تصميم النظم المنطقية باستخدام الدارات المنطقية المبرمجة

المحاضرة الثالثة

د.م. خولة حموي
khawla.hamwi@gmail.com

العام الدراسي: 2023-2024

عناوين المحاضرة

- أنواع المعطيات
- أنواع المعاملات
- أنواع المعاملات مسبقة التعريف
- أنواع المعاملات المعرفة من قبل المستخدم

أنواع المعطيات Object Types

- توجد في لغة VHDL الأغراض (المعطيات) التالية:
- **الإشارة Signal:** تمثل الإشارة أسلاك الربط التي تربط بين مكونات الدارة الرقمية مع بعضها البعض. وتعتبر المنافذ التي يصرح عنها في قسم الكيان (ports) عبارة عن متحول نوع الإشارة. يتم التصريح عنها بالشكل:

```
SIGNAL signal_name: signal_type [:=initial_value]
```

Example: Signal a,b,c:std_logic<='0';

- **المتغيرات Variables:** تستخدم لتخصيص حجم ذاكرة محلي لقيم المتحولات المؤقتة وهي مرئية فقط من داخل العملية PROCCESS. يتم التصريح عنها بالشكل:

```
VARIABLE variable_name, variable_name,...: data-type
```

Example: variable index:integer:=0; **Example:** variable sum, average :real;

- **الثوابت Constants:** تستخدم لتخصيص قيمة محددة لأنواع المعطيات يتم التصريح عنها بالشكل:

```
CONSTANT constant_name{constant_name]:type_name[:=vallue]
```

Example: Constant PI:real:=3.1414; **Example:** constant prop_delay:time:=3ns

أنواع المعاملات Operators Types

1. معاملات التعيين Assignment Operators

يستخدم لتخصيص قيم لمتحول نوع إشارة signal	<=
يستخدم لتخصيص قيم لمتحول نوع Variable أو Constant أو Generic كما يستخدم لإعطاء قيم أولية لأنواع المعطيات	:=
يستخدم لتحديد قيم رقمية معينة ضمن عناصر Vector مع الكلمة المحجوزة others	=>

أمثلة:

```
SIGNAL x : STD_LOGIC;
VARIABLE y : STD_LOGIC_VECTOR(3 DOWNTO 0);   -- Leftmost bit is MSB
SIGNAL w: STD_LOGIC_VECTOR(0 TO 7);          -- Rightmost bit is MSB

x <= '1';          -- '1' is assigned to SIGNAL x using "<="
y := "0000";       -- "0000" is assigned to VARIABLE y using ":="
w <= "10000000";                    -- LSB is '1', the others are '0'
w <= (0 =>'1', OTHERS =>'0');       -- LSB is '1', the others are '0'
```

أنواع المعاملات Operators Types

2. المعاملات المنطقية Logical operators

تتيح لغة VHDL المعاملات المنطقية الأساسية التالية:

NOT, AND, NAND, OR, NOR, XOR, XNOR

أمثلة:

```
y <= NOT a AND b;        -- (a'.b)
y <= NOT (a AND b);      -- (a.b)'
y <= a NAND b;           -- (a.b)'
```

3. المعاملات الحسابية Arithmetic operators

+	Addition	**	Exponentiation
–	Subtraction	MOD	Modulus
*	Multiplication	REM	Remainder
/	Division	ABS	Absolute value

أمثلة:

X<=(a+b)**N;

Y<=ABS(a)+ABS(b);

Z<= a/(a+b);

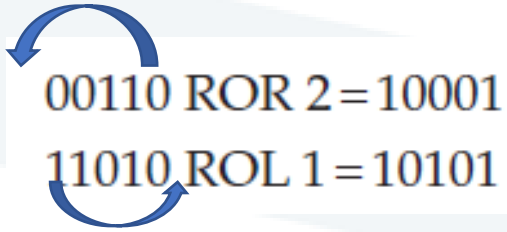
أنواع المعاملات Operators Types

4. معاملات الإزاحة Shift operators

SLL	الإزاحة المنطقية نحو اليسار، تزاح المعطيات نحو اليسار ويملأ مكانها '0'
SRL	الإزاحة المنطقية نحو اليمين، تزاح المعطيات نحو اليمين ويملأ مكانها '0'
SLA	الإزاحة الحسابية نحو اليسار، تزاح المعطيات نحو اليسار ويملأ مكانها الخانة التي على اليمين
SRA	الإزاحة الحسابية نحو اليمين، تزاح المعطيات نحو اليمين ويملأ مكانها الخانة التي على اليسار
ROL	دوران من اليسار (إزاحة دائرية من اليسار باتجاه اليمين)
ROR	دوران من اليمين (إزاحة دائرية من اليمين باتجاه اليسار)

أنواع المعاملات Operators Types

4. معاملات الإزاحة Shift operators



أمثلة:

a<="11001";

X<= a SLL 2; -- result: x<="00100"

y<= a SLA 2; -- result: y<="00111"

z<= a SRL 3; -- result: z<="00011"

"01011" SRL 2 = "00010"

"10010" SRL 2 = "00100"

"11001" SLL 1 = "10010"

"11001" SRL −1 = "10010"

x<="1100"; Y<= x rol 2; -- result: Y<="0011"

x<= "01001"; Y<= x rol 2; -- result: Y<="00101"

x= "110010"; Y<= x rol 1; -- result: Y<="100101"

x= "110010"; Y<= x ror 2; -- result: Y<="101100"

x= "110010"; Y<=x ror -3; -- result: Y<="010110"

Y<= x rol 3

أنواع المعاملات Operators Types

5. معاملات المقارنة Comparison operators

=	Equal to	>	Greater than
/=	Not equal to	<=	Less than or equal to
<	Less than	>=	Greater than or equal to

Operators.

Operator type	Operators	Data types
Assignment	<=, :=, =>	Any
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

أنواع المعاملات Operators Types

أمثلة

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;   -- extra package necessary
...
SIGNAL a: IN SIGNED (7 DOWNTO 0);
SIGNAL b: IN SIGNED (7 DOWNTO 0);
SIGNAL x: OUT SIGNED (7 DOWNTO 0);
...
v <= a + b;        -- legal (arithmetic operation OK)
w <= a AND b;      -- illegal (logical operation not OK)
```

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;     -- no extra package required
...
SIGNAL a: IN STD_LOGIC_VECTOR (7 DOWNTO 0);
SIGNAL b: IN STD_LOGIC_VECTOR (7 DOWNTO 0);
SIGNAL x: OUT STD_LOGIC_VECTOR (7 DOWNTO 0);
...
v <= a + b;        -- illegal (arithmetic operation not OK)
w <= a AND b;      -- legal (logical operation OK)
```

أنواع المعطيات مسبقة التعريف

- تؤمن لغة VHDL مجموعة كبيرة من أنواع المعطيات المسبقة التعريف خلال المعيار IEEE Standard 1076والمعيار IEEE Standard1164
- توجد أنواع المعطيات المسبقة التعريف ضمن الحزم التالية:

1. **الحزمة Standard من المكتبة std:** تعرف أنواع معطيات مثل Bit, Boolean, Real, Integer
2. **الحزمة std_logic_1164 من المكتبة ieee:** تعرف أنواع معطيات مثل Std_logic, std_ulogic
3. **الحزمة std_logic_arith من المكتبة ieee:** تعرف أنواع معطيات مثل Signed, Unsignedومجموعة من توابع تحويل أنواع المعطيات
4. **الحزمة std_logic_signed والحزمة std_logic_unsigned من المكتبة ieee:** تحتوي على مجموعة من التوابع functions تسمح بالتعامل مع نوع المعطيات std_logic_vector التي تنفذ على المعطيات signed, unsigned

أنواع المعطيات مسبقة التعريف

1. BIT (and BIT_VECTOR)

أمثلة:

```
SIGNAL x: BIT;
-- x is declared as a one-digit signal of type BIT.

SIGNAL y: BIT_VECTOR (3 DOWNTO 0);
-- y is a 4-bit vector, with the leftmost bit being the MSB.

x <= '1';
-- x is a single-bit signal (as specified above), whose value is
-- '1'. Notice that single quotes (' ') are used for a single bit.

y <= "0111";
-- y is a 4-bit signal (as specified above), whose value is "0111"
-- (MSB='0'). Notice that double quotes (" ") are used for
-- vectors.

w <= "01110001";
-- w is an 8-bit signal, whose value is "01110001" (MSB='1').
```

أنواع المعطيات مسبقة التعريف

2. STD_LOGIC (and STD_LOGIC_VECTOR) (8-level logic)

يأخذ هذا النوع من المعطيات القيم المنطقية التالية:

'X'	Forcing Unknown	(synthesizable unknown)
'0'	Forcing Low	(synthesizable logic '1')
'1'	Forcing High	(synthesizable logic '0')
'Z'	High impedance	(synthesizable tri-state buffer)
'W'	Weak unknown	
'L'	Weak low	
'H'	Weak high	
'–'	Don't care	

أنواع المعطيات مسبقة التعريف

2. STD_LOGIC (and STD_LOGIC_VECTOR) (8-level logic)

تستخدم مستويات الـ std_logic في المحاكاة فقط

في حال ربط إشارتين من نوع STD_LOGICإلى نفس العقدة وكان هناك اختلاف في مستوى الإشارة المنطقي، يُحل هذا التعارض بشكل أتوماتيكي بالاعتماد على الجدول التالي

Resolved logic system (STD_LOGIC).

	X	0	1	Z	W	L	H	–
X	X	X	X	X	X	X	X	X
0	X	0	X	0	0	0	0	X
1	X	X	1	1	1	1	1	X
Z	X	0	1	Z	W	L	H	X
W	X	0	1	W	W	W	W	X
L	X	0	1	L	W	L	W	X
H	X	0	1	H	W	W	H	X
–	X	X	X	X	X	X	X	X

أمثلة:

```
SIGNAL x: STD_LOGIC;
-- x is declared as a one-digit (scalar) signal of type STD_LOGIC.

SIGNAL y: STD_LOGIC_VECTOR (3 DOWNTO 0) := "0001";
-- y is declared as a 4-bit vector, with the leftmost bit being
-- the MSB. The initial value (optional) of y is "0001". Notice
-- that the ":=" operator is used to establish the initial value.
```

أنواع المعطيات مسبقة التعريف

3. STD_ULOGIC (STD_ULOGIC_VECTOR)(9-level logic)

- يضاف إلى القيم الثمانية السابقة الحالة 'U' التي تعني دون تهيئة. القيم التسعة لنوع المعطيات std_ulogic مبينة على النحو الآتي:

'X'Forcing Unknown	'0'Forcing Low
'1' Forcing High	'Z' High impedance
'W' Weak unknown	'L' Weak low
'H' Weak High	'_' Don't care
'U' uninitiated or unresolved	

- بالنسبة إلى نوع المعطيات STD_ULOGIC(STD_ULOGIC_VECTOR) فإن التعارض بين المستويات المنطقية لا يحل بشكل آلي. في هذه الحالة لايجوز ربط المنافذ مع بعضها البعض بشكل مباشر.
- عندما يكون لدينا منفذين لا يرتبطان أبداً، يمكن استخدام هذا النوع من المعطيات لتحديد أخطاء النظام

أنواع المعطيات مسبقة التعريف

4. *BOOLEAN: True, False.*
5. *INTEGER: 32-bit integers (from -2,147,483,647 to +2,147,483,647.*
6. *NATURAL: Non-negative integers (from 0 to +2,147,483,647.*
7. *REAL: Real numbers ranging from -1.0E38 to +1.0E38. Not synthesizable.*
8. *Physical literals: Used to inform physical quantities, like time, voltage, etc. Useful in simulations. Not synthesizable.*
9. *Character literals: Single ASCII character or a string of such characters. Not synthesizable.*
10. *SIGNED and UNSIGNED: data types defined in the std_logic_arith package of the ieee library. They have the appearance of STD_LOGIC_VECTOR, but accept arithmetic operations, which are typical of INTEGER data types (SIGNED and UNSIGNED).*