تصميم النظم المنطقية باستخدام الدارات المنطقية المبرمجة

المحاضرة السادسة

د.م. خولة حموي
khawla.hamwi@gmail.com

العام الدراسي: 2024-2023

عناوين المحاضرة

- العبارات التسلسلية
  - عبارة wait
  - عبارة Loop
- العبارات التزامنية
  - باستخدام المعاملات
  - تعليمات when
  - تعليمة Generate

العبارات التسلسلية

3. عبارة WAIT

تسمح هذه العبارة بتعليق تنفيذ العبارات التسلسلية أو البرامج الجزئية لفترة زمنية معينة أو لتحقق شرط معين أو لحدوث تغير في بعض الإشارات. يستخدم لهذا الغرض ثلاثة أحرف جر كما هو موضح في الصيغة العامة لعبارة WAIT

```
wait_statement ::=
        [ label : ] wait [ sensitivity_clause ] [ condition_clause ] [ timeout_clause ] ;
sensitivity_clause ::= on sensitivity_list
sensitivity_list ::= signal_name { , signal_name }
condition_clause ::= until condition
condition ::= boolean_expression
timeout_clause ::= for time_expression
```

العبارة WAIT ON: تستخدم لحدوث تغير في متحول الإشارة وهذه العبارة تكافئ تماماً لائحة الحساسية لعبارة PROCESS أي في حال استخدام عبارة WAIT ON فلا يجوز كتابة لائحة الحساسية لعبارة PROCESS وكذلك في حال كتابة لائحة الحساسية لعبارة PROCESS فلا يجوز استخدام عبارة WAIT ON

العبارة WAIT UNITL: تستخدم العبارة لتعليق تنفيذ برنامج حتى تحقق شرط معين (نتيجة اختبار الشرط true)

العبارة WAIT FOR: تستخدم العبارة لتعليق تنفيذ برنامج لفترة زمنية تحدد بعد حرف الجر FOR تستخدم عادة في المحاكاة

العبارات التسلسلية

3. عبارة WAIT

أمثلة

8-Bit register

وصف دارة قلاب D باستخدام العبارة WAIT

```
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;
```

```
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;
```

```
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;
```

العبارات التسلسلية

4. عبارة LOOP

تستخدم هذه العبارة لتنفيذ مجموعة من العبارات بشكل متكرر. لهذه العبارة أشكال عدة كما هو موضح بالشكل:

```
loop_statement ⇐
    [ loop_label : ]
    while boolean_expression loop
        { sequential_statement }
    end loop [ loop_label ] ;
```

1. WHILE/LOOP تتكرر الحلقة طالما الشرط بعد كلمة WHILEمحقق

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

2. FOR/LOOP تتكرر الحلقة عدد محدود من المرات

```
loop_statement ⇐
    [ loop_label : ]
    for identifier in discrete_range loop
        { sequential_statement }
    end loop [ loop_label ] ;
```

```
FOR i IN x'RANGE LOOP
        x(i)<= a(M-i) AND b(i);
END LOOP;
```

```
for count_value in 0 to 127 loop
    count_out <= count_value;
    wait for 5 ns;
end loop;
```

العبارات التسلسلية

4. عبارة LOOP

3. EXITتستخدم للخروج من الحلقة

```
... LOOP [label:] EXIT
[loop_label]
        [WHEN condition];
END LOOP;
```

```
Temp:=0;
FOR i IN N-1 DOWNTO 0 LOOP
        EXIT WHEN x(i)='1';
        temp:= temp+1;
END LOOP;
```

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

4. NEXT تستخدم لتجاهل خطوة من الخطوات المكررة ضمن الحلقة

```
... LOOP [label:] NEXT
[loop_label]
        [WHEN condition];
END LOOP;
```

```
Temp:=0;
FOR i IN N-1 DOWNTO 0 LOOP
        NEXT WHEN x(i)='1';
        temp:= temp+1;
END LOOP;
```

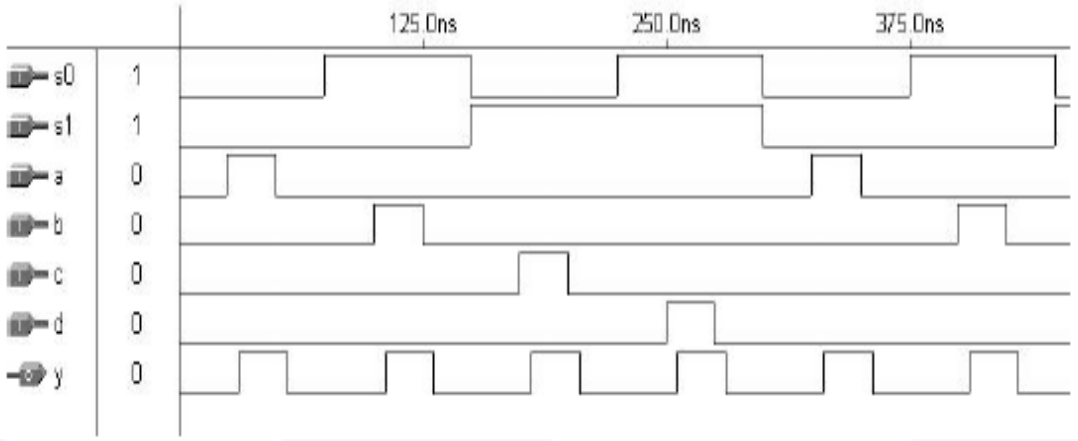
العبارات التزامنية

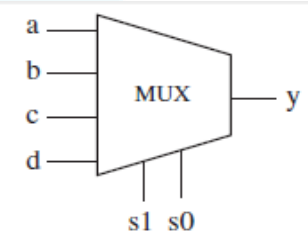
1. باستخدام المعاملات:

Operator type	Operators	Data types
Logical	NOT, AND, NAND, OR, NOR, XOR, XNOR	BIT, BIT_VECTOR, STD_LOGIC, STD_LOGIC_VECTOR, STD_ULOGIC, STD_ULOGIC_VECTOR
Arithmetic	+, −, *, /, ** (mod, rem, abs)	INTEGER, SIGNED, UNSIGNED
Comparison	=, /=, <, >, <=, >=	All above
Shift	sll, srl, sla, sra, rol, ror	BIT_VECTOR
Concatenation	&, (,,,)	Same as for logical operators, plus SIGNED and UNSIGNED

مثال: دارة ناخب بأربعة مداخل

```
1  ----------------------------------
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  ----------------------------------
5  ENTITY mux IS
6     PORT ( a, b, c, d, s0, s1: IN STD_LOGIC;
7            y: OUT STD_LOGIC);
8  END mux;
9  ----------------------------------
10 ARCHITECTURE pure_logic OF mux IS
11 BEGIN
12    y <=  (a AND NOT s1 AND NOT s0) OR
13          (b AND NOT s1 AND s0) OR
14          (c AND s1 AND NOT s0) OR
15          (d AND s1 AND s0);
16 END pure_logic;
17 ----------------------------------
```





العبارات التزامنية

2. باستخدام تعليمات WHEN:

- توجد في لغة VHDL العبارات الشرطية التزامنية التالية:
- عبارة WHEN/ELSE تشابه عبارة IF
- عبارة WITH/SELECT/WHEN تشابه عبارة CASE

WHEN / ELSE:

```
assignment WHEN condition ELSE
assignment WHEN condition ELSE
...;
```

WITH / SELECT / WHEN:

```
WITH identifier SELECT
assignment WHEN value,
assignment WHEN value,
...;
```

عند استخدام هذه التعليمة يجب مناقشة جميع الحالات
وبالتالي عادة ما نستخدم الكلمة المفتاحية others

```
------ With WHEN/ELSE -----------------------
outp <= "000" WHEN (inp='0' OR reset='1') ELSE
        "001" WHEN ctl='1' ELSE
        "010";
```

```
---- With WITH/SELECT/WHEN -------------------
WITH control SELECT
   output <= "000" WHEN reset,
             "111" WHEN set,
             UNAFFECTED WHEN OTHERS;
-----------------------------------------------
```

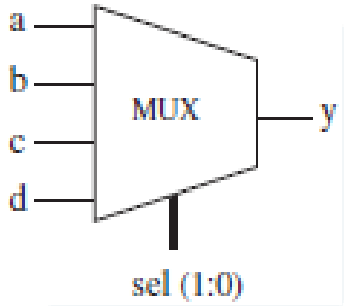
فواصل غير منقوطة

كلمة مفتاحية تستخدم عندما لا يوجد أي حدث كي ينفذ

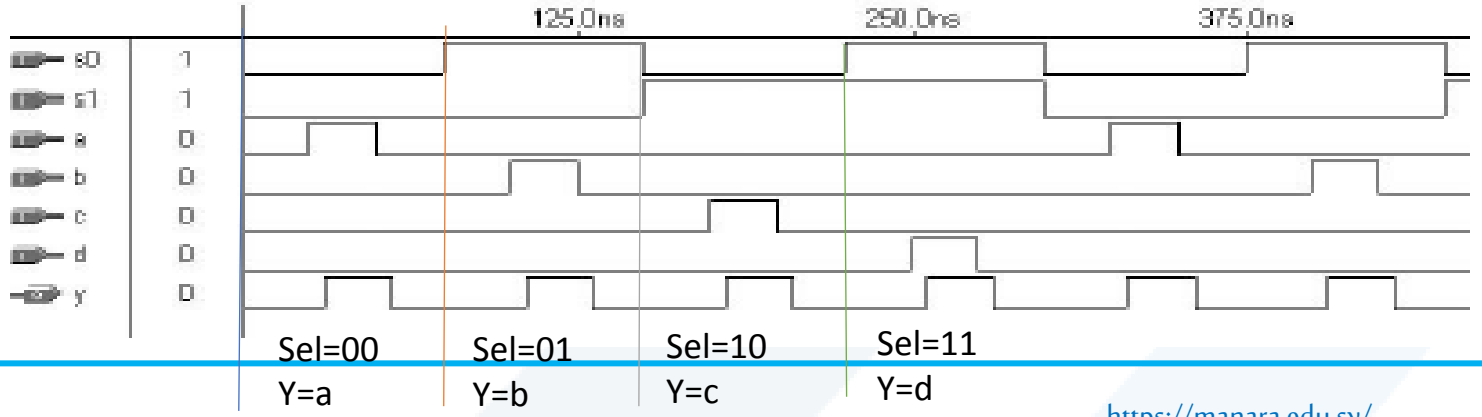
العبارات التزامنية

2. باستخدام تعليمات WHEN:

```
1  ------- Solution 1: with WHEN/ELSE --------
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  ---------------------------------------
5  ENTITY mux IS
6     PORT ( a, b, c, d: IN STD_LOGIC;
7            sel: IN STD_LOGIC_VECTOR (1 DOWNTO 0);
8            y: OUT STD_LOGIC);
9  END mux;
10 ---------------------------------------
11 ARCHITECTURE mux1 OF mux IS
12 BEGIN
13    y <=  a WHEN sel="00" ELSE
14          b WHEN sel="01" ELSE
15          c WHEN sel="10" ELSE
16          d;
17 END mux1;
18 ---------------------------------------
```




```
1  --- Solution 2: with WITH/SELECT/WHEN -----
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  ---------------------------------------
5  ENTITY mux IS
6     PORT ( a, b, c, d: IN STD_LOGIC;
7            sel: IN STD_LOGIC_VECTOR (1 DOWNTO 0);
8            y: OUT STD_LOGIC);
9  END mux;
10 ---------------------------------------
11 ARCHITECTURE mux2 OF mux IS
12 BEGIN
13    WITH sel SELECT
14       y <=  a WHEN "00",      -- notice "," instead of ";"
15             b WHEN "01",
16             c WHEN "10",
17             d WHEN OTHERS;    -- cannot be "d WHEN "11" "
18 END mux2;
19 ---------------------------------------
```



العبارات التزامنية

3. عبارة GENERATE

- تسمح عبارة GENERATE بإعادة تنفيذ شيفرة محددة عدة مرات بشكل مشابه لعبارات الحلقة التسلسلية LOOP.

```
SIGNAL x: BIT_VECTOR (7 DOWNTO 0);
SIGNAL y: BIT_VECTOR (15 DOWNTO 0);
SIGNAL z: BIT_VECTOR (7 DOWNTO 0);
...
G1: FOR i IN x'RANGE GENERATE
    z(i) <= x(i) AND y(i+8);
END GENERATE;
```

- الصيغة العامة لعبارة GENERATE مع عبارة FOR على النحو التالي:

```
label: FOR identifier IN range GENERATE
  (concurrent assignments)
END GENERATE;
```

- يمكن استخدام عبارة GENENRATE مع عبارة IF ولكن في هذه الحالة لا يمكن استخدام ELSE.
- يمكن استخدام الصيغتين السابقتين بشكل متداخل.

```
label1: FOR identifier IN range GENERATE
   ...
   label2: IF condition GENERATE
      (concurrent assignments)
   END GENERATE;
   ...
END GENERATE;
```

العبارات التزامنية

3. عبارة GENERATE

- مثال: Vector shifter

```
1  ------------------------------------------
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  ------------------------------------------
5  ENTITY shifter IS
6     PORT ( inp: IN STD_LOGIC_VECTOR (3 DOWNTO 0);
7            sel: IN INTEGER RANGE 0 TO 4;
8            outp: OUT STD_LOGIC_VECTOR (7 DOWNTO 0));
9  END shifter;
10 ------------------------------------------
11 ARCHITECTURE shifter OF shifter IS
12    SUBTYPE vector IS STD_LOGIC_VECTOR (7 DOWNTO 0);
13    TYPE matrix IS ARRAY (4 DOWNTO 0) OF vector;
14    SIGNAL row: matrix;
15 BEGIN
16    row(0) <= "0000" & inp;
17    G1: FOR i IN 1 TO 4 GENERATE
18       row(i) <= row(i-1)(6 DOWNTO 0) & '0';
19    END GENERATE;
20    outp <= row(sel);
21 END shifter;
22 ------------------------------------------
```

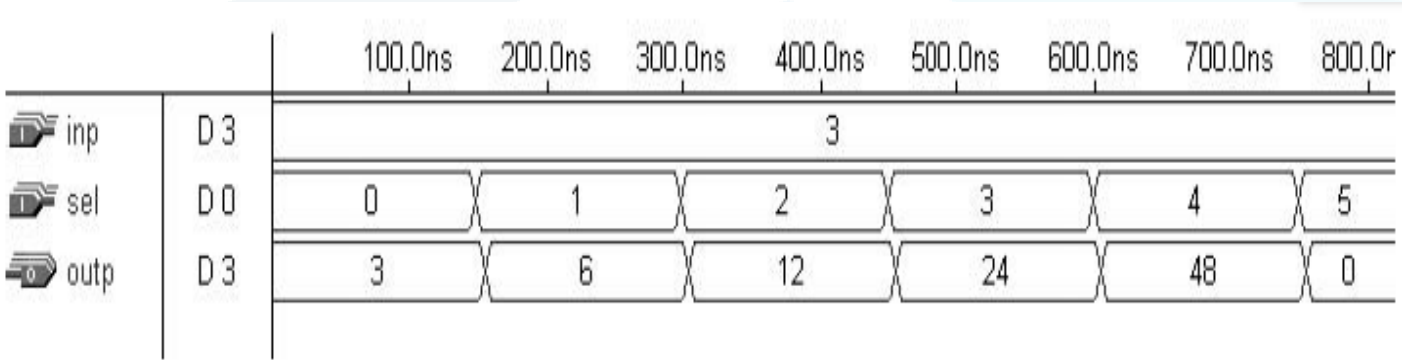
row(0): 0 0 0 0 1 1 1 1

row(1): 0 0 0 1 1 1 1 0

row(2): 0 0 1 1 1 1 0 0

row(3): 0 1 1 1 1 0 0 0

row(4): 1 1 1 1 0 0 0 0



طبقنا على الدخل الإشارة التالية (Decimal 3) Inp= "0011"

عندما إشارة الاختيار sel=0 يعني لا يوجد إزاحة وبالتالي إشارة الخرج (decimal=3)outp="00000011"

عندما إشارة الاختيار sel=1 تمت الإزاحة خانة واحدة وبالتالي إشارة الخرج (decimal=6)outp="00000110"

عندما إشارة الاختيار sel=2 تمت الإزاحة خانتين وبالتالي إشارة الخرج (decimal=12)outp="00001100"