ادامه‌ی درس

sequential coding

PROCESS

- with sensitivity list
  - Synthesizable
  - Run in zero nanoseconds
- without sensitivity list
  - Not Syntesizable
  - Runs in non-zero nanoseconds

یادآوری process بدون لیست حساسیت

ما در این حالت ۴ نوع wait داریم.

مثال:

نکته: هر جمله‌ای در VHDL، مستقل از concurrent یا sequential بودن، یک بار در زمان صفر simulation اجرا می‌شوند و در دفعات بعدی تنها در زمانی که ورودی آن تغییر کند اجرا می‌شود. البته process بدون لیست حساسیت حلقه‌ی بی‌نهایت است.

```
PROCESS BEGIN
    x<='0'
    WAIT FOR 10ns;
    x<='1';
    WAIT FOR 10ns;
END PROCESS;
```

خروجی مثال روبرو: x 

مثال:

مشابه مثال قبل ۱۰۰۰ بار همان کار را تکرار می‌کند اما بعد از آن برای همیشه متوقف می‌شود.

```
PROCESS BEGIN
    FOR i IN 1 TO 1000 LOOP
        // کد مثال قبل
    END LOOP;
    WAIT;
END PROCESS
```

مثال:

```
PROCESS BEGIN
    WAIT ON CLK;
    IF CIK='1' THEN
        q<=d;
    END IF;
END PROCESS;
```

شبیه سازی رفتار flip-flop است.

مثال:

```
PROCESS BEGIN
    WAIT UNTIL CLK='0';
    WAIT UNTIL CLK='1'; → باید یک event رخ دهد و شرط روبرو برقرار باشد
    q<=d;                تا کد به خط بعدی برود. به همین دلیل این خط کافی
END PROCESS;             است.
```

مشابه مثال قبل است.

کاربردهای ساختارهای غیرقابل سنتز:

- تست (توصیف ساختارهای پیچیده و active)
- prototyping (ساخت نمونه‌ی اولیه)

نکته در modelsim:

- برای ساخت کلاک می‌توان از دستور روبرو استفاده کرد که باید سیگنال مقدار اولیه داشته باشد:

```
x<=not x after 5ns;
```

- می‌توان در کدهای sequential از break point استفاده نمود و توسط View→local variable می‌توان متغیرهای local را دید.

ادامه‌ی درس

یادآوری:

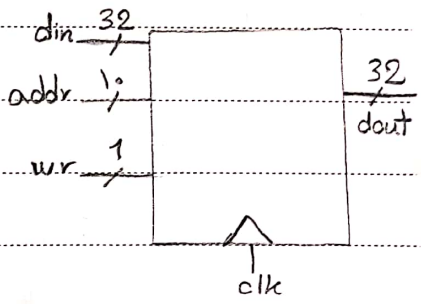
- var is defined inside a process while signal is define inside whole architecture
- var has no timing structure and its value is assigned immediately signal is valued at the end of process

برای یک signal یک signal driver داریم که variable ندارد.

x <= '0', '1' AFTER 10ns, '0' AFTER 20ns

x:	0ns	0	10ns	1	20ns	0

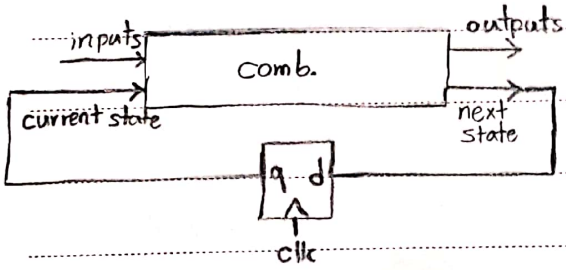
مثال: مدل کردن 1024x32 SRAM:



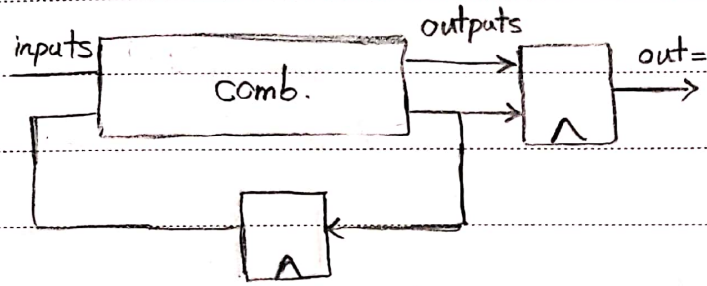

```
ENTITY sram1024x32 IS
    GENERIC (a : INTEGER := 10;
             n : INTEGER := 1024;
             w : INTEGER := 19);
    PORT(clk, wr : IN std_logic;
         din     : IN std_logic_vector(w-1 DOWNTO 0);
         addr    : IN std_logic_vector(a-1 DOWNTO 0);
         dout    : OUT std_logic_vector(w-1 DOWNTO 0));
END sram1024x32;
ARCHITECTURE test OF sram1024x32 IS
    TYPE mem_bank IS ARRAY (0 to n-1) OF std_logic_vector (w-1 DOWNTO 0);
    SIGNAL bank : mem_bank;
BEGIN
    PROCESS (clk)
    BEGIN
        IF clk='1' THEN
            IF wr='1' THEN
                bank(conv_integer(addr)) <= din;
            END IF;
        END IF;
    END PROCESS;
    dout <= bank(conv_integer(addr));
END test;
```

ماشین حالت

مدل رو برو، مدل Hoffman برای مدارهای seq است.

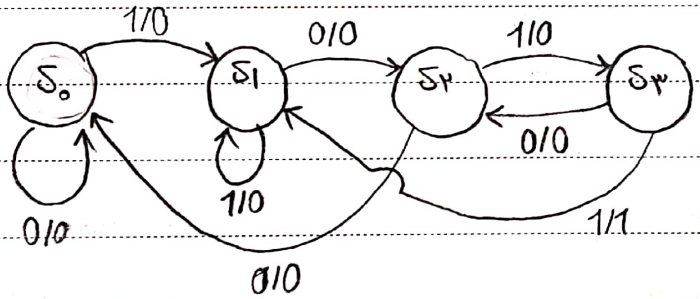


در مدارهای mealy، خروجی به صورت $out = f(input, current\_state)$ است ولی در مدارهای moore تنها تابعی از current_state است و به صورت زیر است:

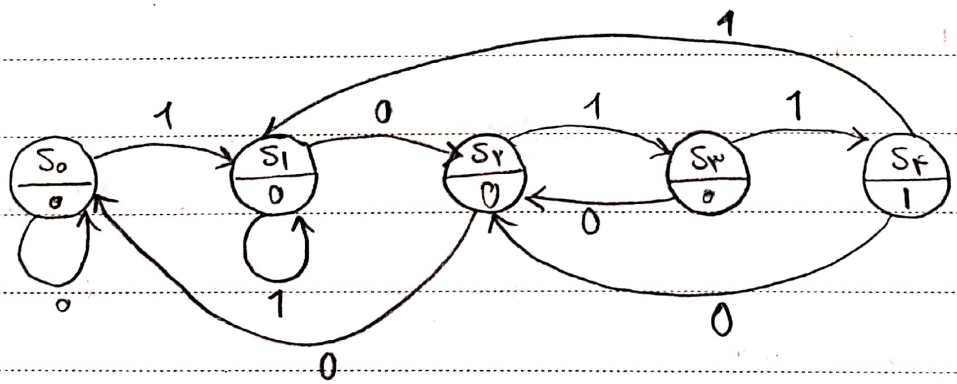


مثال: طراحی sequence detector: 1011

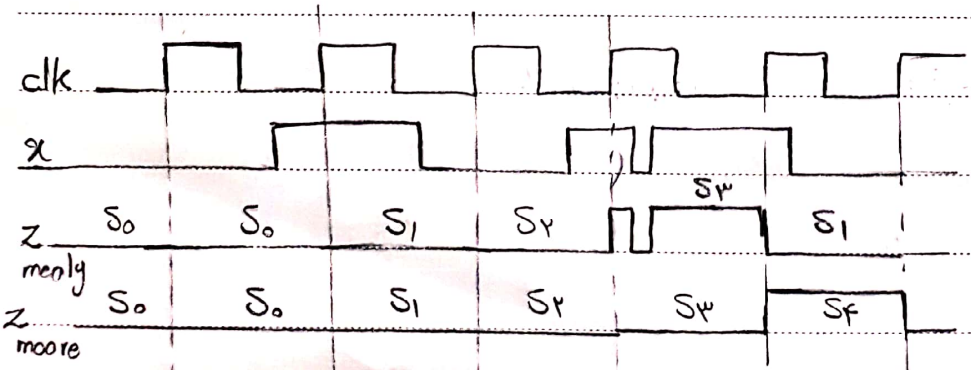
mealy:



moore



بررسی:



```
ENTITY detector_1011 IS
    PORT (clk, nrst, x : IN std_logic;
          z            : OUT std_logic);
END detector_1011;
ARCHITECTURE mealy OF detector_1101 IS
    TYPE state IS (S0, S1, S2, S3);
    SIGNAL cur_state, next_state : state;
BEGIN
    comb : PROCESS (x, cur_state) BEGIN
        IF cur_state = S0 THEN
            IF x = '1' THEN
                next_state <= S1;
            END IF;
            z <= '0';
        ELSIF cur_state = S1 THEN
            IF x = '0' THEN
                next_state <= S2;
            END IF;
            z <= '0';
        ELSIF cur_state = S2 THEN
            IF x = '1' THEN
                next_state <= S3;
            ELSE
                next_state <= S0;
            END IF;
            z <= '0';
        ELSE -- S3
            IF x = '0' THEN
                next_state <= S2;
                z <= '0';
            ELSE
                next_state <= S1;
                z <= '1'
            END IF;
        END IF
    END PROCESS comb;
    seq : PROCESS (clk)
    BEGIN
        IF clk = '1' THEN
            IF nrst = '0' THEN
                cur_state <= S0;
            ELSE
                cur_state <= next_state;
            END
        END IF;
    END PROCESS seq;
END mealy
```

مثال حلسة قبل

در صورت تبدیل آن به حالت moore، تعداد state ها ۴ تا می‌شود (S0...S4)

هم‌چنین z تنها به current state وابسته است و خارج از حالت‌بندی if برای x و درون آن است. برای مثال در حالت S4 جدید که به صورت زیر است:

```
...
ELSIF current_state = S4 THEN
    IF x='0' THEN
        next_state <= S0;
    ELSE
        next_state <= S1;
    ENDIF;
    z<='1'; -- current تنها وابسته به
ENDIF;
...
```

نکته:

which conditions are necessary for a combinational Process?

(a combinational process is a process that implements a combinational circuit)

- All inputs should be listed in sensitivity list
- All states of condition must be checked
- In all states, output must be set. i.e. in one execution of a process, all of the output must set.

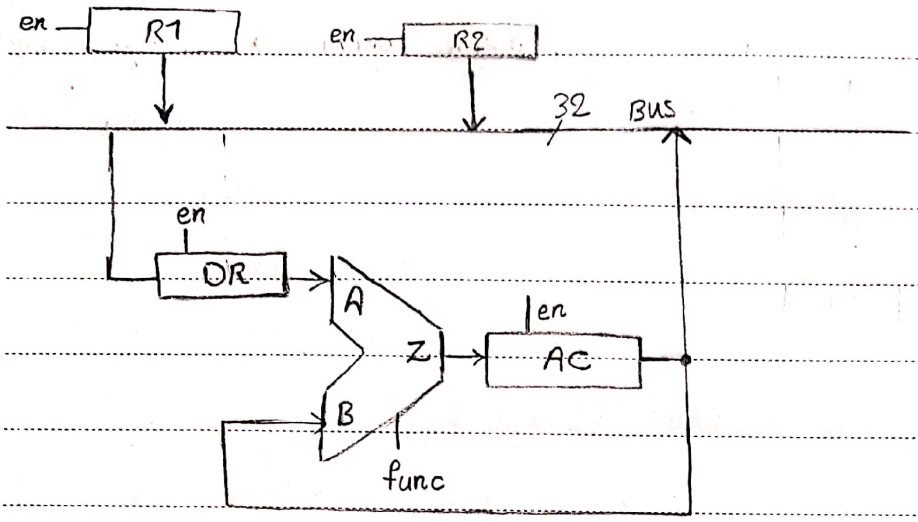
نکته:

- input of process: all the signal that their values are used inside process.
- output of process: all the signals that their values changed inside the process.

example: PROCESS (a,b,c,d, sel) BEGIN → it is no more combinational

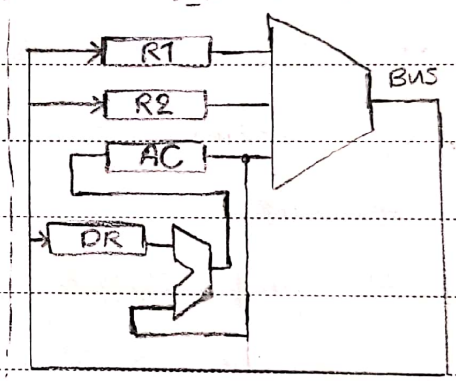
```
    IF sel = "00" THEN
        z<='a'; → اگر حذف شود در یک حالت مقدار متغیر می‌شود.
    EISIF
    ...
    EISE → EISIF sel='11' THEN X
        z<=d; ← حالت‌های 'U', 'Z', ... را داریم پس std_logic
    ENDIF; ترکیبی نیست.
END PROCESS;
```

ادامه‌ی درس



func	operation
000	Z=A+B
001	Z=A>>1
010	Z=A-B
011	Z=A
100	Z=B

مثال: واضحیت bus به صورت زیر است:



MO1 (micro operation 1): R1 → R2

cycle 1: bus.sel = 00, R2.en = 1

MO2: R1 → AC

cycle 1: R1 → DR (Bus.sel = 00, DR.en = 1)

cycle 2: DR → AC (func = 011, AC.en = 1)

MO3: R2 + R1 → R2

cycle 1: R1 → DR

cycle 2: DR → AC

cycle 3: R2 → DR

cycle 4: AC + DR → AC

cycle 5: AC → R2

تکلیف: دستورات ورودی خروجی و توسط fsm و datapath بالا اجرایی شوند.

نوشتن کدهای sequential:

sequential:

- PROCESS
- FUNCTION → همیشه یک خروجی دارد.
- PROCEDURE → می‌تواند خروجی نداشته باشد یا چند خروجی داشته باشد.

مثال: Full Adder

نکته: توابع بین architecure و begin تعریف می‌شوند.

ENTITY fa IS

PORT (a, b, cin : IN std_logic;

s, cout : OUT std_logic);

END fa;

ARCHITECTURE no1 OF fa IS

FUNCTION sum (a, b, cin : std_logic) RETURN std_logic IS

~~SIGNAL~~ VARIABLE x : std_logic; → می‌تواند variable باشد.

→ نمی‌توان signal تعریف کرد.

BEGIN

x ~~<=~~ := a XOR b XOR cin;

RETURN x;

END FUNCTION sum; ← در ۲۰۰۸ الزام است.

FUNCTION carry (SIGNAL a, b, cin : std_logic) RETURN std_logic IS

~~SIGNAL~~ VARIABLE x : std_logic;

BEGIN

x ~~<=~~ := (carry فرمول) AFTER 3 ns;

RETURN x;

END FUNCTION carry;

BEGIN

-- s <= sum (a, b, cin) AFTER 10ns;

-- cout <= carry (a, b, cin);

PROCESS (a, b, cin) BEGIN

s <= sum (a, b, cin) AFTER 10ns;

cout <= carry (a, b, cin);

END PROCESS;

END no1;

مثال مثل با procedurce :

```
PROCEDURE adding(a,b,cin : IN std_logic; s,cout : OUT std_logic) IS
BEGIN
        s<= ...;
        cout <= ...;
END PROCEDURE;
```

PACKAGE مثال :

```
file : fa_basic_utils IS
    LIBRARY ieee;
    USE ieee.std_logic_1164.ALL;
    PACKAGE fa_basic_utils IS
            CONSTANT n : integer := 32;
            FUNCTION sum (SIGNAL a,b,cin : std_logic) RETURN std_logic;
            FUNCTION carry (SIGNAL a,b,cin : std_logic) RETURN std_logic;
            COMPONENT fa2 IS
                    PORT (a,b,cin : IN std_logic;
                          z,cout : OUT std_logic);
            END COMPONENT;
    END PACKAGE fa_basic_utils;
    PACKAGE BODY fa_basic_utils IS
            FUNCTION sum ...
            FUNCTION carry ...
    END PACKAGE BODY fa_basic_utils;
```

} مثال قبل

```
file : fa2.vhd
    LIBRARY ieee;
    USE ieee.std_logic_1164.ALL;
    USE work.fa_basic_utils.ALL;
```

```
ENTITY fa2 IS
        PORT (a, b, cin : IN std_logic;
              z, cout   : OUT std_logic);
END fa2;
ARCHITECTURE no1 of fa2 IS BEGIN
        z <= sum (a,b,cin) AFTER 10ns;
        cout <= carry (a,b, cin) AFTER 15 ns;
END no1;
```

file: fa2_tb.vhd

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE work.fa_basic_utils.ALL;
ENTITY fa2_tb IS END fa2_tb;
ARCHITECTURE no1 of fa2_tb IS
        SIGNAL a_t, b_t, ci_t, s_t, co_t : std_logic;
BEGIN
        u1: fa2 PORT MAP (a_t, b_t, ci_t, s_t, co_t);
        PROCESS BEGIN
            a_t <= '0'; b_t <= '0'; ci_t <= '0';
            WAIT FOR 50ns;
            a_t <= '1';
            WAIT FOR 50ns;
            b_t <= '1';
            WAIT FOR 50ns;
            ci_t <= '1';
            WAIT FOR 50ns;
            a_t <= '0'; b_t <= '0'; ci_t <= '0';
            WAIT;
        END PROCESS;
END no1;
```