C7 – Les Types

Yann DOUZE

VHDL

自定义类型-枚举类型

Les Types énumérés

- Définition d'un nouveau type de donnée

数据对象 数据类型

```
type Opcode is (Add, Neg, Load, Store, Jmp, Halt);
signal S: Opcode;
```

```
S <= Add;
```

```
process(S)
begin
case S is
when Add =>
      ...
```

枚举类型的综合

Synthèse des types énumérés

虽然编程是会被自定义，但是在综 合后，系统会分配二进制编码， 在FPGA内会被实现为独热码。

编码

- Encodage du type énuméré pour la synthèse

	Binaire	One hot
Add	000	100000
Neg	001	010000
Load	010	001000
Store	011	000100
Jmp	100	000010
Halt	101	000001

ASIC

FPGA

默认类型

Les Types définit par défaut

不用声明，
已默认包括.

```
library STD, WORK;
use STD.STANDARD.all;
```

包括

Inclut par défaut

```
INTEGER      0  1  99
BIT          '0'  '1'
BIT_VECTOR   "101011"
BOOLEAN      FALSE  TRUE
STRING       "Hello World"
TIME         10 NS
REAL         1.345
```

描述程序包的数据类型

Types du package

STD.STANDARD

多值逻辑类型

Les types logiques à valeurs multiple

Dans le package STD.STANDARD

```
type BOLEAN is (False, True);

type BIT is (‘0’, ‘1’);
```

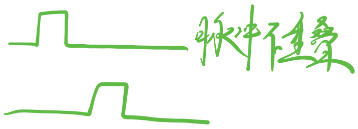
Dans le package IEEE.STD_LOGIC_1164

```
type STD_ULOGIC is (
‘U’,   -- non initialisé (par défaut)
‘X’,   -- état inconnu
‘0’,   -- 0 puissant
‘1’,   -- 1 puissant
‘Z’,   -- Haute impédance
‘W’,   -- État inconnu mais faible
‘L’,   -- 0 faible
‘H’,   -- 1 faible
‘-’);  -- indifférent (pour la synthèse)
```

子类型

```
subtype STD_LOGIC is RESOLVED STD_ULOGIC;
```

Std_ulogic不允许两个以上的驱动器



Résolution : définit la priorité

决定优先级

判决函数
决断函数，用于在多驱动信号时解决信号竞争问题。

转换函数，用于从一种数据类型到另一种数据类型的转换。如在元件例化语句中，利用转换函数可允许不同数据类型的信号和端口间，进行映射。
决断函数，用于在多驱动信号时解决信号竞争问题。

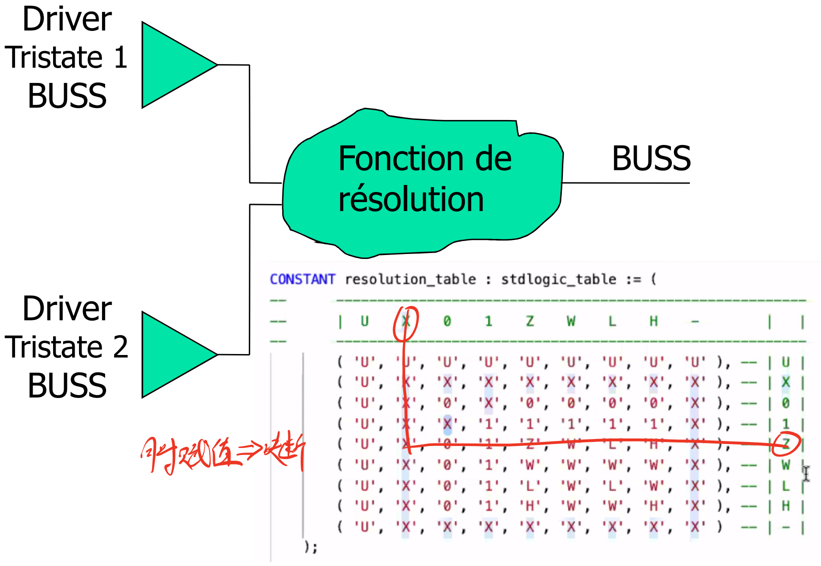
驱动源　决断函数

Drivers et Fonction de Résolution

```
subtype STD_LOGIC is RESOLVED STD_ULOGIC;
```

```
Signal BUSS,ENB1,ENB2,D1,D2: STD_LOGIC;
...
TRISTATE1: process (ENB1,D1)
Begin
if ENB1 = ‘1’ then
          BUSS <= D1;
else
          BUSS <= ‘Z’;
end if;
End process;

TRISTATE2: process (ENB2,D1)
Begin
    if ENB2 = ‘1’ then
       BUSS <= D2;
    else
       BUSS <= ‘Z’;
    end if;
End process;
```

两个进程同时对同一信号赋值
⇒ 决断函数



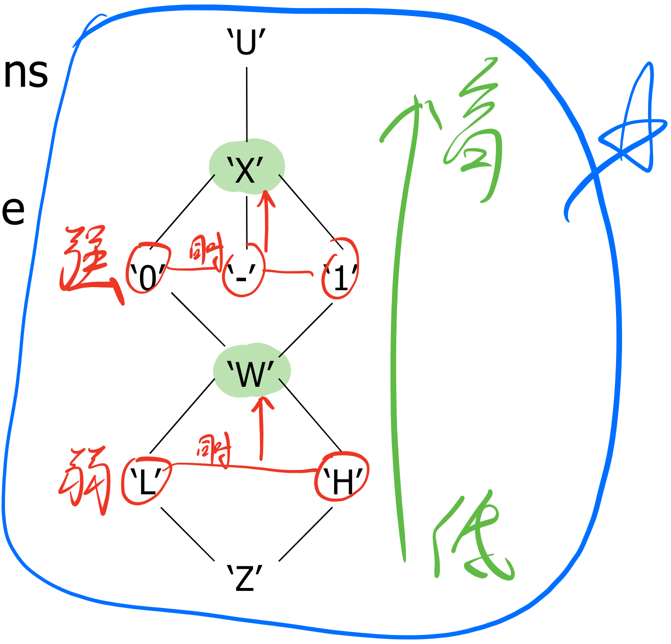
```
CONSTANT resolution_table : stdlogic_table := (
--      ---------------------------------------------------------
--      | U    X    0    1    Z    W    L    H    -         |   |
--      ---------------------------------------------------------
        ( 'U', 'U', 'U', 'U', 'U', 'U', 'U', 'U', 'U' ), -- | U |
        ( 'U', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X' ), -- | X |
        ( 'U', 'X', '0', 'X', '0', '0', '0', '0', 'X' ), -- | 0 |
        ( 'U', 'X', 'X', '1', '1', '1', '1', '1', 'X' ), -- | 1 |
        ( 'U', 'X', '0', '1', 'Z', 'W', 'L', 'H', 'X' ), -- | Z |
        ( 'U', 'X', '0', '1', 'W', 'W', 'W', 'W', 'X' ), -- | W |
        ( 'U', 'X', '0', '1', 'L', 'W', 'L', 'W', 'X' ), -- | L |
        ( 'U', 'X', '0', '1', 'H', 'W', 'W', 'H', 'X' ), -- | H |
        ( 'U', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X' )  -- | - |
    );
```

std_logic的决断

Résolution de STD_LOGIC

- Le type STD_LOGIC est définit dans le package STD_LOGIC_1164
- Les Valeurs les plus hautes dans le schéma sont prioritaires

架构中的较高值优先.



初始值

Valeurs initiales

初始化

- Signaux et variables sont initialisés au début d'une simulation.
- La valeur par défaut est la valeur la plus à gauche du type. 默认值是类型中最左边的值. 即第一个
- Attention : La synthèse ignore les valeurs initiales.

注意:综合时会忽略初始值.

```
type Opcode is (Add, Neg, Load, Store, Jmp, Halt);
signal S: Opcode;
signal CLOCK, RESET: STD_LOGIC;
variable V1: STD_LOGIC_VECTOR(0 to 1);
variable V2: STD_LOGIC_VECTOR(0 to 1):="01";
signal N: Opcode:= Halt;
constant size: INTEGER := 16;
constant ZERO: STD_LOGIC_VECTOR := "0000";
```

Valeur initiale Add

Valeur initiale 'U'

Valeur initiale 'UU'

只在simulation中有效,即仿中.

运算符的内在关系

Relation implicite des opérateurs

- Pour le type STD_LOGIC 关系运算 (比大小)

```
'U' < 'X' < '0' < '1' < 'Z' < 'w' < 'L' < 'H' < '-'
```

- Pour le type STD_LOGIC_VECTOR

```
'0' < "00" < "000" < "001" < "100" < "111" < "1111"
```

低 高

Pour les opérations de comparaison.

算术运算

Opérations arithmétiques

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity ADDER is
   port ( A,B : in STD_LOGIC_VECTOR(7 downto 0);
          SUM : out STD_LOGIC_VECTOR(7 downto 0));
end entity;

architecture A1 of ADDER is
begin
   SUM <= A + B;
end architecture;
```

Erreur: "+" n'est pas définit pour le type STD_LOGIC_VECTOR

Utilisation de NUMERIC_STD

这里改变了端口的类型，变成了无符号数据显示方式。可以进行。

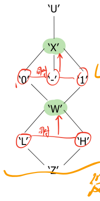
```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.NUMERIC_STD.all;

entity ADDER is
   port (     A,B : in UNSIGNED(7 downto 0);
              SUM : out UNSIGNED(7 downto 0);
end entity;

architecture A1 of ADDER is
begin
   SUM <= A + B;
end architecture;
```

IEEE Std 1076.3

可用的,超载的

Opérateur "+" est surchargé pour les types UNSIGNED et SIGNED

integer



(所以为+仅不支持 std_logic!)

没法加减

问题

Problématique ?

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.NUMERIC_STD.all;

entity ADDER is
   port ( A,B : in STD_LOGIC_VECTOR(7 downto 0);
          SUM : out STD_LOGIC_VECTOR(7 downto 0));
end entity;

architecture A1 of ADDER is
begin


end architecture;
```

Comment faire l'addition de A et B si ils sont de type std_logic_vector ?

设码数（仅数据类型转换）

转换数据类型（除integer之外）

Conversion de type

```
Signal U: UNSIGNED(7 downto 0);
Signal S: SIGNED (7 downto 0);
Signal V: STD_LOGIC_VECTOR(7 downto 0);
```

接近的类型

Conversion entre des types qui sont prochent 接近的类型之间的转换

```
U <= UNSIGNED(S);
S <= SIGNED(U);
U <= UNSIGNED(V);
S <= SIGNED(V);
V <= STD_LOGIC_VECTOR(U);
V <= STD_LOGIC_VECTOR(S);
```

Le nom du type est utilisé pour la conversion de type

类型名称用于类型转换

Solution

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.NUMERIC_STD.all;

entity ADDER is
   port ( A,B : in STD_LOGIC_VECTOR(7 downto 0);
          SUM : out STD_LOGIC_VECTOR(7 downto 0));
end entity;

architecture A1 of ADDER is
begin
 SUM <= STD_LOGIC_VECTOR(UNSIGNED(A) + UNSIGNED(B));
 -- ou SUM <= STD_LOGIC_VECTOR(SIGNED(A) + SIGNED(B));
end architecture;
```

转换函数(与integer相关的)

Fonctions de conversion

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.NUMERIC_STD.all;
```

```
Signal U: UNSIGNED(7 downto 0);
Signal S: SIGNED (7 downto 0);
Signal N: INTEGER;
```

Opérations de conversion

```
N <= TO_INTEGER(U);
N <= TO_INTEGER(S);
U <= TO_UNSIGNED(N,8);
S <= TO_SIGNED(N,8);
```

Taille du vecteur d'arrivée

指定向量

指定位数(位宽)

只有integer整数本身是没有位宽的.

满1位

12345678 1位 10 11 12 ·· 2位

不可直接一次转换

Conversion entre un INTEGER et un STD_LOGIC_VECTOR

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.NUMERIC_STD.all;
```

```
Signal V: STD_LOGIC_VECTOR(7 downto 0);
Signal N: INTEGER;
```

Fonction de conversion

Conversion de type

```
N <= TO_INTEGER(UNSIGNED(V));
```

位宽

```
V <= STD_LOGIC_VECTOR(TO_UNSIGNED(N,8));
```

Conversion de type

Fonction de conversion

例.

整数 整数

```
V <= STD_LOGIC_VECTOR(SIGNED(V)+1);
```

图表摘要

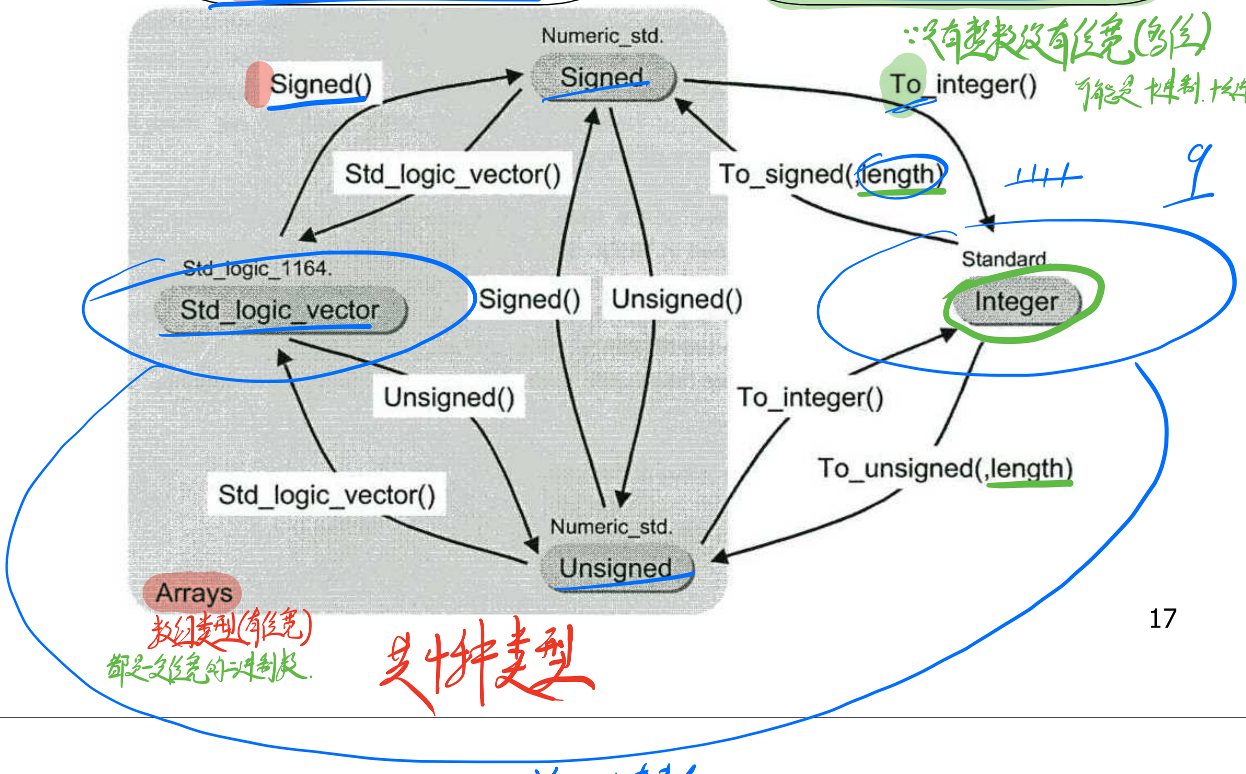
Tableau Récapitulatif

Conversions de types

Fonctions de conversion

∵只有整数没有位宽(多位)

可能是 十进制. 十六进制.



数组类型(有位宽)

都是一定位宽的二进制数.

是4种类型

不可直接一次转化

Sommaire de NUMERIC_STD

```
 +   -   *   /   rem  mod
  <   <=   >   >=   =  /=
~~std_logic~~
UNSIGNED x UNSIGNED
UNSIGNED x NATURAL
NATURAL x UNSIGNED
SIGNED x SIGNED
SIGNED x INTEGER
INTEGER x SIGNED
```

```
sll   srl   rol   ror
位移 opérateur de décalage
UNSIGNED x UNSIGNED
SIGNED x INTEGER
```

```
not   and   or   nand nor xor
xnor
opérateur de logic
UNSIGNED x UNSIGNED
SIGNED x INTEGER
```

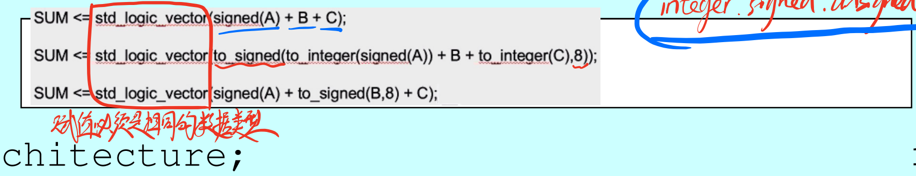
```
TO_INTEGER  [UNSIGNED                 ] return INTEGER
TO_INTEGER  [SIGNED          位数     ] return INTEGER
TO_UNSIGNED [NATURAL,  NATURAL        ] return UNSIGNED
TO_SIGNED   [INTEGER,  NATURAL        ] return SIGNED
RESIZE      [UNSIGNED, NATURAL        ] return UNSIGNED
RESIZE      [SIGNED,   NATURAL        ] return SIGNED
```

Signature (VHDL'93)

Exercice 1 (Addition de A,B et C)

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.NUMERIC_STD.all;

Entity ADDER is
port( A       : in STD_LOGIC_VECTOR(7 downto 0);
      B       : in INTEGER;
      C       : in SIGNED(7 downto 0));
      SUM     : out STD_LOGIC_VECTOR(7 downto 0);
end entity;

Architecture BEHAVIOUR of ADDER is
Begin
SUM <=
```

integer. signed. unsigned之间可共用+号.



```
SUM <= std_logic_vector(signed(A) + B + C);

SUM <= std_logic_vector(to_signed(to_integer(signed(A)) + B + to_integer(C),8));

SUM <= std_logic_vector(signed(A) + to_signed(B,8) + C);
```

赋值必须是相同的数据类型

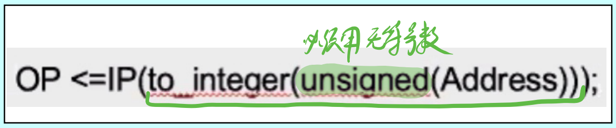
```
End architecture;
```

Exercice 2 (Multiplexeur 8 vers 1)

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.NUMERIC_STD.all;

entity Mux8to1 is
port( Address  :   in  STD_LOGIC_VECTOR(2 downto 0);
      IP       :   in  STD_LOGIC_VECTOR(7 downto 0);
      OP       :   out STD_LOGIC);
end entity;
architecture BEHAVIOUR of Mux8to1 is
begin

OP <=IP(                                 (Address));

end architecture ;
```

一串二进制数

本质上是 std_logic_vector 与 integer 之间的转化



必须用无符号数

```
OP <=IP(to_integer(unsigned(Address)));
```

括号内最终必须是整数

运算符重载

程序包

Package STD_LOGIC_UNSIGNED/SIGNED

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_UNSIGNED.all;
-- use IEEE.STD_LOGIC_SIGNED.all;
```

Ne pas utiliser en même temps

```
+   -

STD_LOGIC_VECTOR
STD_ULOGIC
INTEGER
```

```
*

STD_LOGIC_VECTOR
```

```
<   <=   >   >=   =  /=

STD_LOGIC_VECTOR
INTEGER
```

```
CONV_INTEGER[STD_LOGIC_VECTOR] return INTEGER
```

Un package propriétaire de Synopsis qui est devenu un stantard.
Avantage: surcharge directe du type STD_LOGIC_VECTOR
Inconvénient: ne surcharge pas tout les opérateurs,
Ce package définit une conversion entre le type STD_LOGIC_VECTOR vers INTEGER, mais pour les autres fonctions de type il faut utiliser le type STD_LOGIC_ARITH.

Package STD_LOGIC_ARITH

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_ARITH.all;
```

```
+   -
STD_ULOGIC
UNSIGNED
SIGNED
INTEGER
```

```
*
UNSIGNED
SIGNED
```

```
<   <=   >   >=   =  /=
UNSIGNED
SIGNED
INTEGER
```

```
CONV_INTEGER[INTEGER/UNSIGNED/SIGNED/STD_ULOGIC] return INTEGER
CONV_UNSIGNED[INTEGER/UNSIGNED/SIGNED/STD_ULOGIC, INTEGER] return UNSIGNED
CONV_SIGNED[INTEGER/UNSIGNED/SIGNED/STD_ULOGIC, INTEGER] return SIGNED
CONV_STD_LOGIC_VECTOR[INTEGER/UNSIGNED/SIGNED/STD_ULOGIC, INTEGER] return
STD_LOGIC_VECTOR
EXT[STD_LOGIC_VECTOR, INTEGER] return STD_LOGIC_VECTOR
SXT[STD_LOGIC_VECTOR, INTEGER] return STD_LOGIC_VECTOR
```

Exercice

- Faire l’exercice du C7