SN360/SN361 Introduction à la conception des circuits numériques

Module « Systèmes matériels et logiciels »

Crédit SN360 : 4/4 / SN361 : 2/6

Vincent Beroulle
Bureau : D202
vincent.beroulle@esisar.grenoble-inp.fr

GRENOBLE INP UGA Esisar VALENCE

UGA Université Grenoble Alpes

Plan

II Représentation des nombres en binaire
Codage binaire des entiers positifs

- Le code binaire pour les **entiers positifs** :
  - N bits => 2^n valeurs possibles => de zéro **à 2^n-1**.
  - 8 bits => 2^8 = 256 valeurs => de 0 à 255.

$$X = \sum_{n=0}^{N-1} x_n 2^n$$

  - Exemple de décomposition dans la base 2 :

$$01011010 = 2^7*0+ 2^6*1 + 2^5*0 + 2^4*1 + 2^3*1 + 2^2*0 + 2^1*1 + 2^0*0$$
$$= 128*0 + 64*1 + 32*0 + 16*1 + 8*1 + 4*0 + 2*1 + 1*0$$
$$= 64 + 16 + 8 + 2$$
$$= 90$$

1
2
4
8
16
32
64
128
256
512
1024
2048

Valeur max : 2^n-1 (non signé)

Autres puissances de 2 à connaitre : 2^{10}=1024, 2^{20}=1048576

II Représentation des nombres en binaire

Codage binaire type et objet

VHDL

```
-- type integer
variable i : integer;
i:=10;
-- type bit, par défaut à 0
signal s1, s2 : bit ;
 s1<='0'; s2<='1';
-- type bit_vector
variable v : bit_vector(3 downto 0);
v:="0101";
```

Verilog

```
// wire par défaut à Z
wire w1 w2 w3;
assign w1=1'b0;
assign w2=1'b1;
assign w3=1'bX;

wire [3:0] v1 v2;
assign v1=4'b0110;
assign v2=4'd15;
```

II Représentation des nombres en binaire

Codage binaire des entiers positifs

VHDL

-- type std_logic_vector, par défaut à U (non initialisé)

library IEEE;

use IEEE.std_logic_1164.all;

signal s **:** **std_logic_vector**(5 **downto** 0);

s**<=**"U01XZ-";

-- 9 valeurs différentes possibles (ces valeurs + différentes "forces" utilisées à bas niveau)

Verilog

wire [3:0] v1;

assign v1=4'**b**X01Z;

// d'autres valeurs sont possibles, dont différentes forces (le "don't care" '-' est fusionné dans le X qui représente les valeurs inconnues)

// à X quand non initialisé

II Représentation des nombres en binaire

Hexadécimal

Hexadécimal (Hex) : base 16

- Chiffres : 0-9, A-F
- A-F représentent 10-15
- **Compacte :** Chaque chiffre hexadécimal correspond à quatre chiffres binaires
- Exemple :

7562(décimal)=

1_1101_1000_1010(binaire)

= 1D8A (hexadécimal)

Q: Write 11110000 in hex

F 0

Q: Convert hex A01 to binary

1010 0000 0001

II Représentation des nombres en binaire

Conversion décimal binaire hexadécimal

VHDL

variable v **:**
bit_vector(15 **downto** 0);

v:=x"1D9A";

Verilog

wire [15:0] v1;

assign v1=16**h'**1D9A;

II Représentation des nombres en binaire

Conversion décimal binaire hexadécimal

VHDL

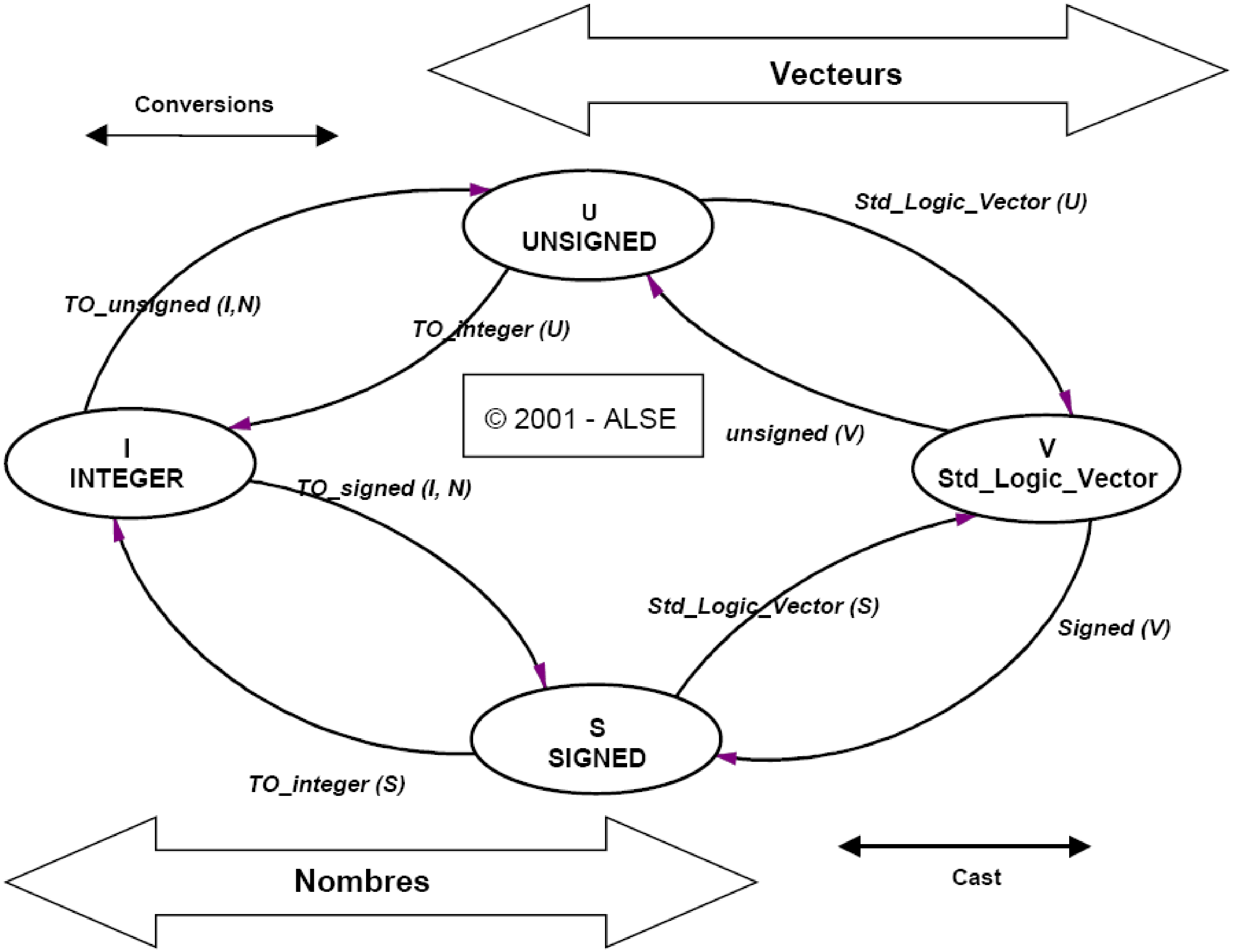
```
-- fonctions de conversion
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
signal s1: integer;
signal s2: std_logic_vector(15 downto 0);
s2<=std_logic_vector(to_unsigned(s1,16));
--et dans l'autre sens
s1<=to_integer(unsigned(s2));
```

Verilog

```
// conversion décimal / binaire
wire [15:0] s1 s2;
 assign s2=16'd7562;
 assign s1=s2;
 //ou par exemple
 assign s1=16'h1D9A
 assign s2=s1;
```

II Représentation des nombres en binaire

Package VHDL – IEEE.numeric_std



II Représentation des nombres en binaire

bit et byte - kilo et kibi

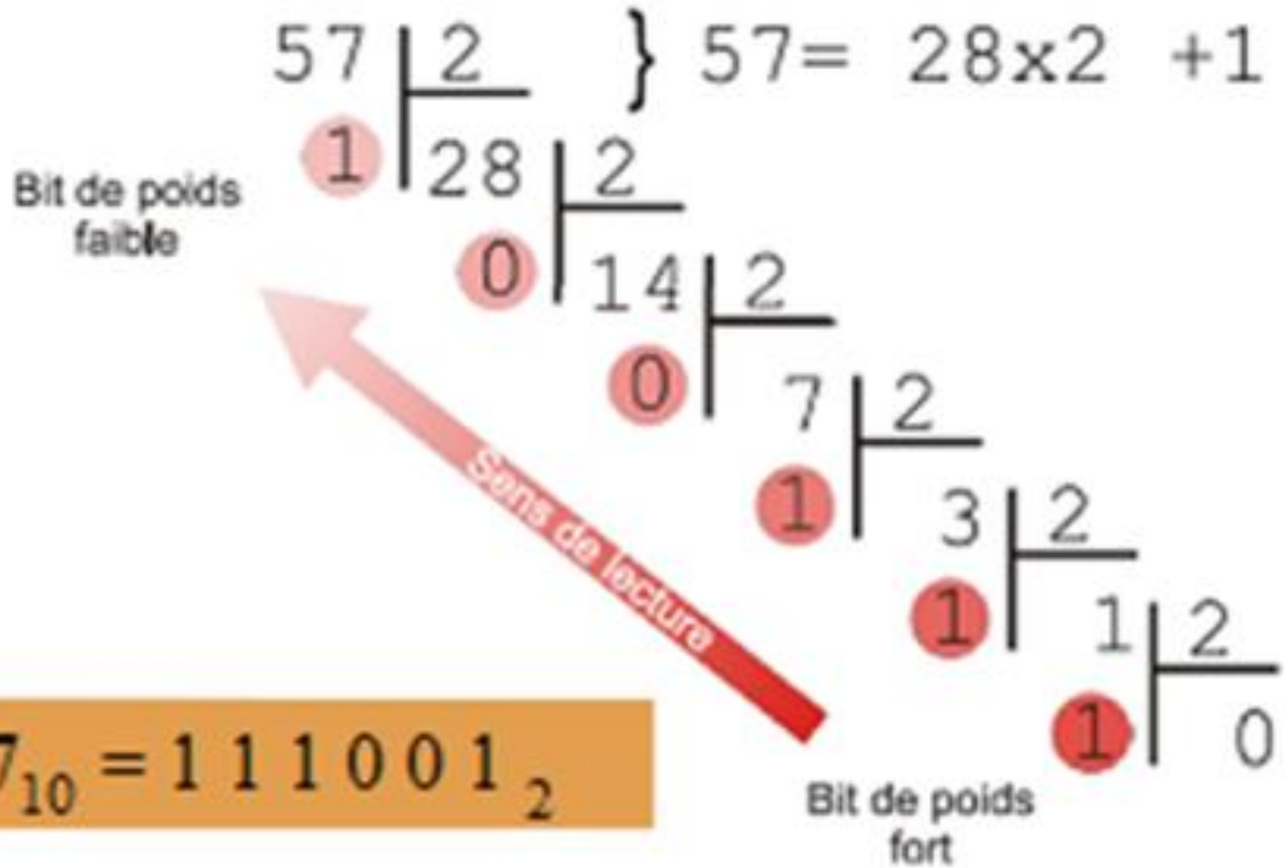
- octet (ou byte en anglais) : mot de 8 bits
- “k**B**” pour kilo**byte** et “kb” pour kilo**bit**
- Kilo (k), Méga (M), Giga (G), Téra (T) :
  - 2^{10} = 1024 => 1 kibibit = 1 Kibit ≈ 1 kilobits = 1 kbit =1000 bits
  - 2^{20}=1048576 => 1 mebibit = 1 Mibit ≈ 1 Mégabits = 1 Mbit = 10^6 bits

Préfixe	Symbole	Facteur
kibi	Ki	$2^{10} = 1\,024$
mébi	Mi	$2^{20} = 1\,048\,576$
gibi	Gi	$2^{30} = 1\,073\,741\,824$
tébi	Ti	$2^{40} = 1\,099\,511\,627\,776$

Mais cette convention n’est pas très utilisée

II Représentation des nombres en binaire

Codage binaire – méthode itérative de conversion



$$57_{10} = 1\,1\,1\,0\,0\,1_{\,2}$$

II Représentation des nombres en binaire

Nombre de bits nécessaires pour les entiers positives

Utiliser la formule : **$n=\lceil \log2(x+1) \rceil$**

Avec :

- n : nombre de bits nécessaires
- $\lceil \cdot \rceil$: **fonction d'arrondi à l'entier supérieur**
- log2 : logarithme en base 2
- x : nombre entier positif à coder

• Exemple :

Pour coder 7 en binaire,
on a besoin de 3 bits : $\lceil \log2(7+1) \rceil = \lceil \log2(8) \rceil = 3$ bits

(pour des valeurs négatives, voir la suite, en complément à 2
il faudra ajouter +1 à la valeur n obtenu)

II Représentation des nombres en binaire
Exercices

- Convertir les nombres binaires positifs en leur équivalent décimal / hexadécimal
  - 00011001 ⬄ ⬄
  - 00010010 ⬄ ⬄
  - 100110011010 ⬄ ⬄
  - ⬄ 23 ⬄
  - ⬄ 42 ⬄
- Quel nombre maximal peut-on atteindre avec 10 bits ? Combien de bits faut-il pour compter jusqu’à 511 ?

II Représentation des nombres en binaire

Codage binaire – entier négatif : **complément à 2**

- **Code complément à 2 :** pour les entiers **positifs et négatifs**
  - Passage d'un nombre vers son opposé :
    - **On complémente chaque bit puis on ajoute 1**
  - **Avantage : addition et soustraction des nombres en complément à 2 (<u>mais pas la multiplication</u>) se font comme en décimal**
  - Par exemple :
    - 5 en binaire = '0101'.
    - On obtient son opposé en inversant les '0' et les '1' puis en ajoutant 1.
    - On obtient -5 ⬄ '1011'

II Représentation des nombres en binaire

Codage binaire - Complément à 2

- Remarques :
  - le bit de poids fort (*Most Significant Bit* ou MSB) donne le signe.
  - Avec N bits on code donc de $\underbrace{\textbf{-2}^{\textbf{N-1}} \textbf{ à } \textbf{2}^{\textbf{N-1}} \textbf{ – 1}}_{\textbf{échelle asymétrique!}}$

$$X = \begin{cases} \sum_{n=0}^{N-2} x_n 2^n & X \geq 0 \\ -2^{N-1} + \sum_{n=0}^{N-2} x_n 2^n & X < 0. \end{cases}$$

Par exemple, « 1011 » vaut $-2^3+2^1+2^0=-8+2+1=-5$

Valeurs max/min : [-2^{N-1} :2^{N-1}-1]
(signé)

II Représentation des nombres en binaire
Complément à 2

VHDL

-- types non signé et signé

library IEEE;

use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

signal s1: **unsigned**(3 **downto** 0);

signal s2: **signed**(3 **downto** 0);

s1**<=to_unsigned**(15,4);

s2**<=to_signed**(-8,4);

Verilog

/* par défaut les wires sont non signés, si on veut un nombre signé en complément à 2 on doit ajouter **signed */**

wire [3:0] s1;

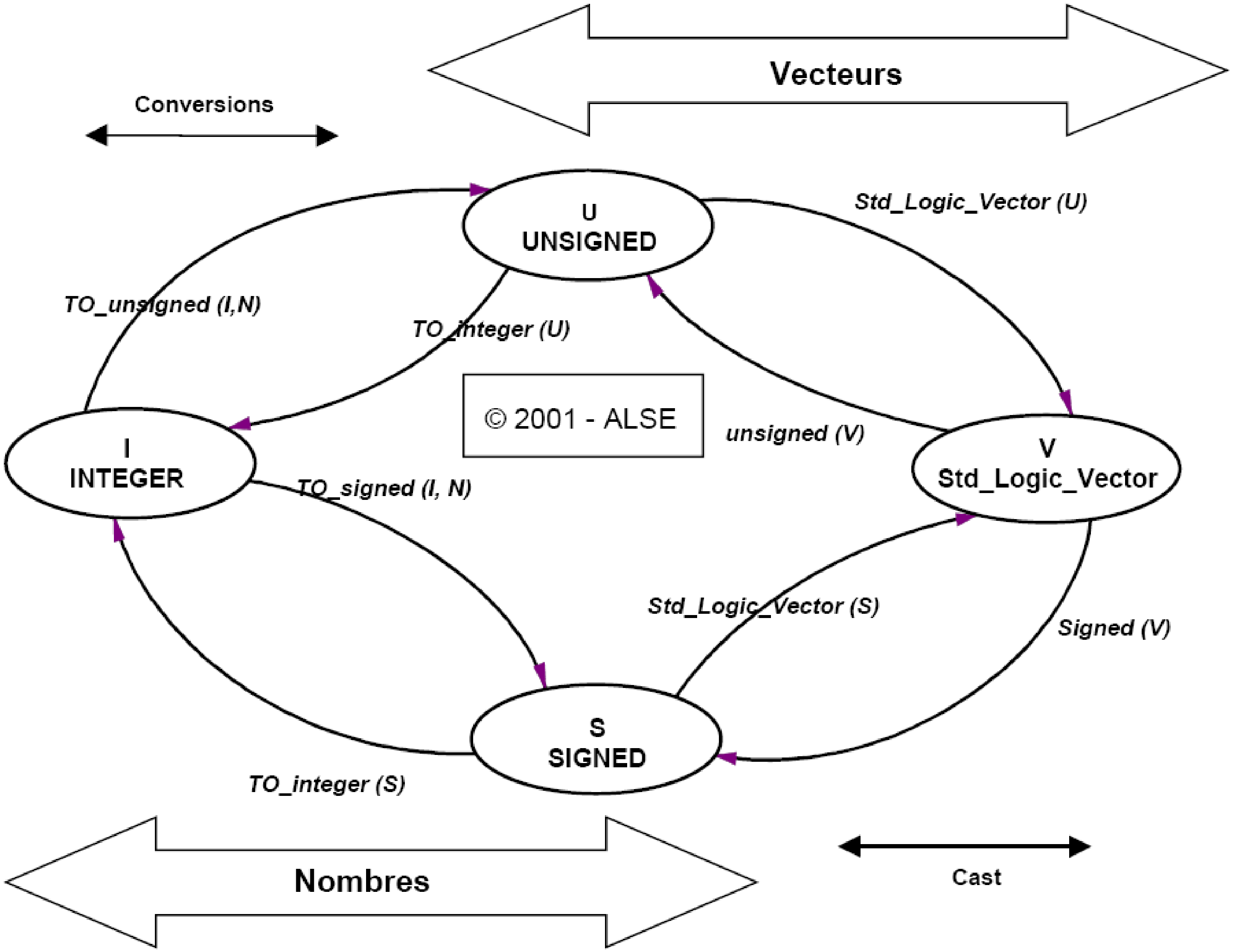
wire signed [3:0] s2;

assign s1**=**4**'b**1111; //15

assign s2**=4'b**1000; //-8

II Représentation des nombres en binaire

Package VHDL – IEEE.numeric_std



II Représentation des nombres en binaire
Exercices

- Convertir les nombres binaires en complément à 2 suivants en leur équivalent décimal ou inversement.
  - 11001 ⇔
  - 10010 ⇔
  - 10011001101 ⇔
  - ⇔ -23
  - ⇔ -42
- Quels nombres maximums positifs et négatifs peut-on atteindre avec 10 bits signé ? Combien de bits faut-il pour compter jusqu'à 512 en complément à 2 ? Et -512?

Faire le QCM : QCM2 SN360/SN361 Représentation des nombres en binaire