Université de Bretagne Occidentale
Département d'Informatique
Master 2 LSE

Adaptation et Ouverture

Introduction au langage VHDL

Ahcène Bounceur

2015

Plan du cours

1. Introduction au langage
2. Prise en main
3. Machine à états
4. Implémentation sur FPGA

Partie 1

INTRODUCTION AU LANGAGE

VHDL

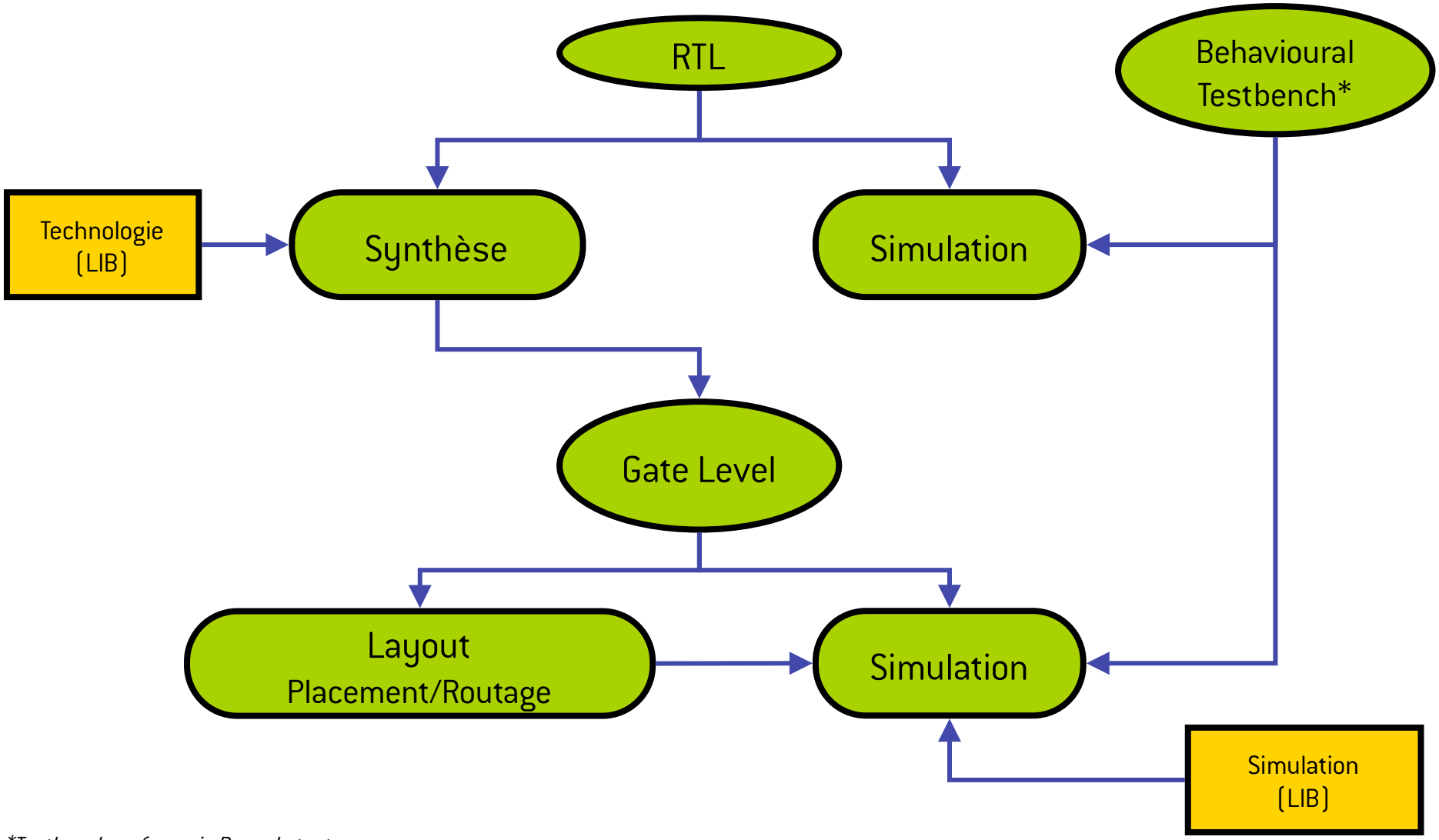
- VHIC (Very High-speed Integration Circuit)
- Hardware
- Description
- Language

1987 normalisé IEEE
1993 revu
1998 revu à nouveau
1998 IEEE std 1076

VHDL

- Conception/simulation de circuits ASIC/FPGA et test
- Développé aux USA dans les années 80, à l'initiative du département de la défense (initiative VHSIC)
- Syntaxe dérivée du langage ADA
- Extension : VHDL-AMS (VHDL-Analog end Mixed Systems).

Flot de conception



*Testbench en français Banc de test

L'En-tête

```
Library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
```

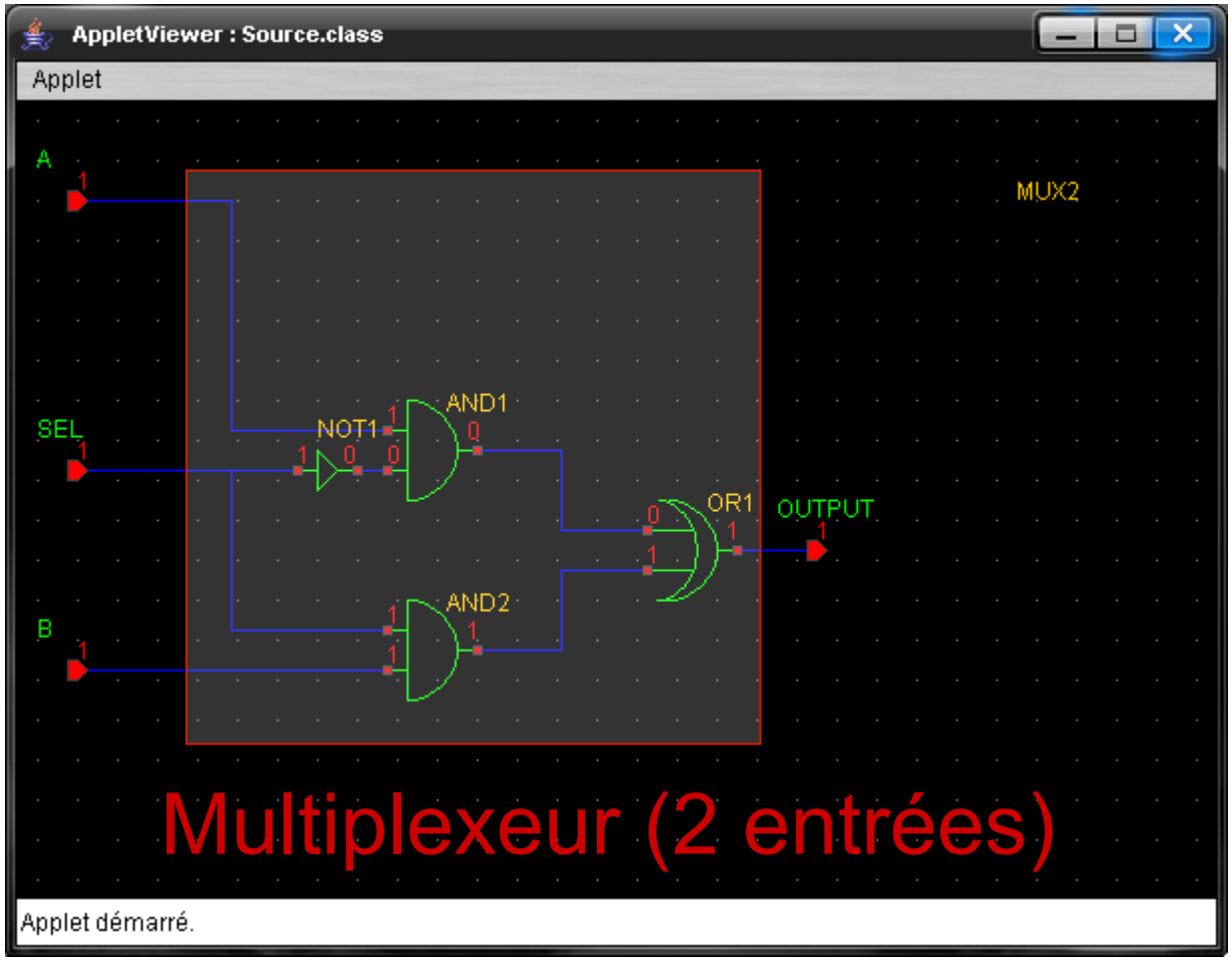
L'En-tête

```
entity DEMIADD is
port (
      A,B : in std_logic;
      R : in std_logic_vector(3 downto 0);
      SOMME, RETEN : out std_logic;
);
end  DEMIADD;
```

L'Architecture

```
architecture ARCHI of DEMIADD is
-- déclarations préalables
begin
    SOMME <= A xor B;
    RETEN <= A and B;
end ARCHI;
```

Le Process


AppletViewer : Source.class
Applet
A
SEL
B
MUX2
NOT1
AND1
AND2
OR1
OUTPUT
Multiplexeur (2 entrées)
Applet démarré.

Le Process

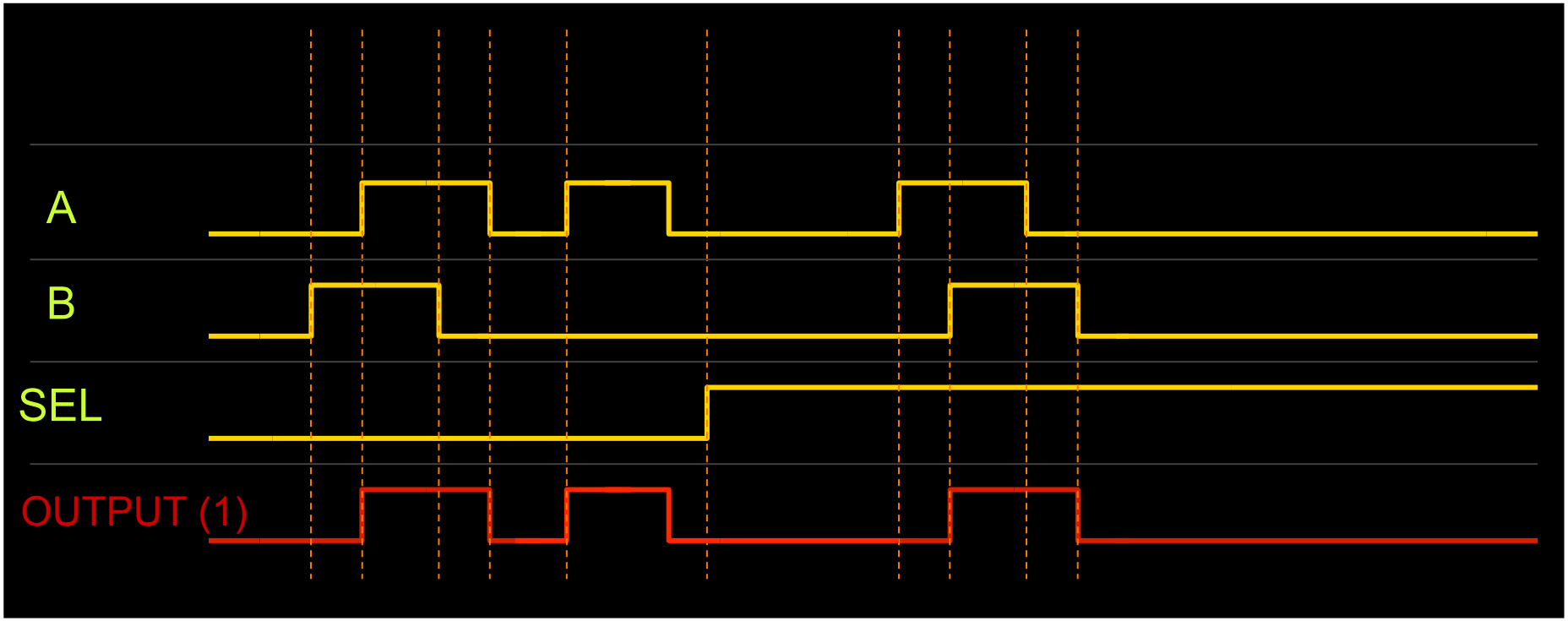
```
Library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;

entity MUX is
port (
          SEL,A,B : in std_logic;
          OUTPUT : out std_logic
);
end  MUX;

architecture ARCHI of MUX is
begin
   MUXPROCESS : process(A,B,SEL)
   begin
        if SEL='1' then
              OUTPUT <= B;
        else
              OUTPUT <= A;
        end if;
   end process MUXPROCESS;
end ARCHI;
```

Le Process

```
Library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;

entity MUX is
port (
          SEL,A,B : in std_logic;
          OUTPUT : out std_logic
);
end  MUX;

architecture A of MUX is
begin
   MUXPROCESS : process(A,B,SEL)
   begin
        if SEL='1' then
              OUTPUT <= B;
        else
              OUTPUT <= A;
        end if;
   end process MUXPROCESS;
end A;
```

Le Process

```
MUXPROCESS : process(A,B,SEL)
```

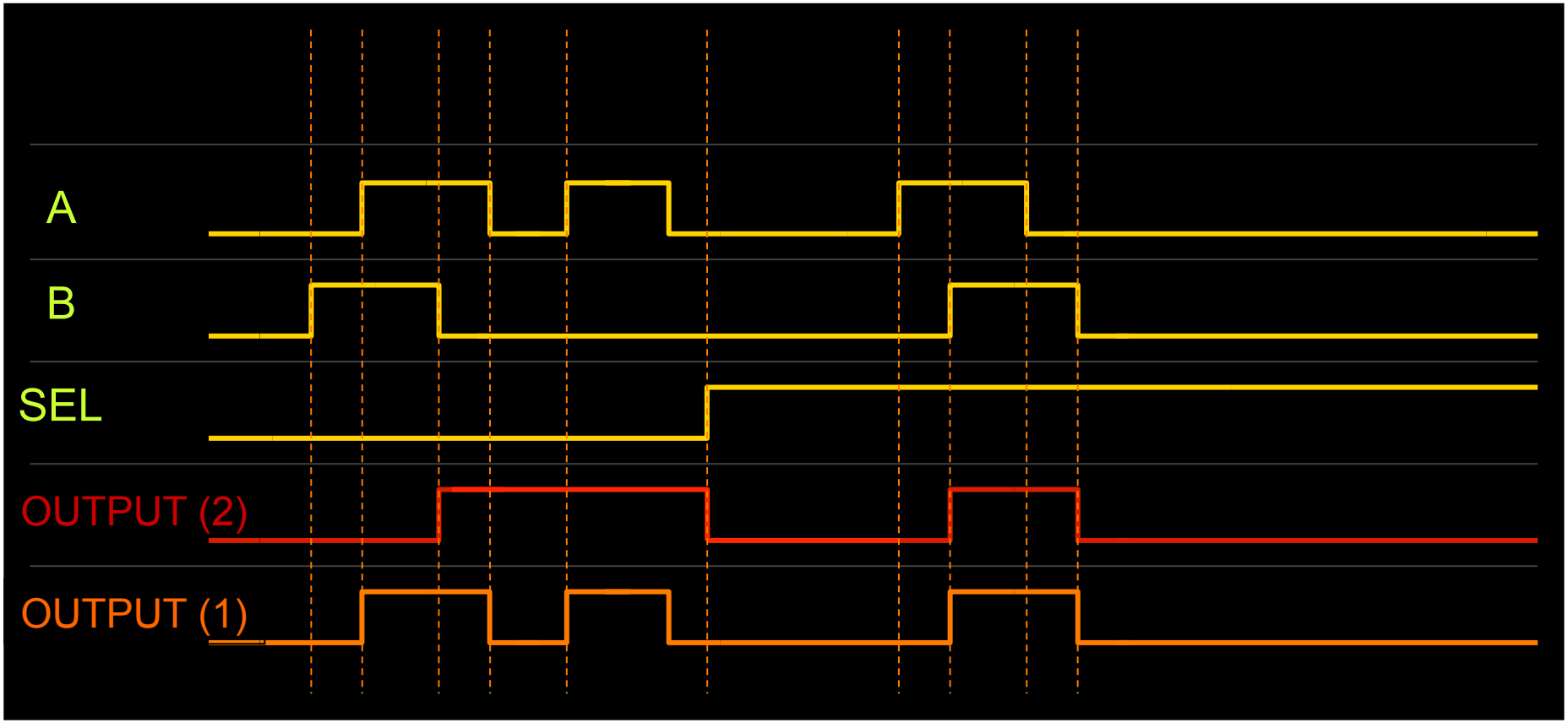


Le Process

```
Library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;

entity MUX is
port (
          SEL,A,B : in std_logic;
          OUTPUT : out std_logic
);
end  MUX;

architecture A of MUX is
begin
   MUXPROCESS : process(A,B,SEL)
   begin
        if SEL='1' then
              OUTPUT <= B;
        else
              OUTPUT <= A;
        end if;
   end process MUXPROCESS;
end A;
```

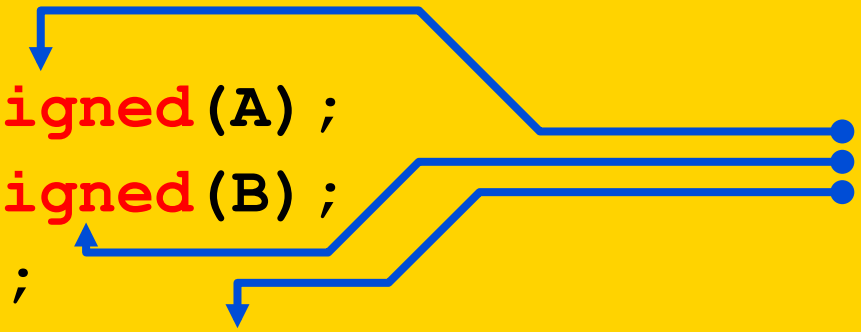
Le Process

```
MUXPROCESS : process(B,SEL)
```



Librairie std_logic_arith

```
architecture archi of my_entity is
signal A,B,C : std_logic_vector(5 downto 0);
signal D,E,F : unsigned(5 downto 0)
begin
  D <= unsigned(A);
  E <= unsigned(B);
  F <= D+E;
  C <= std_logic_vector(F);
end;
```



| + | - | * | / | = |

| < | <= | > | >= | = | /= |

Agrégats et Concaténation

```
architecture archi of my_entity is
signal A,B,C : std_logic;
signal Z : std_logic_vector(1 downto 0)
signal S : std_logic_vector(7 downto 0)
Begin
  Z <= (A,B);
  S <= (7=>'1' , 5 downto 4=>Z , 6=>C , others=> '0');
  S <= '1' & C & Z & "0000";
  S(7) <= '1';
  S(6) <= C;
  S(5 downto 4) <= Z;
  S(3 downto 0) <= "0000";
end;
```

Attributs des signaux et Vecteurs

```
signal A : std_logic_vector(5 downto 0);
```

- **A'high** = 5
- **A'low** = 0
- **A'left** = 5
- **A'right** = 0
- **A'range** = 5 downto 0
- **A'reverse_range** = 0 to 5
- **A'length** = 6
- **A'ascending** = false

Attributs des signaux et Vecteurs

```
signal X : std_logic;
```

X'event = true quand X change de valeur

Règles d'écriture

- Un signal est défini par son type et sa taille
- Tout objet doit être déclaré avant d'être utilisé
- Les noms de signaux, les labels, ... doivent respecter les mêmes règles que les variables en langage soft (C/C++ et Java) et pas d'underscore (_) à la fin
- Le VHDL est insensible à la casse : abc, Abc ou ABC représentent le même nom
- Utiliser -- pour introduire un commentaire et pas de commentaire sur plusieurs lignes
- Ne jamais utiliser X <= X + Y; si X et Y ne figurent pas dans la liste des signaux de la liste de sensibilité

L'écriture séquentielle et parallèle

if

```
if condition then
  -- séquence d'opérations
end if;
```

```
if condition then
  -- séquence 1 d'opérations
else
  -- séquence 2 d'opérations
end if;
```

L'écriture séquentielle et parallèle

if

```
if condition1 then
  -- séquence 1 d'opérations
elseif condition2 then
  -- séquence 2 d'opérations
elseif condition3 then
  -- séquence 3 d'opérations
else
  -- séquence n d'opérations
end if;
```

L'écriture séquentielle et parallèle

case

```
case expression is
when valeur1 =>
  -- séquence 1 d'opérations
when valeur2 to valeur5 =>
  -- séquence 2 d'opérations
when valeur6 | valeur8 =>
  -- séquence 3 d'opérations
when others =>
  -- séquence 4 d'opérations
end case;
```

L'écriture séquentielle et parallèle

for

```
signal X,Y : std_logic_vector(3 downto 0);
--
for I in 0 to 3 loop
  Y(I) <= IP(3-I);
end loop;
```

Assignation Concurrente

```
process(A,B,C,T)
begin
   if T>5 then
      Y <= A;
   elseif T<5 then
      Y <= B;
   else
      Y <= C;
   end if;
end process;
```

if

dans un process

hors process

```
Y <= A when T>5 else B when T<5 else C;
```

Assignation Concurrente

```
process(A,B,C,T)
begin
   case T is
   when 0 to 4 =>
      Y <= B;
   when 5 =>
      Y <= C;
   when others =>
      Y <= A;
   end case;
end process;
```

case

dans un process

hors process

```
with T select
Y <= B when 0 to 4, C when 5, A when others;
```

Variables & Signaux

variables

```
…
signal A,B,P :
  integer;
begin
  process(A,B)
  variable vM,vN : integer
  begin
      vM := A;
      vN := B;
      P <= vM + vN;
  end process;
end ARCHI;
```

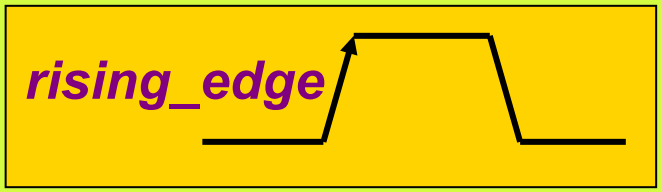
signaux

```
…
signal A,B,P : integer;
signal sM,sN : integer;
begin
  process(A,B,sM,sN)
  begin
      sM <= A;
      sN <= B;
      P <= sM + sN;
  end process;
end ARCHI;
```

Process CLOCKé

```
Library IEEE;
…
entity …
architecture …
signal CLK,RESET :std_logic;
signal count : unsigned(3 downto 0);
begin
process(CLK)
begin
   if CLK'event and CLK='1' then
        if RESET='1' then
                count <= "0000";
        elsif count>=9 then
                count <= "0000";
        else
                count <= count + 1;
        end if;
   end if;
end process;
end architecture;
```

- Machine à état SYNCHRONE
- Process réagissant sur front d'horloge

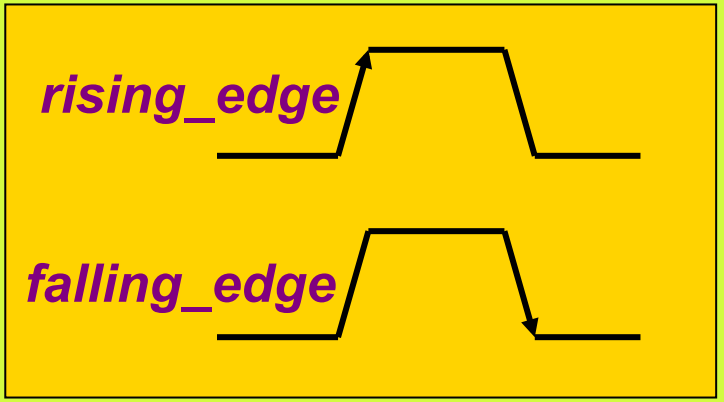



```
if rising_edge(CLK)  then
```

Process CLOCKé

Éviter les if (pour la synthèse)

```
Library IEEE;
…
entity …
architecture …
signal CLK,RESET :std_logic;
signal count : unsigned(3 downto 0);
begin
process(CLK)
begin
   wait until CLK'event and CLK='1';
        if RESET='1' then
                count <= "0000";
        elsif count>=9 then
                count <= "0000";
        else
                count <= count + 1;
        end if;

end process;
end architecture;
```




```
wait until rising_edge(CLK);
```

Reset Asynchrone

```
begin
process(CLK,RESET)
Begin
   if RESET='1' then
        count <= "0000";
   elsif (CLK'event and CLK='1') then
        …
   end if;
end process;
```

```
begin
process(CLK,RESET)
Begin
   if RESET='1' then
        count <= "0000";
   end if;
   wait until CLK'event and CLK='1';
        …
end process;
```

Quelques Remarques

- Ne pas considérer le cas `rising_edge` et `falling_edge` au même temps
- Faire attention de bien écrire un process synchrone

```
wait until CLK'event and (CLK='1' or CLK='0');
if CLK'event and (CLK='1' or CLK='0') then
```

```
wait until CLK'event
if CLK'event then
```

```
wait until CLK='1';
if CLK='1' then
```

Partie 2

PRISE EN MAIN

Les Fonctions combinatoires

Description par une table de vérité

ABCD	F
0000	0
0001	0
0010	0
0011	1
0100	0
0101	0
0110	0
0111	0
1000	1
1001	1
1010	1
1011	1
1100	1
1101	1
1110	1
1111	0

Description par une équation logique

$$F=A\overline{B}\overline{C}+A\overline{C}D+A\overline{B}D+AB\overline{D}+\overline{A}BCD+\overline{A}B\overline{C}\overline{D}$$

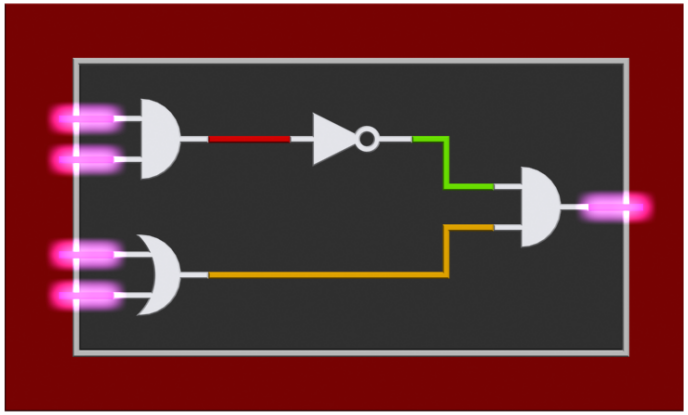
Description et simplification par une table de Karnaugh

AB/CD	00	01	11	10
00			1	
01				
11	1	1		1
10	1	1	1	1

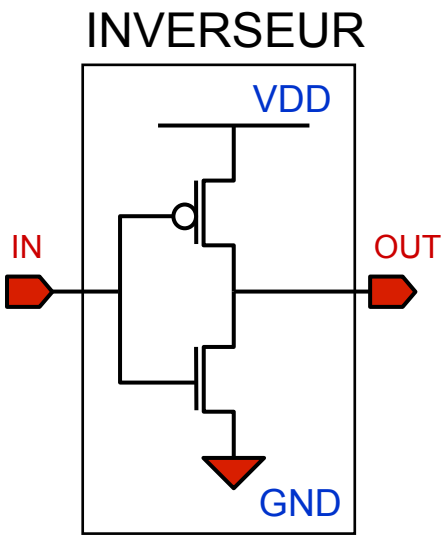
$$F=\overline{A}CD+B\overline{C}D$$

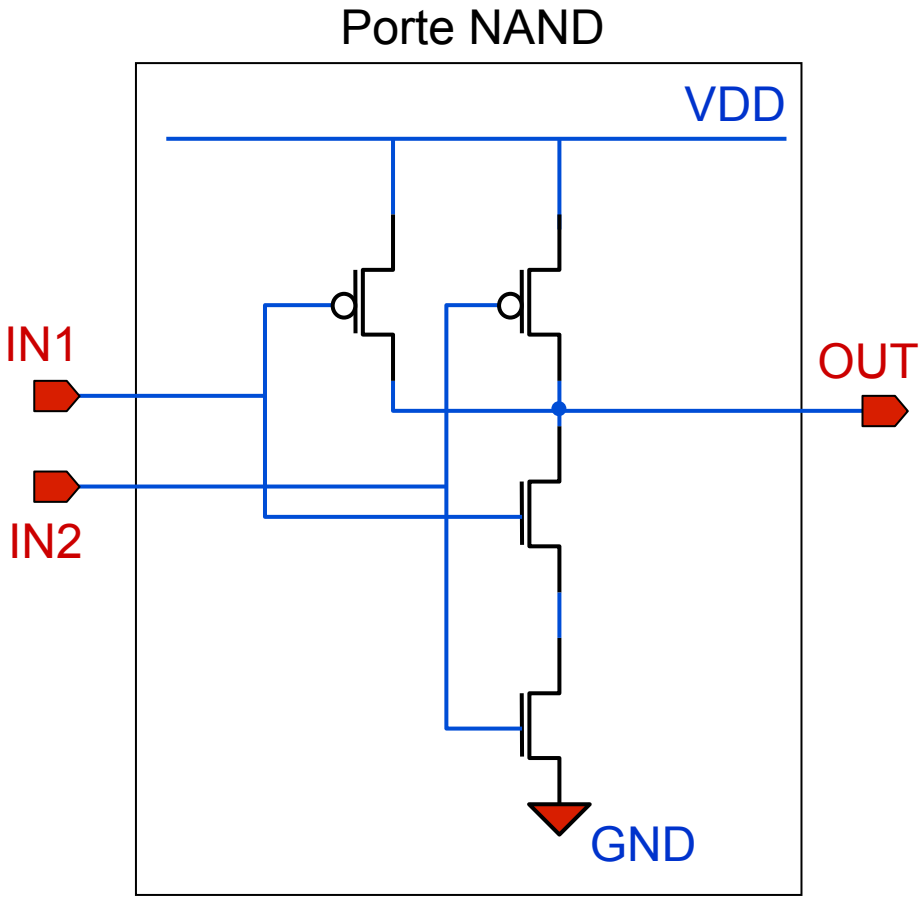
Les Fonctions combinatoires

Description logique en portes



Description niveau transistor

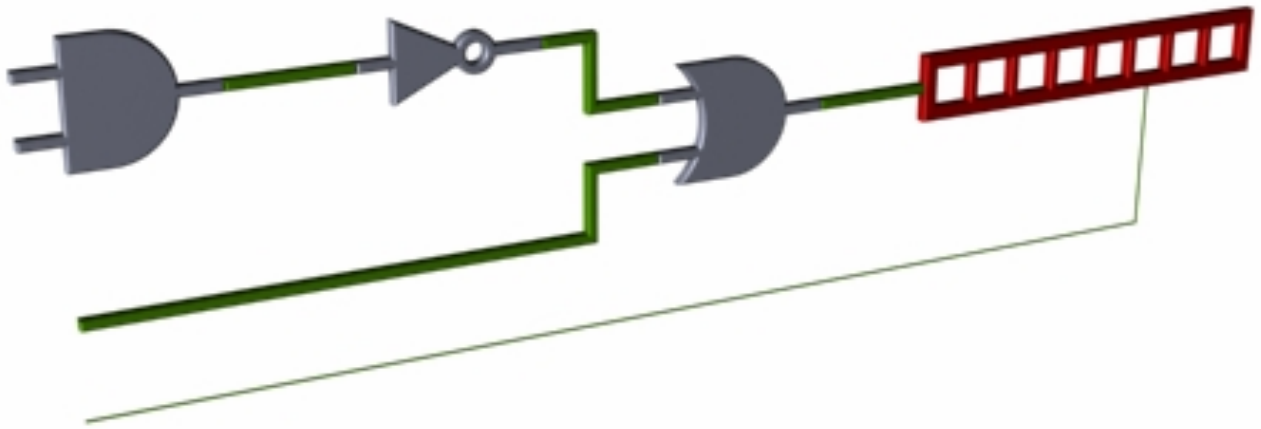


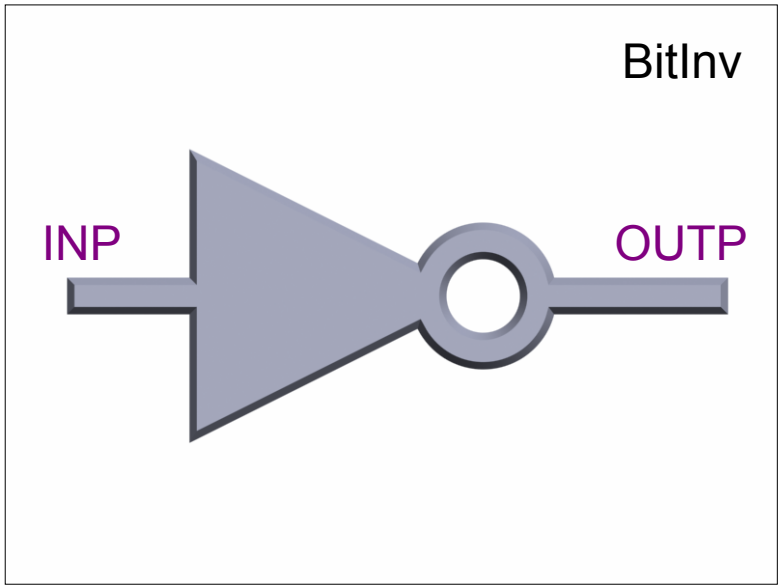


Description d'un circuit

- Code VHDL-AMS/Verilog-A
- Synthèse
- DFT
- Placement/Routage
- Implémentation FPGA/ASIC

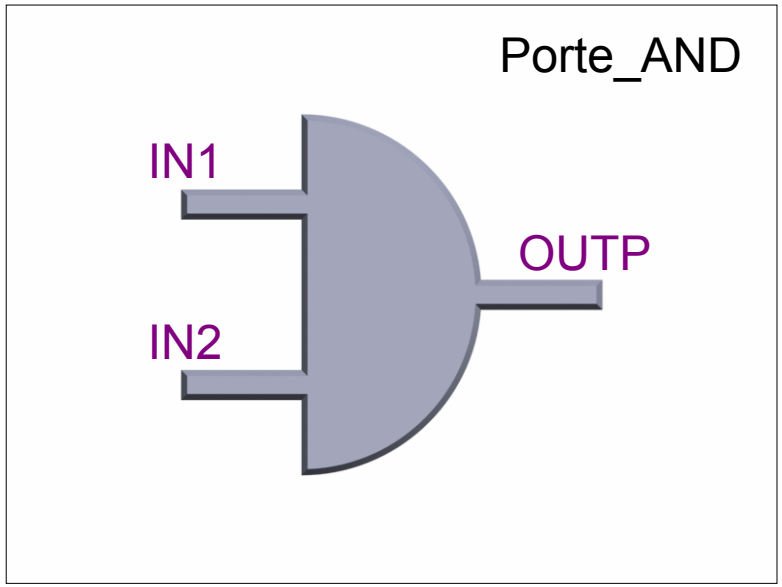
Circuit à décrire

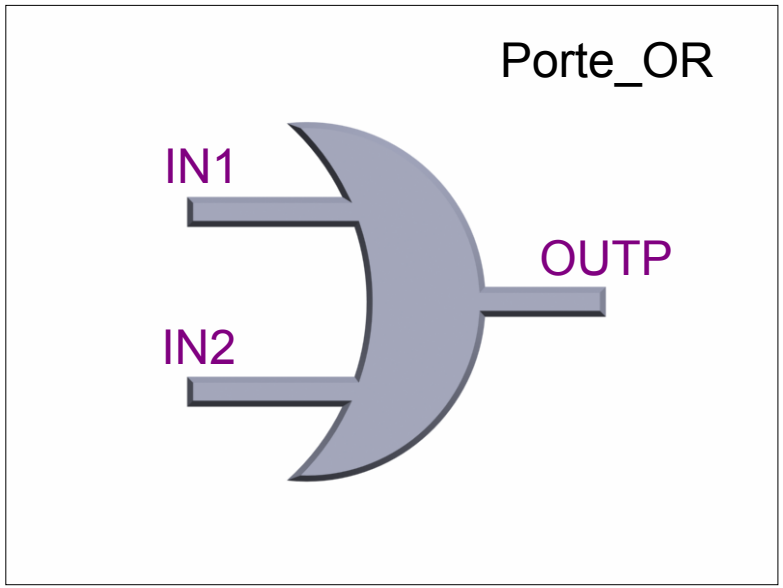




Entités

ENTREES
SORTIES



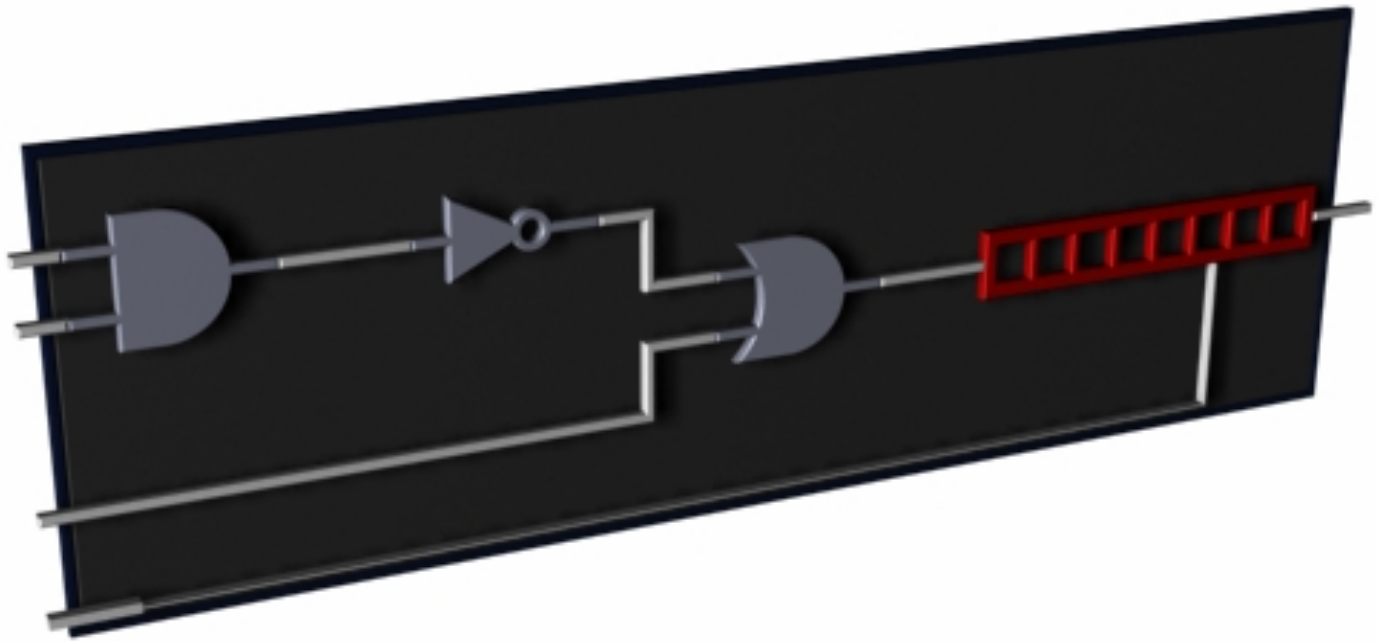


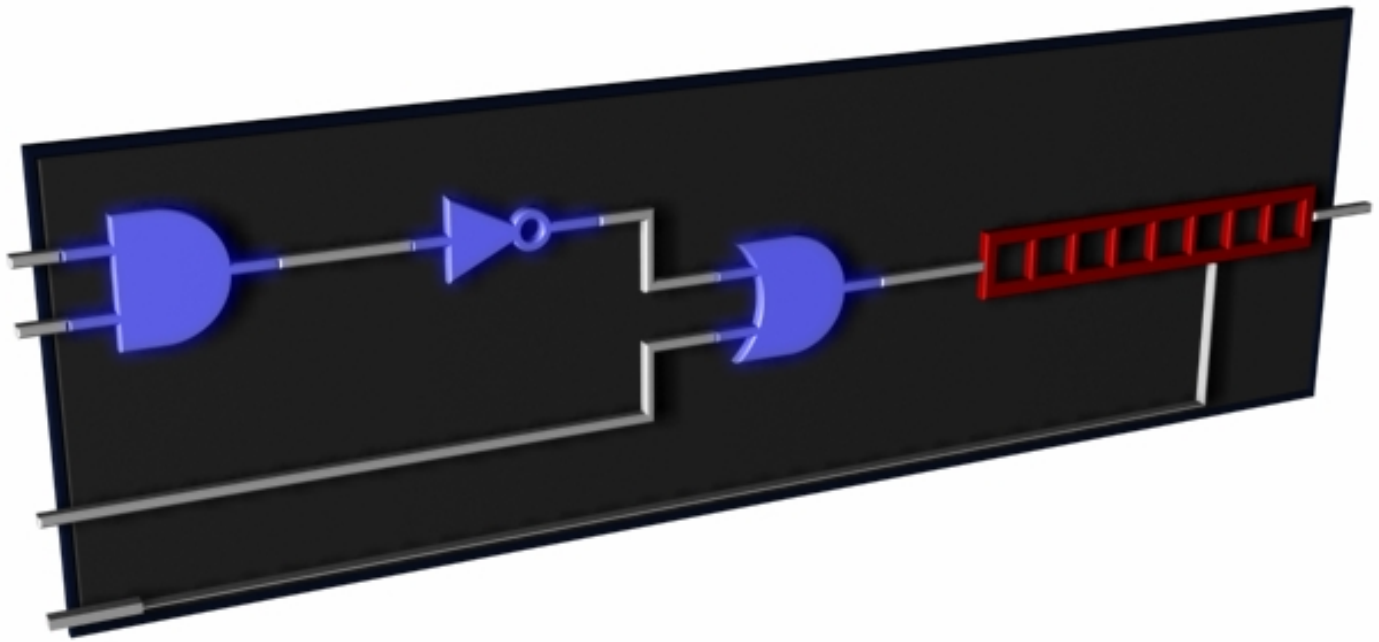
Entité

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

entity Exple is
        port (
                CLK             : in STD_LOGIC;
                RESET           : in STD_LOGIC;
                INPUT           : in STD_LOGIC_VECTOR(3 downto 0);
                OUTPUT          : out STD_LOGIC_VECTOR(3 downto 0);
        );
end Exple;

architecture A of Exple is
begin
        P_Exple : process
        begin
                if RESET='1' then
                        OUTPUT <= "0000";
                end if;
                wait until CLK='1' and CLK'EVENT;
                OUTPUT <= INPUT;
        end process P_Exple;
end A;
```

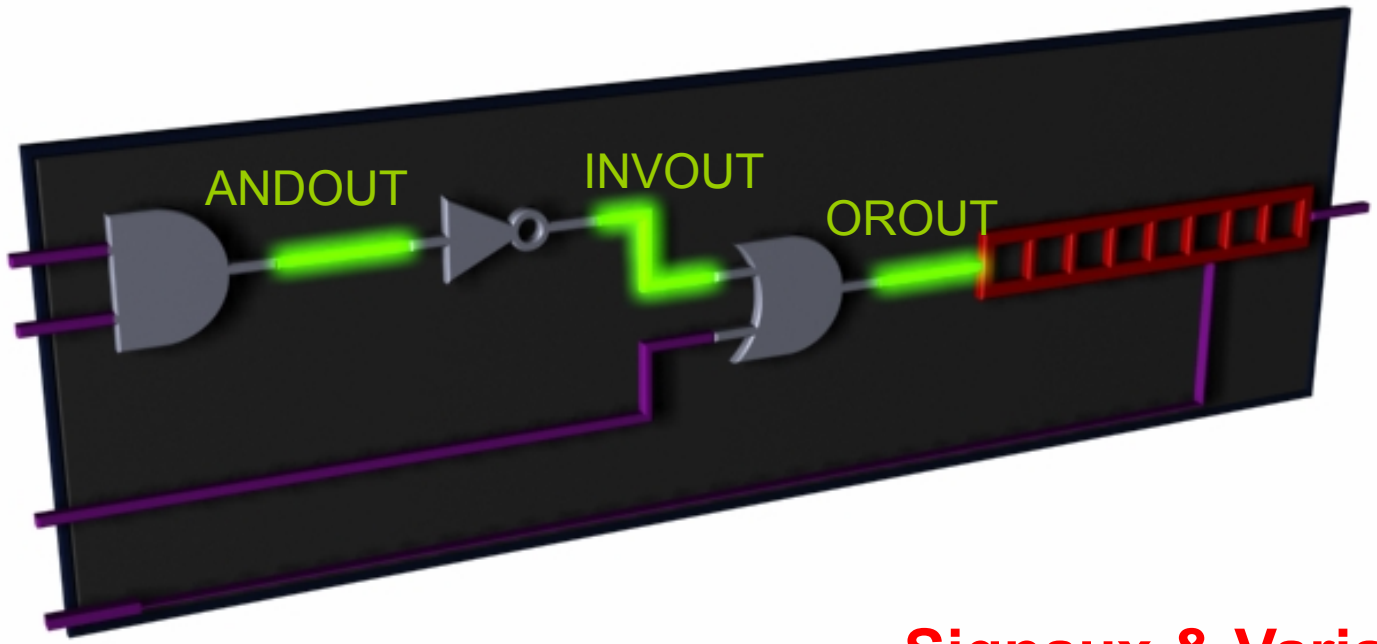
Entité

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

entity Porte is
        port (
        INPUT   : in STD_LOGIC_VECTOR(3 downto 0);
        OUTPUT  : out STD_LOGIC_VECTOR(3 downto 0);
        );
end Porte;

architecture A of Porte is
begin
        P_Porte : process
        begin
                OUTPUT <= INPUT;
        end process P_Porte;
end A;

architecture B of Porte is
begin
        P_Porte : process
        begin
                OUTPUT <= INPUT or '0';
        end process P_Porte;
end B;
```





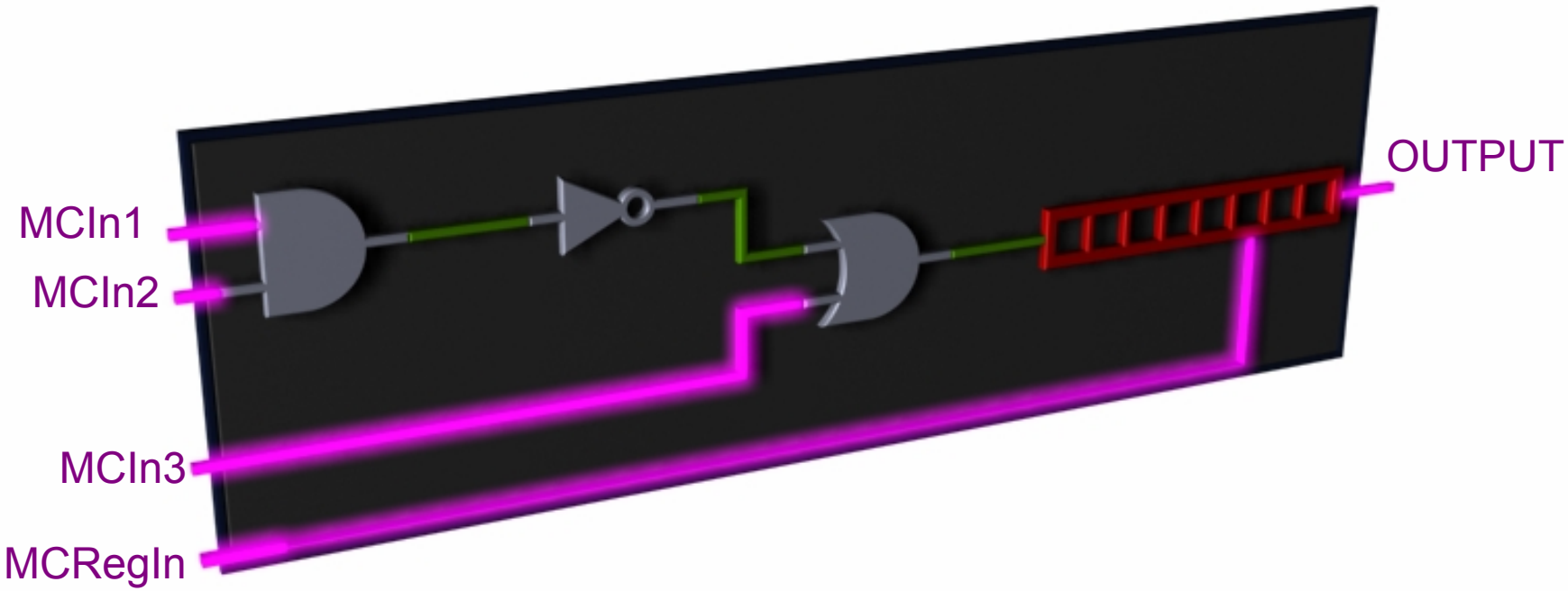
Signaux (internes)

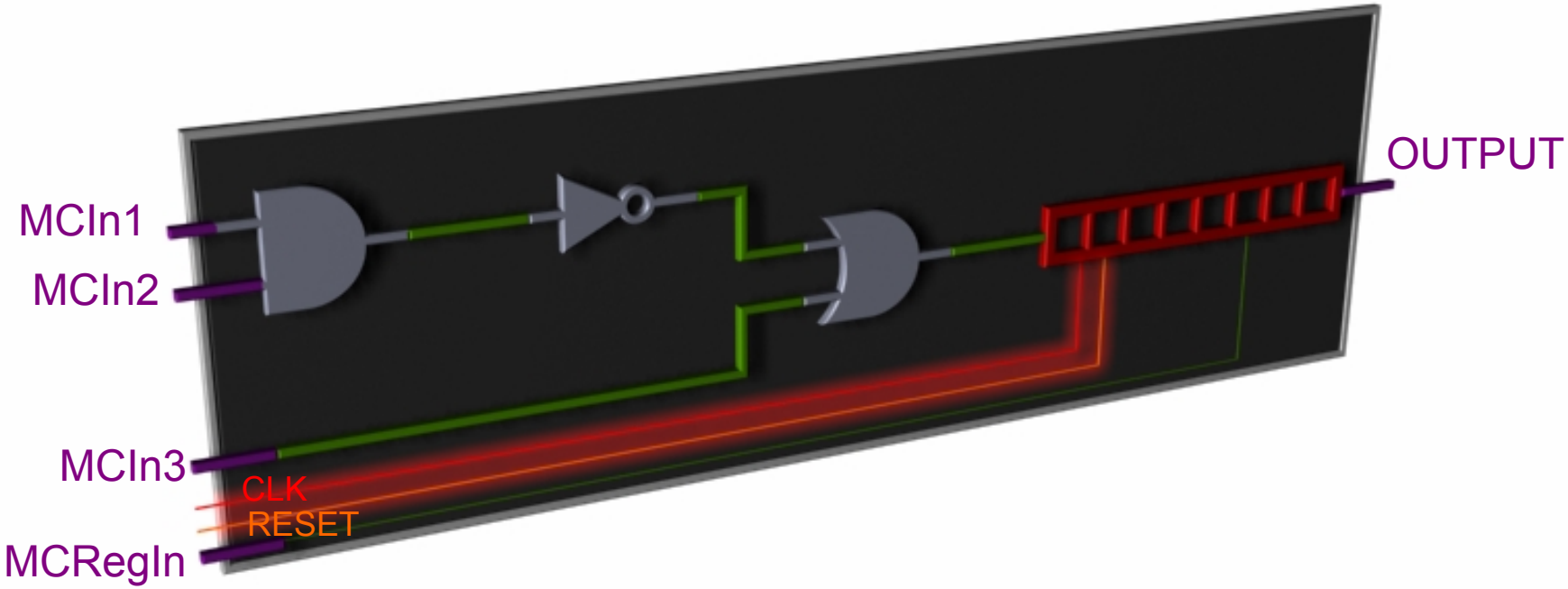


Signaux & Variables

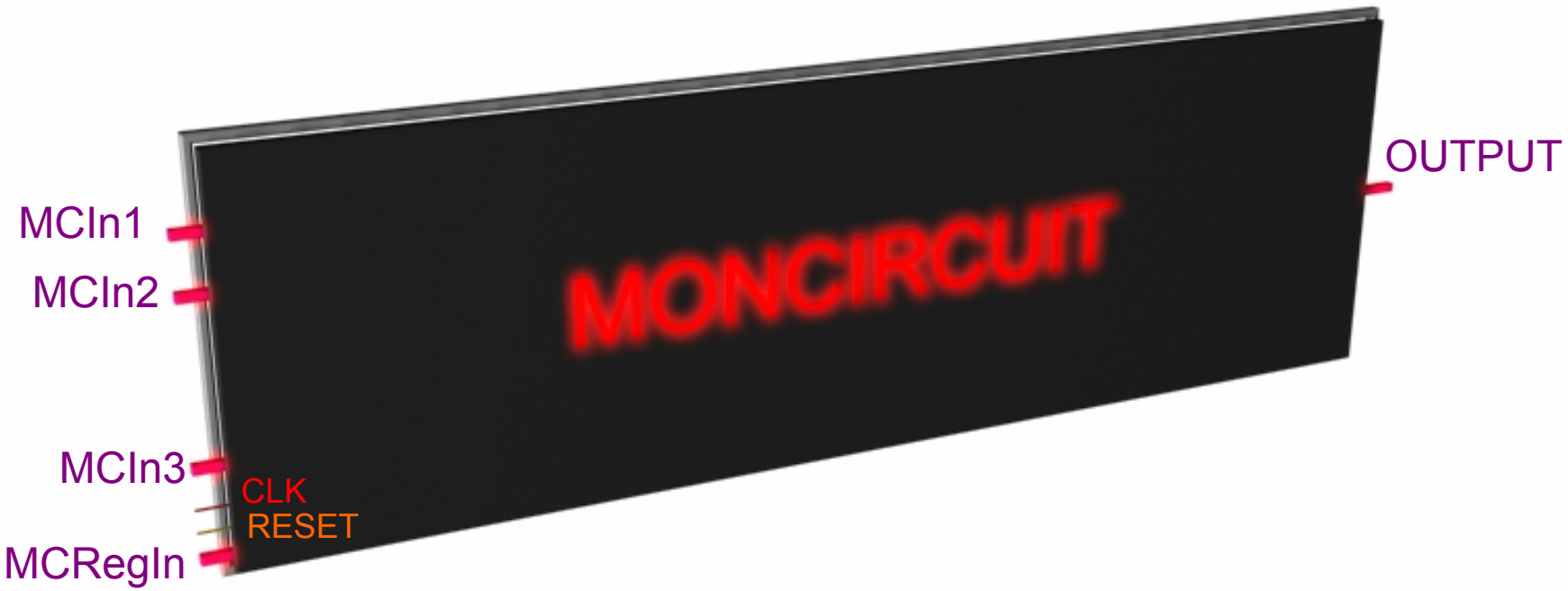
```
Sig <= "0000"
Var := "0000"
```

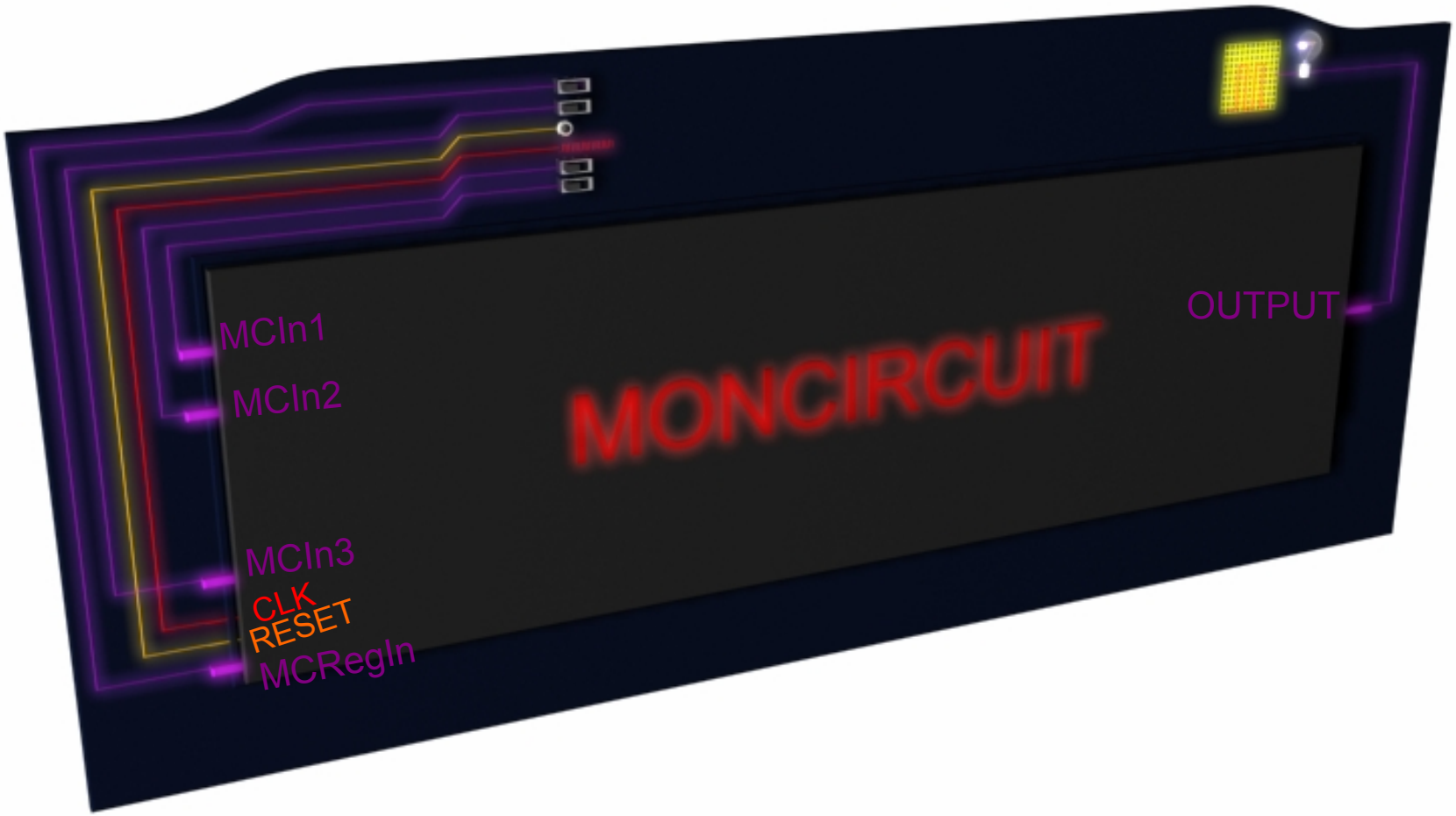
Entrées/Sorties




MCIn1
MCIn2
MCIn3
CLK
RESET
MCRegIn
OUTPUT

MONCIRCUIT





Simulation : Bench

Porte AND

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

entity Porte_AND is
port (IN1          : in STD_LOGIC ;
      IN2          : in STD_LOGIC ;
      OUTP         : out STD_LOGIC);
end Porte_AND;

architecture A of Porte_AND is
begin
      OUTP <= IN1 and IN2;
end A;
```

Porte OR

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

entity Porte_OR is
port (IN1          : in STD_LOGIC ;
      IN2          : in STD_LOGIC ;
      OUTP         : out STD_LOGIC);
end Porte_OR;

architecture A of Porte_OR is
begin
      OUTP <= IN1 or IN2;
end A;
```

Inverseur

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

entity BitInv is
     port (   INP       : in STD_LOGIC ;
          OUTP      : out STD_LOGIC
     );
end BitInv;

architecture A of BitInv is
begin
     OUTP <= not INP;
end A;
```

Registre

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;

entity Registre is
          port (
                    RegIn                : in STD_LOGIC ;
                    RegShift             : in STD_LOGIC ;
                    RESET                : in STD_LOGIC ;
                    CLK                  : in STD_LOGIC ;
                    RegOut               : out STD_LOGIC
          );
end Registre;

architecture ARCHI of Registre is
type RegVecteur is array (0 to 7) of STD_LOGIC ;
signal x : RegVecteur ;

begin
          Process_Registre : process(CLK)
          begin
                    if (CLK'event and CLK='1') then
                       if (RESET = '1') then
                              for i in x'range loop
                                 x(i) <= 'X';
                              end loop ;
                       elsif (RegShift = '1') then
                              x(0) <= RegIn;
                              for i in x'low to (x'high - 1) loop
                                 x(i+1) <= x(i);
                              end loop;
                       end if;
                    end if;
          end process Process_Registre;
          RegOut <= x(7);
end ARCHI;
```

Circuit

```
library IEEE ;
use IEEE.std_logic_1164.ALL ;

entity MONCIRCUIT is
          port(
                    -- Entrées/Sorties du circuit
          ) ;
end MONCIRCUIT;

architecture A of MONCIRCUIT is
-- Déclaration de composants utilisés
component Porte_AND
          port (    IN1                 : in STD_LOGIC ;
                    IN2                 : in STD_LOGIC ;
                    OUTP                : out STD_LOGIC
          );
end component;
…
-- Signaux Internes
signal ANDOUT       : STD_LOGIC ;
signal INVOUT       : STD_LOGIC ;
signal OROUT        : STD_LOGIC ;
-- Les composants instanciés et leurs connections
begin

U1:Porte_AND port map (
 INPUT1   => MCIn1,
 INPUT2   => MCIn2,
 OUTPUT   => ANDOUT
);
…

end A;
```

Bench

```
library IEEE ;
use IEEE.std_logic_1164.ALL ;
use IEEE.STD_LOGIC_ARITH.ALL;

entity bench is
end bench;

architecture A of bench is
component moncircuit
        port(                   MCIn1                   : in STD_LOGIC;
                        MCIn2                   : in STD_LOGIC;
                        MCIn3                   : in STD_LOGIC;
                        MCRegIn                 : in STD_LOGIC;
                        CLK                     : in STD_LOGIC ;
                        RESET                   : in STD_LOGIC ;
                    MCOut                               : out STD_LOGIC ) ;
end component;

signal              sigINPUT1 : STD_LOGIC;
signal              sigINPUT2 : STD_LOGIC;
signal              sigINPUT3 : STD_LOGIC;
signal              sigRegIn  : STD_LOGIC;
signal              sigCLK      : STD_LOGIC := '0';
signal              sigRESET    : STD_LOGIC;
signal              sigOUTPUT : STD_LOGIC;

Begin
U1:MONCIRCUIT port map(sigINPUT1,sigINPUT2,sigINPUT3,sigRegIn,sigCLK,sigRESET,sigOUTPUT);

sigCLK <= not(sigCLK) after 100 ns;
sigRESET <= '1', '0' after 150 ns;
sigINPUT1 <= '0','0' after 250 ns;
sigINPUT2 <= '0','0' after 250 ns;
sigINPUT3 <= '0','0' after 250 ns;
sigRegIn <= '0','1' after 250 ns, '0' after 450 ns;
end;
```

Configuration du bench

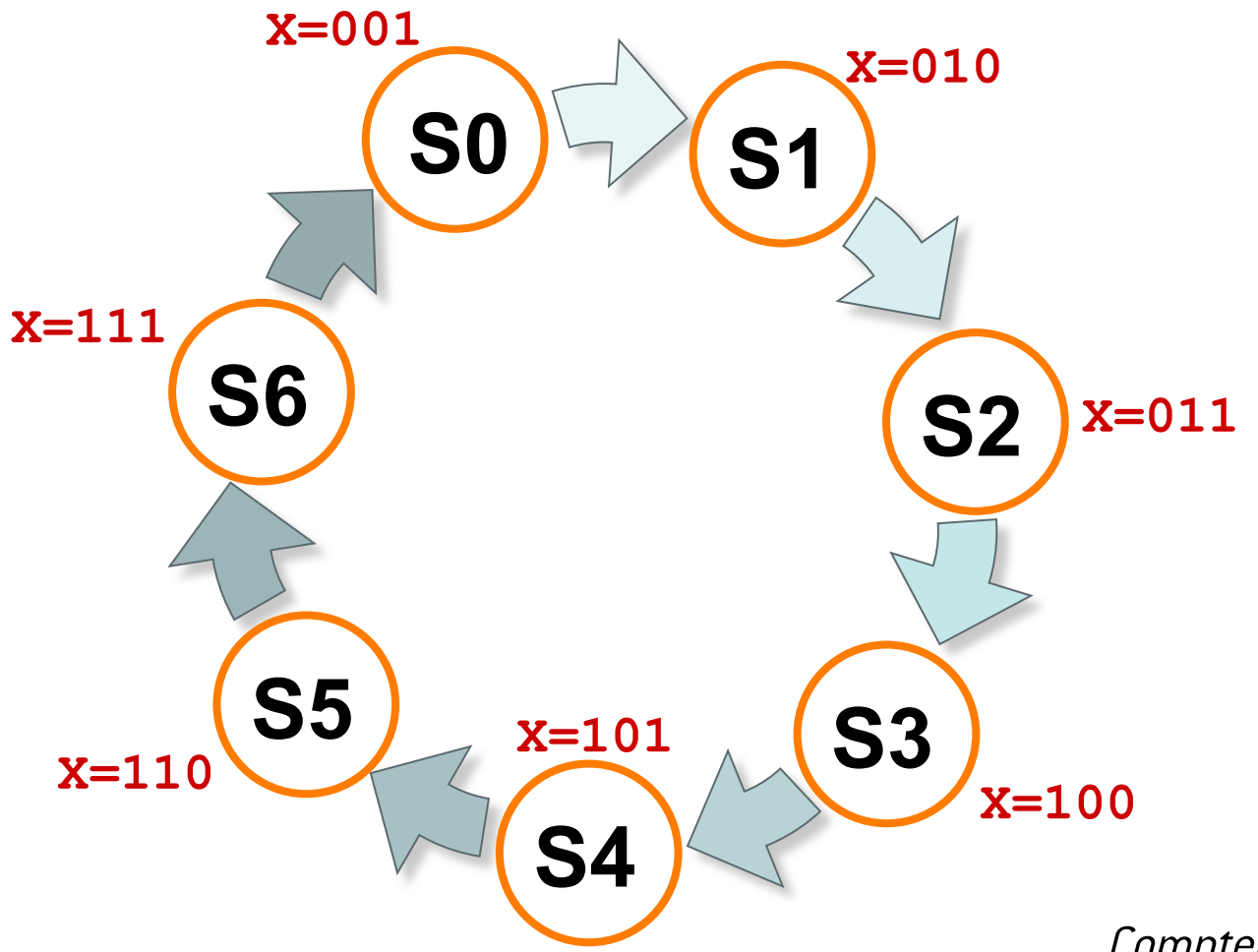
```
library moncir_lib;

configuration bench_cfg_portes of bench is
  for A
       for U1 : moncircuit use entity moncir_lib.moncircuit(A) ;
       end for ;
  end for;
end bench_cfg_portes;
```

Partie 3

MACHINE À ÉTATS

Les Fonctions à mémoire

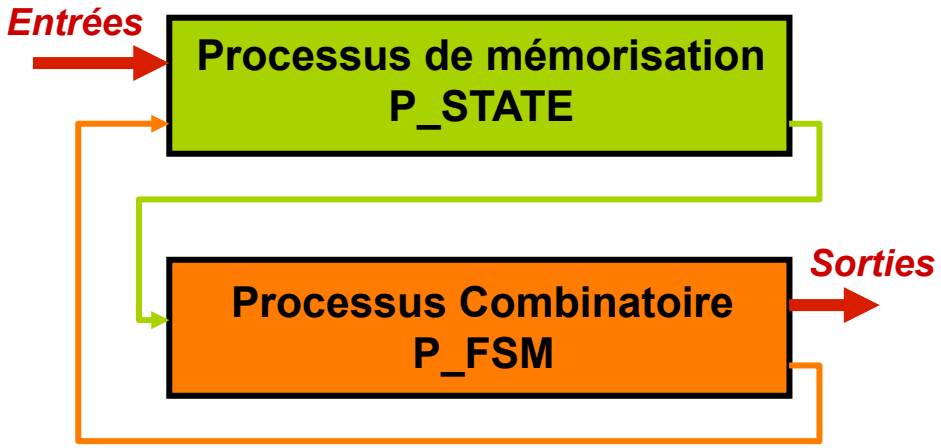
Description d'une fonction séquentielle



Compteur synchrone

(passage d'un état au suivant sur un front d'horloge)

FSM

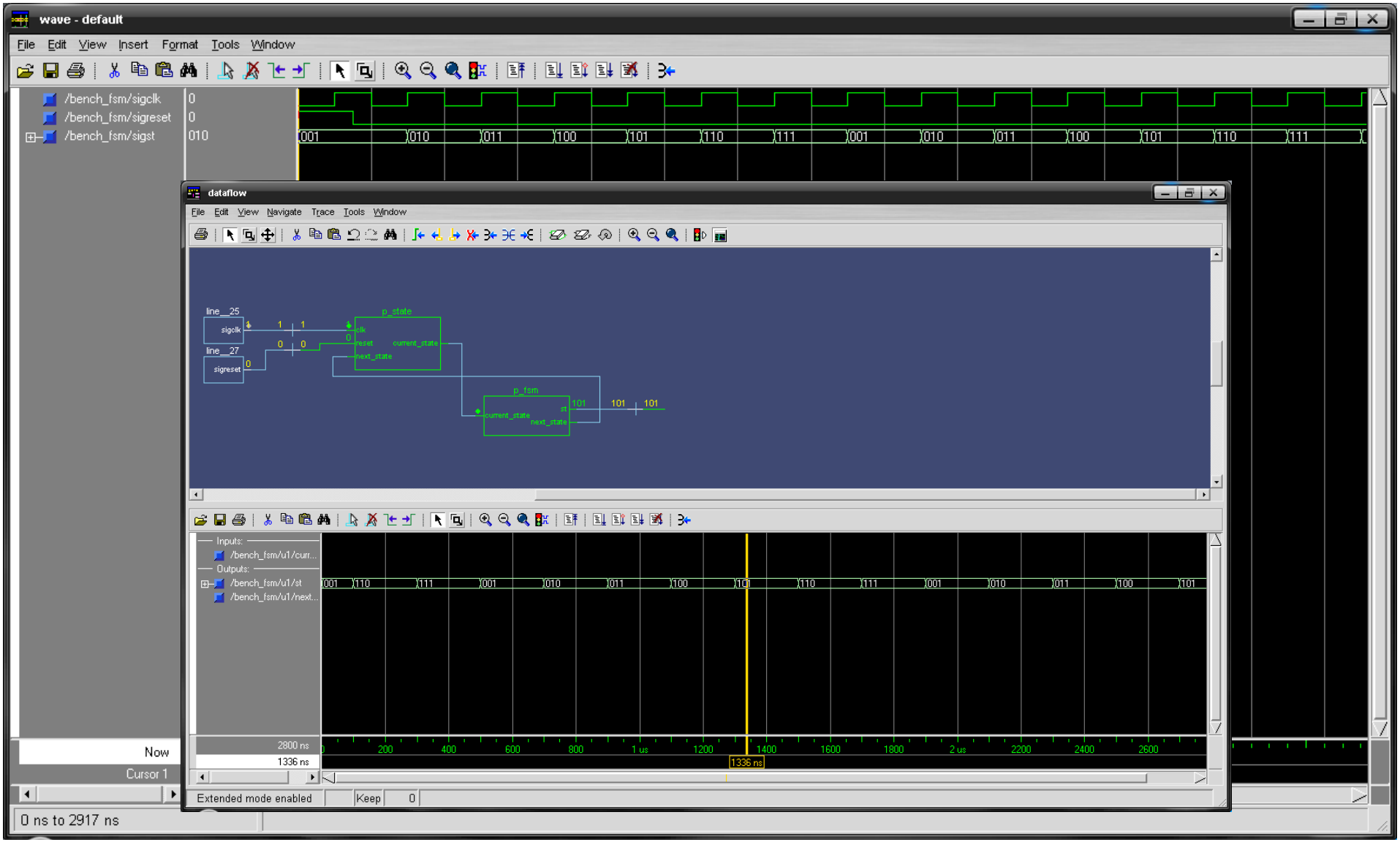



```
library IEEE ;
use IEEE.std_logic_1164.ALL ;
use IEEE.STD_LOGIC_ARITH.ALL;

entity FSM is
        port(
            CLK    : in STD_LOGIC;
            RESET : in STD_LOGIC;
            ST     : out STD_LOGIC_VECTOR(2 downto 0)
            );
end FSM;

architecture A of FSM is
type STATE is (S0,S1,S2,S3,S4,S5,S6);
signal Current_State, Next_State : STATE ;

begin
            -- Processus de Mémorisation
            P_STATE:process(CLK)
            begin
                        if (CLK='1' and CLK'EVENT ) then
                           if (RESET='1') then
                                    Current_State <= S0 ;
                           else
                                    Current_State <= Next_State ;
                           end if ;
                        end if;
            end process P_STATE;

            -- Processus Combinatoire
            P_FSM:process(Current_State)
            begin
            case Current_State is
                         when S0 =>
                                    ST <= "001";
                                    Next_State <= S1;
                         when S1=>
                                     ST <= "010";
                                    Next_State <= S2;
                         when S2 =>
                                     ST <= "011";
                                    Next_State <= S3;
                         when S3 =>
                                     ST <= "100";
                                    Next_State <= S4;
                         when S4 =>
                                     ST <= "101";
                                    Next_State <= S5;
                         when S5 =>
                                     ST <= "110";
                                    Next_State <= S6;
                         when S6 =>
                                     ST <= "111";
                                    Next_State <= S0;
            end case;
            end process P_FSM;

end A;
```

Bench de la FSM (Simulation)

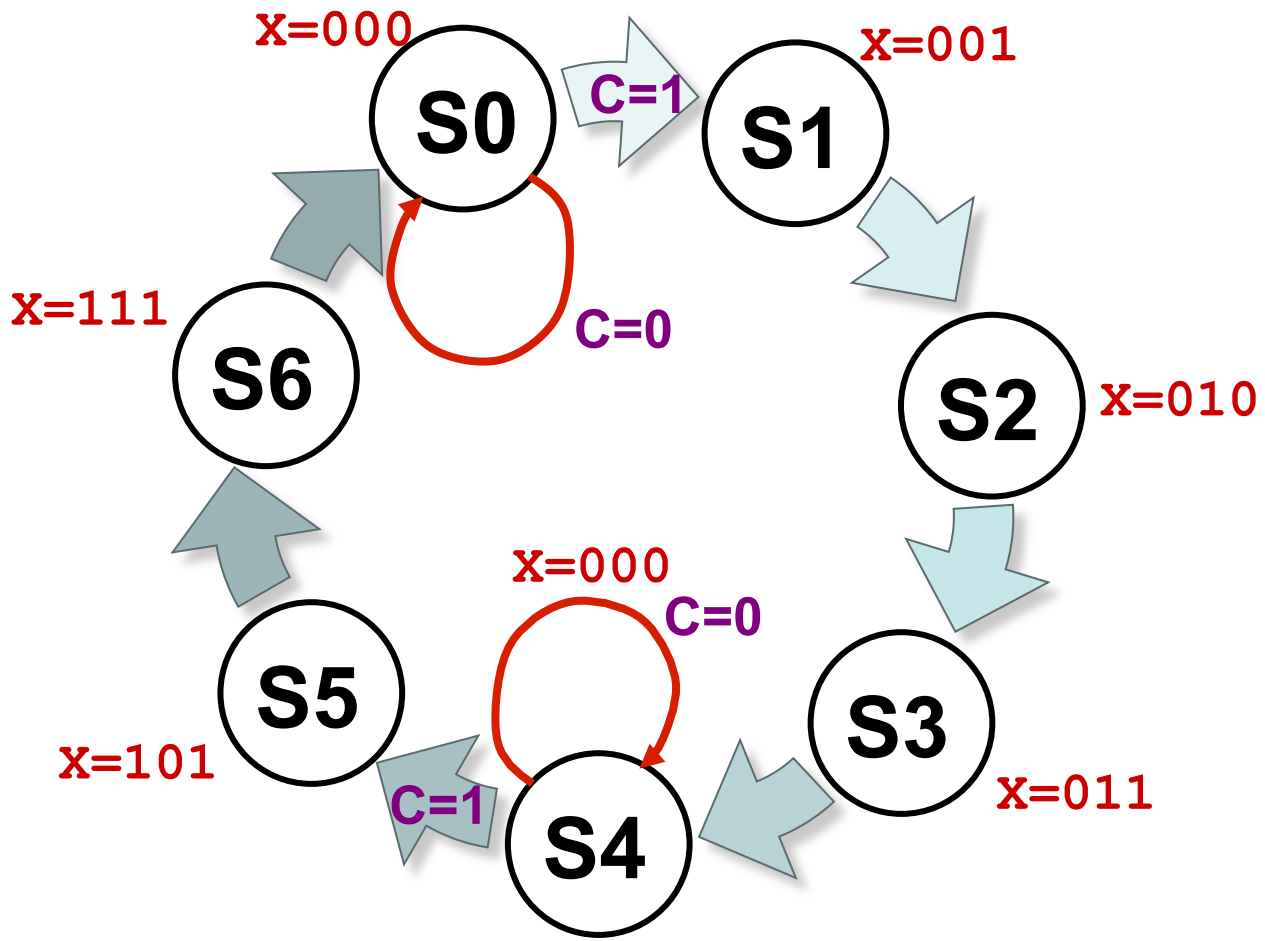
```
library IEEE ;
use IEEE.std_logic_1164.ALL ;
use IEEE.STD_LOGIC_ARITH.ALL;

entity BENCH_FSM is
end BENCH_FSM;

architecture A of BENCH_FSM is

component FSM
        port(    CLK      : in STD_LOGIC;
                 RESET    : in STD_LOGIC;
                 ST       : out STD_LOGIC_VECTOR(2 downto 0)
         );
end  component;

signal           sigCLK           : STD_LOGIC := '0';
signal           sigRESET         : STD_LOGIC;
signal           sigST            : STD_LOGIC_VECTOR(2 downto 0);

begin

U1:FSM port map(CLK,RESET,ST);

sigCLK <= not(CLK) after 100 ns;

sigReset <=      '1',
                 '0' after 150 ns ;
end;
```

Simulation


wave - default
File Edit View Insert Format Tools Window
/bench_fsm/sigclk
/bench_fsm/sigreset
/bench_fsm/sigst
dataflow
File Edit View Navigate Trace Tools Window
line__25
sigclk
line__27
sigreset
p_state
current_state
next_state
p_fsm
Inputs:
/bench_fsm/u1/curr...
Outputs:
/bench_fsm/u1/st
/bench_fsm/u1/next...
2800 ns
1336 ns
Now
Cursor 1
Extended mode enabled
Keep
0 ns to 2917 ns

Les Fonctions à mémoire

Description d'une fonction séquentielle



Compteur synchrone

(passage d'un état au suivant sur un front d'horloge)

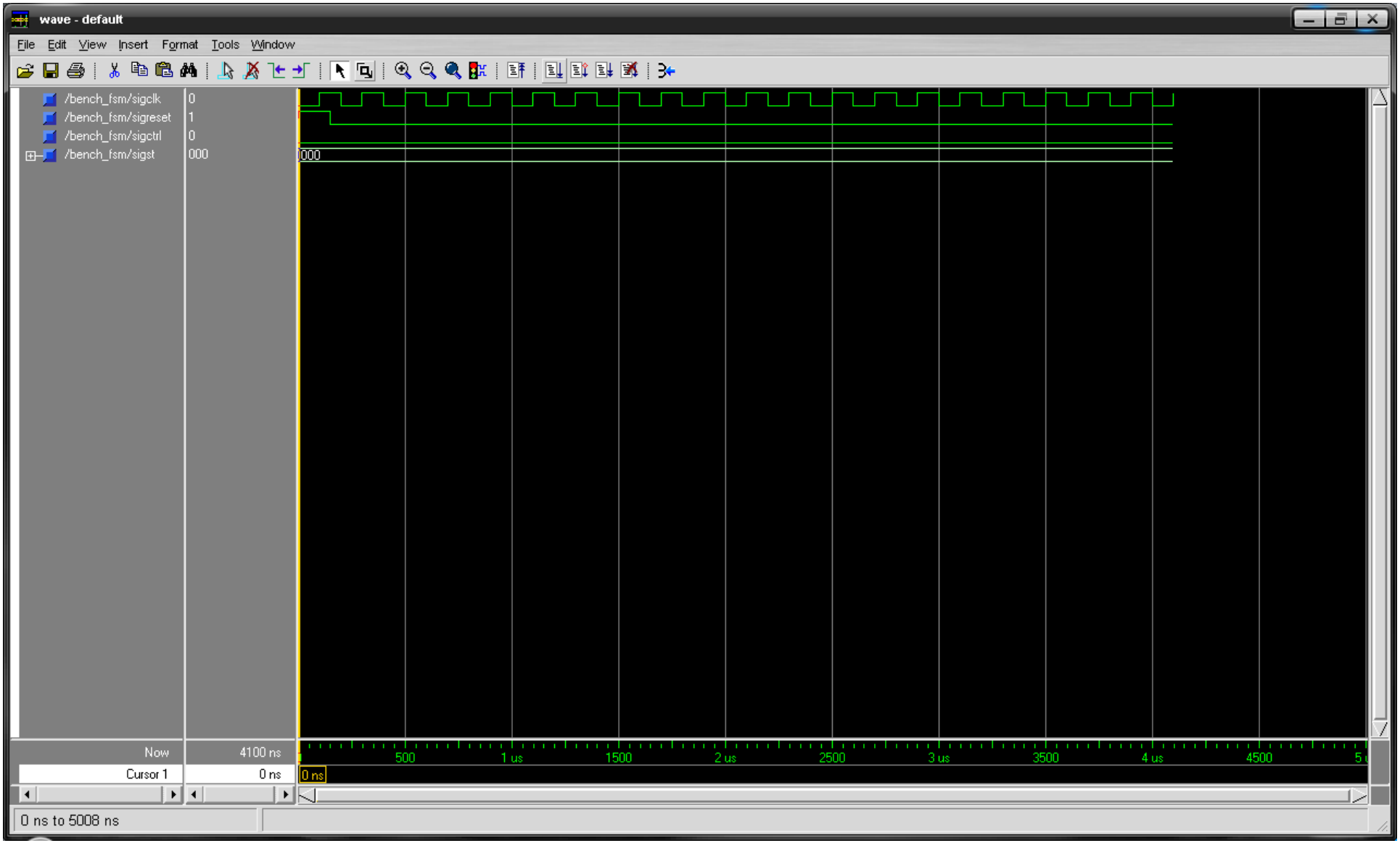
FSM

```
library IEEE ;
use IEEE.std_logic_1164.ALL ;
use IEEE.STD_LOGIC_ARITH.ALL;

entity FSM is
        port(
            CLK          : in STD_LOGIC;
            RESET        : in STD_LOGIC;
            CTRL         : in STD_LOGIC;
            ST           : out STD_LOGIC_VECTOR(2 downto 0)
            );
end FSM;

architecture A of FSM is
type STATE is (S0,S1,S2,S3,S4,S5,S6);
signal Current_State, Next_State : STATE ;

begin
            -- Processus de Mémorisation
            P_STATE:process(CLK)
            begin
            if (CLK='1' and CLK'EVENT ) then
               if (RESET='1') then
                        Current_State <= S0;
               else
                        Current_State <= Next_State ;
               end if ;
            end if;
            end process P_STATE;

            -- Processus Combinatoire
            P_FSM:process(Current_State,CTRL)
            begin
            case Current_State is
             when S0 =>
                        ST <= "000";
                        if CTRL='1' then
                                    Next_State <= S1;
                        else
                                    Next_State <= S0;
                        end if;
             when S1=>
                        ST <= "001";
                        Next_State <= S2;
             when S2 =>
                        ST <= "010";
                        Next_State <= S3;
             when S3 =>
                        ST <= "011";
                        Next_State <= S4;
             when S4 =>
                        ST <= "000";
                        if CTRL='1' then
                                    Next_State <= S5;
                        else
                                    Next_State <= S4;
                        end if;
             when S5 =>
                        ST <= "101";
                        Next_State <= S6;
             when S6 =>
                        ST <= "111";
                        Next_State <= S0;
 end case;
 end process P_FSM;
end A;
```

Bench 1

```
library IEEE ;
use IEEE.std_logic_1164.ALL ;
use IEEE.STD_LOGIC_ARITH.ALL;

entity BENCH_FSM is
end BENCH_FSM;

architecture A of BENCH_FSM is

component FSM
        port(    CLK      : in STD_LOGIC;
                 RESET    : in STD_LOGIC;
                 CTRL     : in STD_LOGIC;
                 ST       : out STD_LOGIC_VECTOR(2 downto 0)
         );
end  component;

signal           sigCLK           : STD_LOGIC := '0';
signal           sigRESET         : STD_LOGIC;
signal           sigCTRL          : STD_LOGIC := '0';
signal           sigST            : STD_LOGIC_VECTOR(2 downto 0);

begin

U1:FSM port map(sigCLK,sigRESET,sigCTRL,sigST);

sigCLK <= not(sigCLK) after 100 ns;

sigReset <=      '1',
                 '0' after 150 ns ;
end;
```

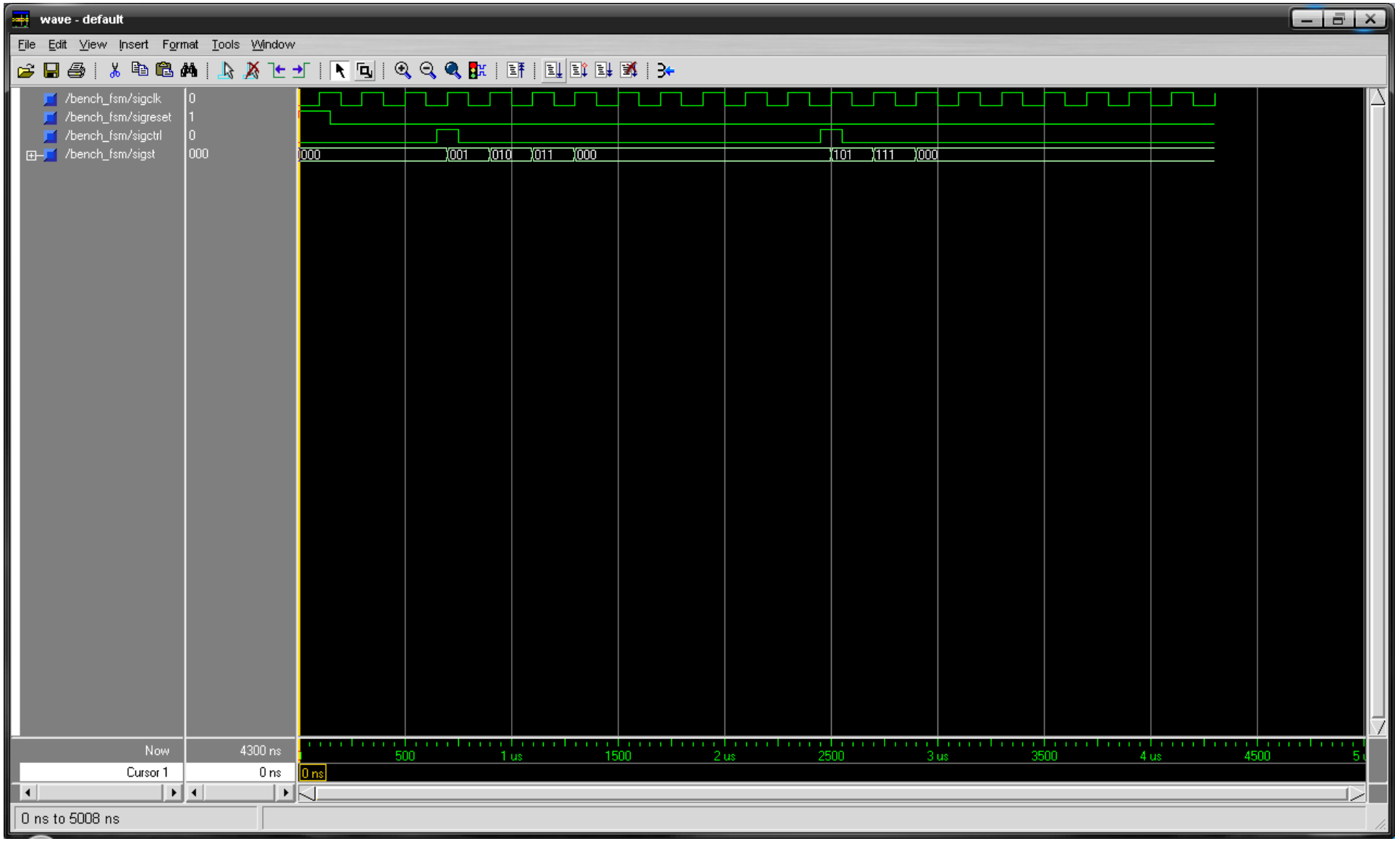
Simulation


wave - default
File Edit View Insert Format Tools Window
/bench_fsm/sigclk
/bench_fsm/sigreset
/bench_fsm/sigctrl
/bench_fsm/sigst
0
1
0
000
000
Now
4100 ns
Cursor 1
0 ns
0 ns
500
1 us
1500
2 us
2500
3 us
3500
4 us
4500
0 ns to 5008 ns

Bench 2

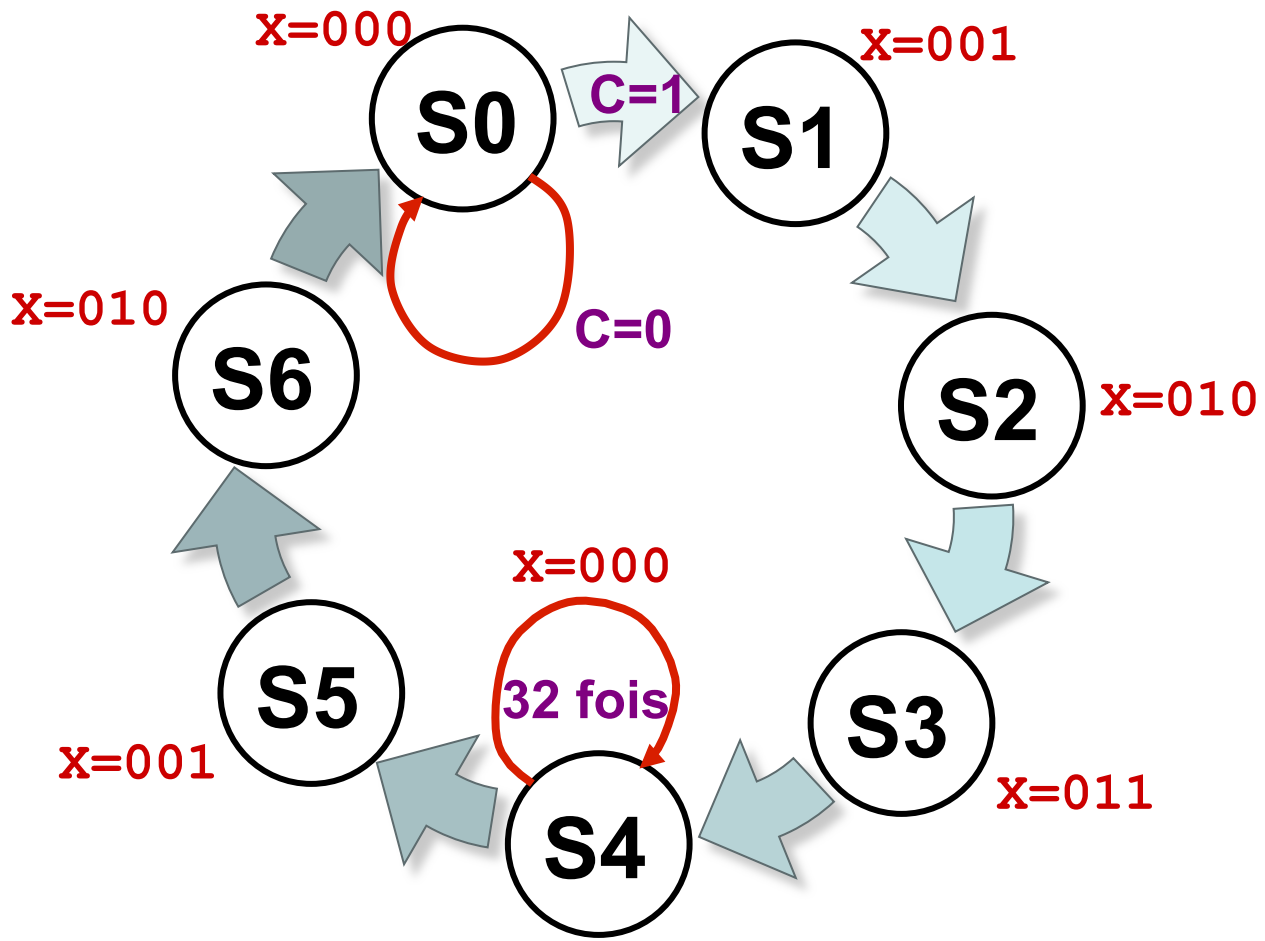
```
library IEEE ;
use IEEE.std_logic_1164.ALL ;
use IEEE.STD_LOGIC_ARITH.ALL;

entity BENCH_FSM is
end BENCH_FSM;

architecture A of BENCH_FSM is

component FSM
        port(           CLK         : in STD_LOGIC;
                        RESET       : in STD_LOGIC;
                        CTRL        : in STD_LOGIC;
                        ST          : out STD_LOGIC_VECTOR(2 downto 0)
            );
end  component;

signal                  sigCLK                  : STD_LOGIC := '0';
signal                  sigRESET                : STD_LOGIC;
signal                  sigCTRL                 : STD_LOGIC := '0';
signal                  sigST                   : STD_LOGIC_VECTOR(2 downto 0);

begin

U1:FSM port map(sigCLK,sigRESET,sigCTRL,sigST);

sigCLK <= not(sigCLK) after 100 ns;

sigReset <= '1',
                        '0' after 150 ns ;
sigCTRL <=      '0',
                '1' after 650 ns,
                '0' after 750 ns,
                '1' after 2450 ns,
                '0' after 2550 ns;
end;
```

Simulation


wave - default
File Edit View Insert Format Tools Window
/bench_fsm/sigclk
/bench_fsm/sigreset
/bench_fsm/sigctrl
/bench_fsm/sigst
0
1
0
000
000
001
010
011
000
101
111
000
Now
4300 ns
Cursor 1
0 ns
0 ns
500
1 us
1500
2 us
2500
3 us
3500
4 us
4500
0 ns to 5008 ns

Exercice

Description d'une fonction séquentielle



Compteur synchrone
(passage d'un état au suivant sur un front d'horloge)

Partie 4

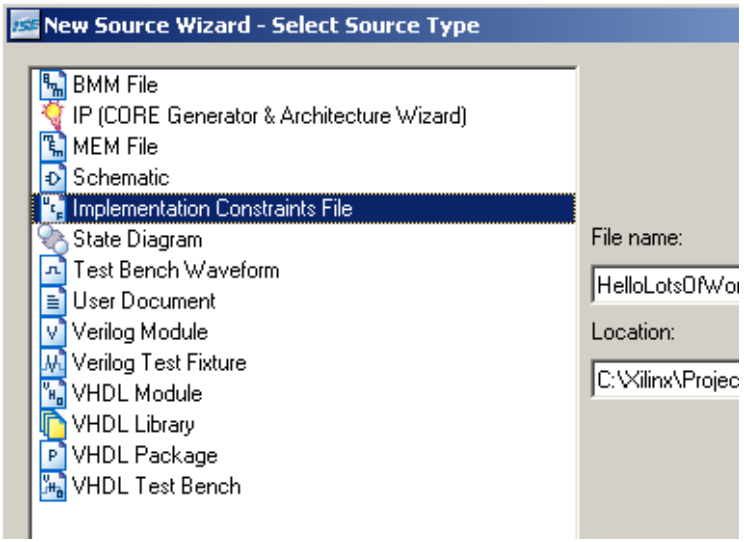
IMPLÉMENTATION SUR FPGA

Synthèse et Placement/Routage

- Partie très importante (indispensable)
- Simulation avec prise en considération des temps de propagation des signaux
- Génération de la netlist (vhdl, verilog, etc.)
- Simulation en utilisant la netlist générée (apparition de retards)
- Résultats différents de la simulation idéale
- Permet de modifier l'architecture du circuit

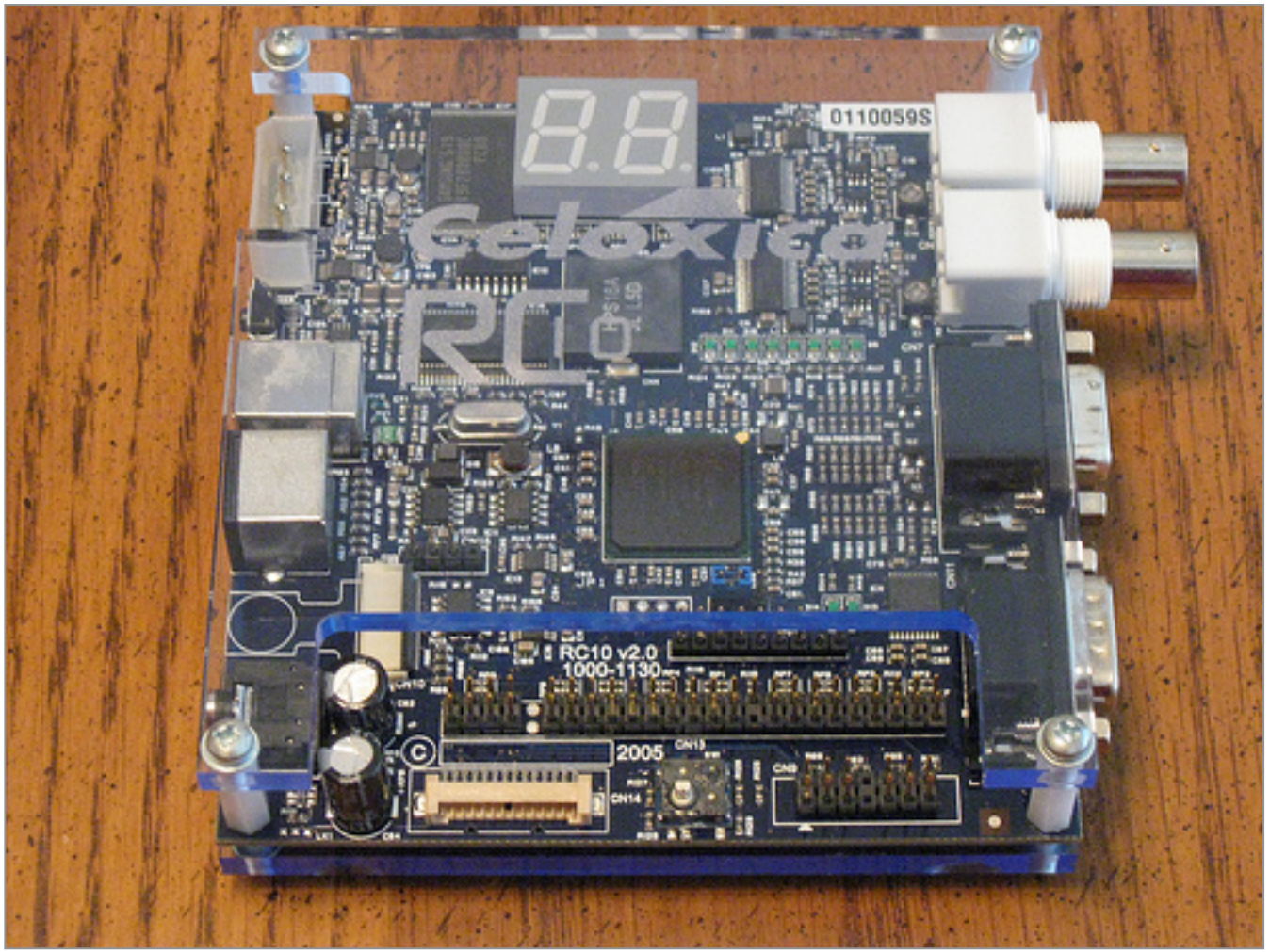
Fichier UCF

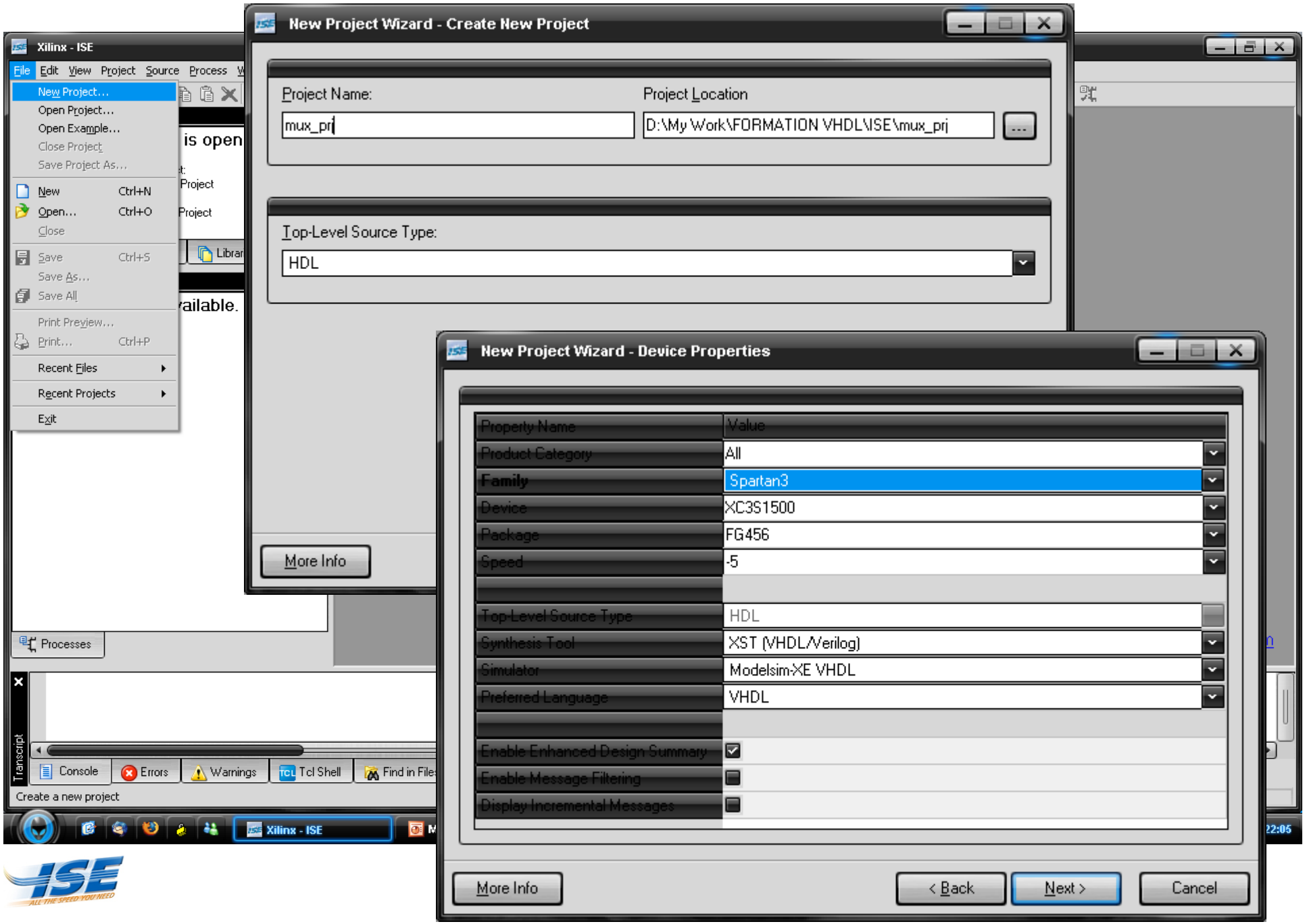
- UCF : User Constraints File
- Affecter les entrées sorties d'un module aux pins de l'FPGA
- Utilisation de la : Data Sheet

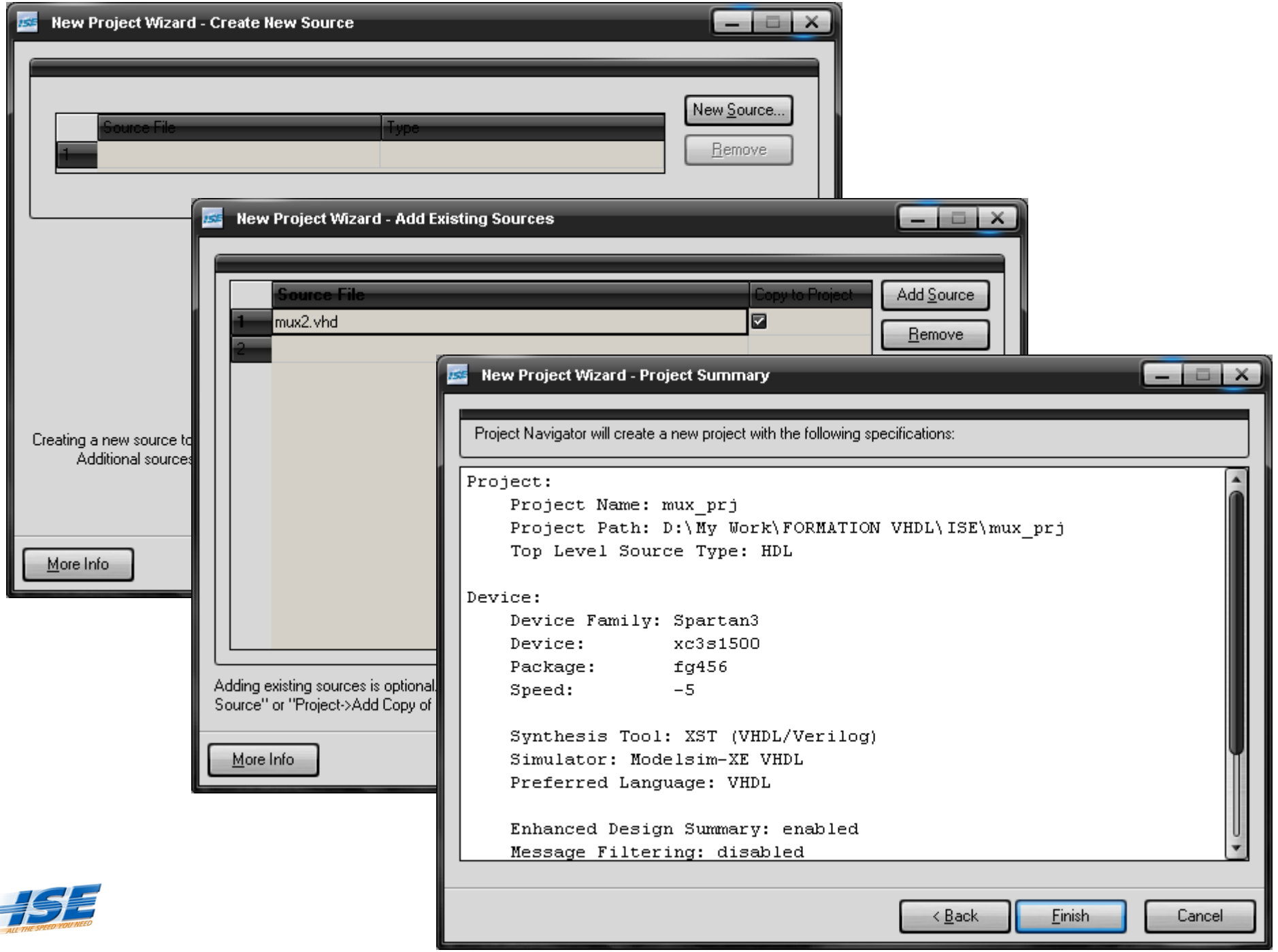


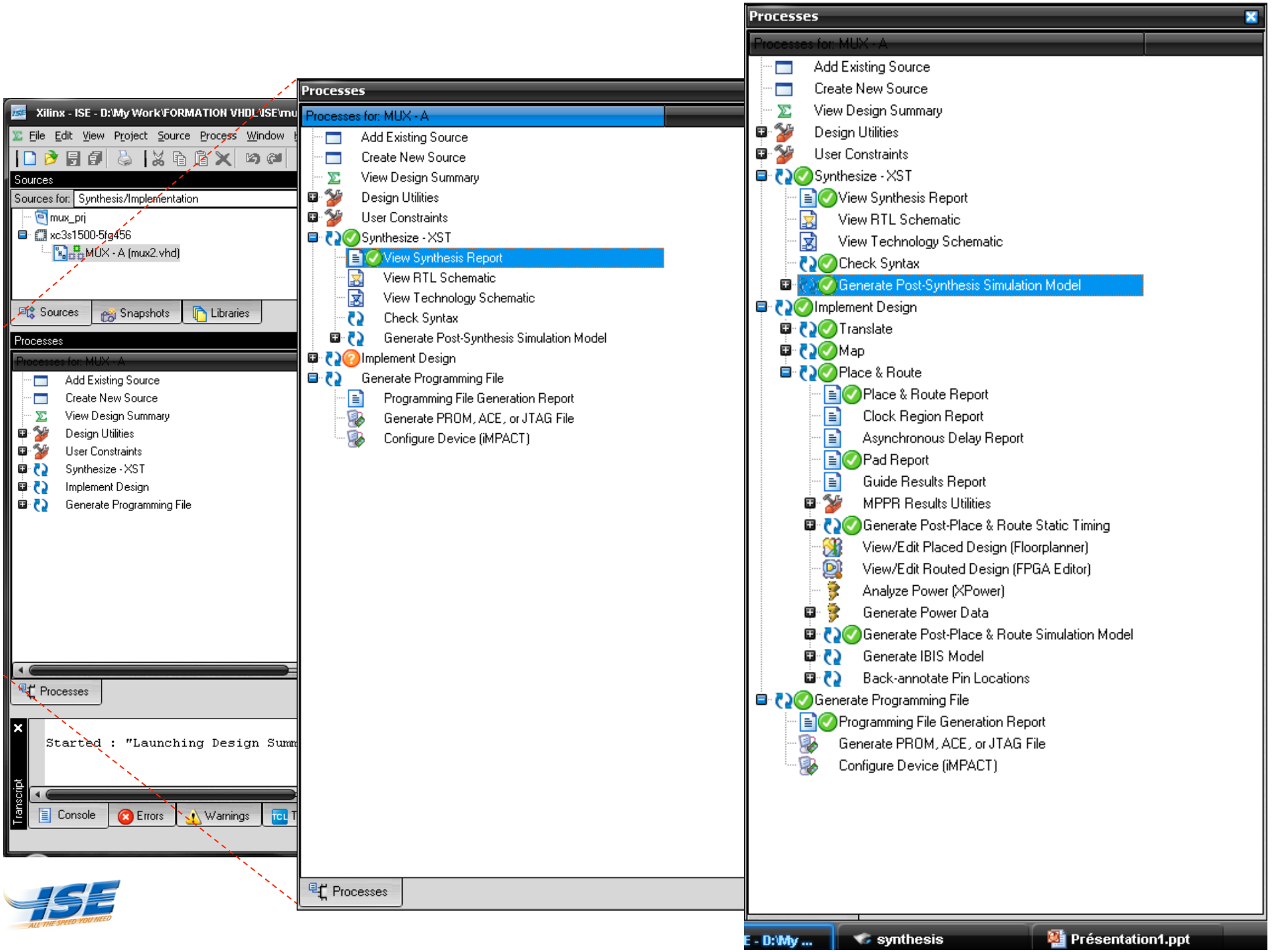

```
1   NET "IN<0>"   LOC = "P38"  ;
2   NET "IN<1>"   LOC = "P36"  ;
3   NET "IN<2>"   LOC = "P29"  ;
4   NET "IN<3>"   LOC = "P24"  ;
5   NET "IN<4>"   LOC = "P18"  ;
6   NET "IN<5>"   LOC = "P12"  ;
7   NET "IN<6>"   LOC = "P10"  ;
8   NET "IN<7>"   LOC = "P6"   ;
9
10  NET "OUT<0>"   LOC = "P15"  ;
11  NET "OUT<1>"   LOC = "P14"  ;
12  NET "OUT<2>"   LOC = "P8"  ;
13  NET "OUT<3>"   LOC = "P7"  ;
14  NET "OUT<4>"   LOC = "P5"  ;
15  NET "OUT<5>"   LOC = "P4"  ;
16  NET "OUT<6>"   LOC = "P3"  ;
17  NET "OUT<7>"   LOC = "P2"  ;
```

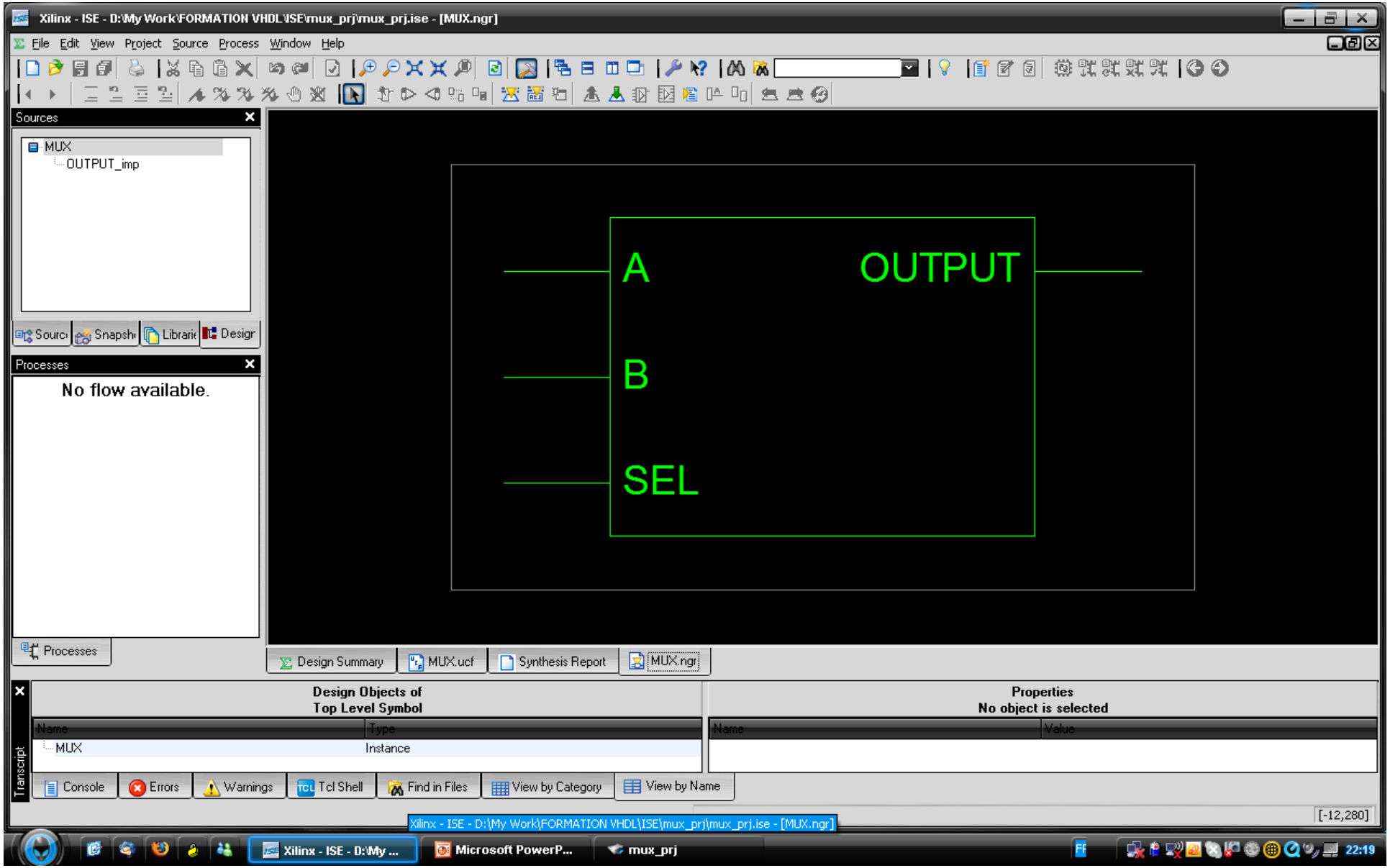
Carte Celoxica RC10

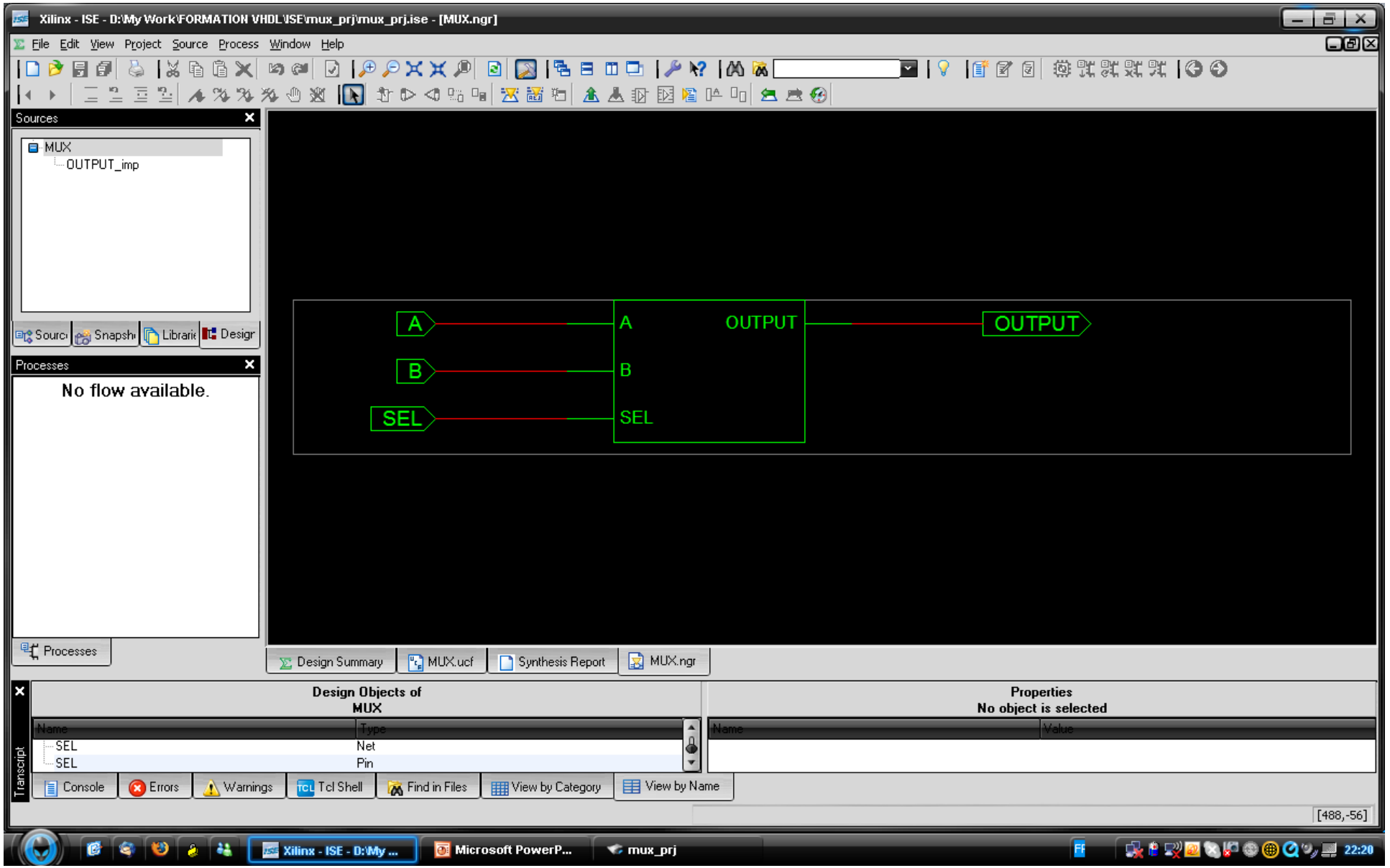


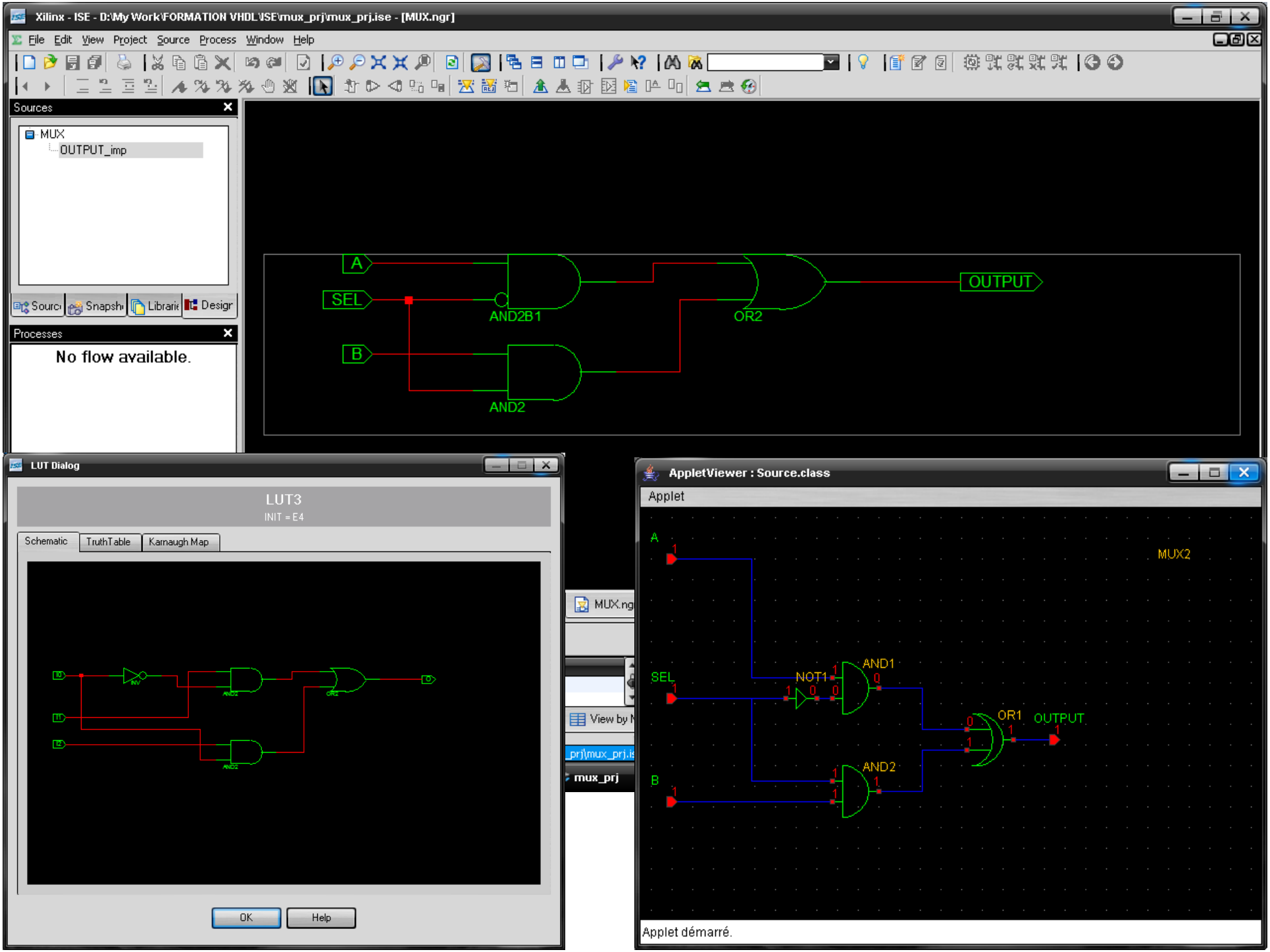

Xilinx - ISE
File Edit View Project Source Process
New Project...
Open Project...
Open Example...
Close Project
Save Project As...
New Ctrl+N
Open... Ctrl+O
Close
Save Ctrl+S
Save As...
Save All
Print Preview...
Print... Ctrl+P
Recent Files
Recent Projects
Exit
Processes
Transcript
Console
Errors
Warnings
Tcl Shell
Find in Files
Create a new project
New Project Wizard - Create New Project
Project Name:
mux_prj
Project Location
D:\My Work\FORMATION VHDL\ISE\mux_prj
Top-Level Source Type:
HDL
More Info
New Project Wizard - Device Properties
Property Name
Value
Product Category
All
Family
Spartan3
Device
XC3S1500
Package
FG456
Speed
-5
Top-Level Source Type
HDL
Synthesis Tool
XST (VHDL/Verilog)
Simulator
Modelsim-XE VHDL
Preferred Language
VHDL
Enable Enhanced Design Summary
Enable Message Filtering
Display Incremental Messages
More Info
< Back
Next >
Cancel


New Project Wizard - Create New Source
Source File
Type
New Source...
Remove
Creating a new source to
Additional sources
More Info
New Project Wizard - Add Existing Sources
Source File
Copy to Project
mux2.vhd
Add Source
Remove
Adding existing sources is optional
Source" or "Project->Add Copy of
More Info
New Project Wizard - Project Summary
Project Navigator will create a new project with the following specifications:
Project:
Project Name: mux_prj
Project Path: D:\My Work\FORMATION VHDL\ISE\mux_prj
Top Level Source Type: HDL
Device:
Device Family: Spartan3
Device: xc3s1500
Package: fg456
Speed: -5
Synthesis Tool: XST (VHDL/Verilog)
Simulator: Modelsim-XE VHDL
Preferred Language: VHDL
Enhanced Design Summary: enabled
Message Filtering: disabled
< Back
Finish
Cancel

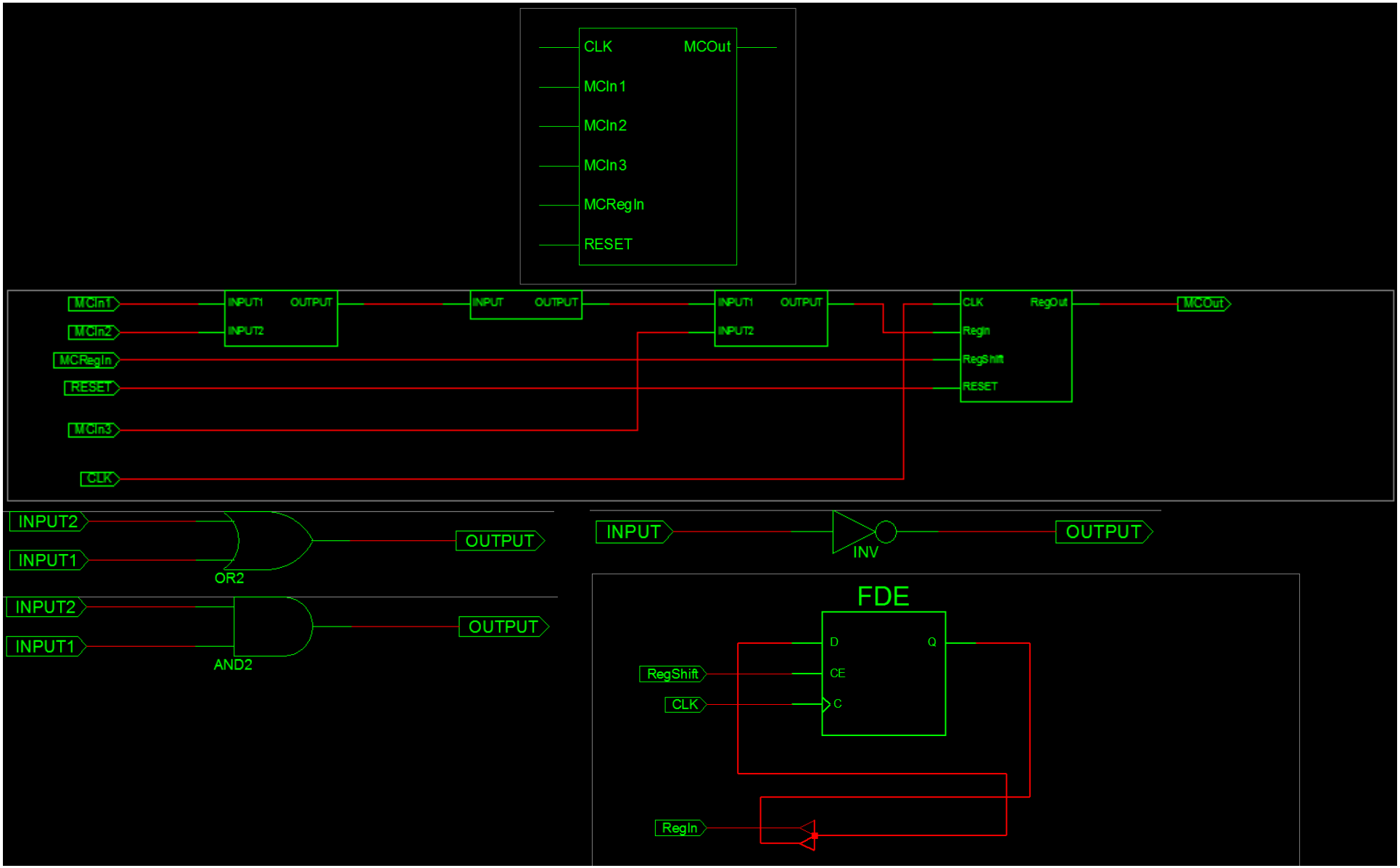

Xilinx - ISE - D:\My Work\FORMATION VHDL\ISE\mu
File Edit View Project Source Process Window
Sources
Sources for: Synthesis/Implementation
mux_prj
xc3s1500-5fg456
MUX - A (mux2.vhd)
Sources Snapshots Libraries
Processes
Processes for: MUX - A
Add Existing Source
Create New Source
View Design Summary
Design Utilities
User Constraints
Synthesize - XST
Implement Design
Generate Programming File
Processes
Started : "Launching Design Summ
Transcript
Console Errors Warnings
Processes
Processes for: MUX - A
Add Existing Source
Create New Source
View Design Summary
Design Utilities
User Constraints
Synthesize - XST
View Synthesis Report
View RTL Schematic
View Technology Schematic
Check Syntax
Generate Post-Synthesis Simulation Model
Implement Design
Generate Programming File
Programming File Generation Report
Generate PROM, ACE, or JTAG File
Configure Device (iMPACT)
Processes
Processes
Processes for: MUX - A
Add Existing Source
Create New Source
View Design Summary
Design Utilities
User Constraints
Synthesize - XST
View Synthesis Report
View RTL Schematic
View Technology Schematic
Check Syntax
Generate Post-Synthesis Simulation Model
Implement Design
Translate
Map
Place & Route
Place & Route Report
Clock Region Report
Asynchronous Delay Report
Pad Report
Guide Results Report
MPPR Results Utilities
Generate Post-Place & Route Static Timing
View/Edit Placed Design (Floorplanner)
View/Edit Routed Design (FPGA Editor)
Analyze Power (XPower)
Generate Power Data
Generate Post-Place & Route Simulation Model
Generate IBIS Model
Back-annotate Pin Locations
Generate Programming File
Programming File Generation Report
Generate PROM, ACE, or JTAG File
Configure Device (iMPACT)
E - D:\My ...
synthesis
Présentation1.ppt


Xilinx - ISE - D:\My Work\FORMATION VHDL\ISE\mux_prj\mux_prj.ise - [MUX.ngr]
File Edit View Project Source Process Window Help
Sources
MUX
OUTPUT_imp
Processes
No flow available.
Processes
A
B
SEL
OUTPUT
Design Summary
MUX.ucf
Synthesis Report
MUX.ngr
Design Objects of
Top Level Symbol
Name
Type
MUX
Instance
Properties
No object is selected
Name
Value
Transcript
Console
Errors
Warnings
Tcl Shell
Find in Files
View by Category
View by Name
[-12,280]
Xilinx - ISE - D:\My Work\FORMATION VHDL\ISE\mux_prj\mux_prj.ise - [MUX.ngr]
Xilinx - ISE - D:\My ...
Microsoft PowerP...
mux_prj
22:19

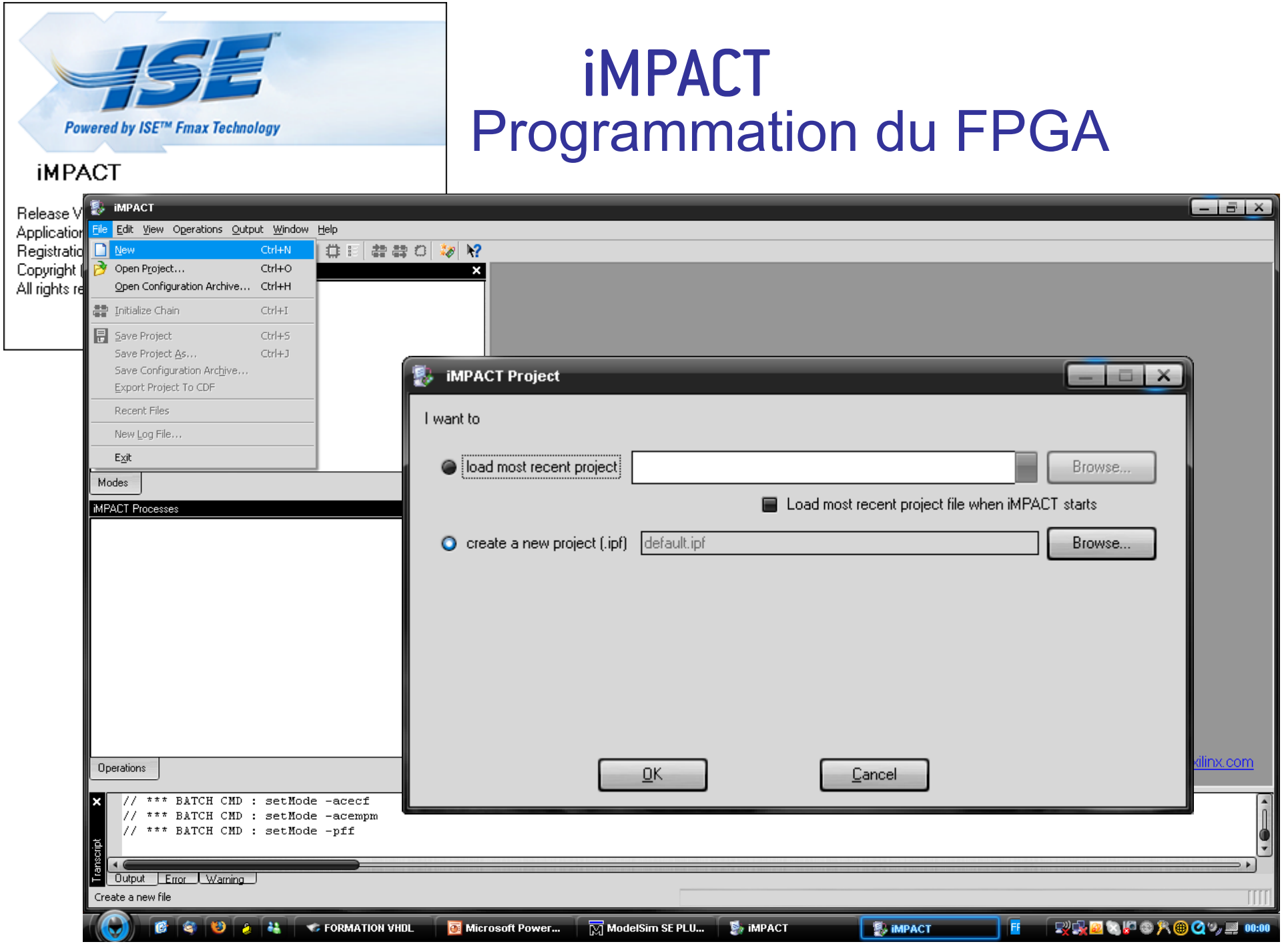

Xilinx - ISE - D:\My Work\FORMATION VHDL\ISE\mux_prj\mux_prj.ise - [MUX.ngr]
File Edit View Project Source Process Window Help
Sources
MUX
OUTPUT_imp
Processes
No flow available.
Processes
A
B
SEL
A
B
SEL
OUTPUT
OUTPUT
Design Summary
MUX.ucf
Synthesis Report
MUX.ngr
Design Objects of MUX
Properties
No object is selected
Name
Type
SEL
Net
SEL
Pin
Name
Value
Transcript
Console
Errors
Warnings
Tcl Shell
Find in Files
View by Category
View by Name
[488,-56]
Xilinx - ISE - D:\My ...
Microsoft PowerP...
mux_prj
22:20


Xilinx - ISE - D:\My Work\FORMATION VHDL\ISE\mux_prj\mux_prj.ise - [MUX.ngr]
File Edit View Project Source Process Window Help
Sources
MUX
OUTPUT_imp
Processes
No flow available.
A
SEL
B
AND2B1
AND2
OR2
OUTPUT
LUT Dialog
LUT3
INIT = E4
Schematic
TruthTable
Karnaugh Map
OK
Help
MUX.ng
View by
_prj\mux_prj.is
mux_prj
AppletViewer : Source.class
Applet
MUX2
A
SEL
B
NOT1
AND1
AND2
OR1
OUTPUT
Applet démarré.

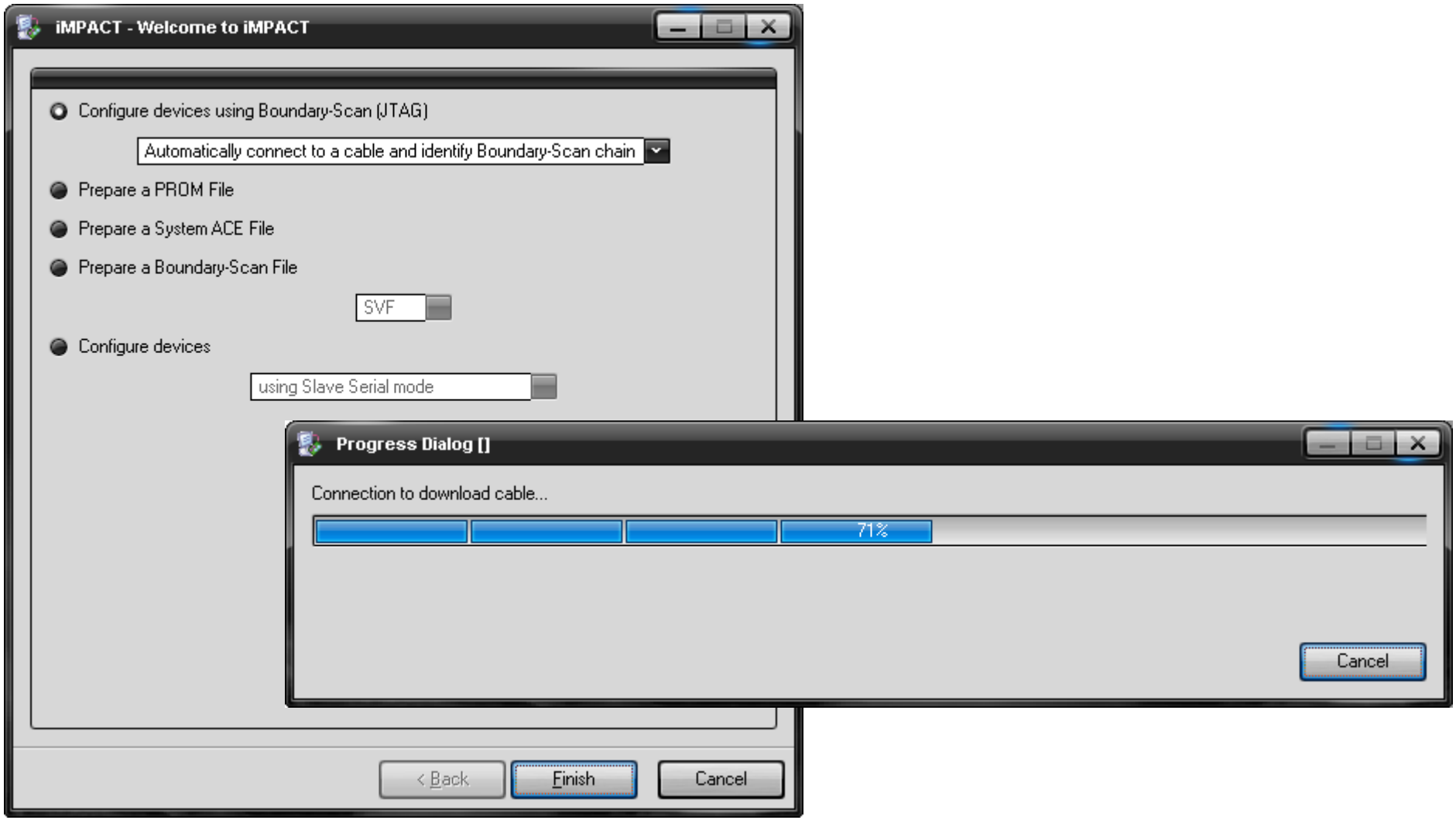
Cas de moncircuit



iMPACT
Programmation du FPGA



Programmation du FPGA



Merci