تصميم النظم المنطقية باستخدام الدارات المنطقية المبرمجة

المحاضرة الخامسة

د.م. خولة حموي
khawla.hamwi@gmail.com

العام الدراسي: 2024-2023

عناوين المحاضرة

- الخاصيات مسبقة التعريف
- المنطق التسلسلي والمنطق التفرعي
- عبارة Process
- العبارات التسلسلية

الخاصيات مسبقة التعريف

- الخاصية attribute عبارة عن وظيفة مسبقة التعريف في لغة VHDL تساهم في إعطاء فعالية لبرنامج مكتوب بلغة VHDL. تصنف الخاصيات في لغة VHDL إلى ثلاثة أصناف:

I. **الخاصيات المتعلقة بالمجال range –related attributes:**

1. الخاصية d'LOW تعيد الدليل الأدنى للمصفوفة
2. الخاصية d'HIGH تعيد الدليل الأعلى للمصفوفة
3. الخاصية d'LEFT تعيد الدليل الأيسر للمصفوفة
4. الخاصية d'RIGHT تعيد الدليل الأيمن للمصفوفة
5. الخاصية d'LENGTH تعيد طول الشعاع
6. الخاصية d'RANGE تعيد مجال الشعاع
7. الخاصية d'REVERSE_RANGE تعيد مجال معكوس للشعاع
8. الخاصية d'ASCENDING [(N)] تعيد قيمة بولیانية (true) إذا كان ترتيب دليل المصفوفة بشكل تصاعدي

الخاصيات مسبقة التعريف

I. الخاصيات المتعلقة بالمجال range –related attributes:

```
SIGNAL d : STD_LOGIC_VECTOR (7 DOWNTO 0);
```

Then:

```
d'LOW=0, d'HIGH=7, d'LEFT=7, d'RIGHT=0, d'LENGTH=8,
d'RANGE=(7 downto 0), d'REVERSE_RANGE=(0 to 7).
```

```
---------------------------------
TYPE my integer IS RANGE 0 TO 255;
x1 <= my_integer'LEFT;       --result=0 (type of x1 must be my_integer)
x2 <= my_integer'RIGHT;      --result=255 (type of x2 must be my_integer)
x3 <= my_integer'LOW;        --result=0 (type of x3 must be my_integer)
x4 <= my_integer'HIGH;       --result=255 (type of x4 must be my_integer)
y <= my_integer'ASCENDING;   --result=TRUE (type of y must be BOOLEAN)
-----------------------------------------------------------------------
```

الخاصيات مسبقة التعريف

II. الخاصيات المتعلقة بالموقع position–related attributes:

1. الخاصية T'VAL (X) تعيد القيمة من النوع T الموجودة في الموقع X
2. الخاصية T'POS (X) تعيد موقع القيمة X
3. الخاصية T'LEFTOF (X) تعيد القيمةالموجودة على يسارالقيمة Xضمن المجال المدروس
4. الخاصية T'RIGHTOF(X) تعيد القيمةالموجودة على يمين القيمة Xضمن المجال المدروس

```
TYPE state IS (reset, start, count);

        state'VAL(0)= reset
```

```
TYPE state IS (a, b, c);
 x1 <= state'LEFT;    --result=a
 x2 <= state'RIGHT;   --result=c
 x3 <= state'LOW;     --result=a
 x4 <= state'HIGH;    --result=c
 y <= state'POS(b);   --result=1
 z <= state'VAL(1);   --result=b
 ------------------------------
```

الخاصيات مسبقة التعريف

III. الخاصيات المتعلقة بالأحداث event–related attributes:

تستخدم هذه الخاصيات لمراقبة التغيرات في الإشارة (كالانتقال من المستوى 1 إلى المستوى المنطقي 0 وبالعكس)

1. الخاصية S'EVENT تعيد True عند حدوث حدث على الإشارة S
2. الخاصية S'STABLE[(T)] تنشئ إشارة جديدة نوع المعطيات Boolean والتي تعيد القيمة true طالما أن الإشارة لم تغير قيمتها, وتصبح هذه القيمة false في حال غيرت الإشارات قيمتها، وتبقى هذه القيمة false خلال الفترة الزمنية T.
3. الخاصية S'DELAYED[(T)] تنشئ نسخة من الإشارة S ومتأخرة عنها بالمقدار الزمني T.
4. الخاصية S'QUIET[(T)] تعمل تماماً مثل الخاصية S'STABLE[(T)]، لكن تتفاعل من أجل كل تحديثات الإشارة S
5. الخاصية S'TRANSACTION تنشئ إشارة جديدة نوع BIT التي تتغير قيمتها مع كل تغير للإشارة S.

الخاصيات مسبقة التعريف

III. الخاصيات المتعلقة بالأحداث event–related attributes:

تستخدم هذه الخاصيات لمراقبة التغيرات في الإشارة (كالانتقال من المستوى 1 إلى المستوى المنطقي 0 وبالعكس)

6. الخاصية S'ACTIVE تعمل تماماً مثل الخاصية S'EVENT لكن تتفاعل من أجل كل تحديثات الإشارة S.

7. الخاصية S'LAST_EVENT تعيد الزمن الذي يحدث فيه آخر تغير للإشارة S.

8. الخاصية S'LAST_ACTIVE تعيد الزمن الذي يحدث فيه آخر تحديث للإشارة S.

9. الخاصية S'LAST_VALUE تعيد قيمة الإشارة S قبل حدوث تغير في قيمتها.

أمثلة

```
IF (clk'EVENT AND clk='1')...          -- EVENT attribute used
                                       -- with IF
IF (NOT clk'STABLE AND clk='1')...     -- STABLE attribute used
                                       -- with IF
WAIT UNTIL (clk'EVENT AND clk='1');    -- EVENT attribute used
                                       -- with WAIT
```

عبارة Generic

- تستخدم هذه العبارة مع عبارة port ضمن الكيان ENITITY لتخصيص متحول عام يسهل عملية تعديل هذا المتحول.
- الصيغة العامة لعبارة GENERIC على النحو التالي:

```
GENERIC (parameter_name : parameter_type := parameter_value);
```

مثال

```
ENTITY my_entity IS
   GENERIC (n : INTEGER := 8);
   PORT (...);
END my_entity;
ARCHITECTURE my_architecture OF my_entity IS
   ...
END my_architecture:
```

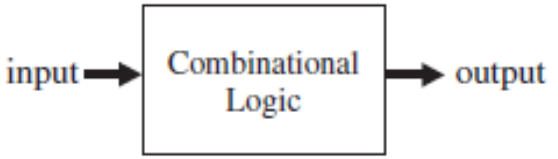
- تخصص عبارة GENERIC متحول n نوع INTEGER والقيمة الأولية لهذا المتحول تساوي 8 بحيث تستبدل n بالقيمة 8 أينما وردت سواء في البنيان أم في الكيان
- يمكن التصريح عن أكثر من متحول ضمن عبارة GENERIC على النحو التالي:

GENERIC (n:INTEGER:=8;vector:BIT_VECTOR:="00001111");

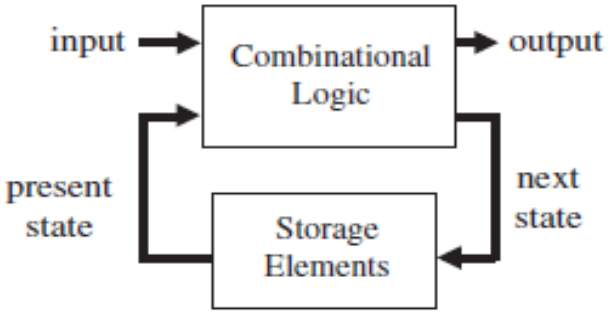
المنطق التسلسلي والمنطق التفرعي

الفرق بين المنطق التسلسلي (sequential logic) والمنطق التفرعي أو المتزامن(combinational logic):

يعتمد خرج المنطق التفرعي (المتزامن) على الدخل الحالي وبالتالي لا يحتاج إلى عنصر تخزين

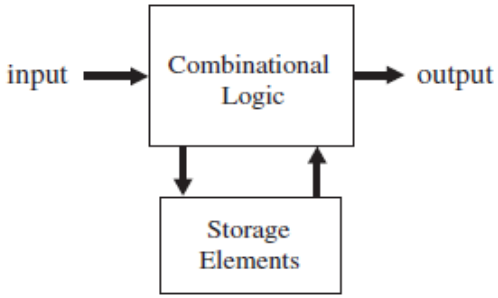


يعتمد المنطق التسلسلي على وجود عنصر تخزين في حلقة تغذية عكسية مع المنطق التفرعي



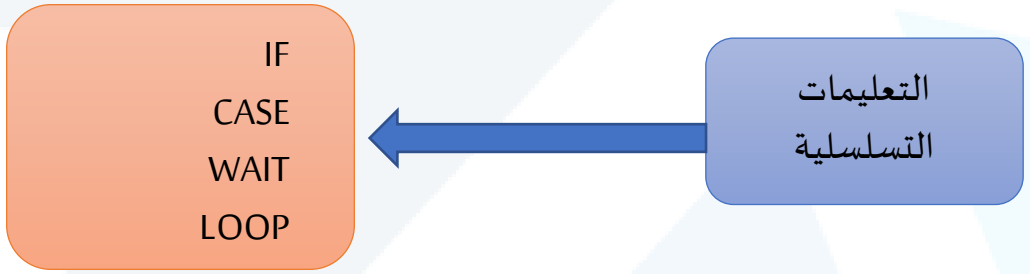
ملاحظة:

ليس كل دارة تحتوي عنصر تخزين هي دارة تسلسلية. مثل RAM هي دارة تفرعية تحتوي على عنصر تخزين في حلقة تغذية أمامية

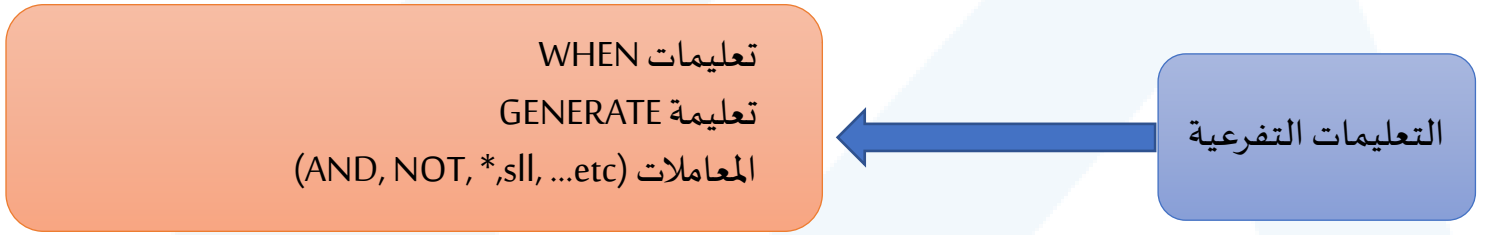


المنطق التسلسلي والمنطق التفرعي

- يمكن أن يكون برنامج VHDL تفرعي (متزامن) أو تسلسلي بناء على التعليمات المستخدمة ضمن البنيان.
- تنفذ الأجزاء المكونة من PROCESSES أو FUNCTIONS أو PROCEDURES بشكل متزامن مع التعليمات الموجودة خارجها



تستخدم هذه التعليمات **داخل** PROCESSES، FUNCTIONS و PROCEDURES



تستخدم هذه التعليمات **خارج** PROCESSES، FUNCTIONS و PROCEDURES

عبارة PROCESS

- هي جزء من برنامج الـ VHDL. تتضمن تعليمات تسلسلية بالإضافة إلى احتوائها على لائحة حساسية (ما عدا في حالة استخدام عبارة WAIT)
- توجد عبارة PROCESS ضمن البرنامج الرئيسي وتنفذ في كل مرة تتغير فيها الإشارات الموجودة في لائحة الحساسية
- تنفذ عبارة PROCESS بشكل تزامني مع التعليمات الموجودة خارجها. أما التعليمات الموجودة ضمنها فهي تنفذ تسلسلياً.

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

- استخدام VARIABLE اختياري وهو مرئي فقط ضمن الPROCESS على عكس SIGNAL التي تكون مرئية على كامل البرنامج (تعرف خارج عبارة PROCESS)

العبارات التسلسلية

1. **عبارة IF:** تعد عبارة IF الأكثر استخداماً في لغة VHDL. تكتب الصيغة العامة لعبارة IF مع مثال توضيحي على الشكل:

```
if_statement ::=
        [ if_label : ]
            if condition then
                sequence_of_statements
            { elsif condition then
                sequence_of_statements }
            [ else
                sequence_of_statements ]
            end if [ if_label ] ;
```

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

2. **عبارة CASE**

تسمح هذه العبارة بتحديد خيار من ضمن مجموعة من الخيارات. وتستخدم غالباً لنمذجة الدارات التركيبية وجداول الحقيقة ومخططات الحالة. الصيغة العامة لعبارة CASE مع مثال توضيحي بالشكل:

```
case_statement ::=
        [ case_label : ]
            case expression is
                case_statement_alternative
                { case_statement_alternative }
            end case [ case_label ] ;

case_statement_alternative ::=
        when choices =>
            sequence_of_statements
```

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

العبارات التسلسلية

مثال: وصف دارة قلاب D باستخدام عبارة CASE

```
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 ------------------------------------------
```