bir mikro işlemci kullanılır.

– Sistemin bazı veya tüm fonksiyonlarını yerine getirmek için bir yazılım çalıştırılır.

– Sık sık bir kontrolör gibi kullanılır.

Kayda değer ilk gömülü sistem MIT Instrumentation Laboratory'de Charles Stark Draper tarafından geliştirilen Apollo Guidance Computer olduğu ileri sürülmektedir.

– Belli bir fonksiyonu yerine getirmek için tasarlanmış yazılım ve donanım kombinasyonudur.

– Büyük bir sistem ~~[illegible]~~ içinde gömülü olarak çalışan yapılardır.

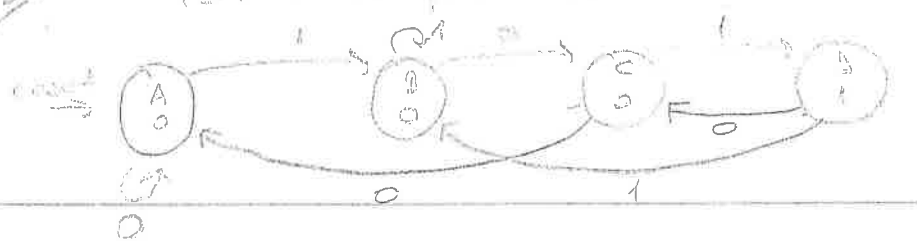
Her bir gömülü sistem

– Gerçek zamanda fiziki ortamdan girişleri alır.

– Gerekli hesaplamaları yapar

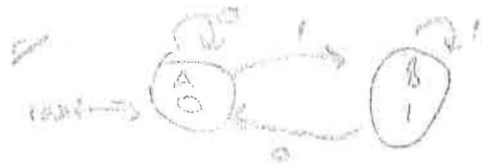
– Olası çıkışları üretir.

55 '101' i tanıyan VHDL



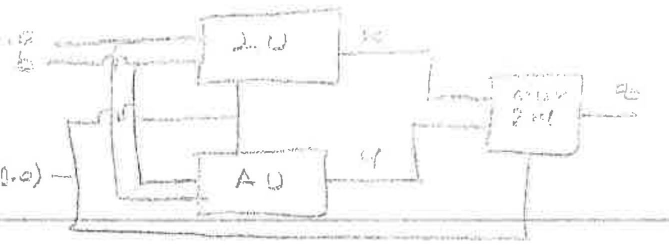
```
entity 101 is
port ( rst, input : in std-logic ;
       cikis : out std-logic );
   end 101;
architecture uyg of 101 is
   type durumlar is (A, B, C, D);
   signal anlikdurum : durumlar;
   begin
     process (rst, input)
       begin
   if rst = '0' then anlikdurum <= A;
   else
        case anlikdurum is
   when A => if input = '0' then anlikdurum <= A;
              else anlikdurum <= B;
                end if;
   when B => if input = '0' then anlikdurum <= C;
              else anlikdurum <= B;
                end if;
   when C => if input = '0' then anlikdurum <= A;
              else anlikdurum <= D;
                end if;
   when D => if input = '0' then anlikdurum <= C;
              else anlikdurum <= [illegible]
                end if;
             end case;
              end if;
              end process;
      if anlikdurum = D then cikis <= '1';
       else cikis <= '0';
              end if;
              end uyg;
```

D



```
      library IEEE;
      use IEEE.std_logic_1164.all;

      entity moore is
   port( rst, clk, input = in std_logic;
          cks = out std_logic);
        end moore;
 architecture uyg of mimari is
      type durum is (A,B);
     signal anlikdurum = durum;
     begin
       process (rst, input)
       begin
    if reset = '0' then anlikdurum <= A;

    else
          case anlikdurum is

       when A => if input = '0' then anlikdurum <= A;
                    else anlikdurum <= B;
                     end if;
       when B => if input = '0' then anlikdurum <= A;
                      else anlikdurum <= B;
                        end if;

                 end case;
            if anlikdurum = B then cikis <= '1';
               else cikis <= '0';
                    end if;
                    end if;
                  end uyg;
```

1/



```
entity ALU is
port ( a, b = in std_logic_vector (7 downto 0);
        sel = in std_logic_vector (2 downto 0);
        clk = in std_logic;
         z = out std_logic_vector (7 downto 0) );
  end ALU;
 achitecture wyg of ALU is
 signal x, y = std_logic_vector (7 downto 0);
  begin
    process (clk, sel)
     begin
   if clk'event and clk='1' then
     case sel(2downto0) is
  when sel (2downto0) = "000" =>      x <= a+1;
                                      y <= aorb;
```

3/ fibonacci ?

```
entity fibonacci is
port ( birinci, ikinci : integer;
        sonuc : integer;
        clk : in std_logic);
   end fibonacci;
  architecture wyg of fibonacci is
   birinci := 0;
   ikinci := 1;
  signal sayac : integer := 0;
   begin process(clk)
          begin
      if ( sayac < 10) then
     sonuc := birinci + ikinci;
      birinci := ikinci;
       ikinci := sonuc;
```

Fibonacci 0 - 1 - 1 - 2 - 3 - 5 - —

```
Library IEEE;
use IEEE.std_logic_1164.all;
entity fib is
port ( ilksayi, ikincisayi : integer;
       sonuc : integer;
       clk : in std_logic );
  end fib;
architecture uyg of fib is
   ilksayi := 0;
   ikincisayi := 1;
  signal sayac : integer := 0;
begin
process (clk)
begin
   if ( sayac < 10 ) then
    sonuc := ilksayi + ikincisayi;
      ilksayi := ikincisayi;
      ikincisayi := sonuc;
      end if;
    sayac := sayac + 1;
      end process;
      end uyg;
```

mikroişlemci

→ Veri yolu yok
→ Hafıza birimleri yok
→ Genel amaçlı kullanılırlar
→ Daha [illegible]

→ Daha az hacim
maliyet
bileşen
hata oranı
az güç tüketimi

üretime değer katıyor

mikrokontrolör

→ Veri yolu var
→ ADC – EEPROM – I/O ports
→ Özel uygulamalar için
→ Veriler ve hafıza komutları
aynı yonga üzerinde
olduğu için hızlıdır

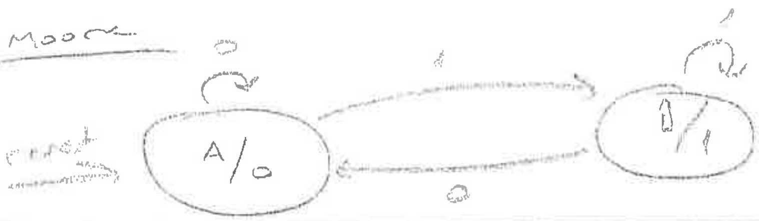
Üniteler arası [illegible]
[illegible]

KARE DALGA

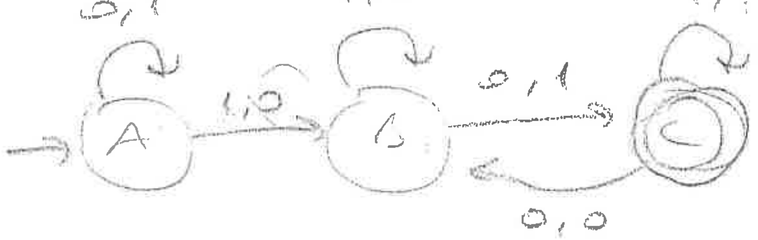
```
→ library IEEE;
  use IEEE.std_logic_1164.all;*
  use IEEE.std_logic_ARITH.all;*
  ----------------------------------------
  use IEEE.std_logic_unsigned.all;

→ entity karedalga is
  port (
    clk : in std_logic;
    cikis : out std_logic);
     end karedalga;

→ architecture karedalgauyg of karedalga is
  signal sayac : integer := 0;
  signal temp : integer := 49;
   begin
   process (clk)
   begin
   if (clk'event and clk='1') then
      sayac <= sayac + 1;          0<x<24
   if ( sayac <= temp/2 ) then
      cikis <= '1';                (24<x<49)
     end if;
   if (sayac > temp/2 and sayac < temp) then
      cikis <= '0';
       end if;

    if (sayac = temp+1) then
      sayac <= 0;
      endif;
      endif;
     end process;
      end karedalgauyg;
```

if içindeki, atama değildir

Moore



```
entity moore is
port ( clk, input, reset : in std_logic;
        cikis : out std_logic );
    end moore;
  architecture uyg of moore is
     type durum is (A, B);
     signal anlikdurum : durum;
      begin
       process (reset, clk)
       begin
        if reset = '0' then anlikdurum <= A;
       elsif rising_edge (clk) then
       case anlikdu
```



```
      case anlikdurum is
when A => if input = '1'  then anlikdurum <= B;
                               cikis <= '0';
```

= ROM =

```
entity ROM is
 port( clk, rst, enable, read : in std_logic;
       Adress : in std_logic_vector(4 downto 0);
       Dataout : out std_logic_vector (8 downto 0)
       );
            end ROM;
```

ROM MODULE

⇒ 32 × 8 Rom module

```
→ Library ieee;
    use ieee.std_logic_1164.all;
    use ieee.std_logic_arith.all;
    use ieee.std_logic_unsigned.all;

→   entity ROM is
      port(  clock   : in std_logic;
             reset   : in std_logic;
             enable  : in std_logic;
             read    : in std_logic;
             Address : in std_logic_vector(4 downto 0);
             Data_out : out std_logic_vector(7 downto 0));
        end ROM;

→   architecture Dyg of ROM is
          type ROM_Bellek is array(0 to 31)
          of std_logic_vector(7downto 0);
       constant Content : ROM_Bellek := (

        { 0 => "00000001",
        { 1 => "00000010",
 32     {    //     //    //
 adet   {    //     //    //
        {    //     //    //
        {
        {OTHERS => "11111111" );

   begin
    process(clock,reset,read,Address)
    begin
        if (reset = '1') then  Data_out <= "ZZZZZZZZ";
      elsif (clk'event and clock = '1') then
           if enable = '1' then
              if (read = '1' ) then  data_out <= Content(conv_integer(Add
        else  Data_out <= "ZZZZZZZZ";  end if end process end
```

```
entity 101 is
part( clk,rst,input : in std_logic;
        cikis : out std_logic);
    end 101;
```

```
architecture myp of 101 is
  type durumlar is (A,B,C,D);
  signal anlikdurum : durumlar;
  begin
     process(clk,rst,input)
     begin
     if rst = '0' then
        anlikdurum <= A;
      elsif  rising_edge(clk)  then
case anlikdurum is;
          when  A => if input ='0' anlikdurum <= A;
                     else anlikdurum <= B;   end if;
          when  B => if  input= '0' anlik durum <= C;
                     else anlikdurum <= B;  end if;
          when  C => if input = '0' anlikdurum <= A;
                     else anlikdurum <= D;  end if;
          when  D => if input = '0' anlikdurum <= C;
                     else anlikdurum <= B;   end if

                     end case;
                     end if;
                     end if;
                     end process;
               if   anlikdurum = D  then   cikis <= '1';
                     else   cikis <= '0';
                                  end if;
                                  end myp;
```

= ROM =

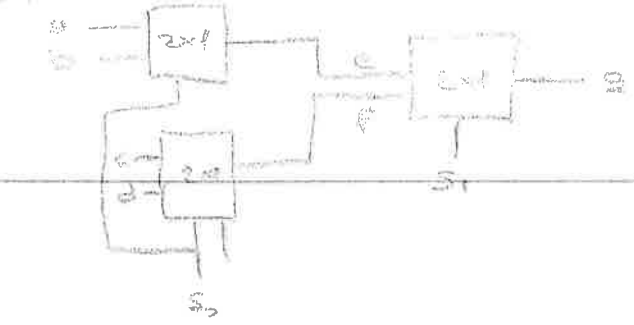
→ 32 × 8 ROM

```
=> Library IEEE;
   use ieee.std_logic_1164.all;
   use ieee.std_logic_arith.all;
   use ieee.std_logic_unsigned.all;
=> entity ROM is
   port (   clock : in std_logic;
            reset : in std_logic;
            enable : in std_logic;
            read : in std_logic;
            address : in std_logic_vector (4 downto 0);
            dataout : out std_logic_vector (7 downto 0));
            end ROM;
   architecture sys of ROM is
            type ROM_Dizi is array (0 to 31)
            of std_logic_vector (7 downto 0);
      constant Content : Rom_Dizi := (
            0 => "00000001",
            1 => "00000010",
               |
               |
            14 => "00001111",
         OTHERS => "11111111");
      begin
        process (clock, reset, read, address)
          begin
             if (reset = '1') then
               dataout <= "ZZZZZZZZ";
             elsif (clock'event and clock = '1') then
                 if enable = '1' then
                 if read = '1' then
                   dataout <= Content (conv_integer (Address));
                   else
                   dataout <= "ZZZZZZZZ";
                   end if;
                 end if;
               end if;
         end process
```

= RAM =

4 x 4 RAM

```
-> library IEEE;
   use ieee.std_logic_1164.all;
   use ieee.std_logic_arith.all;
   use ieee.std_logic_unsigned.all;

-> entity SRAM is
   generic (   width : integer := 4;
               depth : integer := 4;
               addr : integer := 2 );
   port (  clock, enable, read, write : in std_logic;
           read_addr : in std_logic_vector (addr - 1 downto 0);
           write_addr : in std_logic_vector (addr - 1 downto 0);
           data_in : in std_logic_vector (width - 1 downto 0);
           data_out : out std_logic_vector (width - 1 downto 0) );

        end SRAM;

   architecture wyg of SRAM is
     type ram_type is array (0 to depth - 1) of
       std_logic_vector (width - 1 downto 0);
     signal tmp_ram : ram_type;
     begin
 process (clock, read)
  begin
 if (clock'event and clock = '1') then
 if  enable = '1' then
  if  read = '1' then
  data_out <= tmp_ram (conv_integer (read addr));
   else
  data_out <= (Data_out 'range => 'Z');
    end if;
     end if;
    end if;
     end process;

 process ( clock, write )
   begin
    if ( clk'event and clk "1")
      if enable = '1'
       if write = '1'
 tmp_ram (conw integer (write Addr)) <=
                                    Data in;
       end if;
       end if;
      end if;
      end process;
       end wyg;
```

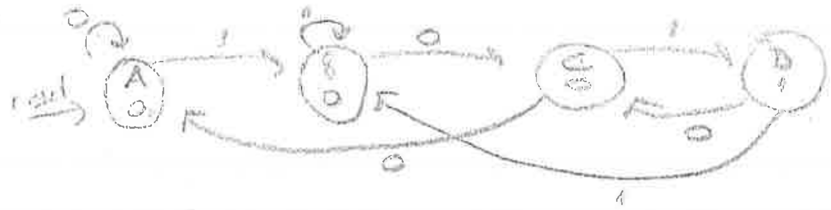


```
 library IEEE ;
 use IEEE . std_logic_1164 . all;
entity 21imux is
port ( a,b,c,d = in std_logic_vector (3downto0);
        s = in std_logic_vektor (1downto);
        g = out std_logic_vektor (3downto0) );

   end 21imux ;
 architecture uyg of 21imux is
    signal e,f= std_logic_vektor(3downto0);

   begin
   process (s)
    begin
   if (s(0)='0') then    e<=a;
                         f<=c;

     else      e<=b;
               f<=d;
        end if;
     if ( s(1) = 1)  then   g<=f;
                else        g<=e;
              end if;
              end process;
              end uyg;
```

sayıcı 3 bitlik ama 6'ya kadar sayacak.

```
 library IEEE;
use IEEE.std_logic_1164.all;
entity sayici is
port ( clk : in std_logic;
        cikis : out std_logic_vektor ( 2 downto 0));
    end sayici;
 architecture uyg of sayici is
  signal sayac : std_logic_vektor (2 downto 0) := "000";
   begin
    process ( clk)
     begin
     if rising_Edge ( clk ) then
         sayac <= sayac + "001";
        if  sayac = "110" then
            sayac <= "000";
             end if;
           end if;
         end process;
           end uyg;
```

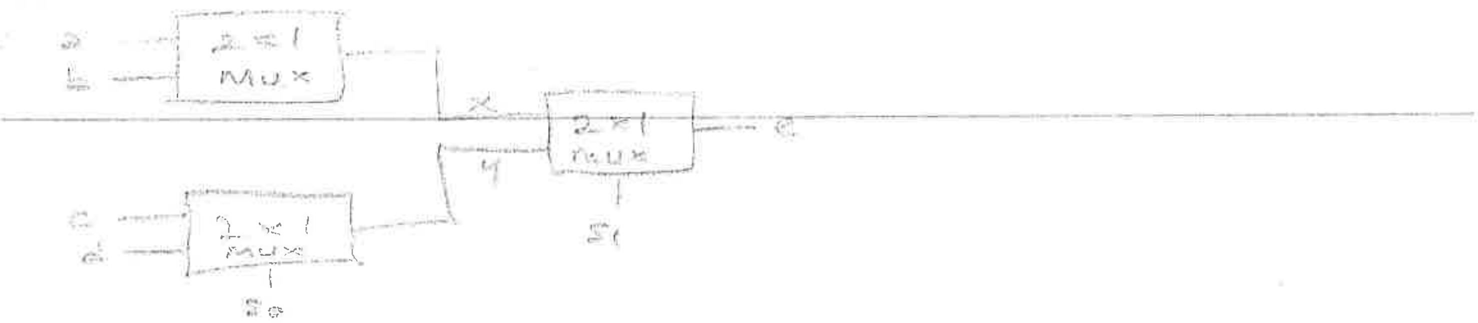
5c



101'i tanıyan
DFA'nın kodu
??

```
library IEEE;
use IEEE.std_logic_1164.all;
entity 101 is
  port ( input, rst, clk : in std_logic;
         cikis : out std_logic);
     end 101;
   architecture uyg of 101 is
    type durumlar is (A,B,C,D);
     signal anlikdurum = durumlar;
    begin
     process(clk,rst,input)
      begin
      if reset='0' then
         anlikdurum <= A;
        elsif rising_edge(clk) then
           case anlikdurum is
         when A =>
              if input='0' then
                  anlikdurum <= A;
              else
                  anlikdurum <= B;
           when B =>
                if input = '0' then
                   anlikdurum <= A;
                else
                   anlikdurum <= B;


              if anlikdurum = A,B,C;
                   cikis <= '0';
               else
                   cikis <= '1'
```

3/ 4x1 Mux'ın 2x1 muxlar ile gerçekleştirilmesi.

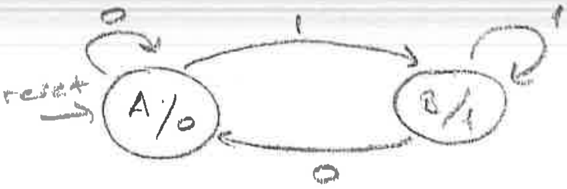


```
library ieee;
use ieee.std_logic_1164.all;
entity muxuygula is
port( a,b,c,d : in std_logic;
        s : in std_logic_vector (1 downto 0);
        e : out std_logic );
  end muxuygula;
architecture uygulama of muxuygula is
  signal x,y : std_logic;
   begin
    process (s)
   begin
   if s(0) = '0' then   x <= a;
                        y <= c;

     else
       x <= b;
       y <= d;
          end if;
       if s(1) = '1' then   e <= x;
        else    e <= y;
          end if;
            end process;
              end uyg;
```

1) sayıcı [illegible]

```
entity sayici is
port ( clk : in std_logic;
       cikis : out std_logic_vektor ( 3 downto 0 ) );
  end sayici;
architecture uyg of sayici is
  signal sayac : std_logic_vektor (3 downto) := "0000";
  begin
    process (clk)
      begin
      if rising_edge (clk) then
        sayac <= sayac + "0001";
        end if;
        end process;
      cikis <= sayac;
      end architecture;
```

5c Moore



```
  entity moore is
port ( clk, input, reset : in std_logic;
     cikis out std_logic );
     end moore;
 architecture uyg of moore is
     type durum is (A,B);
     signal anlikdurum : durum;
     begin
      process (reset, clk)
      begin
       if reset = '0' then  anlikdurum <= A;
      elsif rising_edge (clk) then
         case anlikdurum is
    when   A => if input = '0' then  anlikdurum <= A;
                else  anlikdurum <= B;
                end if;
    when   B => if input = '0' then  anlikdurum <= A;
                else  anlikdurum <= B;
                end if;
             end if;
             end process;
                               if anlikdurum = B then  cikis <= "1";
                               else  cikis <= "0";
                                  end if;
                                     end uyg;
```