دوره مقدماتی آشنایی با FPGA و VHDL

کد ترتیبی

محمدرضا عزیزی

امیرعلی ابراهیمی

بهار ۱۳۹۷

مقدمه (۱)

- همان‌طور که در فصل ۵ اشاره شد، VHDL ذاتا همروند است.
- تنها در بلوک‌های PROCESSES، FUNCTIONS و PROCEDURES می‌توان کد ترتیبی نوشت.
  - به صورت کلی، هر یک از این بلوک‌ها با سایر بخش‌های کد، همروند اند.
- کد ترتیبی به *مدارهای ترتیبی* محدود نمی‌شود و با استفاده از آن می‌توان *مدارهای ترکیبی* را نیز پیاده‌سازی کرد.
- دستورات قابل استفاده در کد ترتیبی شامل موارد زیر هستند:
  - IF
  - CASE
  - LOOP
  - WAIT

مقدمه (۲)

- VARIABLE:
  - تنها داخل کد ترتیبی استفاده می‌شوند.
  - برخلاف SIGNAL نمی‌توانند به صورت سراسری تعریف شود.
  - مقدار آن نمی‌تواند به صورت مستقیم به خارج از بلوک‌های کد ترتیبی، ارسال شود.

- در این فصل بر روی PROCESS تمرکز می‌کنیم.
  - FUNCTION و PROCEDURE مشابه با PROCESS هستند و در بخش دوم درس (طراحی سیستم) بررسی خواهند شد.

(۱) PROCESS

- یک بلوک(بخش) ترتیبی در کد VHDL.
- ساختار:

```
[label:] PROCESS (sensitivity list)
  [VARIABLE name type [range] [:= initial_value;]]
BEGIN
  (sequential code)
END PROCESS [label];
```

- هرگاه یکی از سیگنال‌های موجود در لیست حساسیت(sensitivity list) تغییر کند، PROCESS اجرا می‌شود.
  - در صورت استفاده از WAIT، نیازی به لیست حساسیت نیست و در این صورت، هرگاه شرط WAIT برآورده شود، PROCESS اجرا می‌شود.
  - مقداردهی اولیه سنتزپذیر نیست و فقط برای شبیه‌سازی استفاده می‌شود.

SIGNAL در مقابل VARIABLE

Table 7.1
Comparison between SIGNAL and VARIABLE.

	SIGNAL	VARIABLE
Assignment	<=	:=
Utility	Represents circuit interconnects (wires)	Represents local information
Scope	Can be global (seen by entire code)	Local (visible only inside the corresponding PROCESS, FUNCTION, or PROCEDURE)
Behavior	Update is not immediate in sequential code (new value generally only available at the conclusion of the PROCESS, FUNCTION, or PROCEDURE)	Updated immediately (new value can be used in the next line of code)
Usage	In a PACKAGE, ENTITY, or ARCHITECTURE. In an ENTITY, all PORTS are SIGNALS by default	Only in sequential code, that is, in a PROCESS, FUNCTION, or PROCEDURE

(۱) IF

- یکی از دستورات قابل استفاده در کد ترتیبی.
- تنها در بلوک‌های PROCESSES، FUNCTIONS و PROCEDURES استفاده می‌شود.

- ساختار:

```
IF conditions THEN assignments;
ELSIF conditions THEN assignments;
...
ELSE assignments;
END IF;
```

(۲) IF

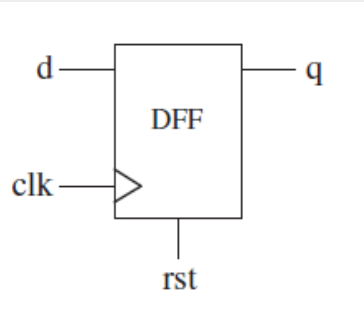
- مثال:

```
IF (x<y) THEN
      temp:="11111111";
ELSIF (x=y AND w='0') THEN
      temp:="11110000";
ELSE
      temp:=(OTHERS =>'0');
```

(۳) IF

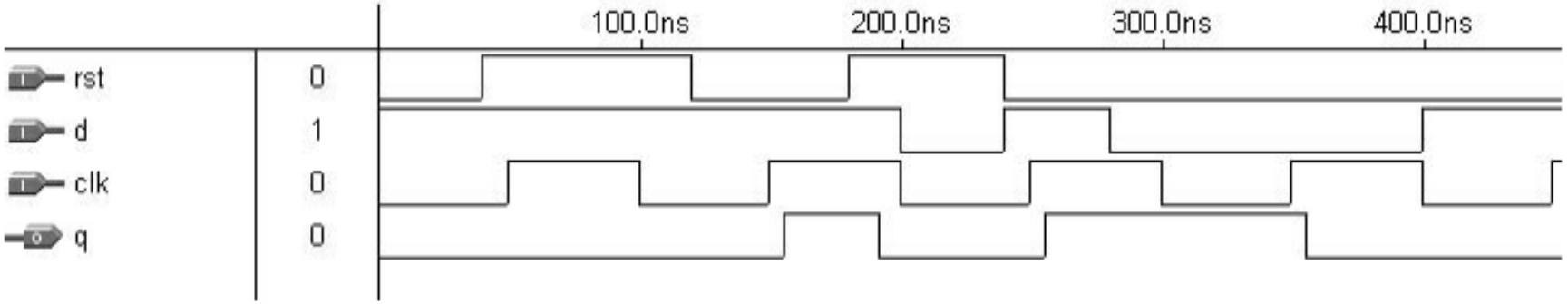
➢ **مثال:** فلیپ‌فلاپ D با ریست آسنکرون #۱

```
1 ------------------------------------
2 LIBRARY ieee;
3 USE ieee.std_logic_1164.all;
4 ------------------------------------
5 ENTITY dff IS
6      PORT (d, clk, rst: IN STD_LOGIC;
7            q: OUT STD_LOGIC);
8 END dff;
9 ------------------------------------
10 ARCHITECTURE behavior OF dff IS
11 BEGIN
12      PROCESS (clk, rst)
13      BEGIN
14           IF (rst='1') THEN
15                q <= '0';
16           ELSIF (clk'EVENT AND clk='1') THEN
17                q <= d;
18           END IF;
19      END PROCESS;
20 END behavior;
21 ------------------------------------
```



(۴) IF

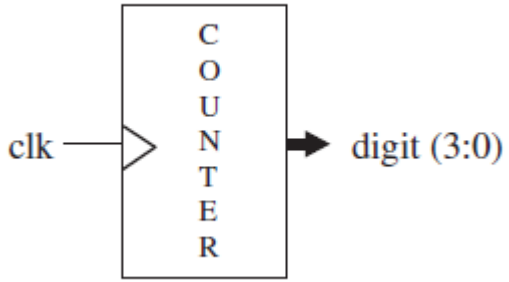
- **مثال:** فلیپ‌فلاپ D با ریست آسنکرون #۱



(۵) IF

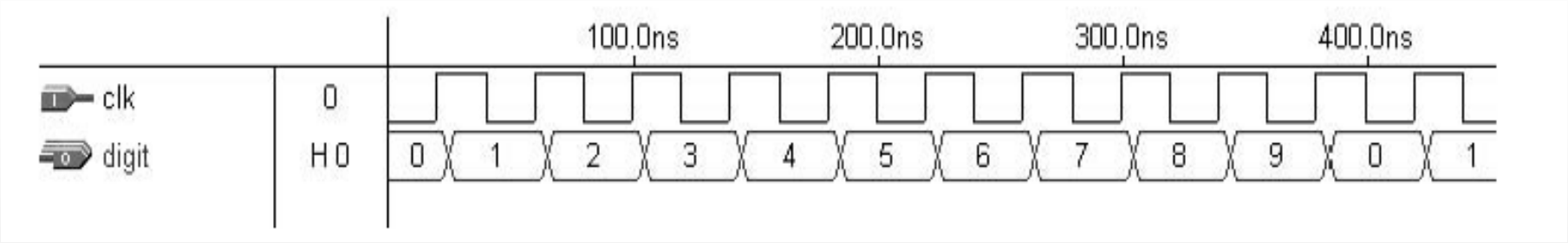
➢ **مثال:** شمارنده ۱ رقمی #۱ (۰ – ۹ – ۰)

```
1 ------------------------------------------
2 LIBRARY ieee;
3 USE ieee.std_logic_1164.all;
4 ------------------------------------------
5 ENTITY counter IS
6      PORT (clk : IN STD_LOGIC;
7            digit : OUT INTEGER RANGE 0 TO 9);
8 END counter;
9 ------------------------------------------
10 ARCHITECTURE counter OF counter IS
11 BEGIN
12      count: PROCESS(clk)
13           VARIABLE temp : INTEGER RANGE 0 TO 10;
14      BEGIN
15           IF (clk'EVENT AND clk='1') THEN
16                temp := temp + 1;
17                IF (temp=10) THEN temp := 0;
18                END IF;
19           END IF;
20           digit <= temp;
21      END PROCESS count;
22 END counter;
23 ------------------------------------------
```



(۶) IF

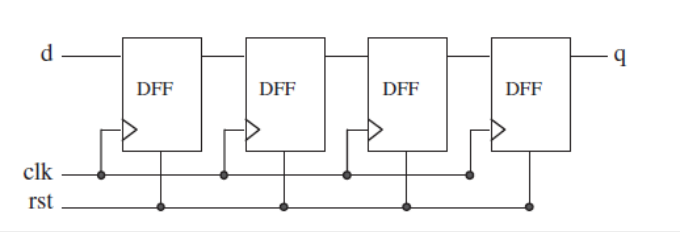
- **مثال:** شمارنده ۱ رقمی #۱ (۰ – ۹ – ۰)



(۷) IF

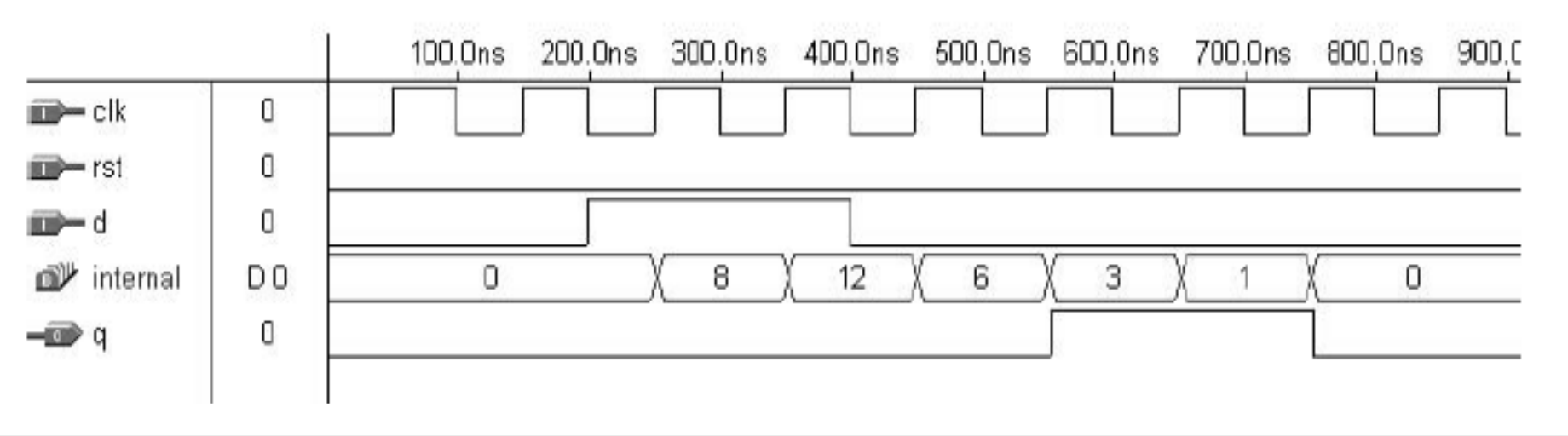
➢ **مثال:** شیفت رجیستر (shift register):

```
1 ------------------------------------------------
2 LIBRARY ieee;
3 USE ieee.std_logic_1164.all;
4 ------------------------------------------------
5 ENTITY shiftreg IS
6      GENERIC (n: INTEGER := 4);          -- # of stages
7      PORT (d, clk, rst: IN STD_LOGIC;
8            q: OUT STD_LOGIC);
9 END shiftreg;
10 ------------------------------------------------
11 ARCHITECTURE behavior OF shiftreg IS
12      SIGNAL internal: STD_LOGIC_VECTOR (n-1 DOWNTO 0);
13 BEGIN
14      PROCESS (clk, rst)
15      BEGIN
16           IF (rst='1') THEN
17                internal <= (OTHERS => '0');
18           ELSIF (clk'EVENT AND clk='1') THEN
19                internal <= d & internal(internal'LEFT DOWNTO 1);
20           END IF;
21      END PROCESS;
22      q <= internal(0);
23 END behavior;
24 ------------------------------------------------
```



(۸) IF

- **مثال:** شیفت رجیستر (shift register):



(۱) WAIT

- عملکرد WAIT گاهی اوقات شبیه IF است.
- هنگامی که از WAIT استفاده می‌شود، نمی‌توانیم لیست حساسیت داشته باشیم.
- ساختار:

```
WAIT UNTIL signal_condition;
```

```
WAIT ON signal1 [, signal2, ... ];
```

```
WAIT FOR time;
```

(۲) WAIT

- **WAIT UNTIL** تنها یک سیگنال قبول می‌کند.
  - بنابراین برای پیاده‌سازی کدهای سنکرون مناسب‌تر است.
- از آن‌جایی که PROCESS در این حالت، لیست حساسیت ندارد، WAIT UNTIL باید به عنوان اولین دستور نوشته شود.
  - PROCESS هر گاه که شرط WAIT UNTIL برقرار شود، اجرا می‌شود.
- **مثال:** ثبات ۸ بیتی با ریست سنکرون:

```
PROCESS       -- no sensitivity list
BEGIN
     WAIT UNTIL (clk'EVENT AND clk='1');
     IF (rst='1') THEN
          output <= "00000000";
     ELSIF (clk'EVENT AND clk='1') THEN
          output <= input;
     END IF;
END PROCESS;
```

(۳) WAIT

- **WAIT ON** چندین سیگنال قبول می‌کند.
- PROCESS تا زمانی که تغییری بر روی یکی از این سیگنال ها رخ ندهد، اجرا نمی‌شود.
- **مثال:** ثبات ۸ بیتی با ریست آسنکرون:

```
PROCESS
BEGIN
     WAIT ON clk, rst;
     IF (rst='1') THEN
          output <= "00000000";
     ELSIF (clk'EVENT AND clk='1') THEN
          output <= input;
     END IF;
END PROCESS;
```

(۴) WAIT

- WAIT FOR تنها برای شبیه‌سازی استفاده می‌شود.
  - ایجاد waveform برای تست‌بنچ ها.

- **مثال:**

```
WAIT FOR 5ns;
```

(۵) WAIT

➢ **مثال:** فلیپ‌فلاپ D با ریست آسنکرون #۲

```
1 ---------------------------------------
2 LIBRARY ieee;
3 USE ieee.std_logic_1164.all;
4 ---------------------------------------
5 ENTITY dff IS
6      PORT (d, clk, rst: IN STD_LOGIC;
7            q: OUT STD_LOGIC);
8 END dff;
9 ---------------------------------------
10 ARCHITECTURE dff OF dff IS
11 BEGIN
12      PROCESS
13      BEGIN
14           WAIT ON rst, clk;
15           IF (rst='1') THEN
16                q <= '0';
17           ELSIF (clk'EVENT AND clk='1') THEN
18                q <= d;
19           END IF;
20      END PROCESS;
21 END dff;
22 ---------------------------------------
```

(۶) WAIT

➢ **مثال:** شمارنده ۱ رقمی ۲# (۰ – ۹ – ۰)

```
1 ------------------------------------------
2 LIBRARY ieee;
3 USE ieee.std_logic_1164.all;
4 ------------------------------------------
5 ENTITY counter IS
6      PORT (clk : IN STD_LOGIC;
7            digit : OUT INTEGER RANGE 0 TO 9);
8 END counter;
9 ------------------------------------------
10 ARCHITECTURE counter OF counter IS
11 BEGIN
12      PROCESS                -- no sensitivity list
13           VARIABLE temp : INTEGER RANGE 0 TO 10;
14      BEGIN
15           WAIT UNTIL (clk'EVENT AND clk='1');
16           temp := temp + 1;
17           IF (temp=10) THEN temp := 0;
18           END IF;
19           digit <= temp;
20      END PROCESS;
21 END counter;
22 ------------------------------------------
```

(۱) CASE

- ساختار:

```
CASE identifier IS
    WHEN value => assignments;
    WHEN value => assignments;
    ...
END CASE;
```

- مثال:

```
CASE control IS
     WHEN "00" => x<=a; y<=b;
     WHEN "01" => x<=b; y<=c;
     WHEN OTHERS => x<="0000"; y<="ZZZZ";
END CASE;
```

(۲) CASE

- مقدار WHEN می‌تواند یکی از حالت‌های زیر باشد:

```
WHEN value                      -- single value
WHEN value1 to value2           -- range, for enumerated data types
                                -- only
WHEN value1 | value2 |...       -- value1 or value2 or ...
```

(۳) CASE

- CASE در مقابل WHEN:

Table 6.1
Comparison between WHEN and CASE.

	WHEN	CASE
Statement type	Concurrent	Sequential
Usage	Only outside PROCESSES, FUNCTIONS, or PROCEDURES	Only inside PROCESSES, FUNCTIONS, or PROCEDURES
All permutations must be tested	Yes for WITH/SELECT/WHEN	Yes
Max. # of assignments per test	1	Any
No-action keyword	UNAFFECTED	NULL

(۴) CASE

- CASE در مقابل WHEN:

```
---- With WHEN: ----------------
WITH sel SELECT
     x <= a WHEN "000",
          b WHEN "001",
          c WHEN "010",
          UNAFFECTED WHEN OTHERS;
---- With CASE: ----------------
CASE sel IS
     WHEN "000" => x<=a;
     WHEN "001" => x<=b;
     WHEN "010" => x<=c;
     WHEN OTHERS => NULL;
END CASE;
--------------------------------
```

(۵) CASE

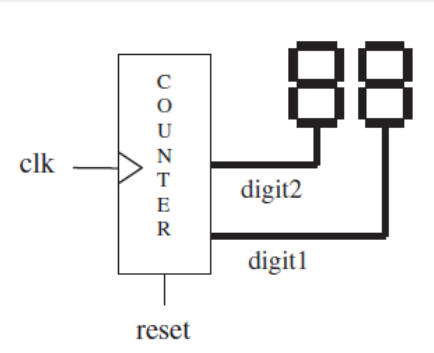
➢ **مثال:** فلیپ‌فلاپ D با ریست آسنکرون #۳

```
1 ------------------------------------------------
2 LIBRARY ieee;                                    -- Unnecessary declaration,
3                                                  -- because
4 USE ieee.std_logic_1164.all;                     -- BIT was used instead of
5                                                  -- STD_LOGIC
6 ------------------------------------------------
7 ENTITY dff IS
8      PORT (d, clk, rst: IN BIT;
9            q: OUT BIT);
10 END dff;
11 ------------------------------------------------
12 ARCHITECTURE dff3 OF dff IS
13 BEGIN
14      PROCESS (clk, rst)
15      BEGIN
16           CASE rst IS
17                WHEN '1' => q<='0';
18                WHEN '0' =>
19                     IF (clk'EVENT AND clk='1') THEN
20                          q <= d;
21                     END IF;
22                WHEN OTHERS => NULL;          -- Unnecessary, rst is of type
23                                               -- BIT
24           END CASE;
25      END PROCESS;
26 END dff3;
27 ------------------------------------------------
```

(۶) CASE

➢ **مثال:** شمارنده دو بیتی با خروجی بر روی نمایشگر ۷ قسمتی (7 segment)

```
1 ------------------------------------------------
2 LIBRARY ieee;
3 USE ieee.std_logic_1164.all;
4 ------------------------------------------------
5 ENTITY counter IS
6      PORT (clk, reset : IN STD_LOGIC;
7            digit1, digit2 : OUT STD_LOGIC_VECTOR (6 DOWNTO 0));
8 END counter;
9 ------------------------------------------------
10 ARCHITECTURE counter OF counter IS
11 BEGIN
12      PROCESS(clk, reset)
13           VARIABLE temp1: INTEGER RANGE 0 TO 10;
14           VARIABLE temp2: INTEGER RANGE 0 TO 10;
15      BEGIN
```

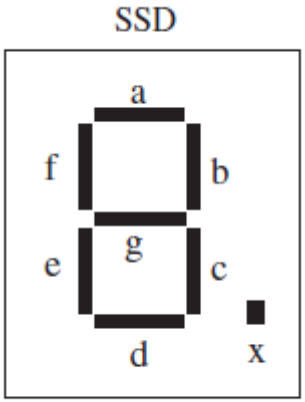


(V) CASE

```
16      ---- counter: ----------------------
17           IF (reset='1') THEN
18                temp1 := 0;
19                temp2 := 0;
20           ELSIF (clk'EVENT AND clk='1') THEN
21                temp1 := temp1 + 1;
22                IF (temp1=10) THEN
23                     temp1 := 0;
24                     temp2 := temp2 + 1;
25                     IF (temp2=10) THEN
26                          temp2 := 0;
27                     END IF;
28                END IF;
29           END IF;
```

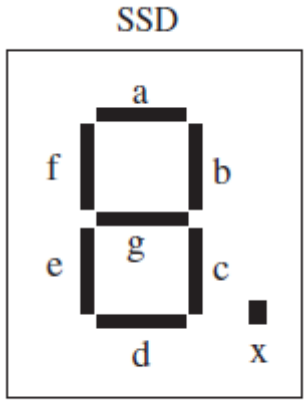
(۸) CASE

```
30          ---- BCD to SSD conversion: --------
31          CASE temp1 IS
32               WHEN 0 => digit1 <= "1111110"; --7E
33               WHEN 1 => digit1 <= "0110000"; --30
34               WHEN 2 => digit1 <= "1101101"; --6D
35               WHEN 3 => digit1 <= "1111001"; --79
36               WHEN 4 => digit1 <= "0110011"; --33
37               WHEN 5 => digit1 <= "1011011"; --5B
38               WHEN 6 => digit1 <= "1011111"; --5F
39               WHEN 7 => digit1 <= "1110000"; --70
40               WHEN 8 => digit1 <= "1111111"; --7F
41               WHEN 9 => digit1 <= "1111011"; --7B
42               WHEN OTHERS => NULL;
43          END CASE;
```



(۹) CASE

```
44            CASE temp2 IS
45                 WHEN 0 => digit2 <= "1111110"; --7E
46                 WHEN 1 => digit2 <= "0110000"; --30
47                 WHEN 2 => digit2 <= "1101101"; --6D
48                 WHEN 3 => digit2 <= "1111001"; --79
49                 WHEN 4 => digit2 <= "0110011"; --33
50                 WHEN 5 => digit2 <= "1011011"; --5B
51                 WHEN 6 => digit2 <= "1011111"; --5F
52                 WHEN 7 => digit2 <= "1110000"; --70
53                 WHEN 8 => digit2 <= "1111111"; --7F
54                 WHEN 9 => digit2 <= "1111011"; --7B
55                 WHEN OTHERS => NULL;
56            END CASE;
57      END PROCESS;
58 END counter;
59 -------------------------------------------------
```



Input: “xabcdefg”

(۱) LOOP

- حلقه – هنگامی که می‌خواهیم یک بخش کد را چند بار اجرا کنیم.
- ساختار FOR/LOOP:

```
[label:] FOR identifier IN range LOOP
  (sequential statements)
END LOOP [label];
```

  - تکرار به تعداد مشخص

- ساختار WHILE/LOOP:

```
[label:] WHILE condition LOOP
  (sequential statements)
END LOOP [label];
```

  - تکرار تا زمانی که (WHILE) شرایط برقرار باشند.

(۲) LOOP

- EXIT:

```
[label:] EXIT [label] [WHEN condition];
```

  - جهت خروج از حلقه

- NEXT:

```
[label:] NEXT [loop_label] [WHEN condition];
```

  - جهت از قلم انداختن (skip) مرحله ای از حلقه

(۳) LOOP

- مثال FOR/LOOP:

```
FOR i IN 0 TO 5 LOOP          -- repeates 6 times
     x(i) <= enable AND w(i+2);
     y(0, i) <= w(i);
END LOOP;
```

- برای این که FOR/LOOP به صورت عمومی سنتزپذیر باشد، باید هر دو حد range آن، استاتیک باشد.
  - به طور مثال، "FOR i IN 0 TO choice LOOP" که در آن choice یک ورودی غیر استاتیک است، به طور عمومی سنتزپذیر نیست.

(۴) LOOP

➢ مثال WHILE/LOOP:

```
WHILE (i < 10) LOOP    -- repeates until i >= 10
     WAIT UNTIL clk'EVENT AND clk='1';
     (other statements)
END LOOP;
```

➢ مثال EXIT:

```
FOR i IN data'RANGE LOOP
     CASE data(i) IS
          WHEN '0' => count:=count+1;
          WHEN OTHERS => EXIT;
     END CASE;
END LOOP;
```

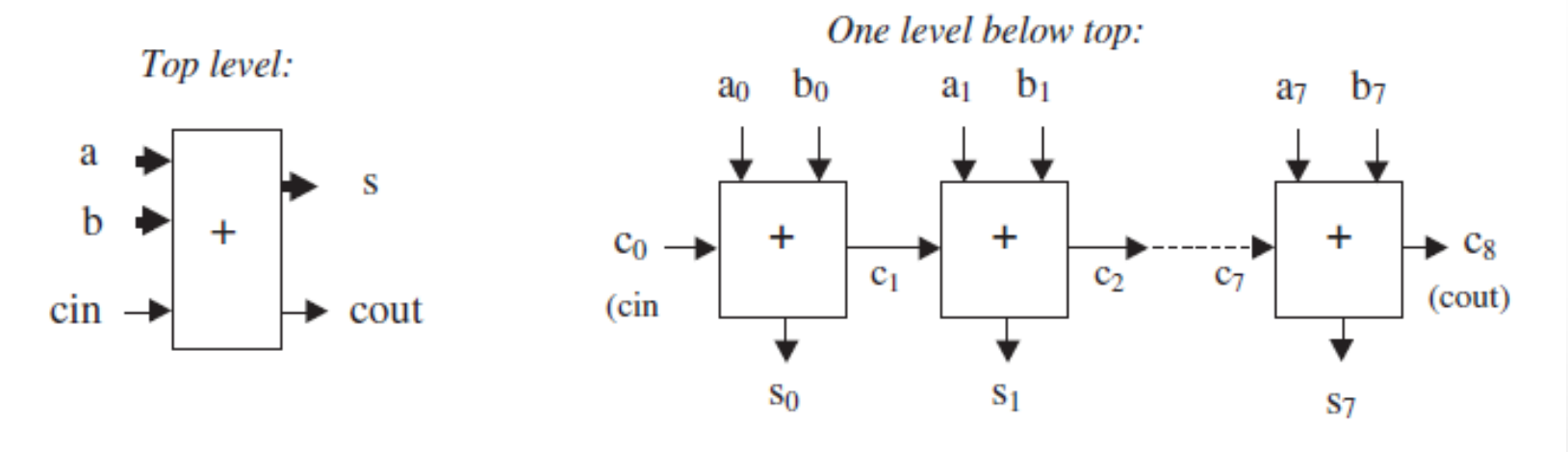
(۵) LOOP

- مثال NEXT:

```
FOR i IN 0 TO 15 LOOP
     NEXT WHEN i=skip;        -- jumps to next iteration
     (...)
END LOOP;
```

(۶) LOOP

➢ **مثال:** Carry Ripple Adder:

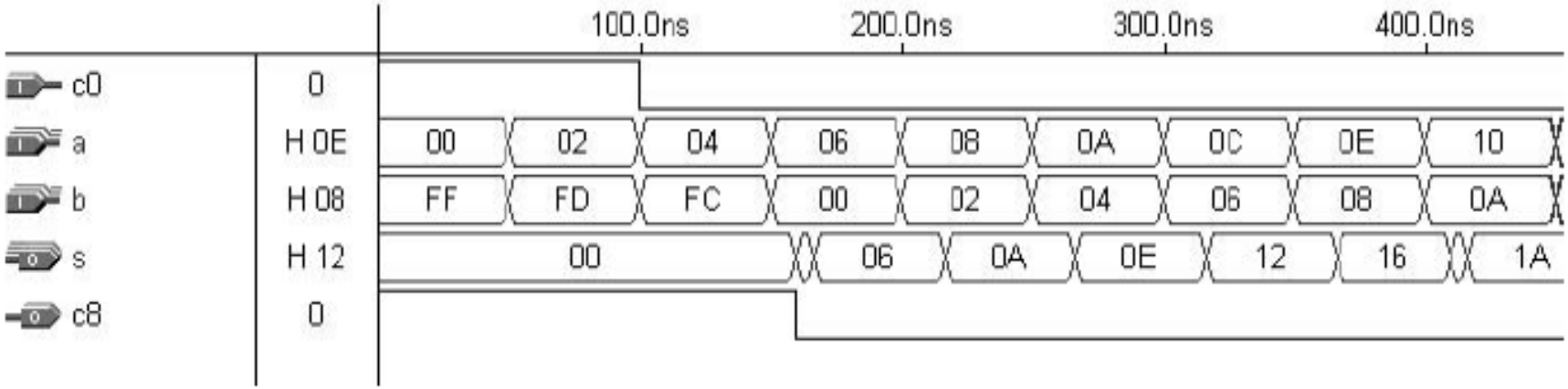


- sj = a_j XOR b_j XOR c_j
- cj1 = (a_j AND b_j) OR (a_j AND c_j) OR (b_j AND c_j)

(۷) LOOP

```
1 ----- Solution 1: Generic, with VECTORS --------
2 LIBRARY ieee;
3 USE ieee.std_logic_1164.all;
4 ---------------------------------------------
5 ENTITY adder IS
6      GENERIC (length : INTEGER := 8);
7      PORT ( a, b: IN STD_LOGIC_VECTOR (length-1 DOWNTO 0);
8             cin: IN STD_LOGIC;
9             s: OUT STD_LOGIC_VECTOR (length-1 DOWNTO 0);
10            cout: OUT STD_LOGIC);
11 END adder;
12 ---------------------------------------------
13 ARCHITECTURE adder OF adder IS
14 BEGIN
15      PROCESS (a, b, cin)
16           VARIABLE carry : STD_LOGIC_VECTOR (length DOWNTO 0);
17      BEGIN
18           carry(0) := cin;
19           FOR i IN 0 TO length-1 LOOP
20                s(i) <= a(i) XOR b(i) XOR carry(i);
21                carry(i+1) := (a(i) AND b(i)) OR (a(i) AND
22                              carry(i)) OR (b(i) AND carry(i));
23           END LOOP;
24           cout <= carry(length);
25      END PROCESS;
26 END adder;
27 ---------------------------------------------
```

(۸) LOOP


100.0ns
200.0ns
300.0ns
400.0ns
c0
0
a
H 0E
00
02
04
06
08
0A
0C
0E
10
b
H 08
FF
FD
FC
00
02
04
06
08
0A
s
H 12
00
06
0A
0E
12
16
1A
c8
0

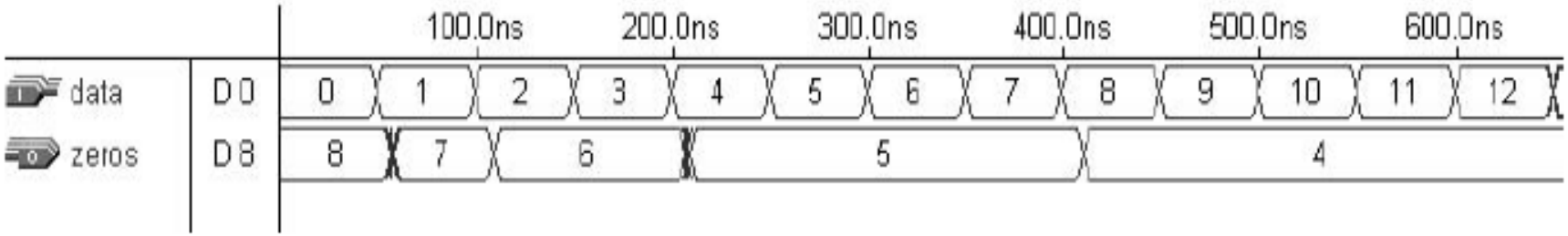
(۹) LOOP

مثال: Leading Zeros:

- شمارش تعداد ۰ های سمت چپ یک وکتور

```
1 ------------------------------------------
2 LIBRARY ieee;
3 USE ieee.std_logic_1164.all;
4 ------------------------------------------
5 ENTITY LeadingZeros IS
6      PORT ( data: IN STD_LOGIC_VECTOR (7 DOWNTO 0);
7             zeros: OUT INTEGER RANGE 0 TO 8);
8 END LeadingZeros;
9 ------------------------------------------
10 ARCHITECTURE behavior OF LeadingZeros IS
11 BEGIN
12      PROCESS (data)
13           VARIABLE count: INTEGER RANGE 0 TO 8;
14      BEGIN
15           count := 0;
16           FOR i IN data'RANGE LOOP
17                CASE data(i) IS
18                     WHEN '0' => count := count + 1;
19                     WHEN OTHERS => EXIT;
20                END CASE;
21           END LOOP;
22           zeros <= count;
23      END PROCESS;
24 END behavior;
25 ------------------------------------------
```

(۱۰) LOOP


100.0ns
200.0ns
300.0ns
400.0ns
500.0ns
600.0ns
data
D 0
0
1
2
3
4
5
6
7
8
9
10
11
12
zeros
D 8
8
7
6
5
4

نکاتی در مورد پالس ساعت (۱)

- کامپایلر معمولا کدهایی که در آن انتساب در دو لبه کلاک صورت گرفته باشد را نمی‌تواند کامپایل کند؛ به خصوص در مواقعی که سخت افزار هدف دارای فلیپ‌فلاپ‌هایی باشد که تنها با یک لبه کلاک می‌توانند کار کنند. مانند CPLDها
  - خطای ‘signal does not hold value after clock edge’ یا مشابه.
- حالتی را در نظر بگیریم که یک شمارنده باید در هر لبه پالس ساعت، مقدارش یک واحد افزایش یابد:

```
PROCESS (clk)
BEGIN
     IF(clk'EVENT AND clk='1') THEN
          counter <= counter + 1;
     ELSIF(clk'EVENT AND clk='0') THEN
          counter <= counter + 1;
     END IF;
     ...
END PROCESS;
```

  - در این حالت، علاوه بر خطای قبلی، ممکن است کامپایلر اعلام کند که counter چندین بار درایو شده است و کامپایل متوقف شود.

نکاتی در مورد پالس ساعت (۲)

- صفت EVENT باید همراه با یک شرط به کار رود.
  - `IF(clk'EVENT AND clk='1')` صحیح است.
  - در صورتی که نوشته شود `IF(clk'EVENT)`، یا کامپایلر شرط یک بودن کلاک را اضافه می‌کند و یا خطای زیر را اعلام می‌کند:
    - "clock not locally stable"

```
PROCESS (clk)
BEGIN
     IF(clk'EVENT) THEN
          counter := counter + 1;
     END IF;
...
END PROCESS;
```

- در این حالت، یا فقط در لبه‌های بالارونده، مقدار `counter` اضافه می‌شود و یا خطا رخ می‌دهد.

نکاتی در مورد پالس ساعت (۳)

- اگر سیگنالی در لیست حساسیت نوشته شود، می‌تواند درون PROCESS استفاده نشود.
- مثال:

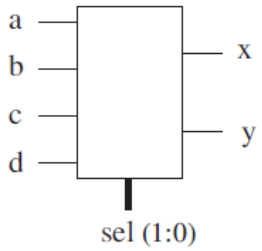
```
-----------------------
PROCESS (clk)
BEGIN
     IF(clk'EVENT AND clk='1') THEN
          x <= d;
     END IF;
END PROCESS;
-----------------------
PROCESS (clk)
BEGIN
     IF(clk'EVENT AND clk='0') THEN
          y <= d;
     END IF;
END PROCESS;
-----------------------
```

- این کد بدون خطا کامپایل می‌شود.
- دقت شود که در هر PROCESS، انتساب به یک سیگنال مجزا انجام شده است.

پیاده‌سازی مدار ترکیبی با کد ترتیبی (۱)

- با رعایت شروط زیر، مدار ساخته شده، یک مدار ترکیبی خواهد بود:
  - تمامی سیگنال‌های ورودی (سیگنال‌هایی که مقدار آن‌ها خوانده خواهد شد) باید در لیست حساسیت PROCESS ظاهر شوند.
    - در صورت رعایت نکردن این شرط، کامپایلر `warning` ای دارای همین مهفوم اعلام می‌کند.
  - تمامی ترکیب‌های سیگنال‌های ورودی/خروجی، با بررسی کد به دست آید.
    - در صورت رعابت نکردن این شرط، لچ ساخته خواهد شد.

- مثال:



sel	x	y
00	a	0
01	b	1
10	c	
11	d	

(b)

sel	x	y
00	a	0
01	b	1
10	c	y
11	d	y

(c)

sel	x	y
00	a	0
01	b	1
10	c	X
11	d	X

(d)

(a)

Figure 6.16
Circuit of example 6.12: (a) top-level diagram, (b) specifications provided, (c) implemented truth-table, and (d) the right approach.

پیاده‌سازی مدار ترکیبی با کد ترتیبی (۲)

```
1 -------------------------------------
2 LIBRARY ieee;
3 USE ieee.std_logic_1164.all;
4 -------------------------------------
5 ENTITY example IS
6     PORT (a, b, c, d: IN STD_LOGIC;
7           sel: IN INTEGER RANGE 0 TO 3;
8           x, y: OUT STD_LOGIC);
9 END example;
10 -------------------------------------
11 ARCHITECTURE example OF example IS
12 BEGIN
13     PROCESS (a, b, c, d, sel)
14     BEGIN
15          IF (sel=0) THEN
16               x<=a;
17               y<='0';
18          ELSIF (sel=1) THEN
19               x<=b;
20               y<='1';
21          ELSIF (sel=2) THEN
22               x<=c;
23          ELSE
24               x<=d;
25          END IF;
26     END PROCESS;
27 END example;
28 -------------------------------------
```

پایان بخش کد ترتیبی