
بسم الله الرحمن الرحیم

دانشگاه پیام نور مرکز ملایر

فصل پنجم

مجموعه مثال های کاربردی از

VHDL

ترم بهار۹۷-۹۶

دانشگاه پیام نور مرکز ملایر

دانشکده فنی و مهندسی

مهندسی کامپیوتر گرایش نرم افزار

عنوان درس: طراحی کامپیوتری سیستمهای دیجیتال

استاد راهنما: سرکار خانم مهندس بابایی

دروازه های منطقی پایه:

- هر توصیف طرح به زبانVHDL شامل حداقل یک جفت ENTITY/ ARCHITECTUREیا یک ENTITY و تعدادیARCHITECTURE میباشد. بخشENTITY برای تعریف درگاه های ورودی خروجی مدار استفاده میشود درحالی که کد توصیف مدار درARCHITECTURE قرار گرفته است. توصیف مدار با استفاده از دستورات مختلف زبانVHDL و به روشهای مختلف صورت میگیرد. در این بخش انواع دروازه های منطقی را ملاحظه خواهید کرد.

EXAMPLE(5-1): DRIVERتوصیف یک درایور(بافر)

```
----------------------------------------
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
----------------------------------------
ENTITY DRIVER IS
PORT(  X: IN STD_LOGIC;
       F: OUT STD_LOGIC );
END DRIVER;
----------------------------------------
ARCHITECTURE BEHV2 OF DRIVER IS
BEGIN
    F <= X;
END BEHV2;
----------------------------------------
```

EXAMPLE(5-1): DRIVER

```
----------------------------------------
ENTITY DRIVER IS
PORT(  X: IN STD_LOGIC;
       F: OUT STD_LOGIC );
END DRIVER;
----------------------------------------
ARCHITECTURE BEHV1 OF DRIVER IS
BEGIN
    PROCESS(X)
    BEGIN
        -- COMPARE TO TRUTH TABLE
        IF (X='1') THEN
            F <= '1';
        ELSE
            F <= '0';
        END IF;
    END PROCESS;
END BEHV1;
----------------------------------------
```

F <= x;

EXAMPLE(5-3): AND GATE توصیف دروازه ANDدو ورودی

```
-------------------------------------
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
-------------------------------------
ENTITY AND_ENT IS
PORT( X: IN STD_LOGIC;
      Y: IN STD_LOGIC;
      F: OUT STD_LOGIC );
END AND_ENT;
-------------------------------------
ARCHITECTURE BEHAV2 OF AND_ENT IS
BEGIN
    F <= X AND Y;
END BEHAV2;
-------------------------------------
```

EXAMPLE(5-3): AND GATE

```
--------------------------------------
ENTITY AND_ENT IS
PORT(   X: IN STD_LOGIC;
        Y: IN STD_LOGIC;
        F: OUT STD_LOGIC );
END AND_ENT;
--------------------------------------
ARCHITECTURE BEHAV1 OF AND_ENT IS
BEGIN
    PROCESS(X, Y)
    BEGIN
        -- COMPARE TO TRUTH TABLE
        IF ((X='1') AND (Y='1')) THEN
           F <= '1';
        ELSE
           F <= '0';
        END IF;
    END PROCESS;
END BEHAV1;;
--------------------------------------
```

F <= x and y;

EXAMPLE(5-6): NOR GATE توصیف یک دو ورودی NOR

```
-------------------------------------
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
-------------------------------------
ENTITY NOR_ENT IS
PORT( X: IN STD_LOGIC;
      Y: IN STD_LOGIC;
      F: OUT STD_LOGIC);
END NOR_ENT;
-------------------------------------
ARCHITECTURE BEHV2 OF NOR_ENT IS
BEGIN
    F <= X NOR Y;
END BEHV2;
-------------------------------------
```

EXAMPLE(5-6): NOR GATE

```
--------------------------------------
ENTITY NOR_ENT IS
PORT(    X: IN STD_LOGIC;
         Y: IN STD_LOGIC;
         F: OUT STD_LOGIC);
END NOR_ENT;
--------------------------------------
ARCHITECTURE BEHV1 OF NOR_ENT IS
BEGIN
    PROCESS(X, Y)
    BEGIN
        -- COMPARE TO TRUTH TABLE
         IF (X='0' AND Y='0') THEN
          F <= '1';
         ELSE
            F <= '0';
         END IF;
    END PROCESS;
END BEHV1;
--------------------------------------
```

F <= x nor y;

EXAMPLE(5-7): XOR GATE توصیف یک XOR دو ورودی

```
--------------------------------------
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
--------------------------------------
ENTITY XOR_ENT IS
PORT( X: IN STD_LOGIC;
      Y: IN STD_LOGIC;
      F: OUT STD_LOGIC);
END XOR_ENT;
--------------------------------------
ARCHITECTURE BEHV2 OF XOR_ENT IS
BEGIN
    F <= X XOR Y;
END BEHV2;
--------------------------------------
```

EXAMPLE(5-8): XNOR توصیف یک XNOR دو ورودی

```
-------------------------------------
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
-------------------------------------
ENTITY XNOR_ENT IS
PORT( X: IN STD_LOGIC;
      Y: IN STD_LOGIC;
      F: OUT STD_LOGIC);
END XNOR_ENT;
-----------------------------------------
ARCHITECTURE BEHV2 OF XNOR_ENT IS
BEGIN
    F <= X XNOR Y;
END BEHV2;
-----------------------------------------
```

EXAMPLE(5-8): XNOR

```
-------------------------------------
ENTITY XNOR_ENT IS
PORT( X: IN STD_LOGIC;
      Y: IN STD_LOGIC;
      F: OUT STD_LOGIC);
END XNOR_ENT;
-------------------------------------
ARCHITECTURE BEHV1 OF XNOR_ENT IS
BEGIN
    PROCESS(X, Y)
    BEGIN
      -- COMPARE TO TRUTH TABLE
      IF (X/=Y) THEN
         F <= '0';
      ELSE
         F <= '1';
      END IF;
    END PROCESS;
END BEHV1;
-------------------------------------
```

```
F <= x xnor y;
```

8_BIT TRI_STATE DRIVER بافر سه حالته ۸ بیتی

```
----------------------------------------------------
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
----------------------------------------------------
ENTITY TRISTATE_DR IS
PORT(  D_IN:   IN STD_LOGIC_VECTOR(7 DOWNTO 0);
       EN:     IN STD_LOGIC;
       D_OUT: OUT STD_LOGIC_VECTOR(7 DOWNTO 0) );
END TRISTATE_DR;
----------------------------------------------------
ARCHITECTURE BEHAVIOR OF TRISTATE_DR IS
BEGIN
     D_OUT <= D_IN WHEN EN='1' ELSE "ZZZZZZZZ";
END BEHAVIOR;
----------------------------------------------------
```

مثال ۵-۱۱) مالتی پلکسر ۴ به ۱ سه بیتی

```
ENTITY MUX IS
PORT(  I3:    IN STD_LOGIC_VECTOR(2 DOWNTO 0);
  I2:  IN STD_LOGIC_VECTOR(2 DOWNTO 0);
  I1:  IN STD_LOGIC_VECTOR(2 DOWNTO 0);
  I0:  IN STD_LOGIC_VECTOR(2 DOWNTO 0);
  S:   IN STD_LOGIC_VECTOR(1 DOWNTO 0);
  O:   OUT STD_LOGIC_VECTOR(2 DOWNTO 0));
END MUX;
-------------------------------------------------
ARCHITECTURE BEHV1 OF MUX IS
BEGIN
    PROCESS(I3,I2,I1,I0,S)
    BEGIN
        -- USE CASE STATEMENT
        CASE S IS
             WHEN "00" =>        O <= I0;
             WHEN "01" =>        O <= I1;
             WHEN "10" =>        O <= I2;
             WHEN "11" =>        O <= I3;
             WHEN OTHERS =>      O <= "ZZZ";
      END CASE;
    END PROCESS;
END BEHV1;
```

مثال ۵-۱۱) مالتی پلکسر ۴ به ۱ سه بیتی

```
ENTITY MUX IS
PORT(  I3:    IN STD_LOGIC_VECTOR(2 DOWNTO 0);
  I2:  IN STD_LOGIC_VECTOR(2 DOWNTO 0);
  I1:  IN STD_LOGIC_VECTOR(2 DOWNTO 0);
  I0:  IN STD_LOGIC_VECTOR(2 DOWNTO 0);
  S:   IN STD_LOGIC_VECTOR(1 DOWNTO 0);
  O:   OUT STD_LOGIC_VECTOR(2 DOWNTO 0));
END MUX;
-------------------------------------------------

ARCHITECTURE BEHV2 OF MUX IS
BEGIN
    -- USE WHEN.. ELSE STATEMENT
    O <=    I0 WHEN S="00" ELSE
            I1 WHEN S="01" ELSE
            I2 WHEN S="10" ELSE
            I3 WHEN S="11" ELSE
            "ZZZ";
END BEHV2;
-------------------------------------------------
```

مثال ۵-۱۴) مقایسه کننده N بیتی

```
-------------------------------------------------
-- EXAMPLE(5-14): N-BIT COMPARATOR
-- TWO N-BIT INPUTS & THREE 1-BIT OUTPUTS
-------------------------------------------------
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
-------------------------------------------------
ENTITY COMPARATOR IS
GENERIC(N: NATURAL :=2);
PORT( A:IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
      B:IN STD_LOGIC_VECTOR(N-1 DOWNTO 0);
      LESS:OUT STD_LOGIC;
      EQUAL:OUT STD_LOGIC;
      GREATER:OUT STD_LOGIC);
END COMPARATOR;
```

مثال ۵-۱۴) مقایسه کننده N بیتی

```
ARCHITECTURE BEHV OF COMPARATOR IS
BEGIN
    PROCESS(A,B)
    BEGIN
      IF (A<B) THEN
      LESS <= '1';
      EQUAL <= '0';
      GREATER <= '0';
      ELSIF (A=B) THEN
      LESS <= '0';
      EQUAL <= '1';
      GREATER <= '0';
      ELSE
      LESS <= '0';
      EQUAL <= '0';
      GREATER <= '1';
      END IF;
   END PROCESS;
END BEHV;
```

طراحی مدارات ترتیبی و ماشین حالت

- مدارات ترتیبی شامل:
  - سیگنالهای ورودی
  - سیگنالهای خروجی
  - سیگنال کلاک
  - احیانا یکسری سیگنالهای کنترلی دیگر (مانند RESET)
- سیگنالهای کنترلی می توانند فعال صفر یا یک باشند
- انتقال از یک وضعیت به وضعیت دیگر لبه سیگنال کلاک
  - (CLOCK 'EVENT AND CLOCK='1')
  - (CLOCK 'EVENT AND CLOCK='0')

مثال ۵-۱۷) نگهدار D

```
-----------------------------------------
ENTITY D_LATCH IS
PORT( DATA_IN:     IN STD_LOGIC;
      ENABLE:      IN STD_LOGIC;
      DATA_OUT:    OUT STD_LOGIC );
END D_LATCH;
-----------------------------------------
ARCHITECTURE BEHV OF D_LATCH IS
BEGIN
  --COMPARE THIS TO D FLIPFLOP
    PROCESS(DATA_IN, ENABLE)
    BEGIN
      IF (ENABLE='1') THEN
            --NO CLOCK SIGNAL HERE
          DATA_OUT <= DATA_IN;
      END IF;
    END PROCESS;
END BEHV;
-----------------------------------------
```

مثال ۵-۱۸) فیلیپ فلاپ D

```
-----------------------------------------
ENTITY DFF IS
PORT( DATA_IN:    IN STD_LOGIC;
      CLOCK:      IN STD_LOGIC;
      DATA_OUT:   OUT STD_LOGIC);
END DFF;
-----------------------------------------
ARCHITECTURE BEHV OF DFF IS
BEGIN
    PROCESS(DATA_IN, CLOCK)
    BEGIN
        -- CLOCK RISING EDGE
      IF (CLOCK='1' AND CLOCK'EVENT) THEN
          DATA_OUT <= DATA_IN;
      END IF;
    END PROCESS;
END BEHV;
-----------------------------------------
```

مثال ۵-۲۱)
ثبات انتقالی

```
-------------------------------------------------
ENTITY SHIFT_REG IS
PORT(  I:     IN STD_LOGIC;
       CLOCK: IN STD_LOGIC;
       SHIFT: IN STD_LOGIC;
       Q:     OUT STD_LOGIC );
END SHIFT_REG;
-------------------------------------------------
ARCHITECTURE BEHV OF SHIFT_REG IS
    -- INITIALIZE THE DECLARED SIGNAL
    SIGNAL S: STD_LOGIC_VECTOR(2 DOWNTO 0):="111";
BEGIN
    PROCESS(I, CLOCK, SHIFT, S)
    BEGIN
  -- EVERYTHING HAPPENS UPON THE CLOCK CHANGING
       IF CLOCK'EVENT AND CLOCK='1' THEN
          IF SHIFT = '1' THEN
              S <= I & S(2 DOWNTO 1);
         END IF;
       END IF;
    END PROCESS;
    -- CONCURRENT ASSIGNMENT
    Q <= S(0);
END BEHV;
-------------------------------------------------
```

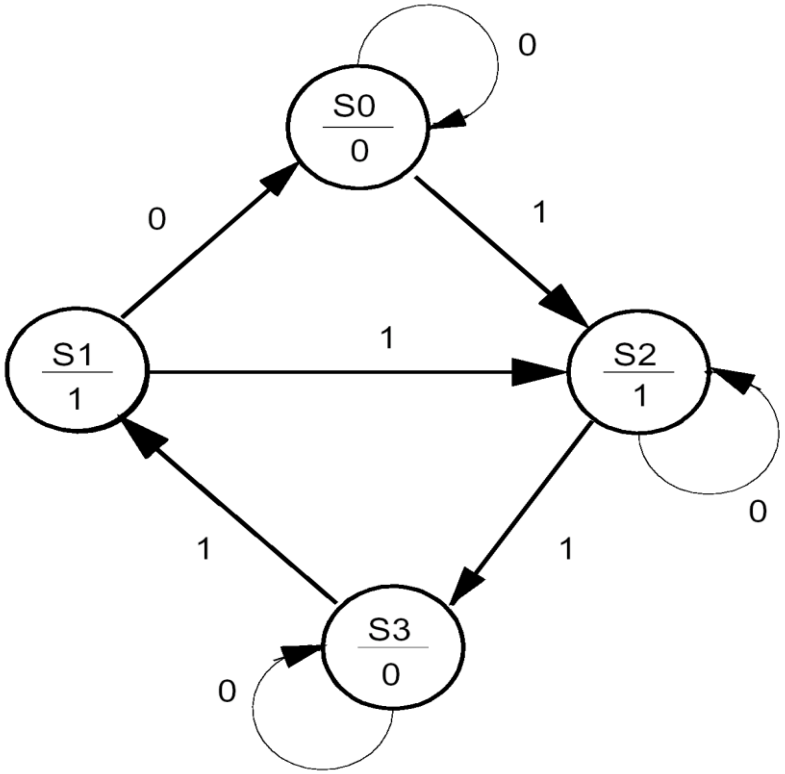
مثال ۵-۲۲)
شمارنده N بیتی

```
-------------------------------------------------------
ENTITY COUNTER IS
GENERIC(N: NATURAL :=2);
PORT(  CLOCK: IN STD_LOGIC;
       CLEAR: IN STD_LOGIC;
       COUNT: IN STD_LOGIC;
       Q:     OUT STD_LOGIC_VECTOR(N-1 DOWNTO 0));
END COUNTER;
-------------------------------------------------------
ARCHITECTURE BEHV OF COUNTER IS
    SIGNAL PRE_Q: STD_LOGIC_VECTOR(N-1 DOWNTO 0);
BEGIN
    -- BEHAVIOR DESCRIBE THE COUNTER
    PROCESS(CLOCK, COUNT, CLEAR)
    BEGIN
       IF CLEAR = '1' THEN
              PRE_Q <= PRE_Q - PRE_Q;
       ELSIF (CLOCK='1' AND CLOCK'EVENT) THEN
             IF COUNT = '1' THEN
                    PRE_Q <= PRE_Q + 1;
              END IF;
      END IF;
    END PROCESS;
    -- CONCURRENT ASSIGNMENT STATEMENT
    Q <= PRE_Q;
END BEHV;
-------------------------------------------------------
```

مثال 5-23) ماشین حالت مور

مقدار خروجی آن فقط وابسته به حالت فعلی است

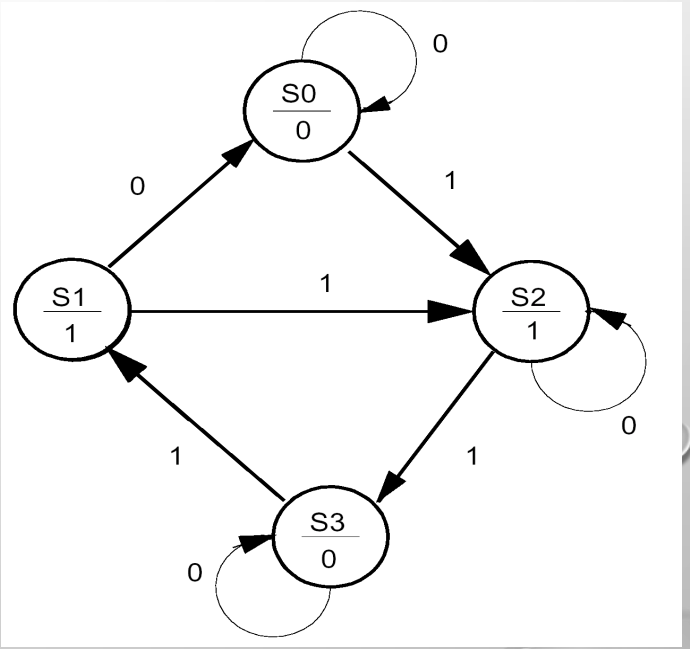
هیچ ورودی دخالت مستقیم در عبارت بولی خروجی ها ندارد



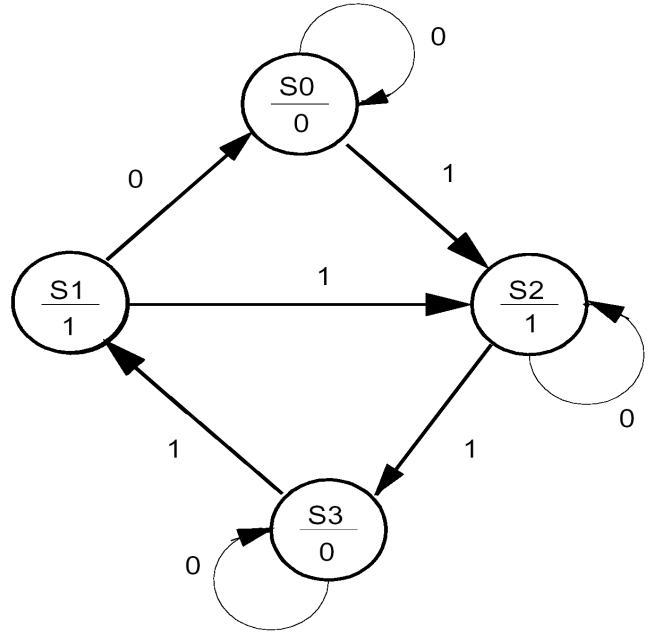
حالت فعلی	حالت بعدی		خروجی
	X=0	X=1	z
S0	S0	S2	0
S1	S0	S2	1
S2	S2	S3	1
S3	S3	S1	0

مثال 5-23) ماشین حالت مور

```
-------------------------------------------------------
ENTITY MOORE IS -- MOORE MACHINE
        PORT(X, CLOCK: IN BIT;
             Z: OUT BIT);
END MOORE;
-------------------------------------------------------
ARCHITECTURE BEHAVIOR OF MOORE IS
    TYPE STATE_TYPE IS (S0, S1, S2, S3);
    SIGNAL CURRENT_STATE, NEXT_STATE: STATE_TYPE;
BEGIN
-------------------------------------------------------
-- PROCESS TO HOLD COMBINATIONAL LOGIC
COMBIN: PROCESS(CURRENT_STATE, X)
BEGIN
  CASE CURRENT_STATE IS
  WHEN S0 =>
       Z <= '0';
       IF X = '0' THEN
          NEXT_STATE <= S0;
       ELSE
          NEXT_STATE <= S2;
       END IF;
  WHEN S1 =>
       Z <= '1';
       IF X = '0' THEN
          NEXT_STATE <= S0;
       ELSE
          NEXT_STATE <= S2;
       END IF;
```



ماشین حالت مور




```
  WHEN S2 =>
        Z <= '1';
        IF X = '0' THEN
           NEXT_STATE <= S2;
        ELSE
           NEXT_STATE <= S3;
        END IF;
    WHEN S3 =>
        Z <= '0';
        IF X ='0' THEN
           NEXT_STATE <= S3;
        ELSE
           NEXT_STATE <= S1;
        END IF;
  END CASE;
END PROCESS COMBIN;


-------------------------------------------------------------
-- PROCESS TO HOLD SYNCHRONOUS ELEMENTS (FLIP-FLOPS)
SYNCH: PROCESS
BEGIN
  WAIT UNTIL CLOCK'EVENT AND CLOCK = '1';
  CURRENT_STATE <= NEXT_STATE;
END PROCESS SYNCH;
END BEHAVIOR;
-------------------------------------------------------------
```

مثال ۵-۲۵) آشکارساز توالی بیت (ماشین مور)

- توصیف رفتاری یک ماشین حالت از نوع مور
- هدف: تشخیص رشته ی بیتی "1110101101" است
- هر آشکار ساز توالی N بیتی به روش مور N+1 حالت مورد نیاز دارد
- چون ۱۰ بیت داریم ۱۱ حالت در نظر می گیریم

مثال 5-25)
آشکارساز توالی بیت

```
-------------------------------------------------------------
ENTITY SEQDETECTOR IS
PORT    (CLK, RESET    :IN  STD_LOGIC;
         INPUT         :IN  STD_LOGIC;
         OUTPUT :OUT STD_LOGIC  );
END SEQDETECTOR ;
-------------------------------------------------------------
ARCHITECTURE STAT OF SEQDETECTOR IS
TYPE STATE_TYPE IS (S0, S1, S2, S3, S4, S5, S6, S7, S8, S9, S10);
SIGNAL STATE, NEXT_STATE : STATE_TYPE;
BEGIN
-------------------------------------------------------------
STATE_REGISTER: PROCESS (CLK, RESET)
BEGIN
  IF RESET = '0' THEN
        STATE <= S0;
        OUTPUT<='0';
  ELSIF CLK'EVENT AND CLK = '1' THEN
        STATE <= NEXT_STATE;
  END IF;
END PROCESS STATE_REGISTER;
-------------------------------------------------------------
```

مثال 5-25)
آشکارساز توالی بیت

```
STATE_LOGIC : PROCESS ( STATE )
  BEGIN
  CASE STATE IS
       WHEN S0 =>                                  --DETECT "1"
              OUTPUT<='0';
              IF (INPUT='1') THEN
              NEXT STATE <=S1;
              ELSIF (INPUT='0') THEN
              NEXT STATE <=S0;
              END IF;
       WHEN S1 =>                                  --DETECT "11"
              OUTPUT<='0';
              IF (INPUT='1') THEN
              NEXT STATE <=S2;
              ELSIF (INPUT='0') THEN
              NEXT STATE <=S0;
              END IF;
       WHEN S2 =>                                  --DETECT "111"
              OUTPUT<='0';
              IF (INPUT='1') THEN
              NEXT STATE <=S3;
              ELSIF (INPUT='0') THEN
              NEXT STATE <=S0;
              END IF;
```

مثال 5-25) آشکارساز توالی بیت

```
WHEN S3 =>                          --DETECT "1110"
OUTPUT<='0';
       IF (INPUT='0') THEN
       NEXT_STATE <=S4;
       ELSIF (INPUT='1') THEN
       NEXT_STATE <=S2;
       END IF;
WHEN S4 =>                          --DETECT "11101"
       OUTPUT<='0';
       IF (INPUT='1') THEN
       NEXT_STATE <=S5;
       ELSIF (INPUT='0') THEN
       NEXT_STATE <=S0;
       END IF;
WHEN S5 =>                          --DETECT "111010"
       OUTPUT<='0';
       IF (INPUT='0') THEN
       NEXT_STATE <=S6;
       ELSIF (INPUT='1') THEN
       NEXT_STATE <=S2;
       END IF;
```

مثال 5-25)
آشکارساز توالی بیت

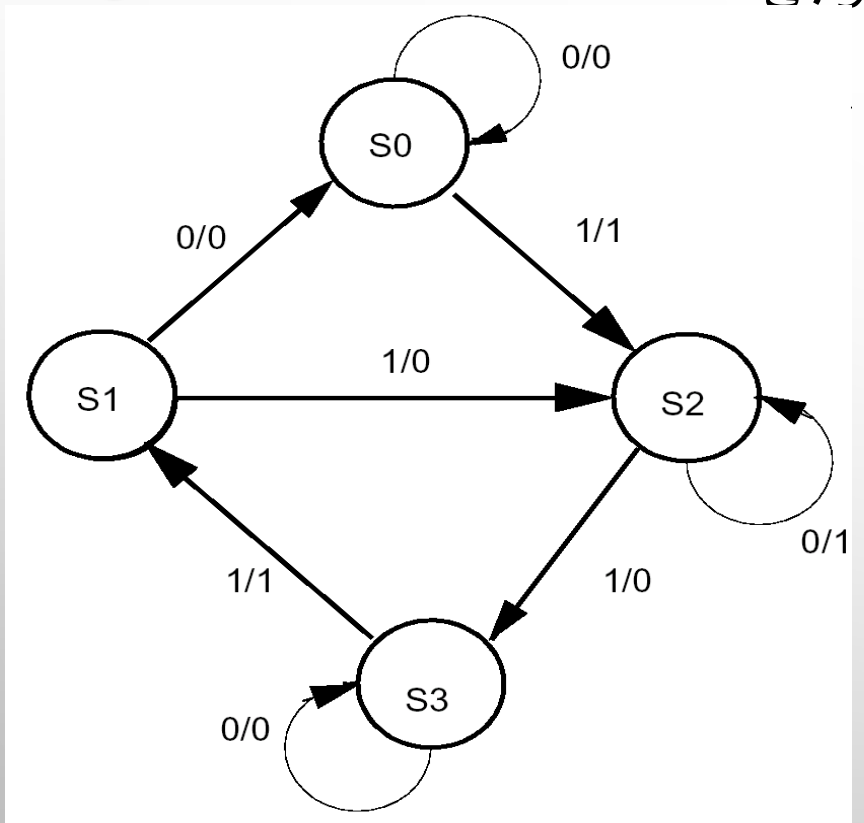
```
WHEN S6 =>                      --DETECT "1110101"
       OUTPUT<='0';
       IF (INPUT='1') THEN
       NEXT_STATE <=S7;
       ELSIF (INPUT='0') THEN
       NEXT_STATE <=S0;
       END IF;
WHEN S7 =>                      --DETECT "11101011"
       OUTPUT<='0';
       IF (INPUT='1') THEN
       NEXT_STATE <=S8;
       ELSIF (INPUT='0') THEN
       NEXT_STATE <=S0;
       END IF;
WHEN S8 =>                      --DETECT "111010110"
       OUTPUT<='0';
       IF (INPUT='0') THEN
       NEXT_STATE <=S9;
       ELSIF (INPUT='1') THEN
       NEXT_STATE <=S2;
       END IF;
```

مثال 5-25)
آشکارساز توالی بیت

```
        WHEN S9 =>                    --DETECT "1110101101"
              OUTPUT<='0';
              IF (INPUT='1') THEN
              NEXT_STATE <=S10;
              ELSIF (INPUT='1') THEN
              NEXT_STATE <=S0;
              END IF;
        WHEN S10 =>                   -- OUTPUT ASSERTED
              OUTPUT<='1';
              NEXT_STATE<=S0;
        END CASE;
        END PROCESS STATE_LOGIC;
END STAT;
```

مثال 5-26) ماشین میلی

مقدار خروجی آن وابسته به حالت فعلی و خروجی است



حالت فعلی	حالت بعدی		خروجی	
	X=0	X=1	X=0	X=1
S0	S0	S2	0	1
S1	S0	S2	0	0
S2	S2	S3	1	0
S3	S3	S1	0	1

مثال 5-26) ماشين ميلي

```
-------------------------------------------------------
ENTITY MEALY IS -- MEALY MACHINE
       PORT(X, CLOCK: IN BIT;
              Z: OUT BIT);
END MEALY;
-------------------------------------------------------
ARCHITECTURE BEHAVIOR OF MEALY IS
     TYPE STATE_TYPE IS (S0, S1, S2, S3);
     SIGNAL CURRENT_STATE, NEXT_STATE: STATE_TYPE;
BEGIN
-------------------------------------------------------
-- PROCESS TO HOLD COMBINATIONAL LOGIC.
  COMBIN: PROCESS(CURRENT_STATE, X)
  BEGIN
  CASE CURRENT_STATE IS
  WHEN S0 =>
       IF X = '0' THEN
          Z <= '0';
          NEXT_STATE <= S0;
       ELSE
          Z <= '1';
          NEXT_STATE <= S2;
       END IF;
  WHEN S1 =>
        IF X = '0' THEN
          Z <= '0';
          NEXT_STATE <= S0;
       ELSE
          Z <= '0';
          NEXT_STATE <= S2;
       END IF;
```

مثال 5-26) ماشين ميلي

```
  WHEN S2 =>
       IF X = '0' THEN
          Z <= '1';
          NEXT_STATE <= S2;
       ELSE
          Z <= '0';
          NEXT_STATE <= S3;
       END IF;
  WHEN S3 =>
       IF X = '0' THEN
          Z <= '0';
          NEXT_STATE <= S3;
       ELSE
          Z <= '1';
          NEXT_STATE <= S1;
       END IF;
END CASE;
END PROCESS COMBIN;
-------------------------------------------------------------
-- PROCESS TO HOLD SYNCHRONOUS ELEMENTS (FLIP-FLOPS)
SYNCH: PROCESS
BEGIN
  WAIT UNTIL CLOCK'EVENT AND CLOCK = '1';
  CURRENT_STATE <= NEXT_STATE;
END PROCESS SYNCH;
END BEHAVIOR;
-------------------------------------------------------------
```

RAM با ورودی و خروجی مجزا

```
-------------------------------------------------------------
-- Example(5-28): RAM with separated input output
-- No RD input (output allways enable)
-------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
-------------------------------------------------------------
ENTITY ram IS
  GENERIC ( bits: INTEGER := 8;     -- # of bits per word
      words: INTEGER := 16); -- # of words in the memory
    PORT ( wr_ena, clk: IN STD_LOGIC;
        addr: IN INTEGER RANGE 0 TO words-1;
        data_in: IN STD_LOGIC_VECTOR (bits-1 DOWNTO 0);
        data_out: OUT STD_LOGIC_VECTOR (bits-1 DOWNTO 0));
END ram;
-------------------------------------------------------------
ARCHITECTURE ram OF ram IS
    TYPE vector_array IS ARRAY (0 TO words-1)OF
          STD_LOGIC_VECTOR(bits-1 DOWNTO 0);
    signal memory: vector_array;
BEGIN
    PROCESS (clk, wr_ena) THEN
    BEGIN
          IF (wr_ena='1')
              IF (clk'EVENT AND clk='1') THEN
                  memory(addr) <= data_in;
              END IF;
          END IF;
    END PROCESS;
    data_out <= memory(addr);
END ram;
-------------------------------------------------------------
```

حافظه فقط خواندنی(ROM)

```
-- Example(5-31): ROM 8×8
library ieee;
use ieee.std_logic_1164.all;
ENTITY rom IS
    GENERIC ( bits: INTEGER := 8;   --# of bits per word
              words: INTEGER := 8);      --# of words in
the memory
    PORT (Addr   : IN INTEGER RANGE 0 TO words-1;
          Enable: in std_logic;
          Read   : in std_logic;
          Data   : OUT STD_LOGIC_VECTOR(bits-1 DOWNTO 0));
END rom;
ARCHITECTURE rom OF rom IS
    TYPE vector_array IS ARRAY (0 TO words-1) OF
         STD_LOGIC_VECTOR (bits-1 DOWNTO 0);
    ONSTANT memory: vector_array :=("10000000",
                                    "01000000",
                                    "00100000",
                                    "00010000",
                                    "00001000",
                                    "00000100",
                                    "00000010",
                                    "00000001");
BEGIN
    process( Enable, Read, Addr)
    begin
        if Enable = '1' then
            if( Read = '1' ) then
                data <= memory(addr);
            else Data <= "ZZZZZZZZ";
            end if;
        else Data <= "ZZZZZZZZ";
        end if;
    end process;
END rom;
```

پردازنده های خاص

در این بخش تفاوت بین مدل های ماشینهای حالت با مسیر داده توکار و مدل ماشین حالت با مسیر داده مجزا را نشان می دهد. از زاویه طراحی در سطح انتقال ثبات هر طرح دیجیتال شامل یک واحد کنترل(FSM)و یک مسیر داده است.

مسیر داده شامل واحد های حافظه مانند ثباتها و حافظه و واحدهای ترکیبی مانند: محاسبه و منطق-جمع کننده-ضرب کننده-ثبات انتقالی و مقایسه کننده می باشد.

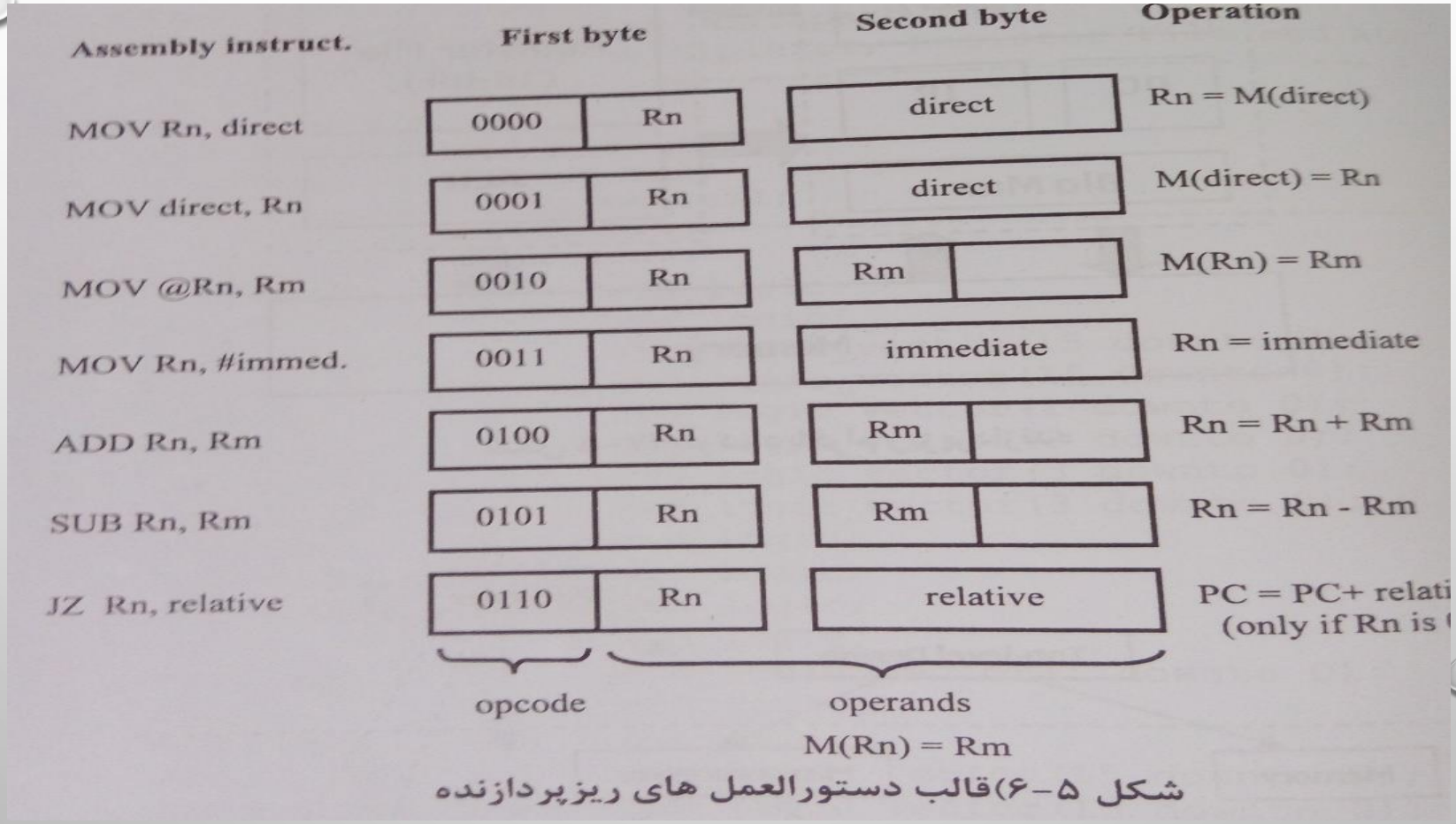
مسیر داده عملوند را از واحد حافظه بر می دارد و با استفاده از واحد های ترکیبی عملیات محاسباتی را انجام می دهد و حاصل را به حافظه باز می گرداند. این فرآیند یک یا دو کلاک طول می کشد.

محاسبه کننده ب.م.م ترکیبی

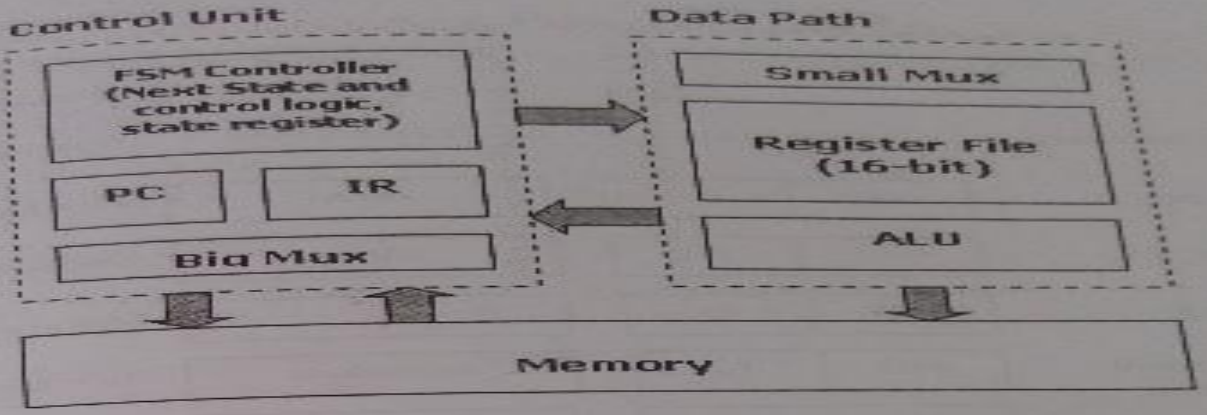
```
-- Example(5-34):Behvaior Design of GCD calculator
-- GCD algorithm behavior modeling(combinational)
-------------------------------------------------------

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use work.all;
```

```
entity gcd1 is
port(   Data_X:      in unsigned(3 downto 0);
        Data_Y:      in unsigned(3 downto 0);
        Data_out:    out unsigned(3 downto 0)
    );
end gcd1;
-------------------------------------------------------
architecture behv of gcd1 is
begin
    process(Data_X, Data_Y)
      variable tmp_X, tmp_Y: unsigned(3 downto 0);
    begin
      tmp_X := Data_X;
      tmp_Y := Data_Y;
      for i in 0 to 15 loop
          if (tmp_X/=tmp_Y) then
            if (tmp_X < tmp_Y) then
                        tmp_Y := tmp_Y - tmp_X;
            else
                tmp_X := tmp_X - tmp_Y;
            end if;
          else
            Data_out <= tmp_X;
          end if;
      end loop;
    end process;
end behv;
-------------------------------------------------------
```

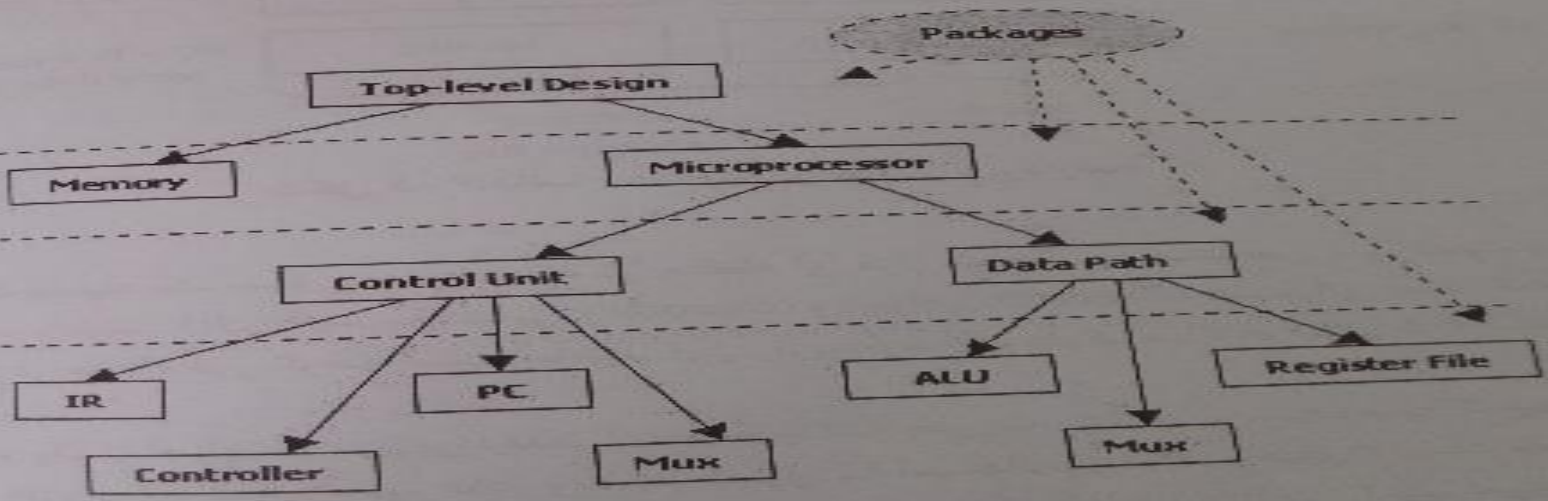
طراحی یک ریز پردازنده ساده



شکل ۵-۶)قالب دستورالعمل های ریزپردازنده



شکل ۵-۷) بلوک دیاگرام ریزپردازنده



شکل ۵-۸) نمودار سلسله مراتبی ریزپردازنده

انتخاب کننده

```
-------------------------------------------------------------
-- Simple Microprocessor Design
-- multiplexor of datapath unit (smallmux.vhd)
-- three 16 bit inputs and one 16 bit output
-------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
-------------------------------------------------------------
entity smallmux is
port(           I0:    in std_logic_vector(15 downto 0);
                I1:    in std_logic_vector(15 downto 0);
                I2:    in std_logic_vector(15 downto 0);
                Sel:   in std_logic_vector(1 downto 0);
                O:     out std_logic_vector(15 downto 0)
      );
end smallmux;
-------------------------------------------------------------
architecture behv of smallmux is
begin
      process(I0, I1, I2, Sel)
    begin
        case Sel is
             when "00" =>       O <= I0;
             when "01" =>       O <= I1;
             when "10" =>       O <= I2;
             when others =>
        end case;
    end process;
end behv;
-------------------------------------------------------------
```

بافر خروجی

```
-------------------------------------------------------------
-- Simple Microprocessor Design
-- Output buffer of Data Path (obuf.vhd)
-------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use work.constant_lib.all;
-------------------------------------------------------------
entity obuf is
port( O_en:        in std_logic;
      obuf_in:     in std_logic_vector(15 downto 0);
      obuf_out:    out std_logic_vector(15 downto 0)
      );
end obuf;
-------------------------------------------------------------
architecture behv of obuf is
begin
  process (O_en, obuf_in)
  begin
    if O_en = '1' then
      obuf_out <= obuf_in;
    else
      obuf_out <= HIRES;
    end if;
  end process;
end behv;
-------------------------------------------------------------
```

ثبات دستور العمل

```
-------------------------------------------------------------
-- Simple Microprocessor Design
-- Instruction Register (IR.vhd)
-------------------------------------------------------------
library      ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
-------------------------------------------------------------
entity IR is
port( IRin:          in std_logic_vector(15 downto 0);
      IRld:          in std_logic;
      dir_addr:      out std_logic_vector(15 downto 0);
      IRout:         out std_logic_vector(15 downto 0)
      );
end IR;
-------------------------------------------------------------
architecture behv of IR is
begin
  process(IRld, IRin)
  begin
    if IRld = '1' then
      IRout <= IRin;
      dir_addr <= "00000000" & IRin(7 downto 0);
    end if;
  end process;
end behv;
-------------------------------------------------------------
```

موفق و پیروز باشید

ز کوشش به هر چیز خواهی رسید

به هر چیز خواهی کماهی رسید

برو کارگر باش و امیدوار

که از یاس جز مرگ ناید به بار

گرت پایداری است در کارها

شود سهل پیش تو دشوار ها

پایان