تصميم النظم المنطقية باستخدام الدارات المنطقية المبرمجة

المحاضرة الرابعة

د.م. خولة حموي
khawla.hamwi@gmail.com

العام الدراسي: 2024-2023

جَامعة المَنارة
MANARA UNIVERSITY

عناوين المحاضرة

- أنواع المعطيات مسبقة التعريف (أمثلة)
- أنواع المعطيات المعرفة من قبل المستخدم

1. صحيح (integer) معرف من قبل المستخدم
2. لائحة معرف من قبل المستخدم
3. فاصلة عائمة (floating point) معرف من قبل المستخدم
4. فيزيائي معرف من قبل المستخدم
5. المصفوفات Arrays

- نوع المعطيات Subtype
- التحويل بين أنواع المعطيات
- الخاصيات مسبقة التعريف
- عبارة Generic

أنواع المعطيات مسبقة التعريف

أمثلة:

بفرض لدينا الإشارات التالية حدد أي العبارات التالية صحيحة وأيها خاطئة مع التعليل

```
SIGNAL a: BIT;
SIGNAL b: BIT_VECTOR(7 DOWNTO 0);
SIGNAL c: STD_LOGIC;
SIGNAL d: STD_LOGIC_VECTOR(7 DOWNTO 0);
SIGNAL e: INTEGER RANGE 0 TO 255;
...
a <= b(5);   -- legal (same scalar type: BIT)
b(0) <= a;   -- legal (same scalar type: BIT)
c <= d(5);   -- legal (same scalar type: STD_LOGIC)
d(0) <= c;   -- legal (same scalar type: STD_LOGIC)
a <= c;      -- illegal (type mismatch: BIT x STD_LOGIC)
b <= d;      -- illegal (type mismatch: BIT_VECTOR x
             -- STD_LOGIC_VECTOR)
e <= b;      -- illegal (type mismatch: INTEGER x BIT_VECTOR)
e <= d;      -- illegal (type mismatch: INTEGER x
             -- STD_LOGIC_VECTOR)
```

أنواع المعطيات المعرفة من قبل المستخدم

- تتيح لغة VHDL للمستخدم تعريف أنواع المعطيات الخاصة به باستخدام الكلمة المحجوزة TYPE.
- يمكن التمييز بين أنواع مختلفة من هذه الأنواع:

1. نوع معطيات صحيح (integer) معرف من قبل المستخدم:

```
TYPE integer IS RANGE -2147483647 TO +2147483647;
-- This is indeed the pre-defined type INTEGER.

TYPE natural IS RANGE 0 TO +2147483647;
-- This is indeed the pre-defined type NATURAL.

TYPE my_integer IS RANGE -32 TO 32;
-- A user-defined subset of integers.

TYPE student_grade IS RANGE 0 TO 100;
-- A user-defined subset of integers or naturals.
```

أنواع المعطيات المعرفة من قبل المستخدم

2. نوع معطيات لائحة معرف من قبل المستخدم:

```
TYPE bit IS ('0', '1');
-- This is indeed the pre-defined type BIT

TYPE my_logic IS ('0', '1', 'Z');
-- A user-defined subset of std_logic.

TYPE bit_vector IS ARRAY (NATURAL RANGE <>) OF BIT;
-- This is indeed the pre-defined type BIT_VECTOR.
-- RANGE <> is used to indicate that the range is unconstrained.
-- NATURAL RANGE <>, on the other hand, indicates that the only
-- restriction is that the range must fall within the NATURAL
-- range.

TYPE state IS (idle, forward, backward, stop);
-- An enumerated data type, typical of finite state machines.

TYPE color IS (red, green, blue, white);
-- Another enumerated data type.
```

يستخدم هذا النوع من المعطيات في توصيف مخططات الحالة

TYPE state IS (idle, forward, backward, stop);

--An anumerated data type, typical of finite state machines

أنواع المعطيات المعرفة من قبل المستخدم

3. نوع معطيات فاصلة عائمة (floating point) معرف من قبل المستخدم:

```
TYPE input_level IS range -10.0 to +10.0;

TYPE propability IS range 0.0 to 1.0;
```

4. نوع معطيات فيزيائي معرف من قبل المستخدم:

```
TYPE time IS range implementation defined
        units
                fs;
                ps=1000fs;
                ns= 1000ps;
                us=1000ns;
                ms= 1000us;
                sec= 1000 ms;
                min= 60sec;
                hr= 60min;
end units;
```

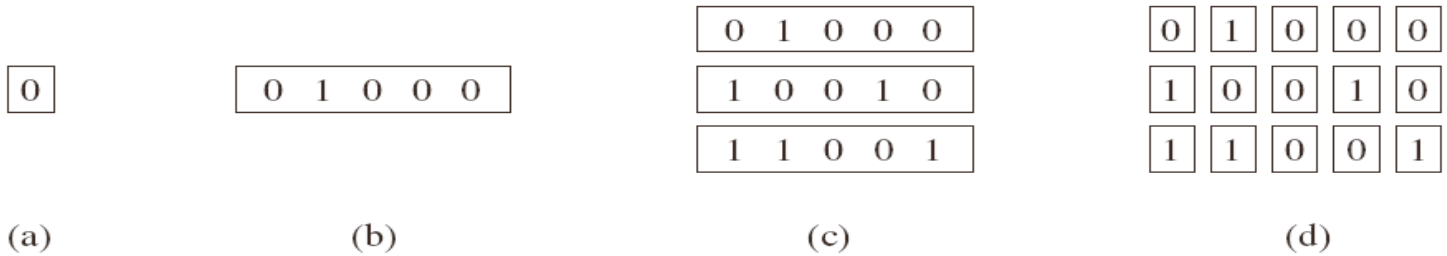
```
TYPE length IS range 0 to 1E9
        units
                um;-- primary unit:micron
                mm=1000um;
                m= 1000mm;
                inch= 25400um;
                foot= 12inch;
end units length;
```

```
TYPE resistance IS range 0 to 1E9
        units
                ohm;
                kohm=1000ohm;
                Mohm= 1000kohm;
end units resistances;
```

أنواع المعطيات المعرفة من قبل المستخدم

5. المصفوفات Arrays

- تشكل المصفوفات من تجمع من نفس بنى المعطيات ويمكن أن تكون أحادية البعد 1D أو ثنائية البعد 2D أو ضرب مصفوتين أحاديتي البعد 1Dx1D.
- تستخدم المصفوفات بشكل رئيسي لوصف الذواكر في لغة VHDL



(a) scalar, (b) 1D, (c) 1Dx1D, and (d) 2D data arrays

الشكل العام لتعريف المصفوفة على النحو الآتي:

```
TYPE type_name IS ARRAY (specification) OF data_type;
```

كي نستخدم هذا النوع يجب تعريف إشارة كالتالي:

```
SIGNAL signal_name: type_name [:= initial_value];
```

أمثلة:

```
TYPE row IS ARRAY (7 DOWNTO 0) OF STD_LOGIC;     -- 1D array
TYPE matrix IS ARRAY (0 TO 3) OF row;             -- 1Dx1D array
SIGNAL x: matrix;                                 -- 1Dx1D signal
```

أنواع المعطيات المعرفة من قبل المستخدم

5. المصفوفات Arrays

```
------- Package: -------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.all;
----------------------------
PACKAGE my_data_types IS
   TYPE vector_array IS ARRAY (NATURAL RANGE <>) OF
      STD_LOGIC_VECTOR(7 DOWNTO 0);
END my_data_types;

------- Main code: -----------------------
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE work.my_data_types.all;      -- user-defined package
--------------------------
ENTITY mux IS
   PORT (inp: IN VECTOR_ARRAY (0 TO 3);
   ... );
END mux;
   ... ;
-----------------------------------------
```

منفذ المصفوفة:

- كما رأينا سابقاً، لا يوجد نوع معطيات معرف مسبقاً لمصفوفة ثنائية.
- قد نحتاج في بعض الأحيان إلى تعريف منافذ للدارة على شكل مصفوفة

وبما أن تعريف (TYPE) **غير مسموح** ضمن الكيان فلا بد من التصريح عن هذا النوع من المعطيات ضمن حزمة خاصة (package) تكون مرئية بالنسبة لكل البرنامج

نوع المعطيات Subtype

- نوع المعطيات Subtype موافق لنوع المعطيات type مع بعض القيود وذلك لحل مشكلة التوافق بين أنواع المعطيات.

أمثلة

```
SUBTYPE my_logic IS STD_LOGIC RANGE '0' TO 'Z';
-- Recall that STD_LOGIC=('X','0','1','Z','W','L','H','-').
-- Therefore, my_logic=('0','1','Z').

SUBTYPE my_color IS color RANGE red TO blue;
-- Since color=(red, green, blue, white), then
-- my_color=(red, green, blue).

SUBTYPE small_integer IS INTEGER RANGE -32 TO 32;
-- A subtype of INTEGER.
```

أمثلة عن الحالات الصحيحة والخاطئة

```
SUBTYPE my_logic IS STD_LOGIC RANGE '0' TO '1';
SIGNAL a: BIT;
SIGNAL b: STD_LOGIC;
SIGNAL c: my_logic;
...
b <= a;     -- illegal (type mismatch: BIT versus STD_LOGIC)
b <= c;     -- legal (same "base" type: STD_LOGIC)
```

التحويل بين أنواع المعطيات

- لا تسمح لغة VHDL بإجراء العمليات (كالحسابية والمنطقية) على أنواع معطيات مختلفة لذلك كان لا بد من تحويل بنى المعطيات من نوع لآخر.
- تتم عملية التحويل وفق طريقتين:
  1. عبر كتابة مجموعة من العبارات بلغة VHDL.
  2. استدعاء مجموعة من الوظائف ضمن حزمة معرفة مسبقاً التي تمكن من عملية التحويل.
- في حالة كون المعطيات تنتمي إلى نفس بنى المعطيات الأساسية فإن الحزمة std_logic_1164 والمكتبة ieee تقوم بشكل مباشر بتحويل المعطيات كما هو موضح بالشكل:

```
TYPE long IS INTEGER RANGE -100 TO 100;
TYPE short IS INTEGER RANGE -10 TO 10;
SIGNAL x : short;
SIGNAL y : long;
...
y <= 2*x + 5;          -- error, type mismatch
y <= long(2*x + 5);    -- OK, result converted into type long
```

التحويل بين أنواع المعطيات

- تتيح لغة VHDL مجموعة من وظائف التحويل ضمن الحزمة std_logic_1164 والمكتبة ieee موضحة كمايلي:

1. **Conv_integer(p)**: تحول هذا الوظيفة البارامتر p من النوع integer أو unsigned أو signed أو std_logic إلى النوع integer.
2. **Conv_unsigned (p,b)** : تحول هذا الوظيفة البارامتر p من النوع integer أو unsigned أو signed أو std_logic إلى النوع unsigned بطول b خانة.
3. **Conv_signed (p,b)** : تحول هذا الوظيفة البارامتر p من النوع integer أو unsigned أو signed أو std_logic إلى النوع signed بطول b خانة.
4. **Conv_std_logic_vector (p,b)** : تحول هذا الوظيفة البارامتر p من النوع integer أو unsigned أو signed أو std_logic إلى النوع std_logic_vector بطول bخانة.

أمثلة:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
...
SIGNAL a: IN UNSIGNED (7 DOWNTO 0);
SIGNAL b: IN UNSIGNED (7 DOWNTO 0);
SIGNAL y: OUT STD_LOGIC_VECTOR (7 DOWNTO 0);
...
y <= CONV_STD_LOGIC_VECTOR ((a+b), 8);
-- Legal operation: a+b is converted from UNSIGNED to an
-- 8-bit STD LOGIC VECTOR value, then assigned to y.
```

التحويل بين أنواع المعطيات

```
1  ----- Solution 1: in/out=SIGNED ---------
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  USE ieee.std_logic_arith.all;
5  ---------------------------------------
6  ENTITY adder1 IS
7     PORT ( a, b : IN SIGNED (3 DOWNTO 0);
8            sum : OUT SIGNED (4 DOWNTO 0));
9  END adder1;
10 ---------------------------------------
11 ARCHITECTURE adder1 OF adder1 IS
12 BEGIN
13    sum <= a + b;
14 END adder1;
15 ---------------------------------------
```

```
1  ------ Solution 2: out=INTEGER ----------
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  USE ieee.std_logic_arith.all;
5  ---------------------------------------
6  ENTITY adder2 IS
7     PORT ( a, b : IN SIGNED (3 DOWNTO 0);
8            sum : OUT INTEGER RANGE -16 TO 15);
9  END adder2;
10 ---------------------------------------
11 ARCHITECTURE adder2 OF adder2 IS
12 BEGIN
13    sum <= CONV_INTEGER(a + b);
14 END adder2;
15 ---------------------------------------
```