ANNEXES :

```
entity cell2 is
port(  i0, i1 : in std_logic ;
       out0 : out std_logic) ;
end entity;

entity cell3 is
port(  i0, i1, i2 : in std_logic ;
       out0 : out std_logic) ;
end entity;
```

Circuit 1:

```
architecture circuit1 of cell3 is
begin
 process (i1)
 begin
       if rising'edge(i1) then
             if (i0 = '1') then
                   out0 <= '0';
             else
                   out0 <= i2;
             end if;
       end if
 end process;
end architecture circuit1;
```

Circuit 2:

```
architecture circuit2 of cell2  is
begin
 process (i0, i1)
 begin
       if (i0= '1') then
             out0 <= i1;
       end if;
 end process;
end architecture;
```

Circuit 3:

```
architecture circuit3 of cell3 is
begin
 process (i0, i1,i2)
 begin
       if (i0 = '1') then
             out0 <= i1;
       else
             out0 <= i2;
       end if;
 end process;
end architecture;
```

Circuit 4:

```
 architecture circuit4 of cell3 is
       signal reg: std_logic_vector(7 downto 0);
 begin
       process(i1)
       begin
             if Rising_edge(i1) then
                   reg(7) <= i2;
                   reg(6 downto 0) <= reg(7 downto 1);
                   out0 <= reg(0);
             end if;
       end process;
 end architecture;
```

Circuit 5 :

```
architecture V1 of LOGIC is
begin
      A <= not D after 2 NS;
      F <= C xor D after 4 NS;
      C <= A xor B after 4 NS;
end architecture V1;
```

Circuit 6 : Additionneur

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.NUMERIC_STD.all;
entity ADDER is
      port  (A : in SIGNED(8 downto 0);
             B : in INTEGER;
             C : in STD_LOGIC_VECTOR(8 downto 0);
             SUM : out STD_LOGIC_VECTOR(8 downto 0));
end entity;

architecture BEHAVIOUR of ADDER is
begin

end architecture;
```

Circuit 7 : Multiplexeur 64 vers 1

```
entity MUX64V1 is
      port  (D : out STD_LOGIC;
             SEL : in STD_LOGIC_VECTOR(5 downto 0);
             Y : in STD_LOGIC_VECTOR(63 downto 0));
end entity;

architecture RTL of MUX64V1 is
begin


end architecture;
```

Circuit 8 : Démultiplexeur 1 vers 32

```
entity DEMUX1V32 is
      port  (D : in STD_LOGIC;
             SEL : in STD_LOGIC_VECTOR(4 downto 0);
             Y : out STD_LOGIC_VECTOR(31 downto 0));
end entity;

architecture RTL of DEMUX1V32 is
begin


end architecture;
```

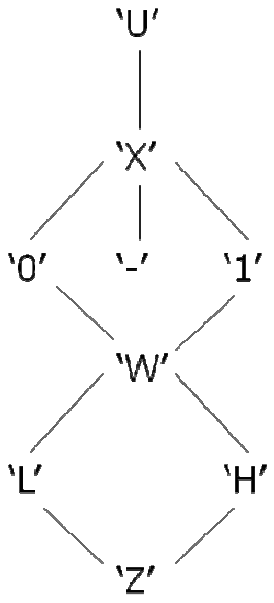
Circuit 9 : conversion de type et fonction de conversion

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

Exemples :

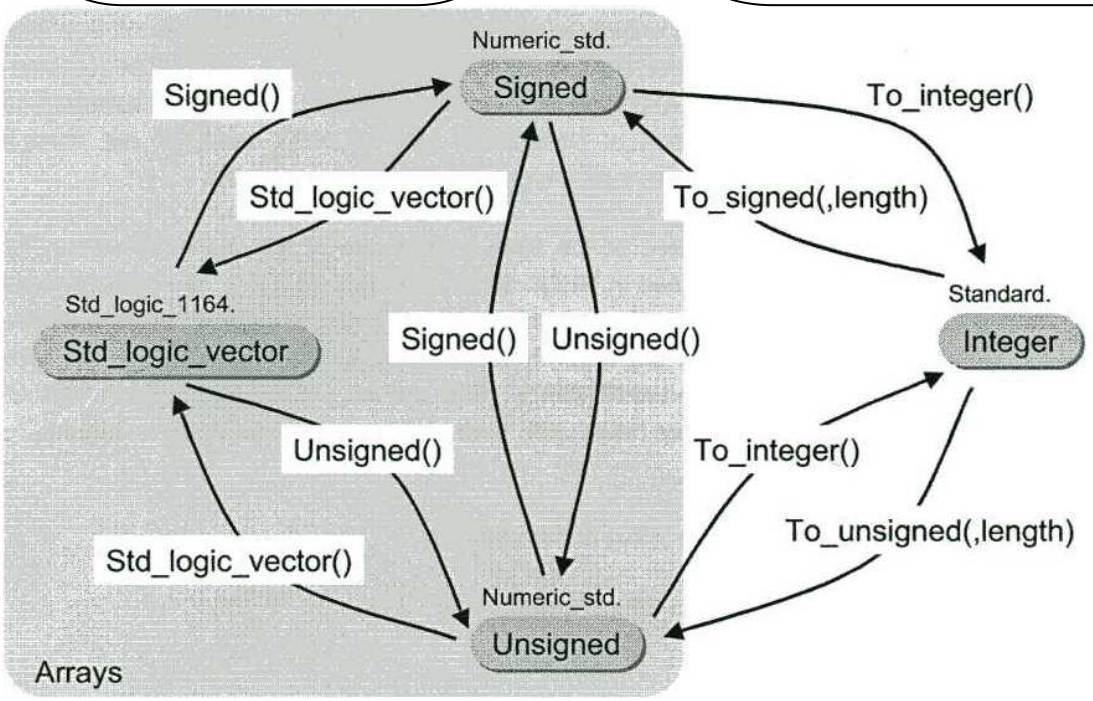
```
U <= UNSIGNED(V);     -- conversion STD_LOGIC_VECTOR → UNSIGNED
N <= TO_INTEGER(S);   -- conversion SIGNED → INTEGER
```

Fonction de résolution du type std_logic :



Conversions de types

Fonctions de conversion



Librairie NUMERIC_STD :

```
+  -  *  /  rem  mod
<  <=  >  >=  = /=

UNSIGNED x UNSIGNED
UNSIGNED x NATURAL
NATURAL x UNSIGNED
SIGNED x SIGNED
SIGNED x INTEGER
INTEGER x SIGNED
```

```
sll  srl  rol  ror

UNSIGNED x UNSIGNED
SIGNED x INTEGER
```

```
not  and  or  nand nor xor
xnor

UNSIGNED x UNSIGNED
SIGNED x INTEGER
```

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