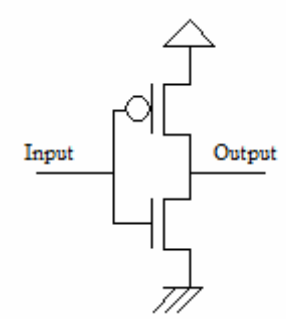
TRƯỜNG ĐẠI HỌC ĐÀ LẠT
NGÀNH ĐIỆN TỬ - VIỄN THÔNG

---ꕥ🕮ꕥ---

Lê Văn Tùng

Tài Liệu Hướng Dẫn
THỰC HÀNH THIẾT KẾ VI MẠCH



Đà Lạt, 2011

Bài 1
GIỚI THIỆU

1.1 Mục đích

Bài thực hành đầu tiên được thiết kế nhằm trang bị cho sinh viên những kiến thức cần thiết để làm việc với Board FPGA Xilinx Spartan3. Do thời gian thực tập không nhiều, tài liệu này chỉ cung cấp những thông tin cơ bản về lập trình FPGA bằng ngôn ngữ VHDL dưới sự hỗ trợ của phần mềm ISE Design Suite, phiên bản 12.1. Đây là gói phần mềm chuyên dụng của hãng Xilinx. Để có thể nắm vững, đầy đủ tính năng của ISE Design Suite thì thời gian nghiên cứu ngoài giờ thực tập là cần thiết.

Để có thể làm việc với Board Xilinx Spartan 3, phần mềm cần có là ISE Xilinx Design Suite, Xilinx PlanAhead và Adept. Trong đó, Adept là phần mềm được cung cấp bởi Digilent để nạp chương trình vào FPGA. Tất cả các phần mềm này cũng như giấy phép sử dụng (license) có thể tải trực tiếp từ trang chủ của nhà sản xuất.

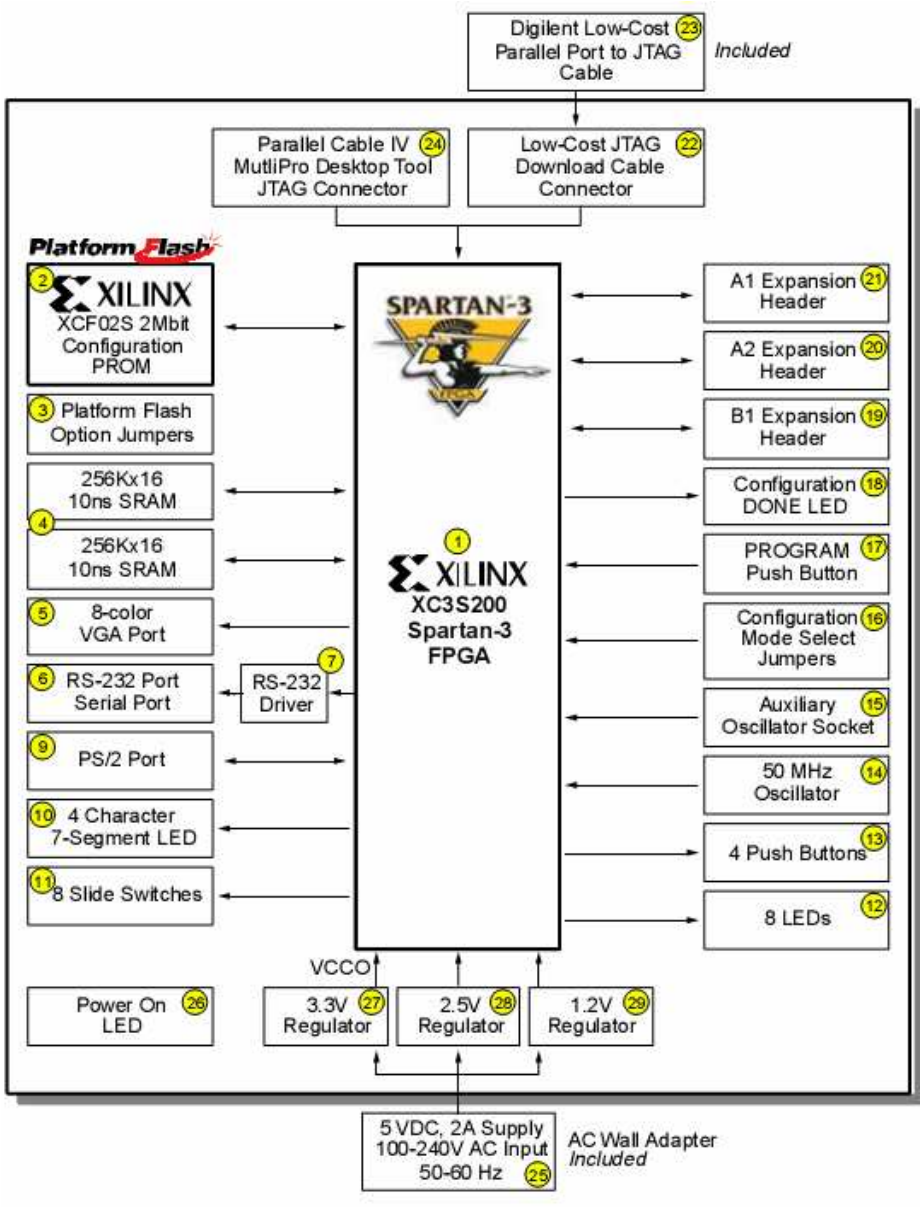
1.2 Quy trình thực tập

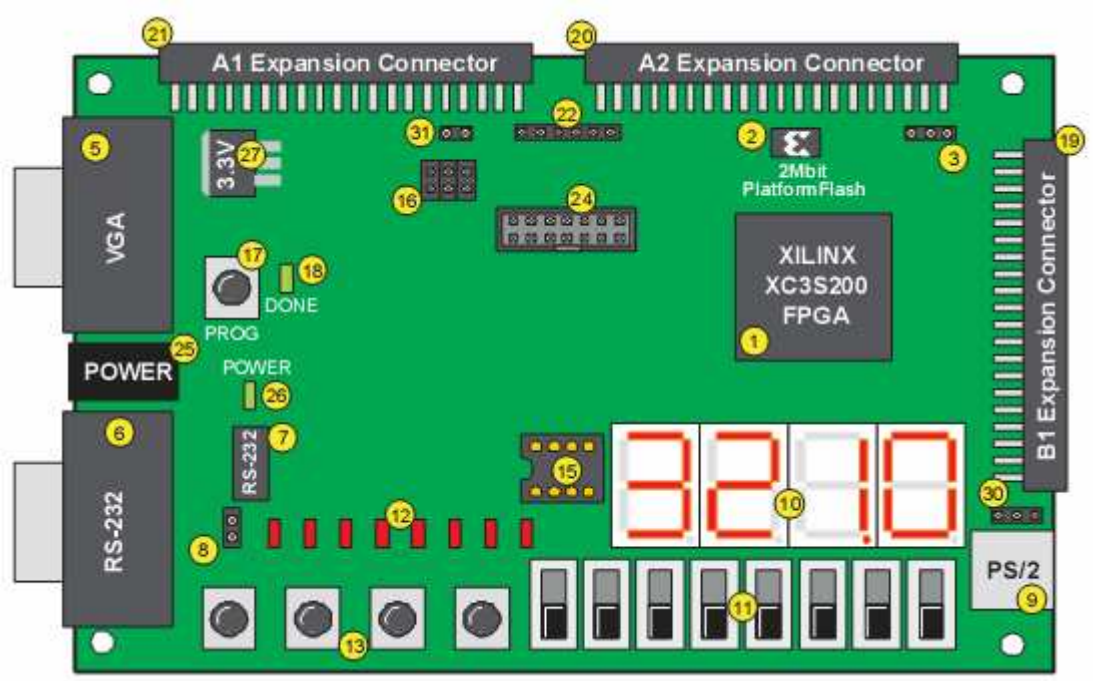
Trong bài 1, quy trình làm việc cơ bản sẽ được trình bày chi tiết. Từ bài thứ 2 trở đi, các bước này sẽ không được nhắc lại. Ví dụ sử dụng để minh họa các bước thực tập là bật tắt đèn LED bằng công tắc SW.

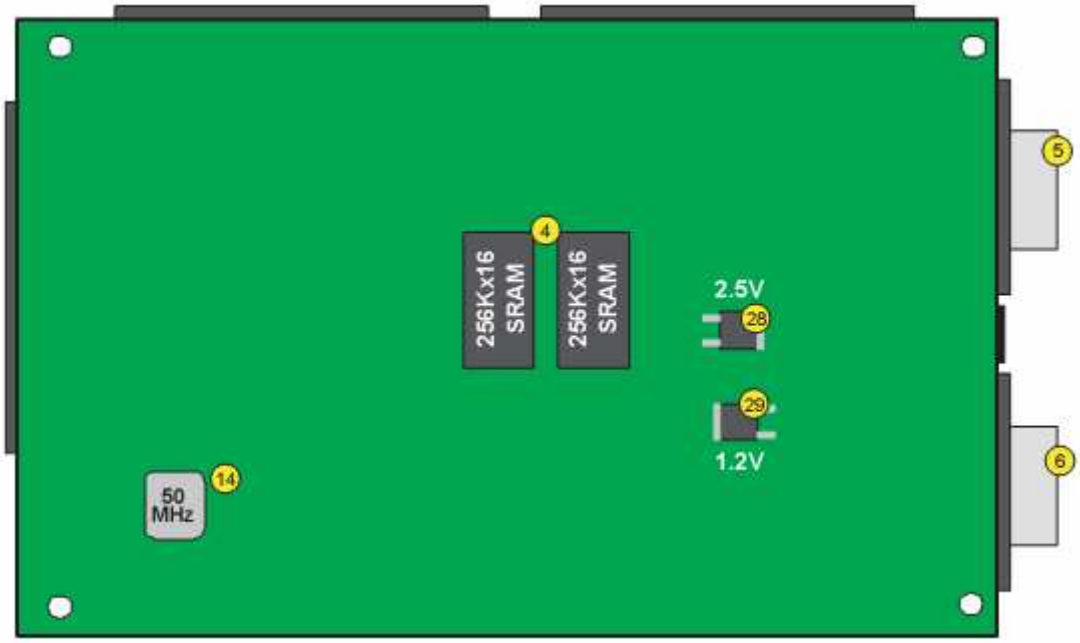
Chú ý: "Báo cáo Thực tập" được viết bao gồm các ý chính sau:

- Quy trình thực tập
- Kết quả từng phần
- Nhận xét & Kết luận

1.2.1 Thiết kế

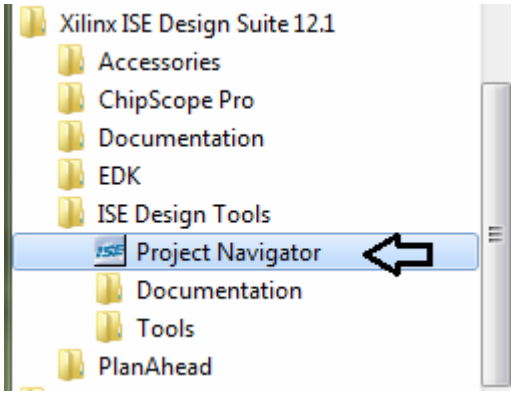

Digilent Low-Cost 23 Parallel Port to JTAG Cable
Included
Parallel Cable IV 24 MutliPro Desktop Tool JTAG Connector
Low-Cost JTAG 22 Download Cable Connector
Platform Flash
2 XILINX XCF02S 2Mbit Configuration PROM
3 Platform Flash Option Jumpers
256Kx16 10ns SRAM
4 256Kx16 10ns SRAM
5 8-color VGA Port
6 RS-232 Port Serial Port
7 RS-232 Driver
9 PS/2 Port
10 4 Character 7-Segment LED
11 8 Slide Switches
SPARTAN-3
1 XILINX XC3S200 Spartan-3 FPGA
A1 Expansion 21 Header
A2 Expansion 20 Header
B1 Expansion 19 Header
Configuration 18 DONE LED
PROGRAM 17 Push Button
Configuration 16 Mode Select Jumpers
Auxiliary 15 Oscillator Socket
50 MHz 14 Oscillator
4 Push Buttons 13
8 LEDs 12
VCCO
Power On 26 LED
3.3V 27 Regulator
2.5V 28 Regulator
1.2V 29 Regulator
5 VDC, 2A Supply 100-240V AC Input 50-60 Hz 25
AC Wall Adapter Included


A1 Expansion Connector
A2 Expansion Connector
B1 Expansion Connector
VGA
RS-232
POWER
PROG
DONE
POWER
3.3V
RS-232
PS/2
2Mbit
PlatformFlash
XILINX
XC3S200
FPGA

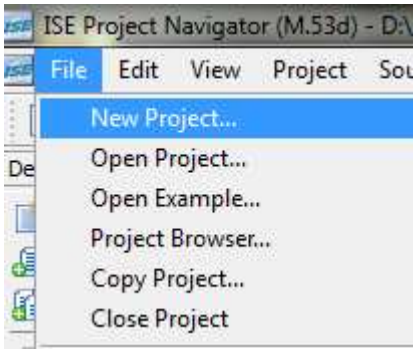

256Kx16
SRAM
256Kx16
SRAM
2.5V
1.2V
50
MHz

Trước tiên, Board S3 cần được cắm nguồn và kết nối với máy tính thông qua cổng USB. Nếu máy tính không nhận ra thiết bị, người dùng cần phải tải Driver về và cài đặt.

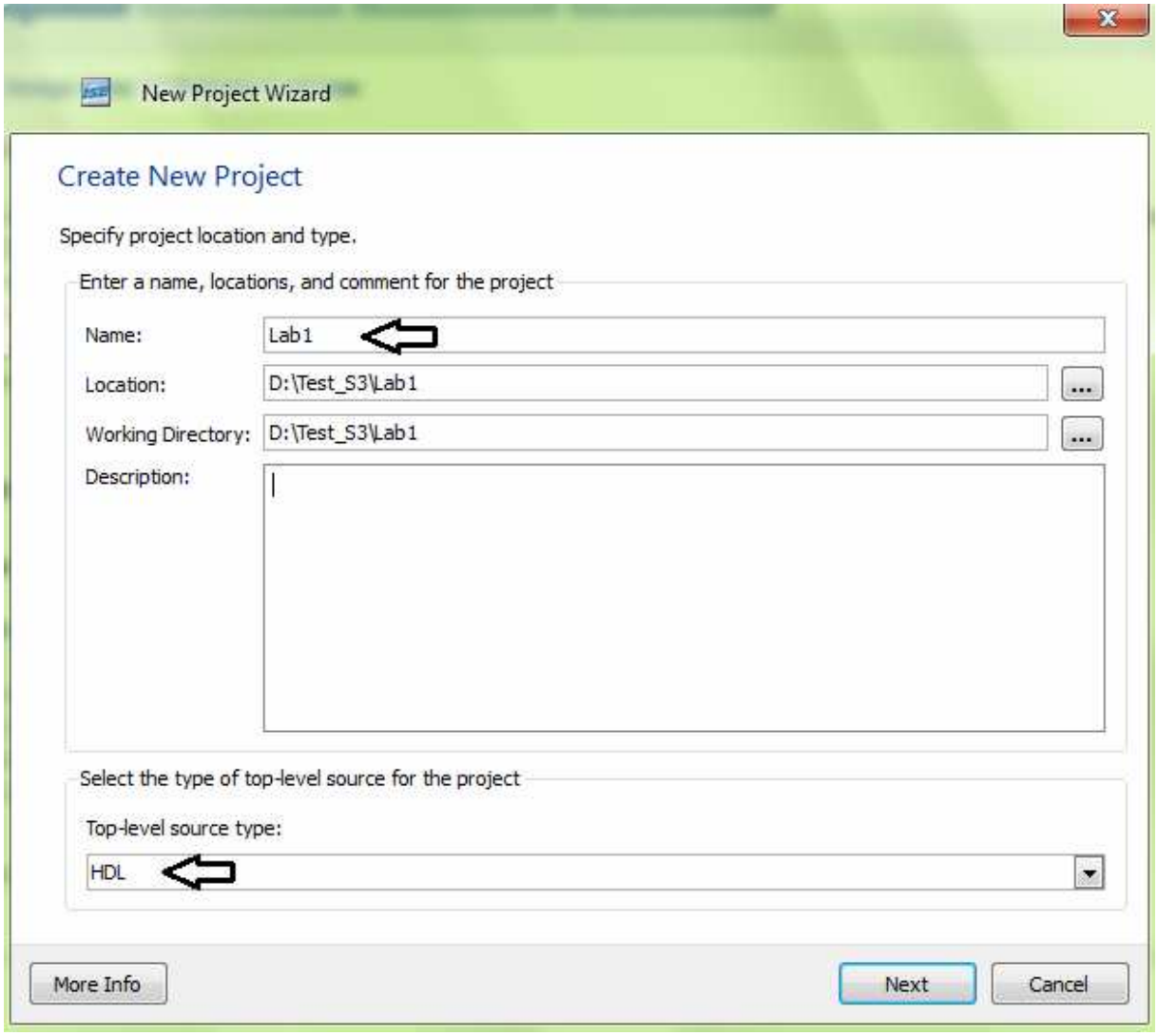
Khởi động chương trình chính: ***Project Navigator***. Chương trình nằm trong *Program List* hoặc trên màn hình dưới dạng *Shortcut.*



- Tạo một thiết kế mới bằng cách chọn: ***File → New Project…***

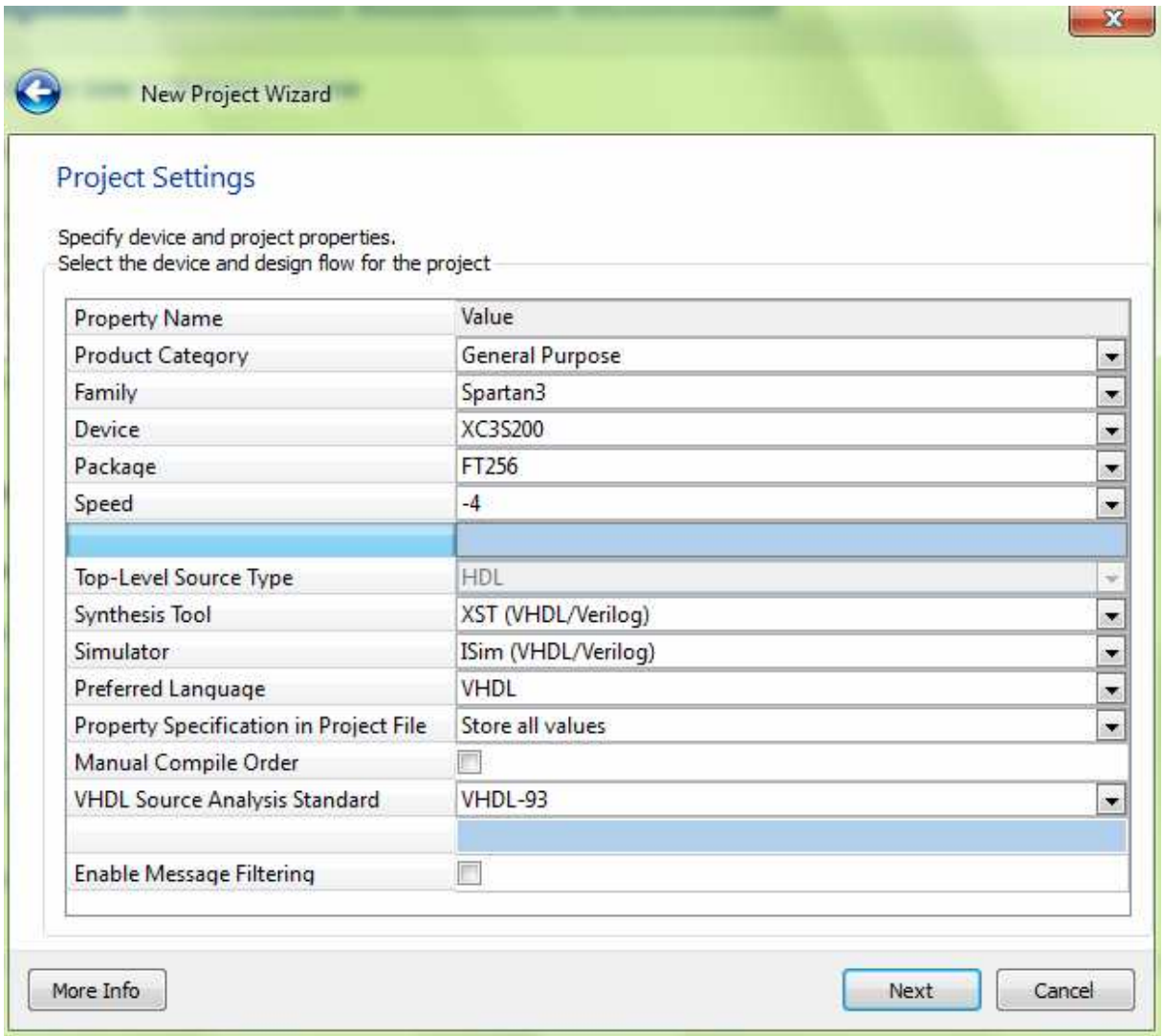


Chương trình sẽ hiện ra cửa sổ hỗ trợ, đặt tên và chọn vị trí lưu cho chương trình. Chú ý rằng loại ngôn ngữ cấp cao là HDL.



Sau khi nhấn ***Next***, cửa sổ tiếp theo cho phép lựa chọn các thông số và thuộc tính của chương trình đang thực hiện:

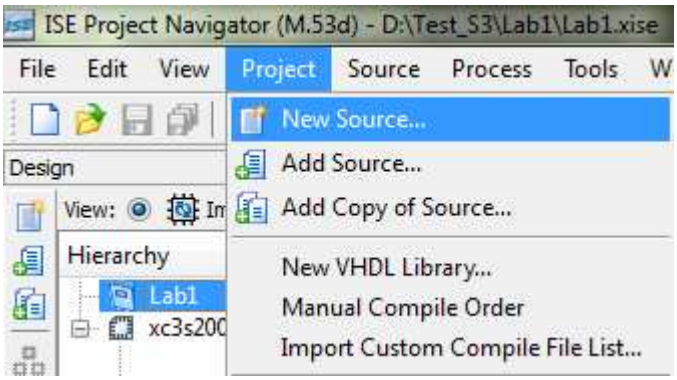
- Họ thiết bị: Spartan 3
- Thiết bị: XC3S200
- Gói: FT256 (FPGA có 256 chân)
- Tốc độ: -4 (tốc độ thông thường)
- Công cụ mô phỏng: ISim (ModelSim XE không còn được hỗ trợ bởi Xilinx)
- Ngôn ngữ : VHDL



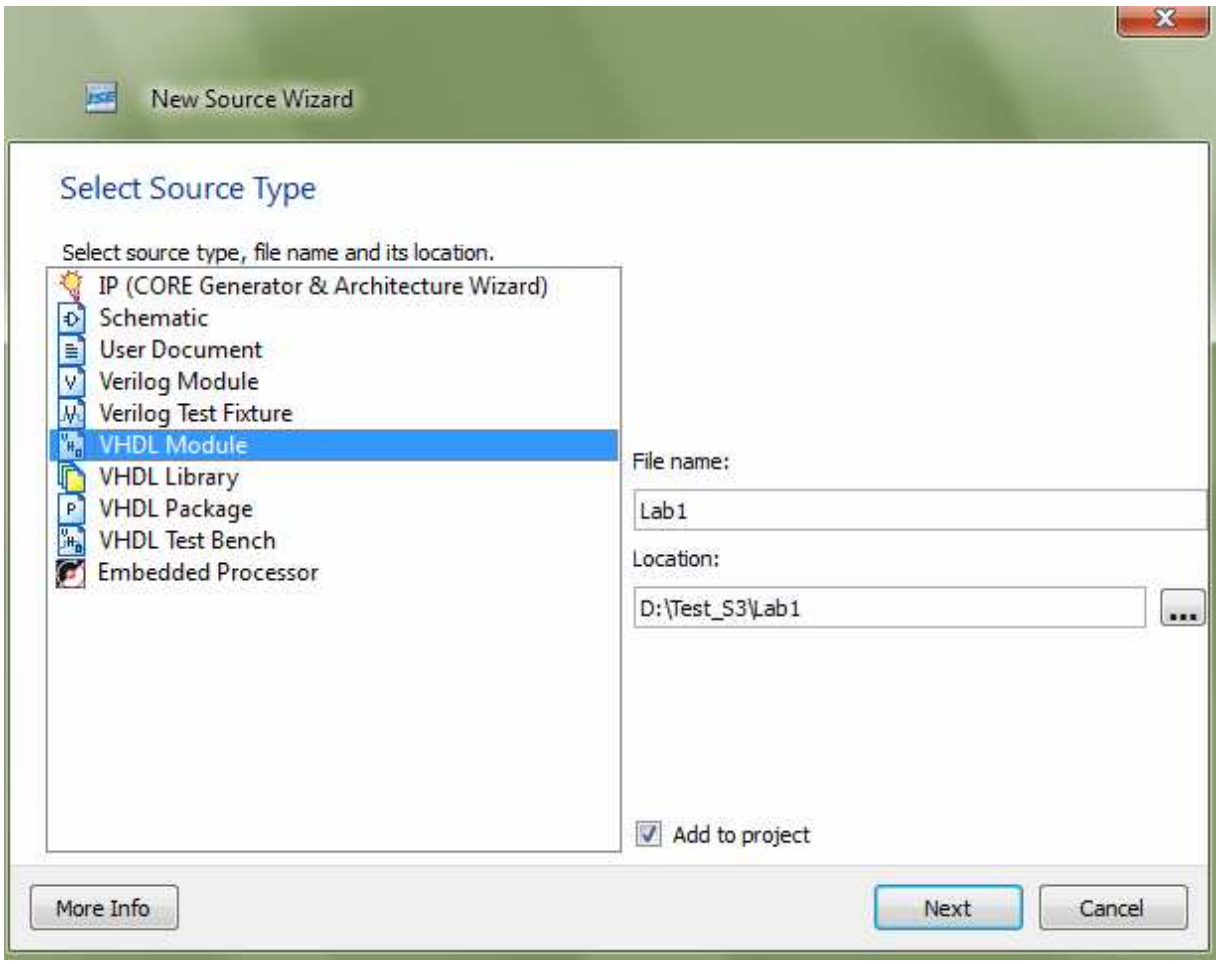
Nhấn ***Next*** để sang cửa sổ tóm tắt; tiếp tục nhấn ***Finish*** để kết thúc.

Cửa sổ làm việc chính **ISE Project Navigator** được khởi động.

- Chọn ***Project → New Source…***



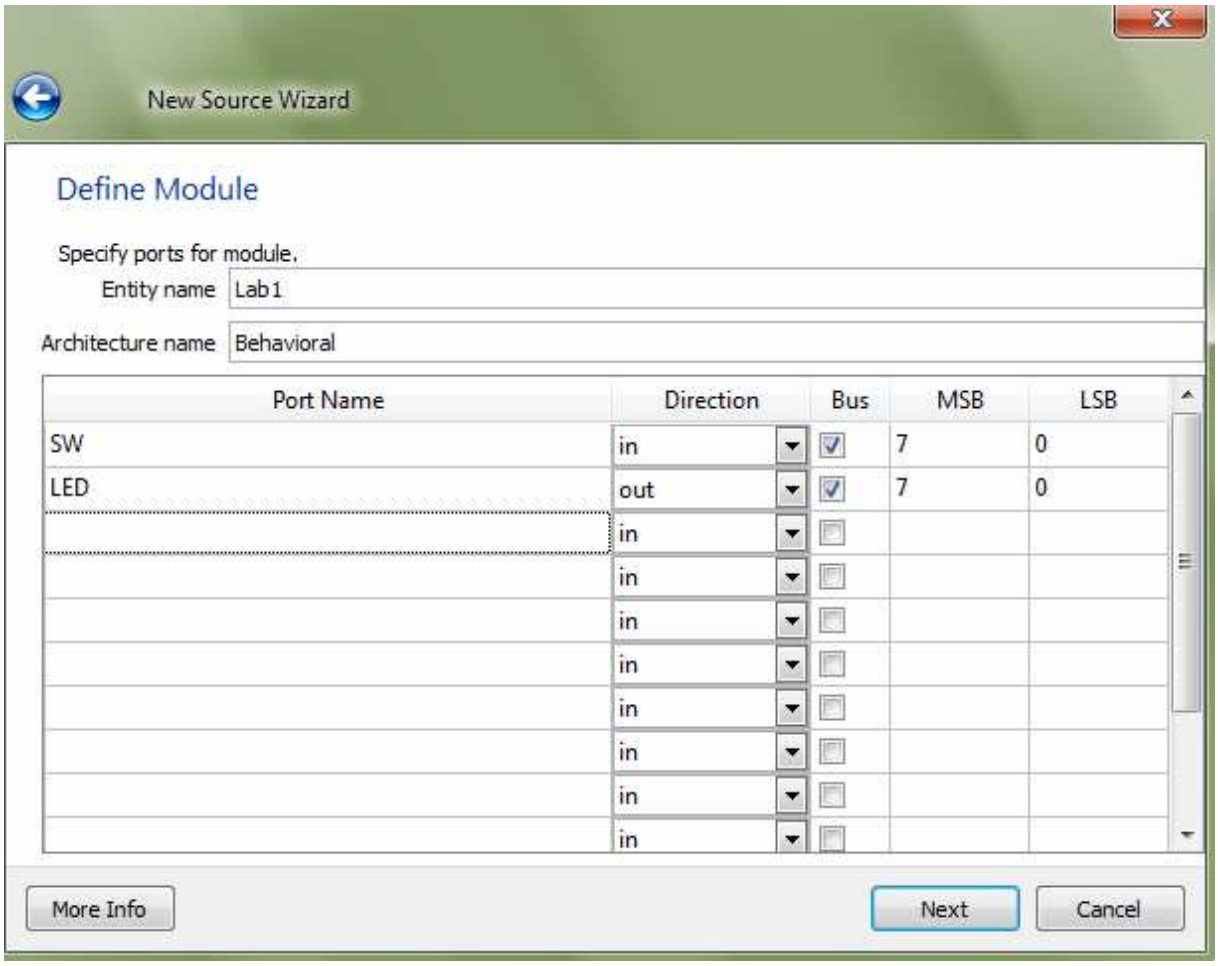
- Chọn ***VHDL Module*** trong cửa sổ tiếp theo và đặt tên cho tập tin.



- Định nghĩa thông số cần thiết cho thiết kế.

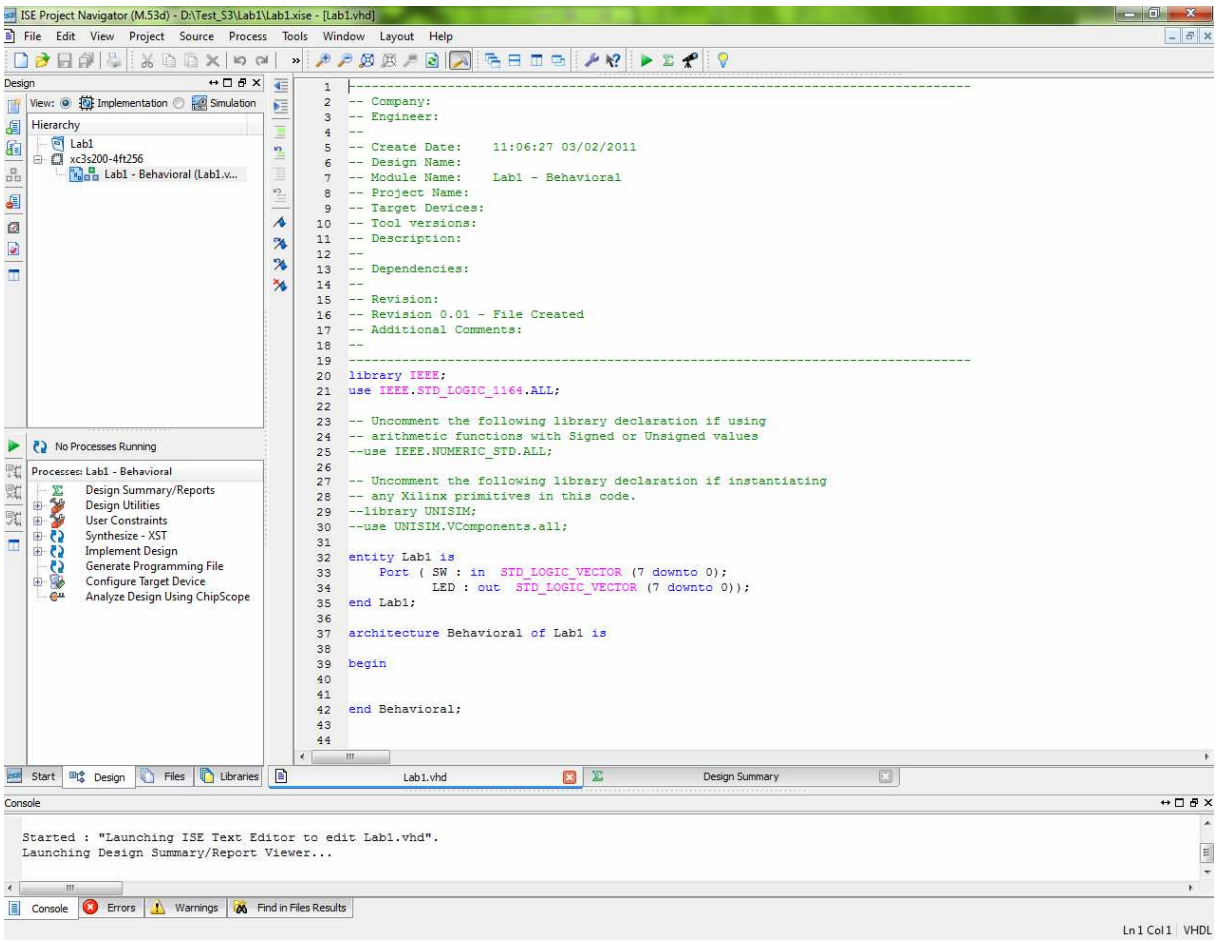
Ví dụ: Đầu vào 8 công tắc, SW được xem như đường bus có 8 bit.

Đầu ra 8 đèn LED, tương tự như trên.



Nhấn ***Next*** để sang cửa sổ tóm tắt, tiếp tục nhấn ***Finish*** để hoàn tất.

- Giao diện chương trình chính



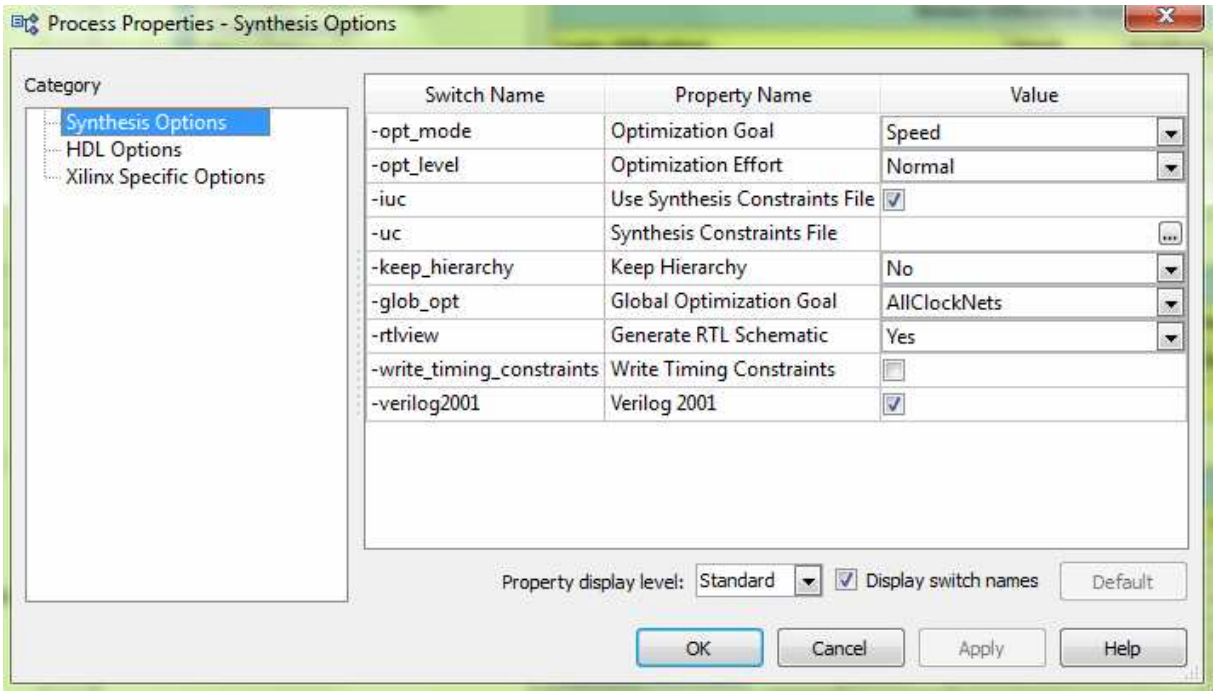

```
1  ----------------------------------------------------------------------------------
2  -- Company:
3  -- Engineer:
4  --
5  -- Create Date:    11:06:27 03/02/2011
6  -- Design Name:
7  -- Module Name:    Lab1 - Behavioral
8  -- Project Name:
9  -- Target Devices:
10 -- Tool versions:
11 -- Description:
12 --
13 -- Dependencies:
14 --
15 -- Revision:
16 -- Revision 0.01 - File Created
17 -- Additional Comments:
18 --
19 ----------------------------------------------------------------------------------
20 library IEEE;
21 use IEEE.STD_LOGIC_1164.ALL;
22
23 -- Uncomment the following library declaration if using
24 -- arithmetic functions with Signed or Unsigned values
25 --use IEEE.NUMERIC_STD.ALL;
26
27 -- Uncomment the following library declaration if instantiating
28 -- any Xilinx primitives in this code.
29 --library UNISIM;
30 --use UNISIM.VComponents.all;
31
32 entity Lab1 is
33     Port ( SW : in  STD_LOGIC_VECTOR (7 downto 0);
34            LED : out  STD_LOGIC_VECTOR (7 downto 0));
35 end Lab1;
36
37 architecture Behavioral of Lab1 is
38
39 begin
40
41
42 end Behavioral;
43
44
```

Trong bài ví dụ đầu tiên, thiết kế đơn giản chỉ là bật tắt đèn LED bằng SW. Chương trình như sau:

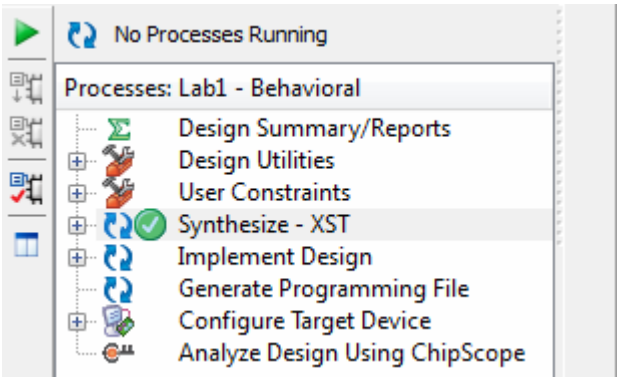
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity Lab1 is
        Port ( SW : in STD_LOGIC_VECTOR (7 downto 0);
               LED : out STD_LOGIC_VECTOR (7 downto 0));
end Lab1;
architecture Behavioral of Lab1 is
begin
        LED <= SW;
end Behavioral;
```

- Sau khi viết xong chương trình, bước kế tiếp là **Tổng hợp** thiết kế.

Trước khi tổng hợp, có thể kích chuột phải vào ***Synthesize-XST → Process Properties…*** để thay đổi đặc tính theo ý muốn. Đối với những thiết kế đơn giản, không cần thay đổi Process Properties.

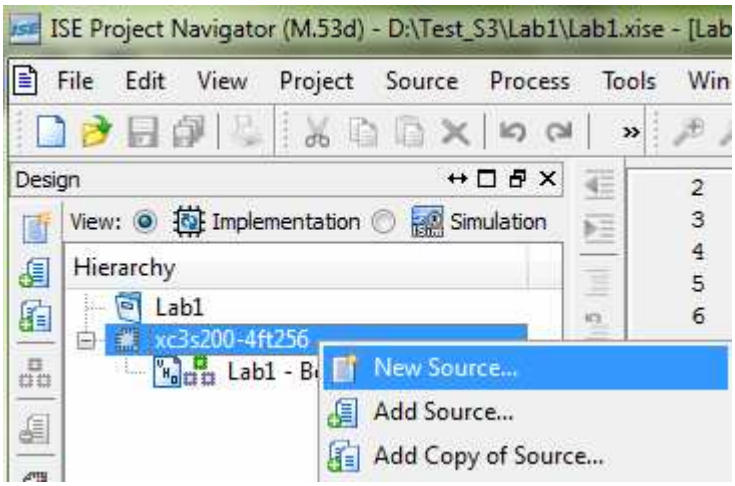


Để tổng hợp thiết kế, kích trái chuột liên tiếp 2 cái vào dòng ***Synthesize-XST*** để chạy công đoạn tổng hợp**.** Nếu thiết kế có lỗi, cảnh báo lỗi hiện ra và thiết kế cần phải sửa lại cho đúng.

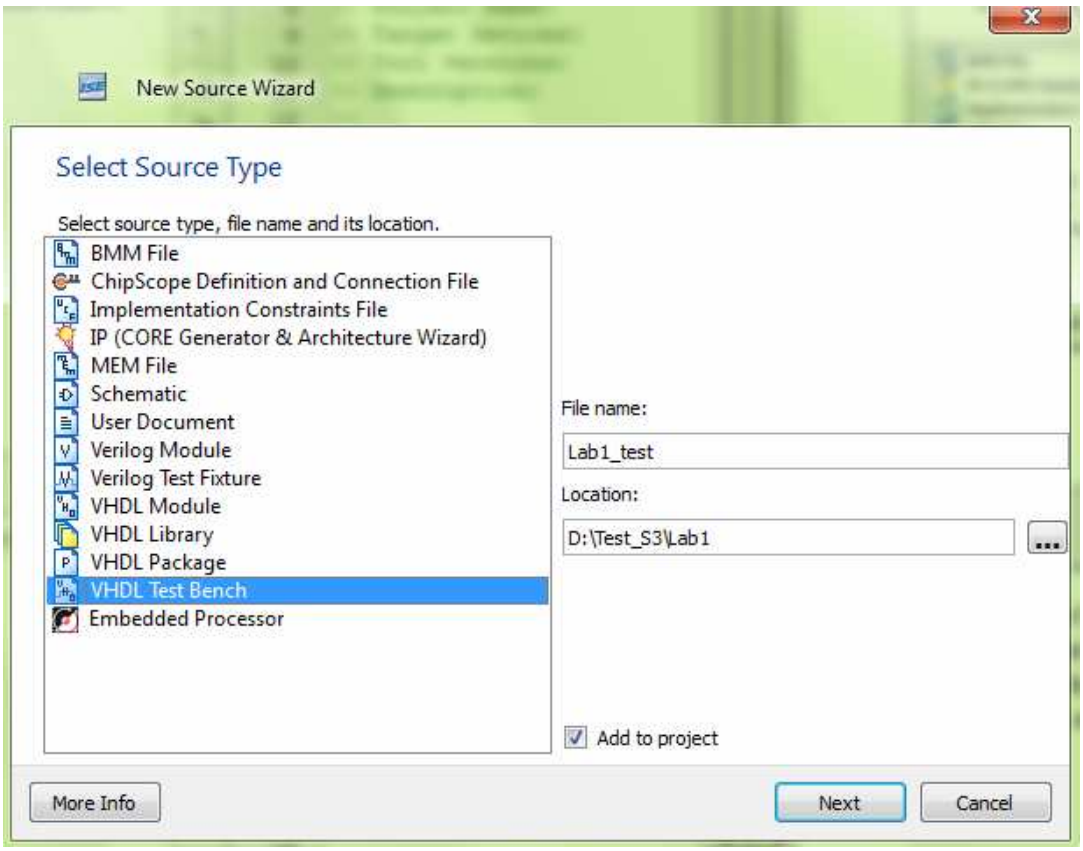


Sau khi tổng hợp thành công, bước tiếp theo là **Mô phỏng**. Quá trình mô phỏng sẽ cho thấy kết quả ở dạng sóng tín hiệu; dựa vào dạng sóng để kiểm tra lỗi, sửa đổi hoặc cải tiến thiết kế.

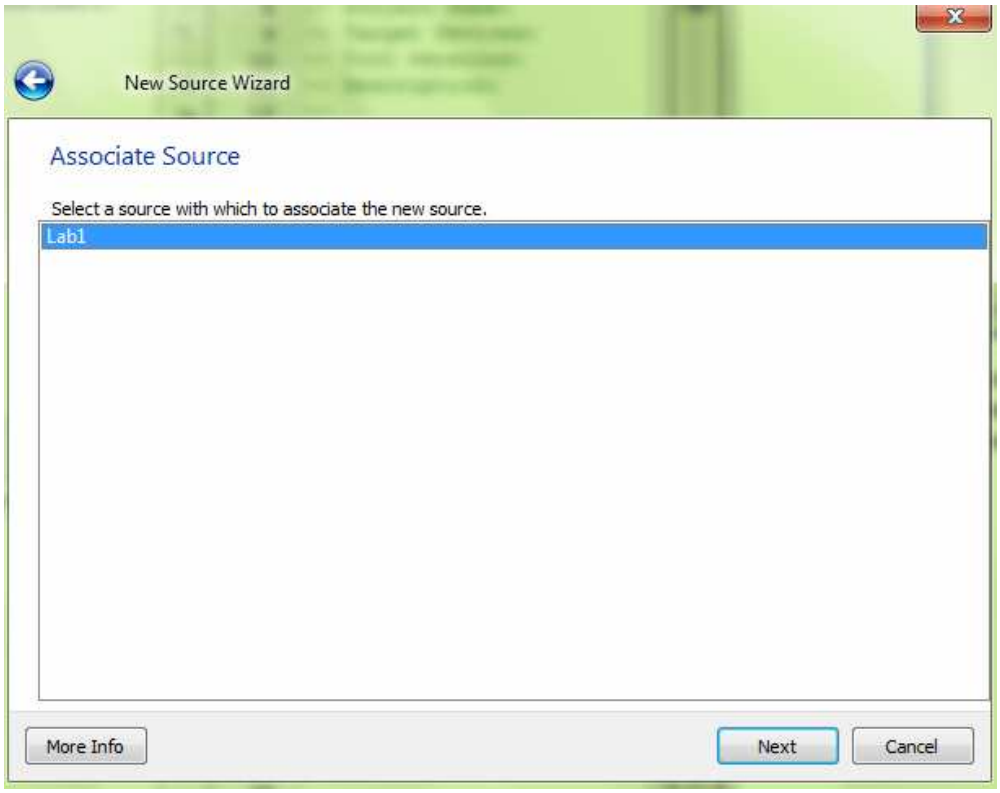
- Chọn dòng có tên thiết bị, kích phải chuột rồi chọn ***New Source…***



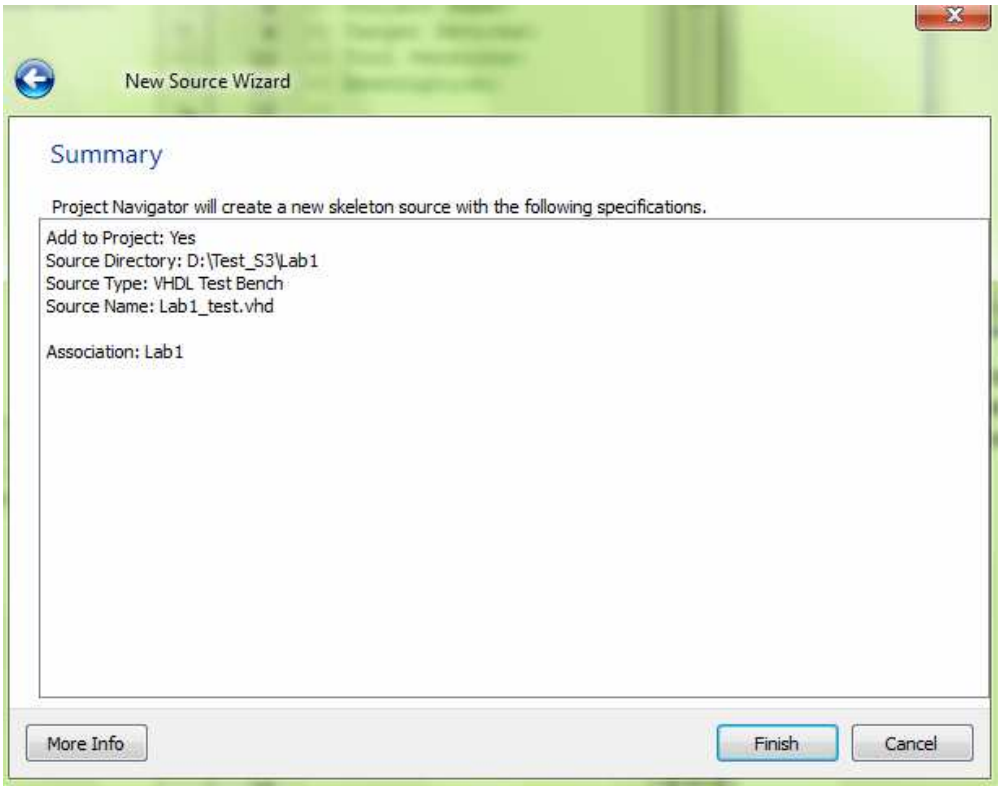
- Nhấn chọn ***VHDL Test Bench***, đặt tên cho tập tin mô phỏng



Nếu chương trình có nhiều tập tin nguồn, cần chọn tập tin chính cần mô phỏng. Trong ví dụ này, chỉ có 1 tập tin là *Lab1* cần mô phỏng.

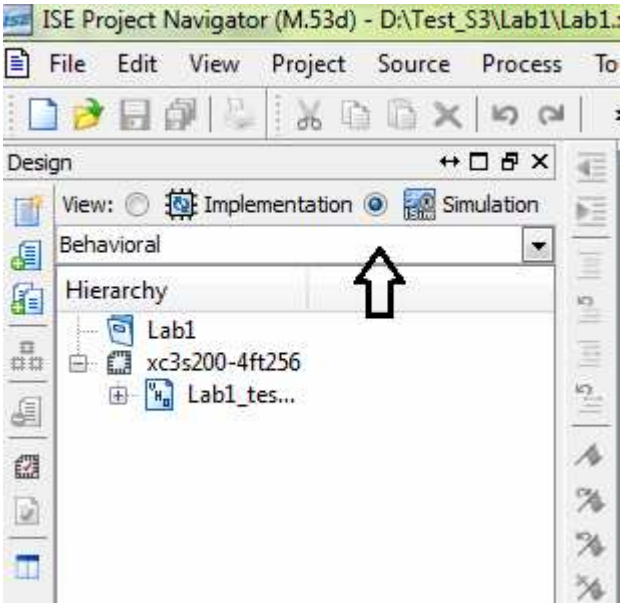


Nhấn ***Next***

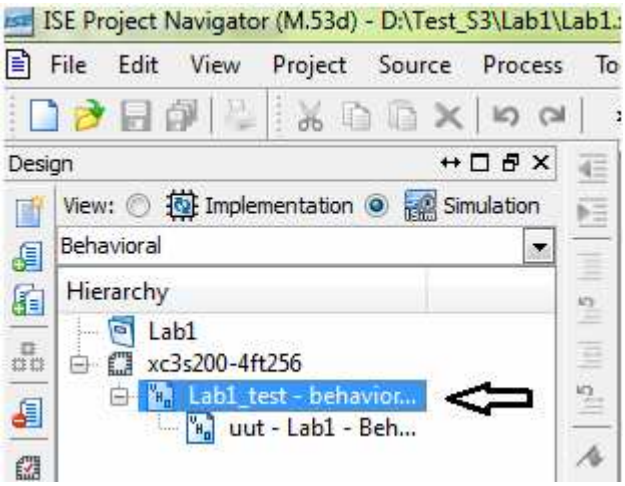


Nhấn ***Finish***

Quay lại chương trình chính, chọn nút ***Simulation***

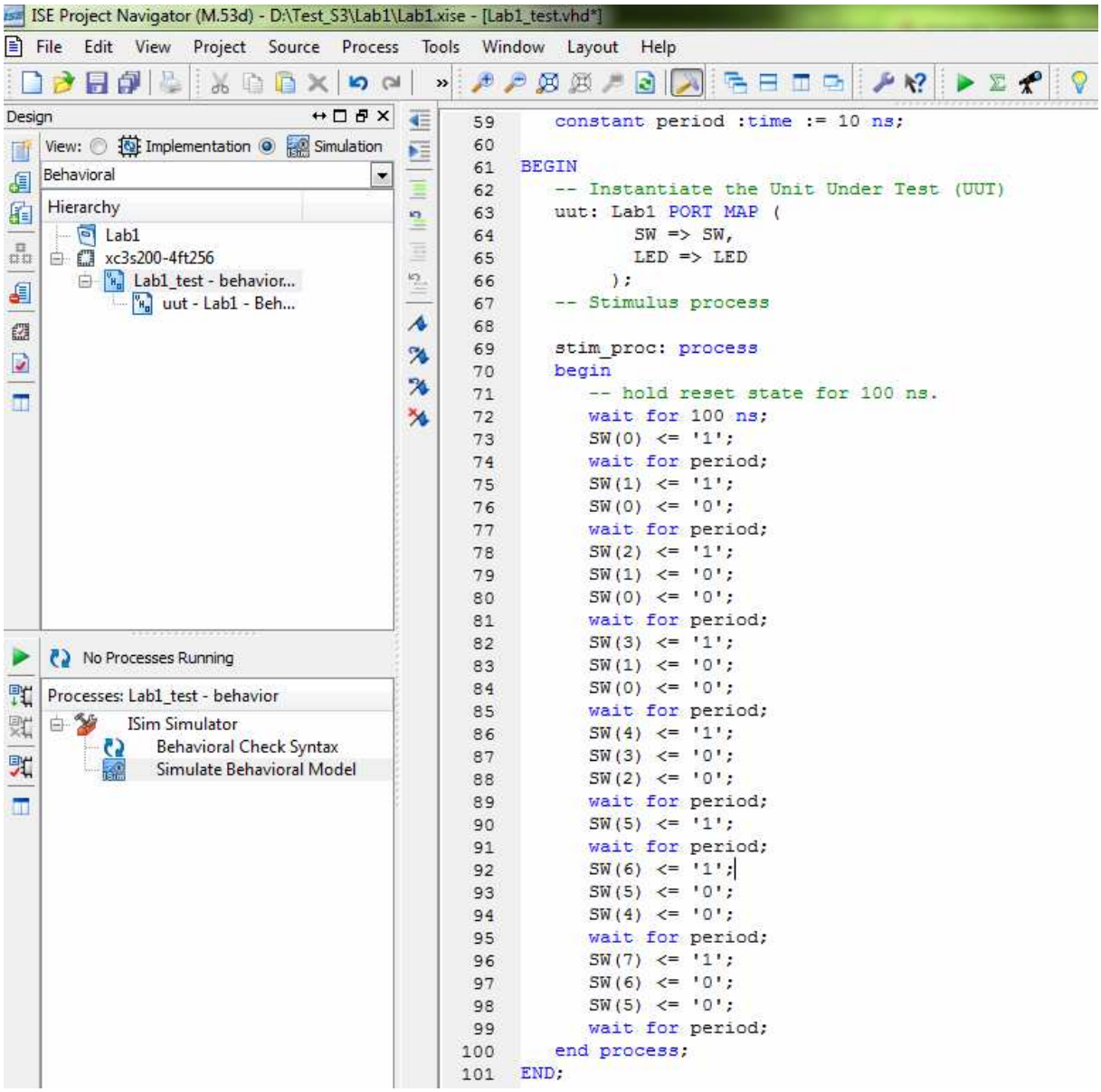


Chọn tập tin mô phỏng



- Viết chương trình mô phỏng.

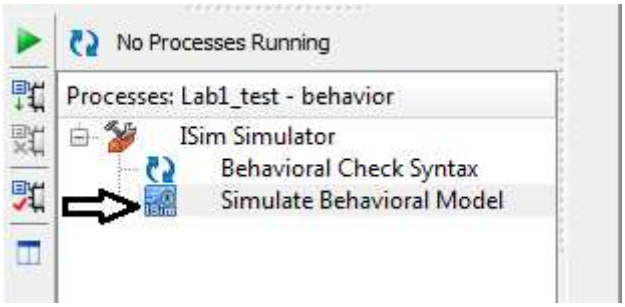
Ví dụ:



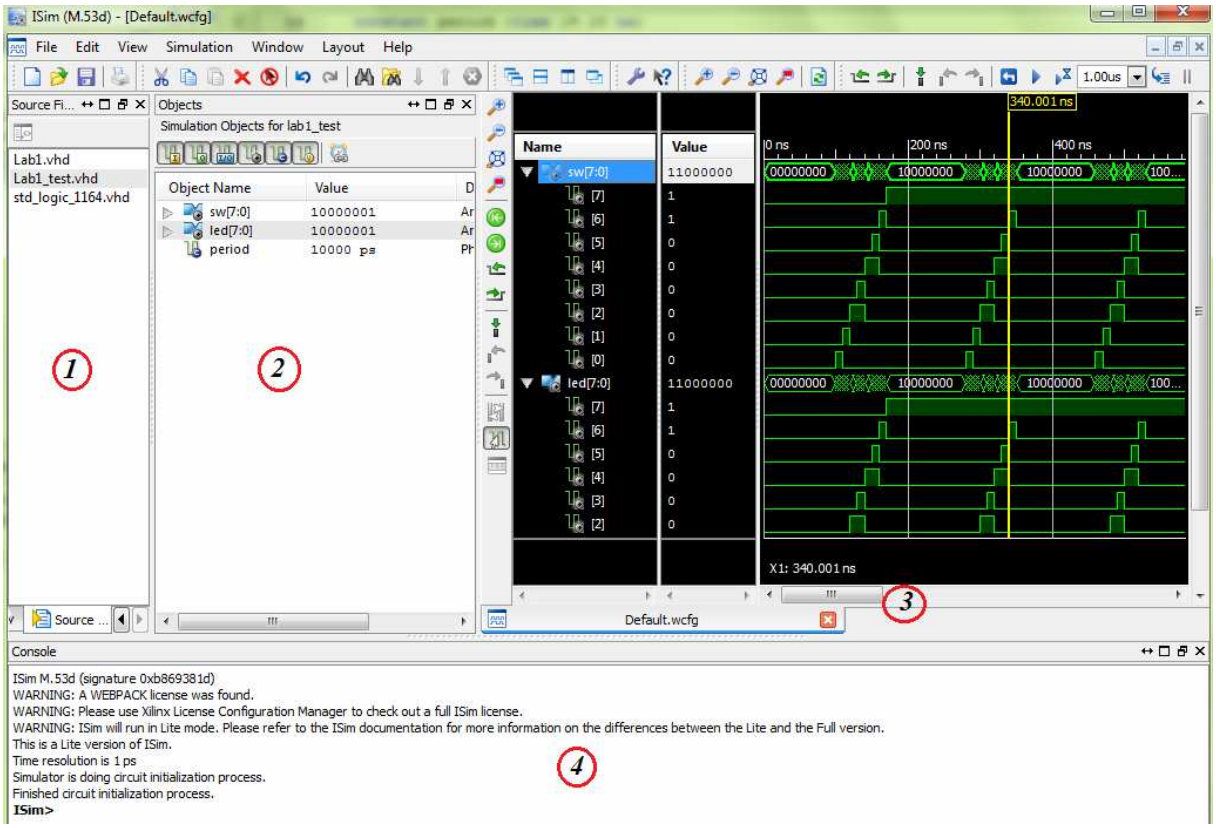
Chương trình mô phỏng mẫu được tự động tạo ra nhưng thường có nhiều thành phần hơn cần thiết. Trong ví dụ này, phần chính của tập tin mô phỏng chỉ như hình trên.

Cửa sổ phía dưới có biểu tượng ***ISim Simulator***

- Kích hoạt chương trình ***Simulate Behavioral Model***



Chương trình mô phỏng Isim được khởi động

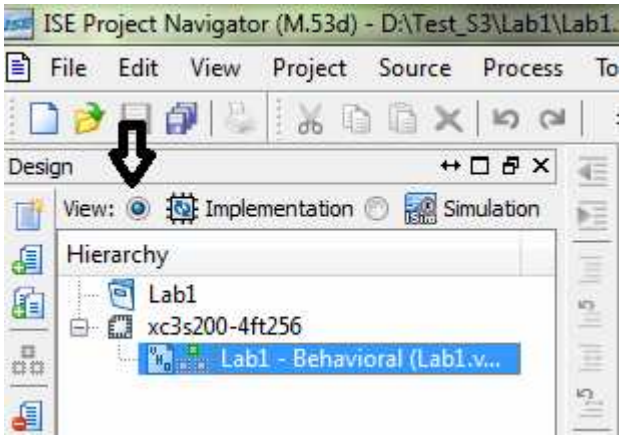


Cửa sổ chính có bốn phần, phần số 3 hiển thị dạng sóng. Trên đó, có các công cụ chính như là *Phóng to, Thu nhỏ, Xem toàn bộ và Xem vùng được đánh dấu*.

Bên cạnh là các nút *Mô phỏng lại, Chạy mô phỏng, Chạy đến thời gian đã chọn, Chạy từng bước* và *Dừng* .

Sau khi mô phỏng, sửa đổi để kết quả như yêu cầu đặt ra. Bước tiếp theo là đặt các chế độ thời gian và gán chân cho FPGA.

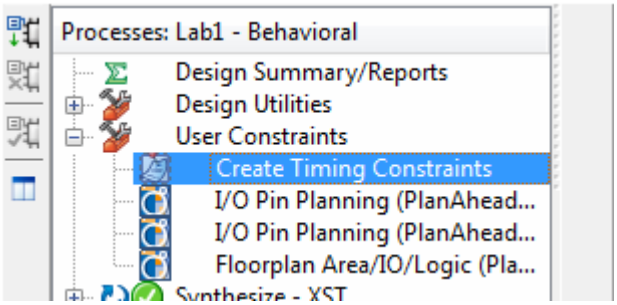
- Tắt chương trình mô phỏng ***ISim***.
- Quay lại chương trình chính, chọn lại chế độ ***Implementation***

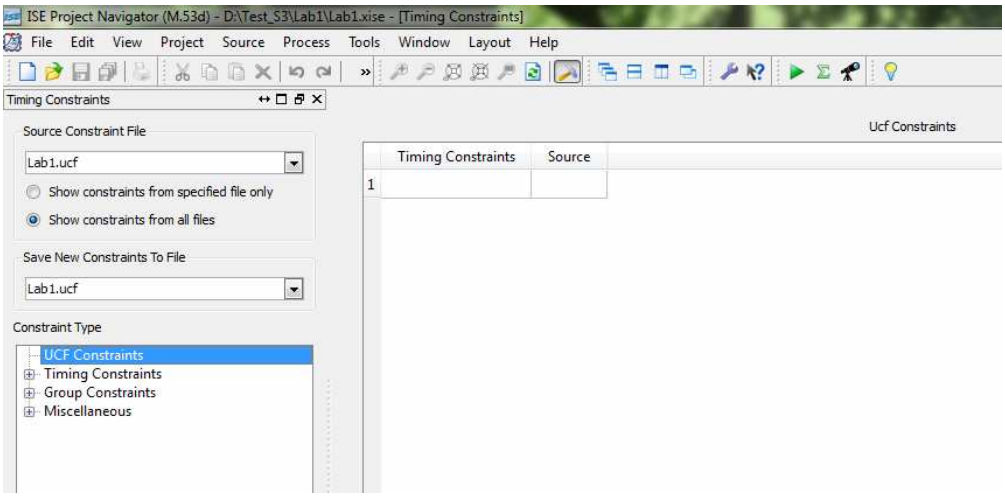


- Kích hoạt công cụ ***Create Timing Constraints*** trong phần ***User Constraints***

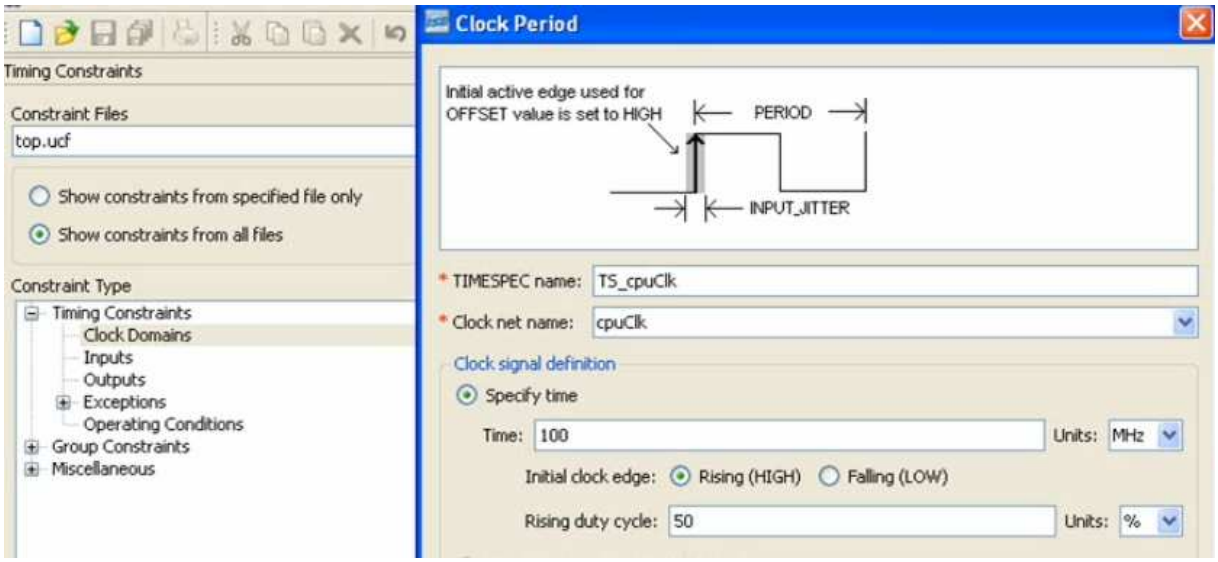
Nếu tập tin (*.ucf) chưa được tạo ra, chương trình sẽ tự động tạo và thông báo.

- Nhấn ***Yes*** để chấp nhận tạo tập tin (*.ucf) mới





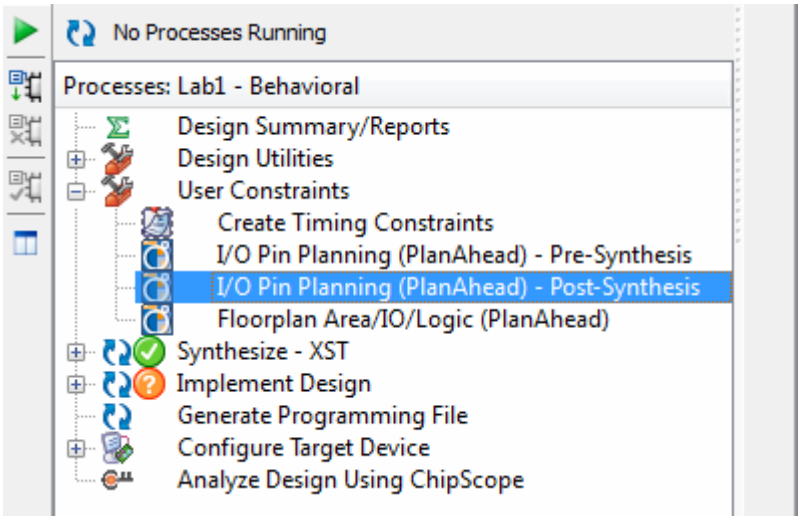
Đặt các chế độ liên quan đến xử lí xung nhịp, thời gian trễ,…



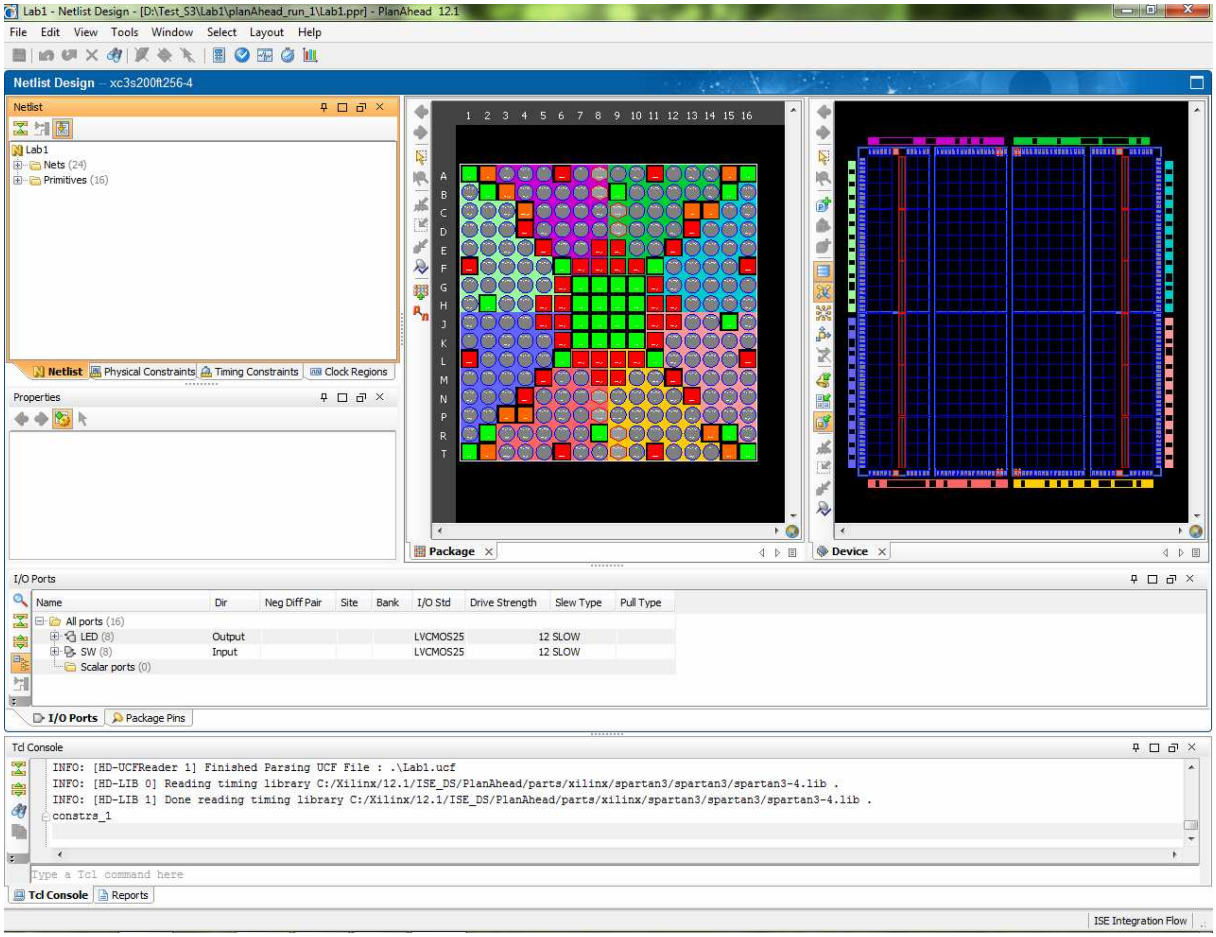
Trong phần ví dụ này không sử dụng xung CLK nên tập tin (*.ucf) không cần sửa đổi.

Phần tiếp theo là gán chân cho FPGA.

- Khởi động công cụ ***I/O Pin Planning (PlanAhead) – Post-Synthesis***



- Chương trình PlanAhead



Chương trình cho phép gán chân linh hoạt, chỉ cần kéo và thả chân cần gán vào vị trí được minh họa.

Bảng chân định sẵn của công tắc SW

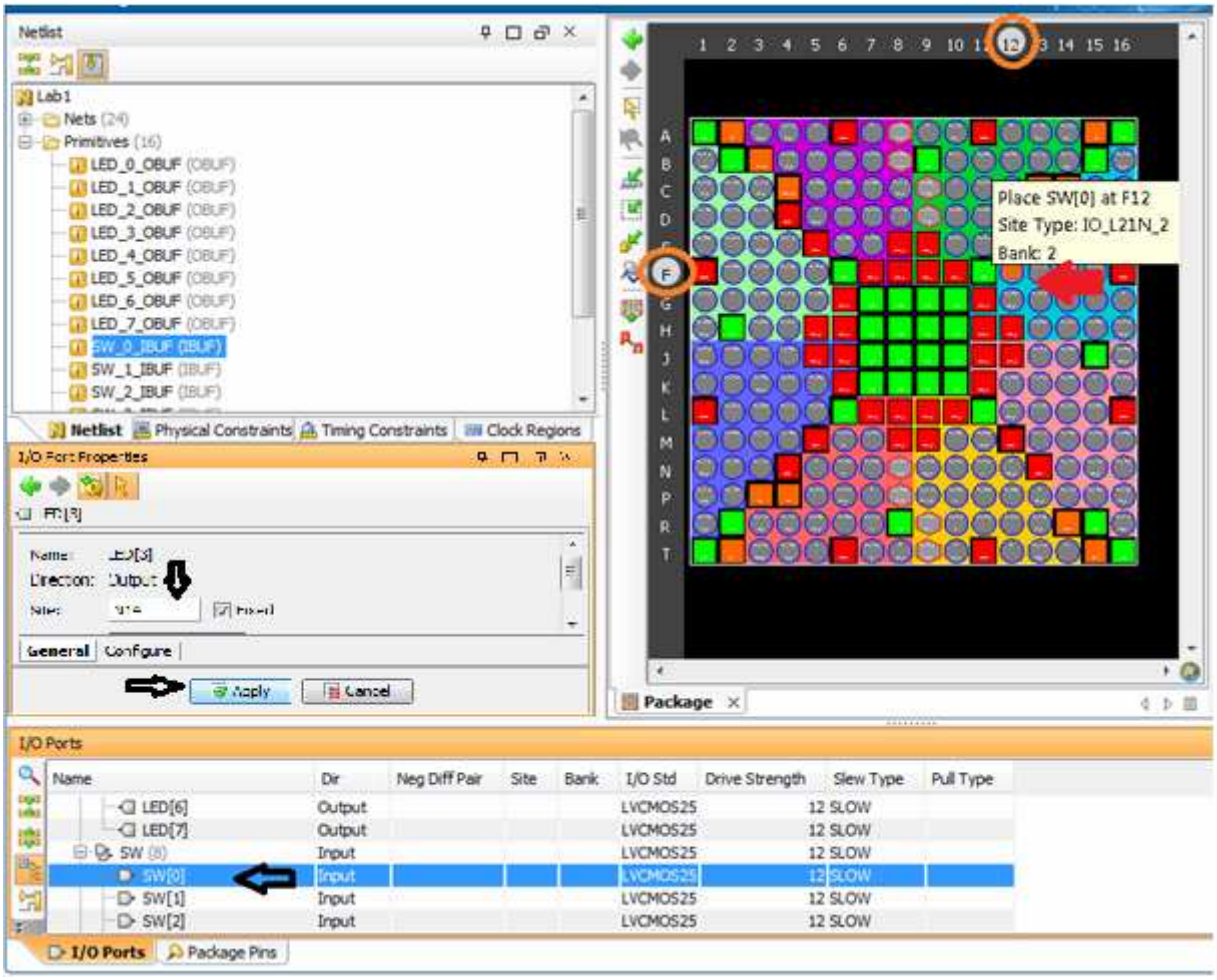
Switch	SW7	SW6	SW5	SW4	SW3	SW2	SW1	SW0
FPGA Pin	K13	K14	J13	J14	H13	H14	G12	F12

Bảng chân định sẵn của đèn LED

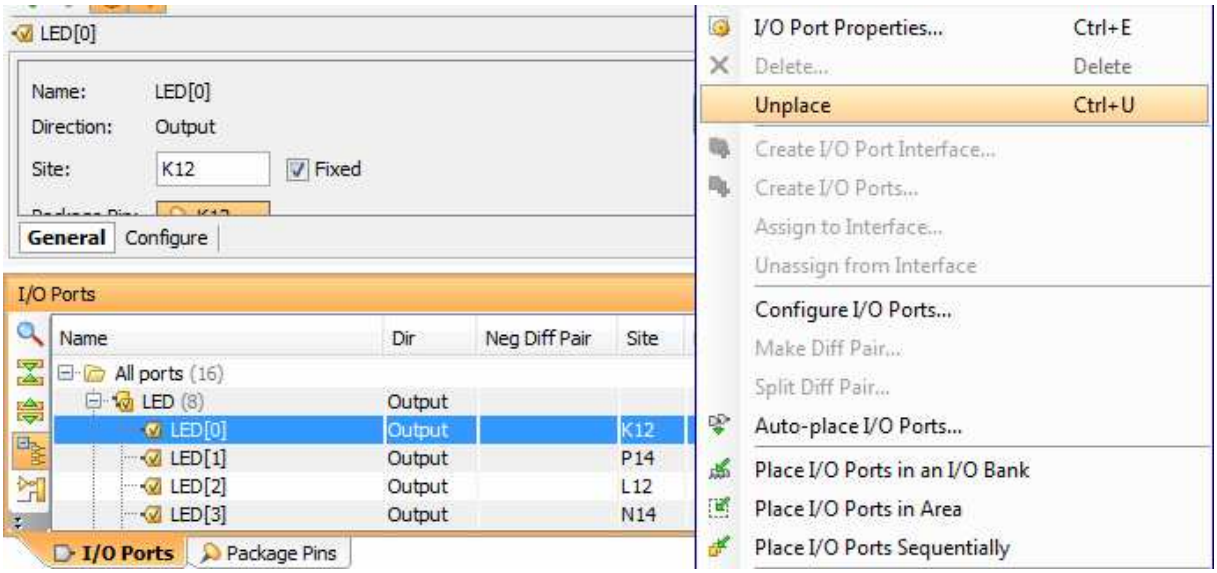
LED	LD7	LD6	LD5	LD4	LD3	LD2	LD1	LD0
FPGA Pin	P11	P12	N12	P13	N14	L12	P14	K12

- Hoặc là nhập tên chân vào ô ***Site*** trong cửa sổ ***I/O Port Properties***, nhấn ***Apply***

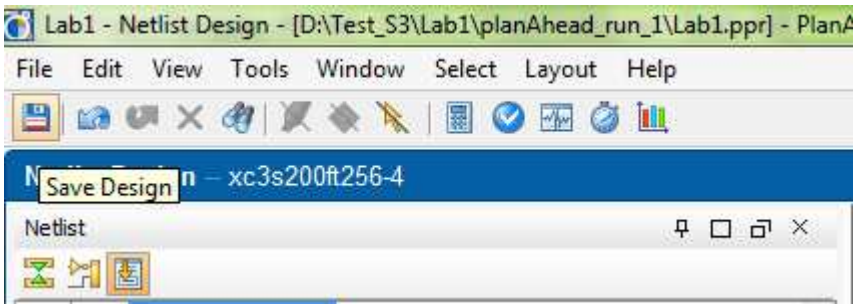
Ví dụ: SW[0] được quy định là chân F12.



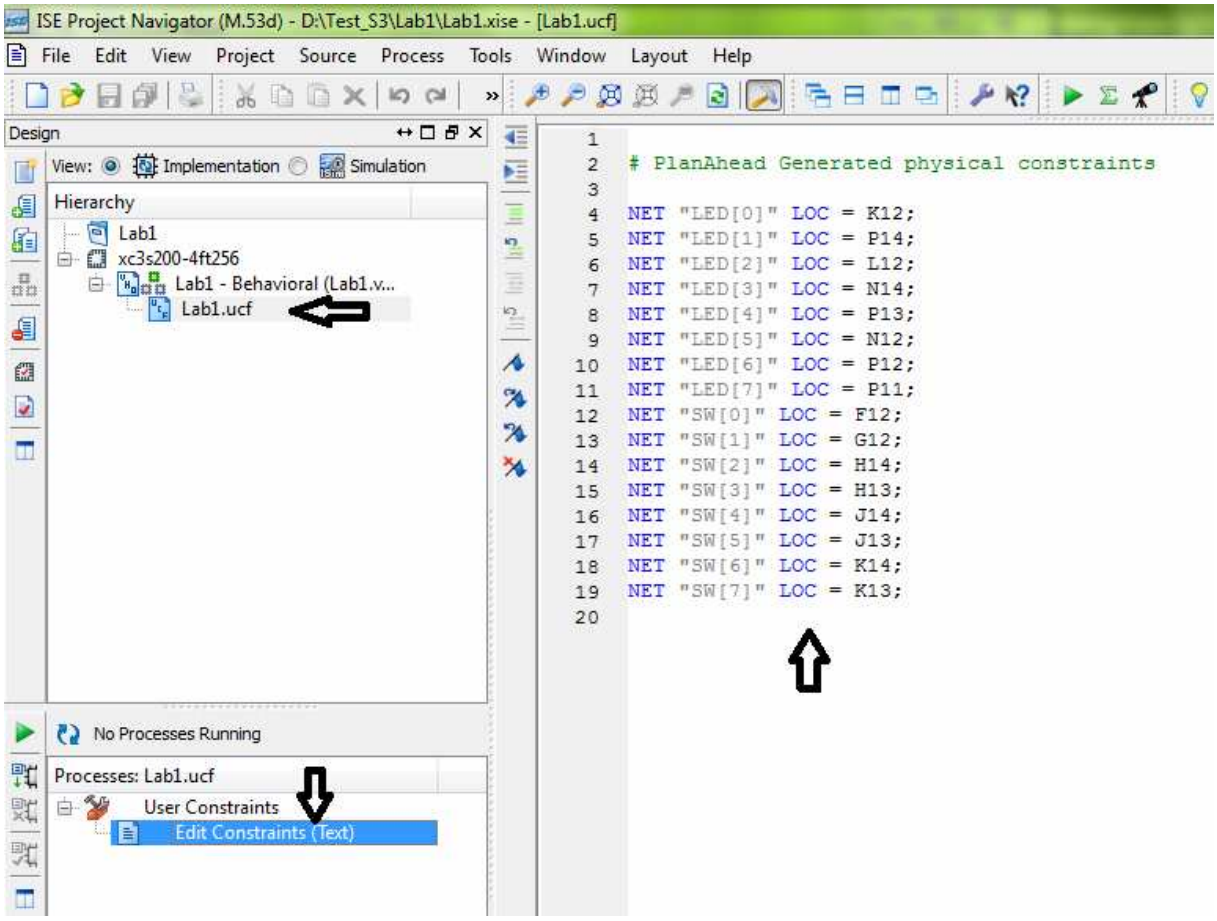
Nếu gán sai chân, kích phải chuột vào chân đó và chọn ***Unplace***



Sau khi hoàn tất việc gán chân, lưu dữ liệu rồi quay lại chương trình chính.

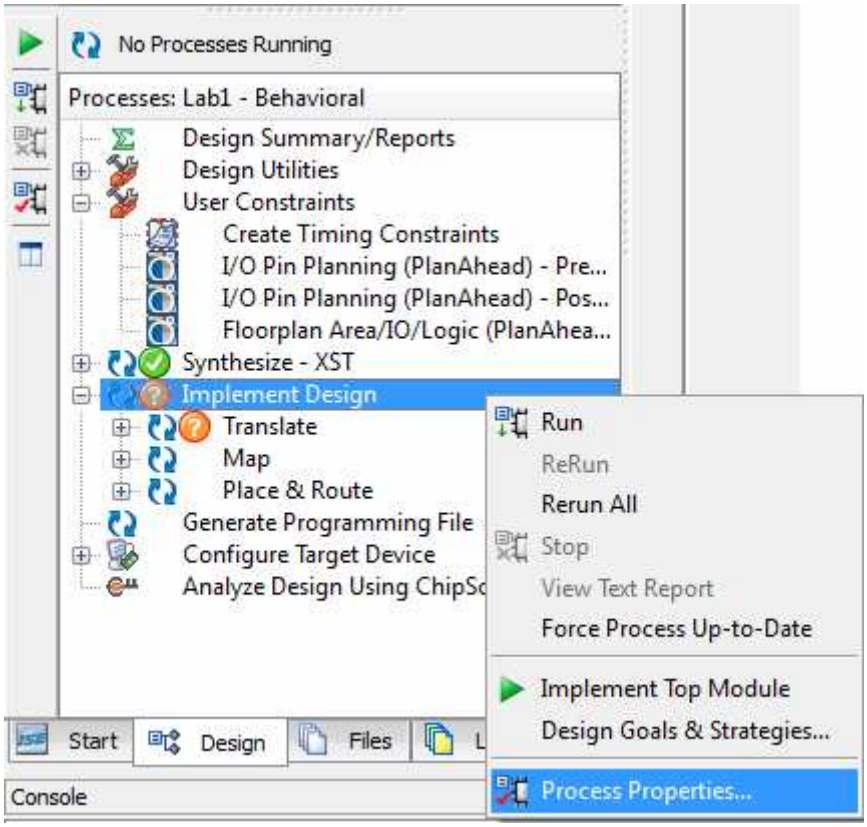


Có thể xem lại tập tin (*.ucf) bằng cách chọn ***Edit Constraints (Text)***



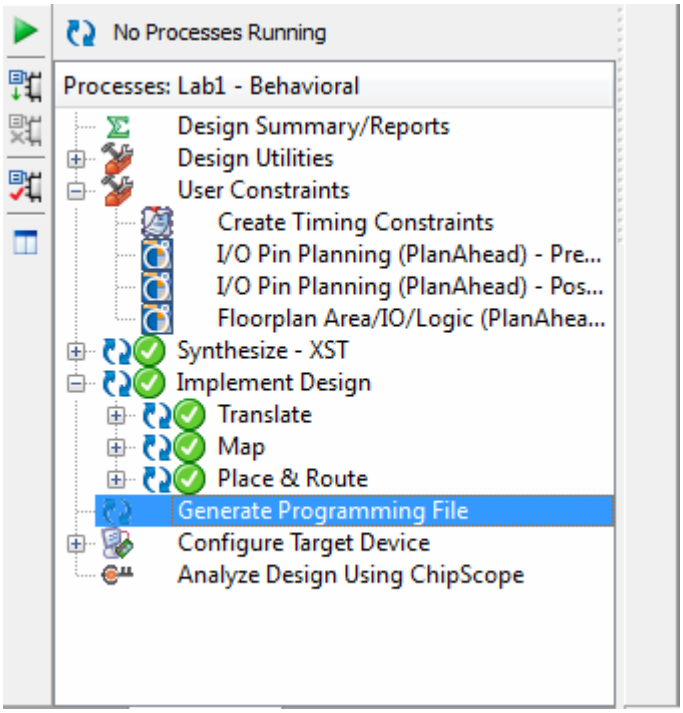
Công đoạn tiếp theo là **Thực hiện** thiết kế.

Đối với những thiết kế phức tạp, kích phải trên dòng ***Implement Design*** rồi chọn **Design Goals & Strategies**… hay **Process Properties**… để chọn các chế độ phù hợp.



Trong bài này, không cần can thiệp 2 phần trên.

- Kích hoạt chương trình ***Implement Design***

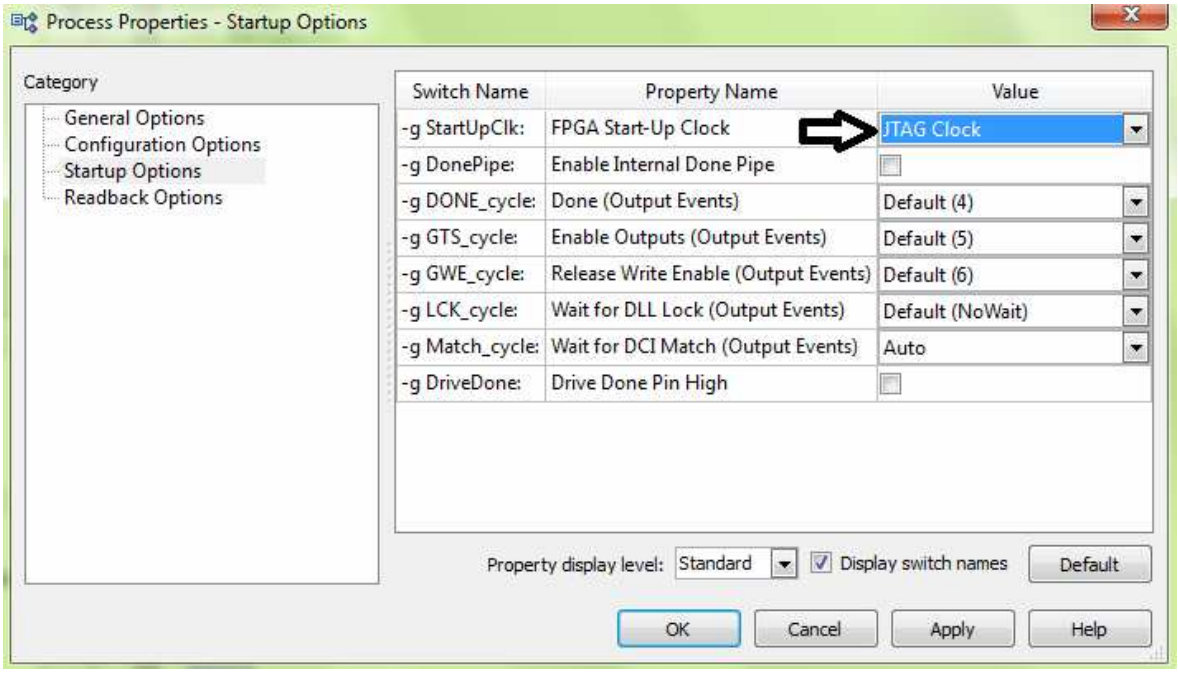


Tạo tập tin nạp vào FPGA.

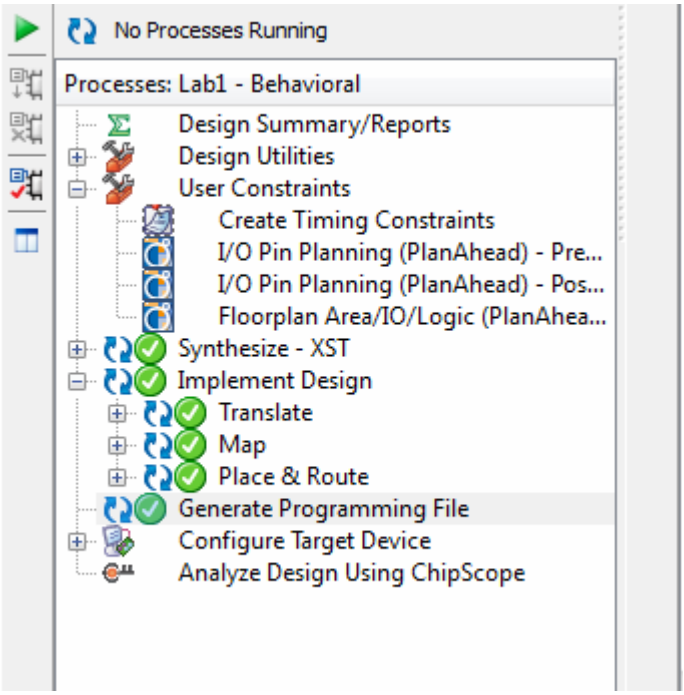
- Kích chuột phải trên dòng ***Generate Programming File → Process Properties***

Vì chương trình sẽ được nạp bằng chuẩn JTAG-USB nên phải chọn

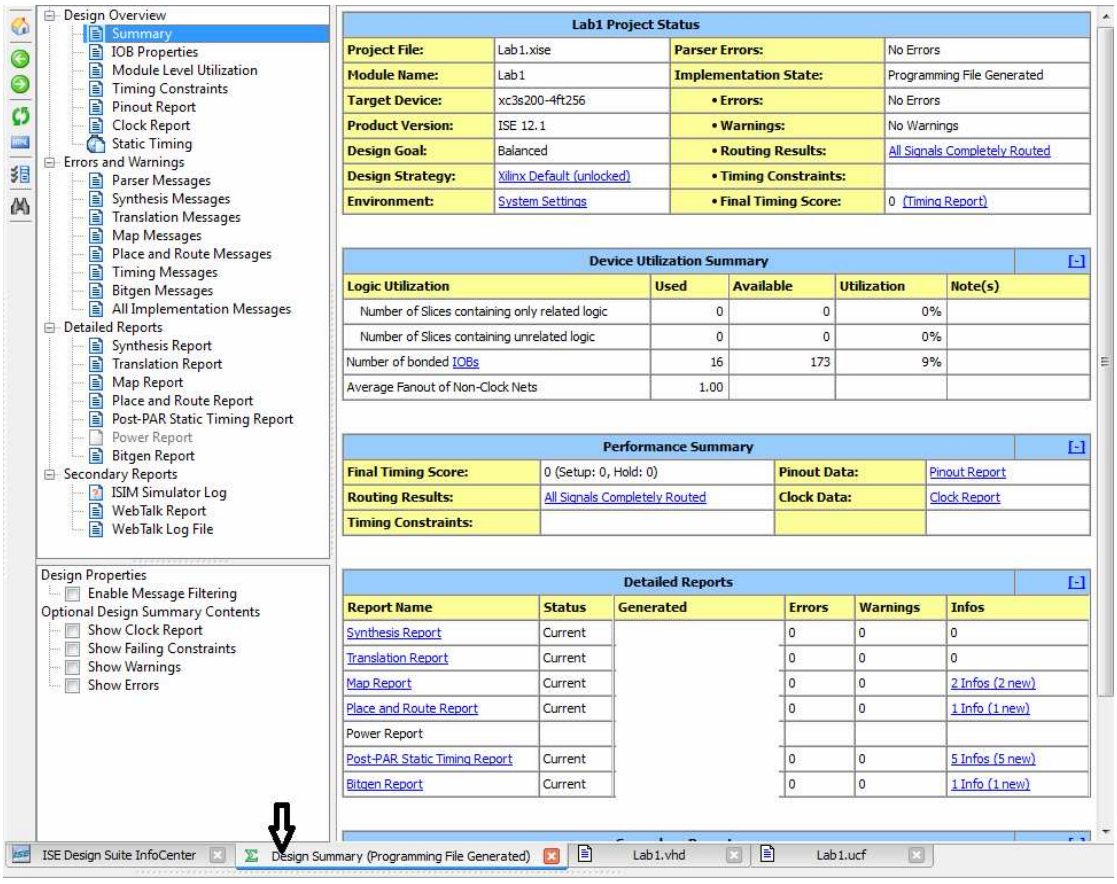
Startup Options → FPGA Start-Up Clock → JTAG Clock



Kích hoạt chương trình ***Generate Programming File***



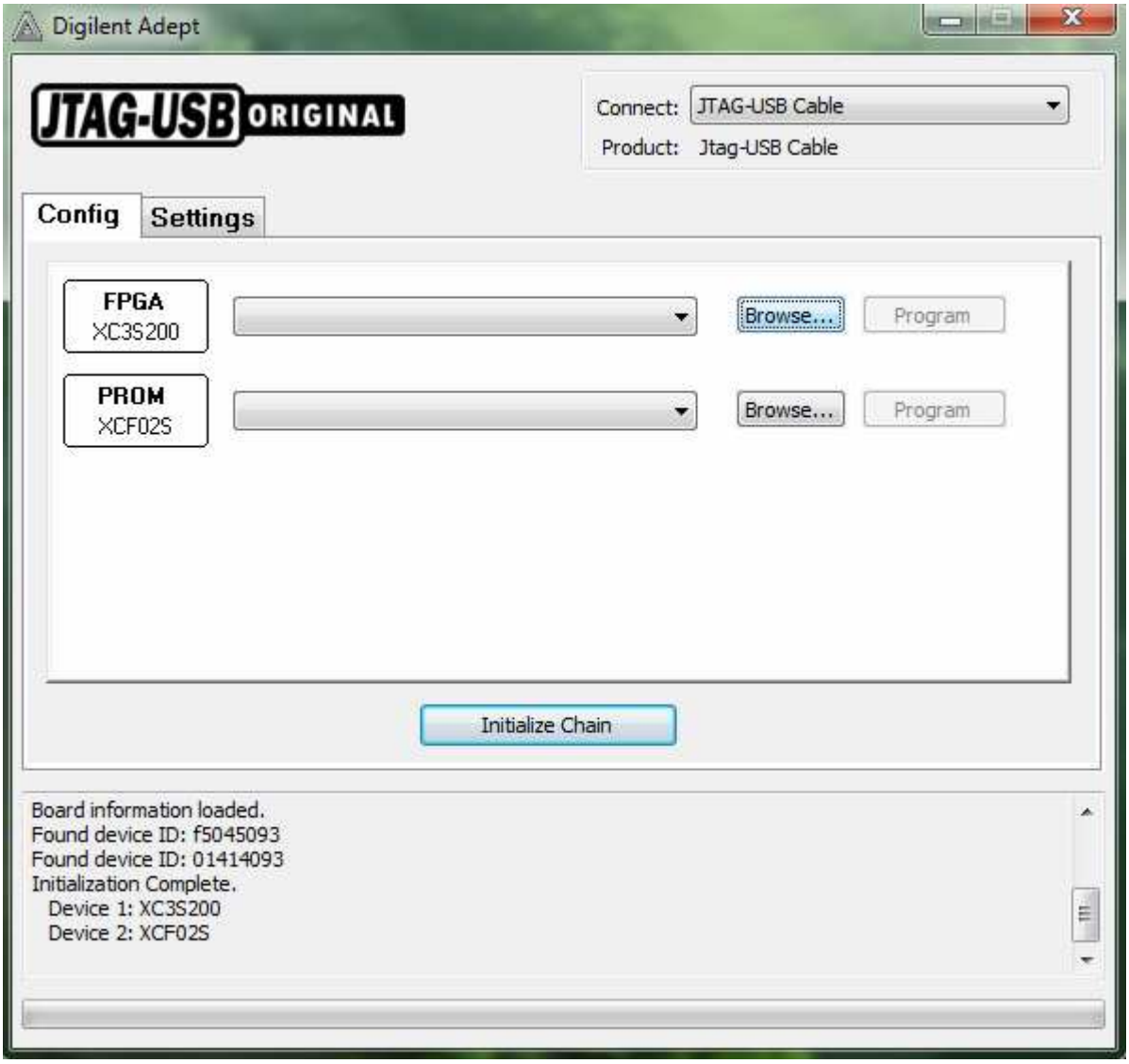
Sau khi hoàn tất, có thể kiểm tra sơ lược thiết kế qua bảng tóm tắt ở thẻ phía dưới của cửa sổ chương trình chính.



1.2.2 Nạp chương trình

Nếu sử dụng cáp của Xilinx, phần mềm iMPACT sẽ được sử dụng để nạp FPGA. Tuy nhiên, cáp JTAG-USB đi kèm thiết bị là của hãng Digilent; do vậy, phần mềm nạp FPGA là Adept. Phần mềm này và driver cho cáp USB có thể tải trực tiếp từ trang chủ của hãng Digilent.

- Khởi động chương trình Digilent Adept



Thông tin về thiết bị được hiển thị trên cửa sổ chương trình Adept.

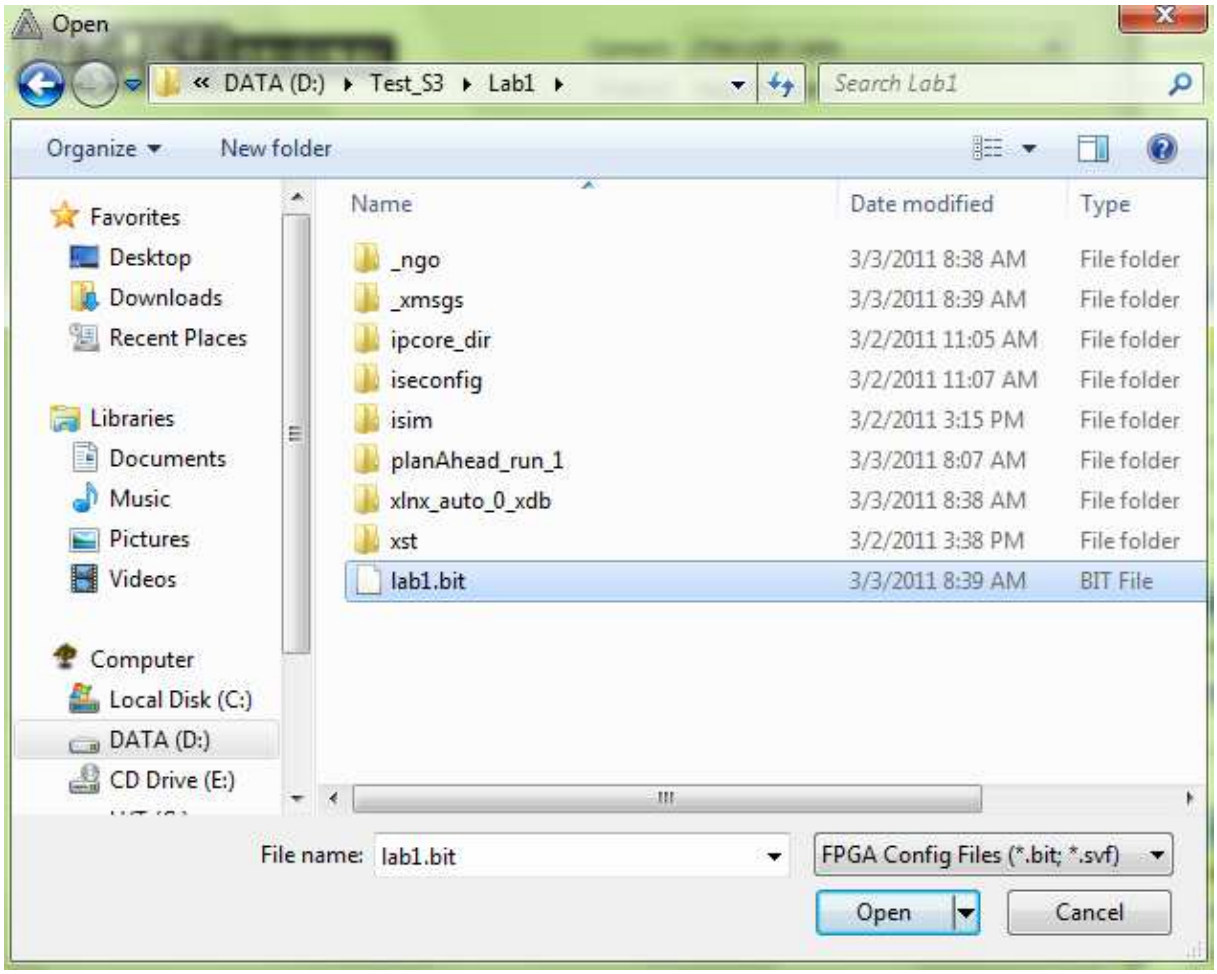
- Thiết bị 1: FPGA XC3S200
- Thiết bị 2: PROM XCF02S

Chương trình nạp cho FPGA sẽ mất đi khi ngắt điện. Để chương trình có thể hoạt động khi cấp nguồn, cần nạp chương trình vào PROM, FPGA sẽ lấy dữ liệu từ PROM mỗi lần khởi động.

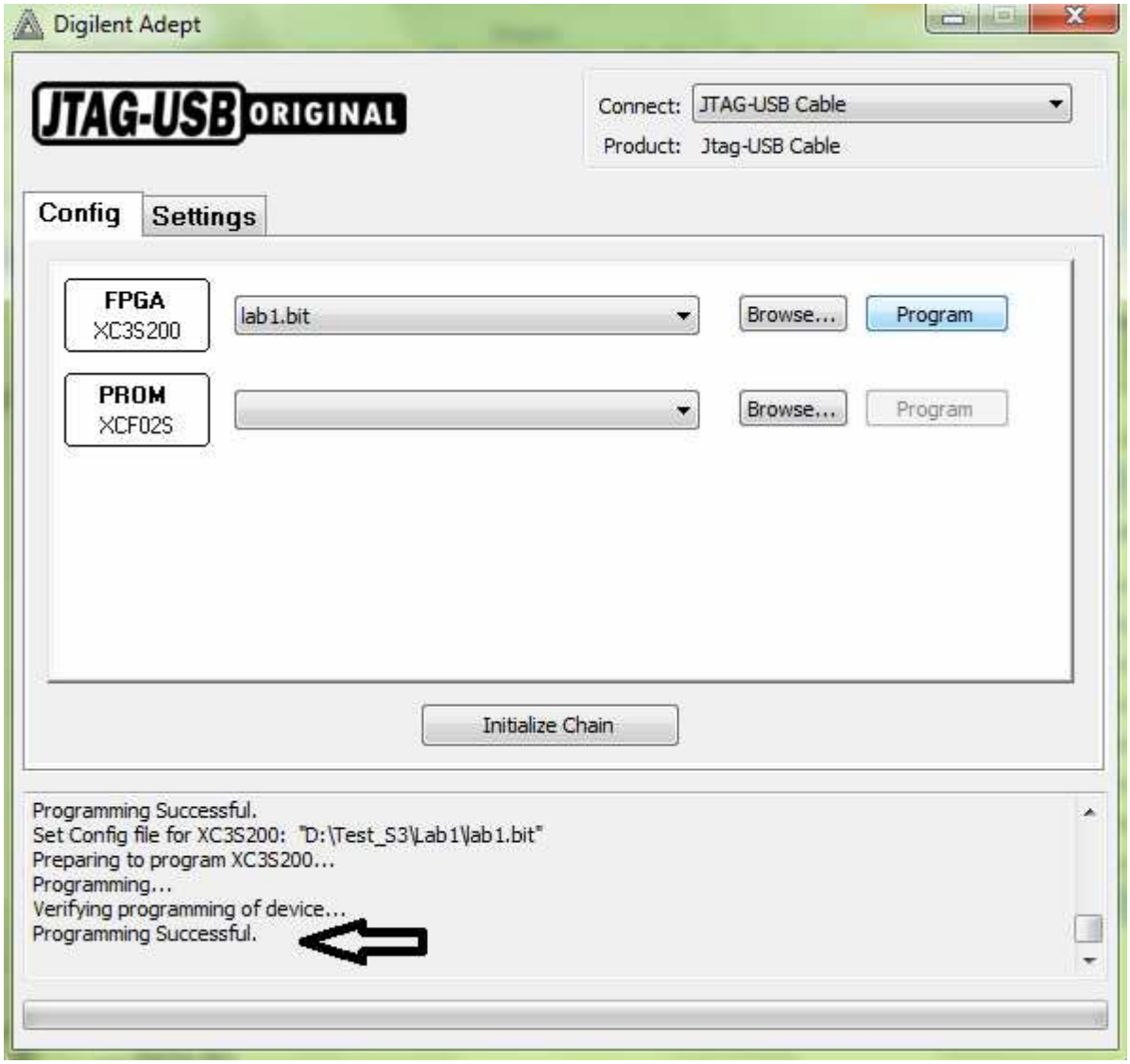
Chú ý: Số lần nạp cho FPGA là không giới hạn nhưng số lần nạp cho PROM là có giới hạn. Trong khi thực tập, KHÔNG nạp chương trình cho PROM.

- Để nạp chương trình cho FPGA, chọn nút ***Browse***

Chọn tập tin cần nạp (*.bit) trong cửa sổ tiếp theo



- Nhấn nút ***Program*** để nạp chương trình. Kết quả được báo trên dòng thông báo.



- Quan sát kết quả thực hiện trên board S3.

Bài 2

LEDs & SWITCHs

2.1 Giới thiệu

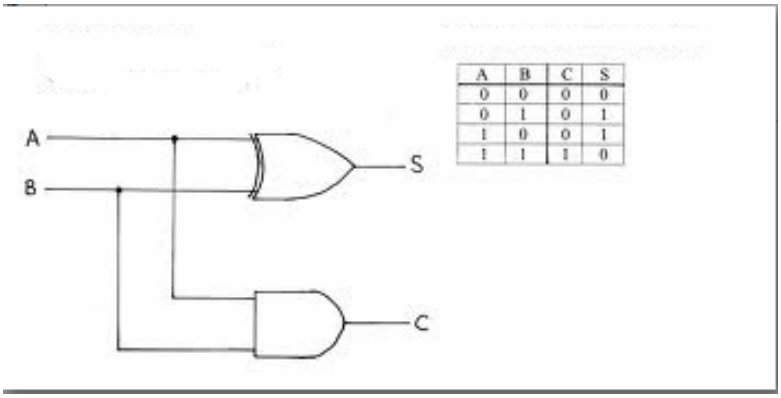
Trong bài này, sinh viên sẽ làm việc với 2 thành phần cơ bản của Xilinx S3 FPGA Starter Board là LED và Switch.

Những thao tác làm việc với ISE Design Suite sẽ không được nhắc lại.

2.2 Thực hành

2.2.1 Half Adder

- Sơ đồ nguyên lý và bảng sự thật



A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

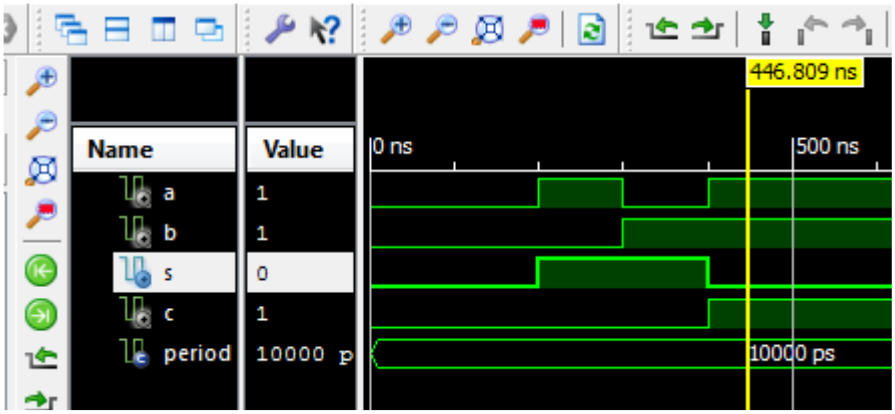
- Chương trình

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity H_Adder is
      Port ( A : in  STD_LOGIC;
             B : in  STD_LOGIC;
             S : out  STD_LOGIC;
             C : out  STD_LOGIC);
   end H_Adder;

   architecture Behavioral of H_Adder is
   begin
          S <= A XOR B;
          C <= A AND B;
   end Behavioral;
```

- Viết chương trình mô phỏng (**VHDL Testbench**) theo hướng dẫn tại Bài 1.
- Chương trình gợi ý như sau:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
 ENTITY HA_test IS
END HA_test;
ARCHITECTURE behavior OF HA_test IS
        COMPONENT H_Adder
                PORT(       A : IN  std_logic;
                            B : IN  std_logic;
                            S : OUT  std_logic;
                            C : OUT  std_logic    );
        END COMPONENT;
signal A : std_logic := '0';
signal B : std_logic := '0';
signal S : std_logic;
signal C : std_logic;
constant period : time := 10 ns;
BEGIN
  uut: H_Adder PORT MAP (
     A => A,
     B => B,
     S => S,
     C => C    );

 stim_proc: process
begin
        wait for 100 ns;
                A <= '0';
                B <= '0';
        wait for 100 ns;
                A <= '1';
                B <= '0';
        wait for 100 ns;
                A <= '0';
                B <= '1';
        wait for 100 ns;
                A <= '1';
                B <= '1';
        wait;
end process;
END;
```

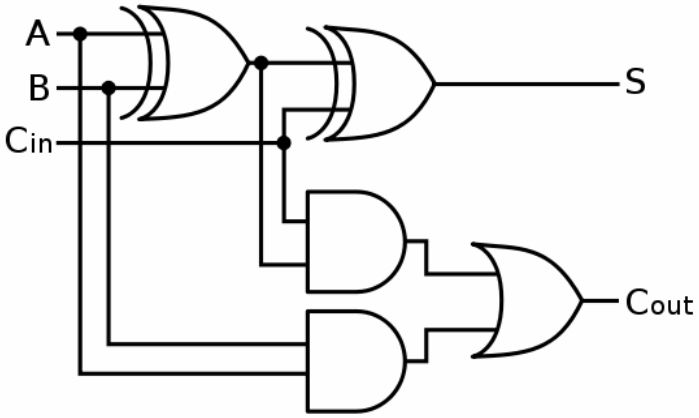
- Giản đồ xung trong ISim



- Thực hiện gán chân cho các tín hiệu đầu vào là SW và đầu ra là LED.

2.2.2 Full Adder

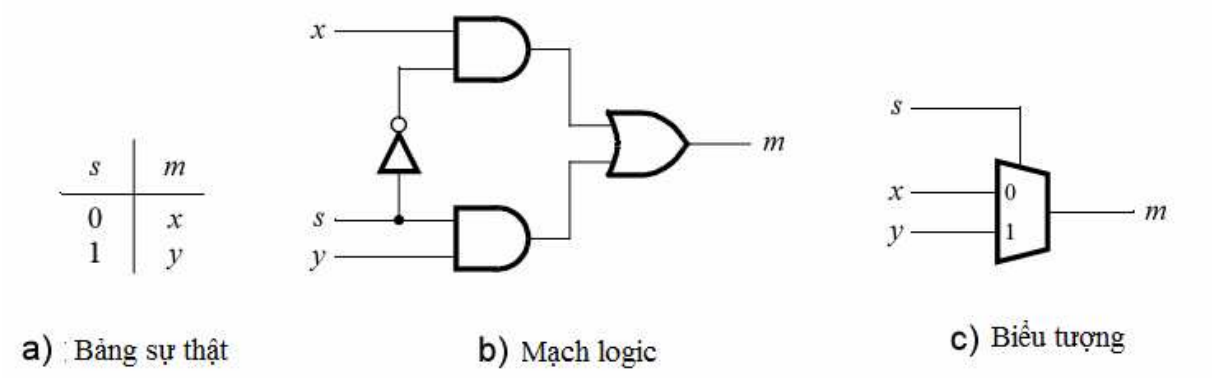
- Sơ đồ nguyên lí và bảng sự thật



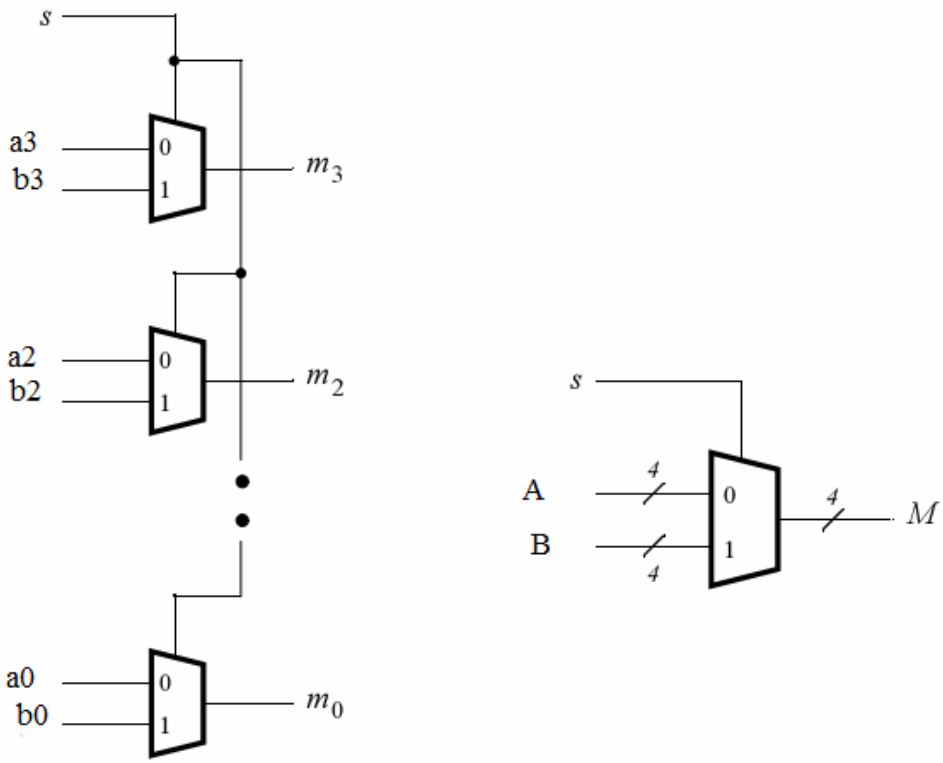
Inputs			Outputs	
A	*B*	C_{in}	C_{out}	*S*
0	0	0	0	0
1	0	0	0	1
0	1	0	0	1
1	1	0	1	0
0	0	1	0	1
1	0	1	1	0
0	1	1	1	0
1	1	1	1	1

- Viết chương trình thiết kế Full Adder.
- Viết chương trình thực hiện mô phỏng
- Gán chân cho các tín hiệu, quan sát kết quả trên board S3.

2.2.3 Multiplexer



Dựa theo sơ đồ mạch trên, thiết kế 1 bộ multiplexer có độ rộng 4 bit.



- Chương trình gợi ý

```
library ieee;
use ieee.std_logic_1164.all;
Entity Lab21 is
        Port (  SW    : in    STD_LOGIC_VECTOR(7 downto 0);
                BT    : in    STD_LOGIC;
                LED   : out   STD_LOGIC_VECTOR(4 downto 0));
End Lab21;
```

```
Architecture Behavioral of Lab21 is
        signal Sel      :       STD_LOGIC;
        signal A, B, M:         STD_LOGIC_VECTOR(3 downto 0);
Begin
        A <= SW (3 downto 0);
        B <= SW(7 downto 4);
        Sel <= BT;

        M(0) <= (NOT(Sel) AND A(0)) OR (Sel AND B(0));
        M(1) <= (NOT(Sel) AND A(1)) OR (Sel AND B(1));
        M(2) <= (NOT(Sel) AND A(2)) OR (Sel AND B(2));
        M(3) <= (NOT(Sel) AND A(3)) OR (Sel AND B(3));

        LED(3 downto 0) <=M;
        LED(4) <= BT;
End Behavioral;
```

- Thực hiện mô phỏng bằng ISim
- Gán chân SW, LED, BT tự chọn theo bảng sau:

Switch	SW7	SW6	SW5	SW4	SW3	SW2	SW1	SW0
FPGA Pin	K13	K14	J13	J14	H13	H14	G12	F12

LED	LD7	LD6	LD5	LD4	LD3	LD2	LD1	LD0
FPGA Pin	P11	P12	N12	P13	N14	L12	P14	K12

Push Button	BTN3 (User Reset)	BTN2	BTN1	BTN0
FPGA Pin	L14	L13	M14	M13

- Nạp chương trình và quan sát kết quả trên board S3

Bài 3

7 SEGMENTs DISPLAY

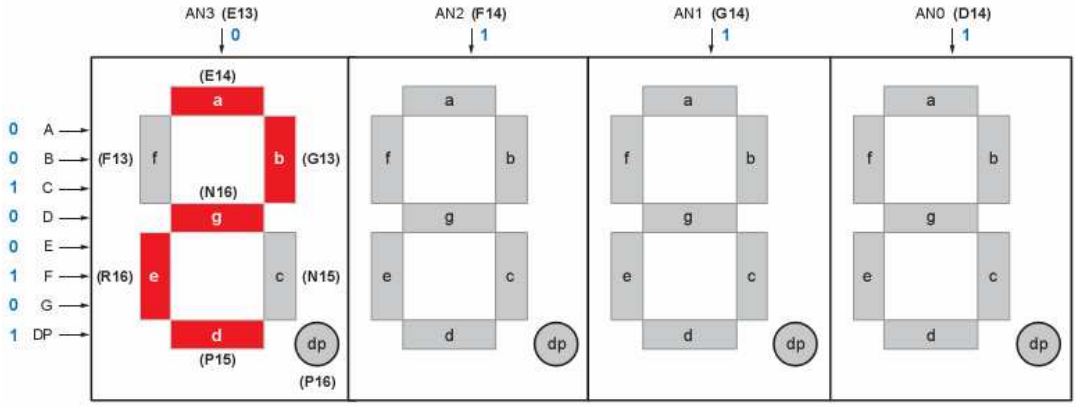
3.1 Giới thiệu

Trong bài này, sinh viên sẽ làm việc với bốn đèn LED 7 đoạn có sẵn trên board S3. Chúng được kết nối với FPGA thông qua một bộ phân kênh, do vậy mỗi lần chỉ có thể xuất dữ liệu ra một đèn LED thông qua tín hiệu chọn đèn.

Những đèn 7 đoạn này hoạt động tại mức thấp, nghĩa là đèn sáng khi nhận bit '0' và tắt khi nhận bit '1'.

3.2 Thực tập

- Sơ đồ bố trí đèn



- Bảng gán chân chọn đèn và thanh

FPGA Connections to Seven-Segment Display (Active Low)

Segment	FPGA Pin
A	E14
B	G13
C	N15
D	P15
E	R16
F	F13
G	N16
DP	P16

Digit Enable (Anode Control) Signals (Active Low)

Anode Control	AN3	AN2	AN1	AN0
FPGA Pin	E13	F14	G14	D14

3.2.1 Điều khiển một đèn LED 7 đoạn

Hiển thị lần lượt chữ (**d L U .**) dựa vào giá trị nhập từ 2 công tắc SW.

S1	*S0*	*Kí tự*
0	**0**	**d**
0	**1**	**L**
1	**0**	**U**
1	**1**	**.**

- Chương trình gợi ý

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

    entity dLU is
      Port ( SW : in  STD_LOGIC_VECTOR (1 downto 0);
             AN : out      STD_LOGIC_VECTOR (3 downto 0);
             HEX : out  STD_LOGIC_VECTOR (7 downto 0);
             LED : out  STD_LOGIC_VECTOR (1 downto 0));
    end dLU;

    architecture Behavioral of dLU is
        signal S: STD_LOGIC_VECTOR (1 downto 0);

    begin
        AN <= "1110";
        LED <= SW;
        S(1 downto 0) <= SW(1 downto 0);

        HEX(0) <= '1';
        HEX(1) <= S(0);
        HEX(2) <= S(0);
        HEX(3) <= S(0) AND S(1);
        HEX(4) <= S(0) AND S(1);
        HEX(5) <= S(0) XNOR S(1);
        HEX(6) <= S(0) OR S(1);
        HEX(7) <= S(0) NAND S(1);

    end Behavioral;
```

- Gán chân, nạp chương trình, quan sát kết quả.

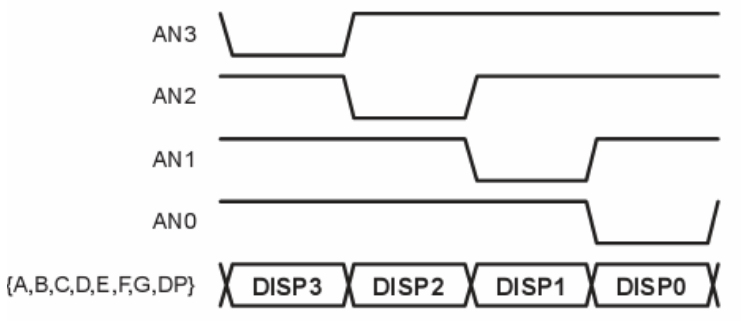
3.2.2 Điều khiển bốn đèn LED 7 đoạn

- Chương trình hiện chữ **CAFE** đồng thời

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity Lab2 is
   Port (     iCLK : in  STD_LOGIC;
              AN3 : inout  STD_LOGIC;
              AN2 : inout  STD_LOGIC;
              AN1 : inout  STD_LOGIC;
              AN0 : inout  STD_LOGIC;
              HEX : out  STD_LOGIC_VECTOR (7 downto 0));
end Lab2;

architecture Behavioral of Lab2 is

signal Counter: STD_LOGIC_VECTOR ( 15 downto 0);

begin
      process (iCLK, Counter)
      begin

            if iCLK'event and iCLK ='1' then
                  Counter <= Counter + "1";
                  if (Counter ="0000000000000000") then
                              HEX <= "11000110";        --C
                              AN0 <= '1';
                              AN1 <= '1';
                              AN2 <= '1';
                              AN3 <= '0';
                  elsif (Counter ="0011111111111111") then
                              AN0 <= '1';
                              AN1 <= '1';
                              AN2 <= '0';
                              AN3 <= '1';
                              HEX <= "10001000";        --A
                  elsif (Counter ="0111111111111111") then
                              AN0 <= '1';
                              AN1 <= '0';
                              AN2 <= '1';
                              AN3 <= '1';
                              HEX <= "10001110";        --F
```

```
            elsif (Counter ="1100000000000000") then
                        AN0 <= '0';
                        AN1 <= '1';
                        AN2 <= '1';
                        AN3 <= '1';
                        HEX <= "10000110";        --E
            end if;
        end if;
    end process;
end Behavioral;
```

- Giản đồ xung hoạt động 4 đèn LED 7 đoạn



- Chân cấp xung chính

Clock Oscillator Sources

Oscillator Source	FPGA Pin
50 MHz (IC4)	T9
Socket (IC8)	D9

- Gán chân và quan sát kết quả.

3.2.3 Hiển thị lần lượt từng đèn LED

- Viết chương trình chạy chữ CAFE lần lượt từng đèn
- Gán chân và quan sát kết quả.

Bài 4

COUNTERs

4.1 Giới thiệu

Trong bài thực tập này, dựa trên những ví dụ có sẵn, sinh viên thiết kế các bộ đếm theo yêu cầu và tự chọn.

4.2 Thực hành

4.2.1 Bộ đếm lên 0 – 9

- Chương trình sử dụng CLK là nút bấm, mỗi lần nhấn nút sẽ đếm.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

    entity Counter9 is
      Port ( CLK : in  STD_LOGIC;
             RST : in  STD_LOGIC;
            COUT : out  STD_LOGIC_VECTOR (6 downto 0);
            AN : out  STD_LOGIC_VECTOR (3 downto 0));
    end Counter9;

    architecture Behavioral of Counter9 is

    signal number : STD_LOGIC_VECTOR (3 downto 0);

    begin
        process (CLK, RST)
        begin
            if (RST = '1') then
                number <= "0000";
            elsif (CLK'event and CLK = '0') then
                if (number < "1001") then
                    number <= number + "0001";
                else number <= "0000";
                end if;
            end if;
        end process;
```

```
    process (CLK, number)
        begin
            case number is
                when "0000" => COUT <= "1000000";   --0
                when "0001" => COUT <= "1111001";   --1
                when "0010" => COUT <= "0100100";   --2
                when "0011" => COUT <= "0110000";
                when "0100" => COUT <= "0011001";
                when "0101" => COUT <= "0010010";
                when "0110" => COUT <= "0000010";
                when "0111" => COUT <= "1111000";
                when "1000" => COUT <= "0000000";
                when "1001" => COUT <= "0010000";   --9
                when others => null;
            end case;
        end process;
                AN <= "1110";       -- 1 LED
end Behavioral;
```

- Mô phỏng chương trình sử dụng ISim.
- Gán chân, quan sát và so sánh với mô phỏng.

4.2.2 Bộ đếm lên xuống 0 – 9

- Viết chương trình đếm có khả năng đếm lên hoặc xuống.
- Mô phỏng trong ISim
- Gán chân, quan sát và so sánh với mô phỏng.

4.2.3 Bộ đếm lên xuống 0 – F

- Viết chương trình đếm có khả năng đếm lên xuống hệ thập lục phân.
- Mô phỏng trong ISim
- Gán chân, quan sát và so sánh với mô phỏng.

Bài 5

CLOCKs & TIMERs

5.1 Giới thiệu

Tổng hợp từ những bài thực tập trước, trong bài này sinh viên sẽ viết chương trình có khả năng sử dụng bộ dao động có sẵn để thiết kế những mạch định giờ.

5.2 Thực tập

5.2.1 Đồng hồ đơn giản

- Chương trình đồng hồ đếm từ 0 – F

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;


entity Clock is
        Port ( CLK : in  STD_LOGIC;
          RST : in  STD_LOGIC;
        HEX : out  STD_LOGIC_VECTOR (6 downto 0);
        AN : out  STD_LOGIC_VECTOR (3 downto 0));
end Clock;

architecture Behavioral of Clock is
        signal count : STD_LOGIC_VECTOR (25 downto 0);
        signal number: STD_LOGIC_VECTOR (3 downto 0);

begin
        process (CLK)
        begin
                if (CLK'event and CLK = '1') then
                        if (RST = '1') then
                         number <= "0000";
                         count <= "00000000000000000000000000";
                        else
                         count <= count + '1';
                        end if;
```

```
                if (count = "11111111111111111111111111") then
                    number <= number + '1';
                end if;
            end if;
        end process;
        process (number)
        begin
            case number is
                when "0000" => HEX <= "1000000";
                when "0001" => HEX <= "1111001";
                when "0010" => HEX <= "0100100";
                when "0011" => HEX <= "0110000";
                when "0100" => HEX <= "0011001";
                when "0101" => HEX <= "0010010";
                when "0110" => HEX <= "0000010";
                when "0111" => HEX <= "1111000";
                when "1000" => HEX <= "0000000";
                when "1001" => HEX <= "0010000";
                when "1010" => HEX <= "0001000";
                when "1011" => HEX <= "0000011";
                when "1100" => HEX <= "1000110";
                when "1101" => HEX <= "0100001";
                when "1110" => HEX <= "0000110";
                when "1111" => HEX <= "0001110";
                when others => null;
            end case;
        end process;

        AN <= "1110";

end Behavioral;
```

- Viết chương trình mô phỏng ISim
- Gán chân, quan sát kết quả

5.2.2 Đồng hồ đếm lên xuống từ 0 – 9

- Viết chương trình thiết kế đồng hồ đếm lên xuống từ 0 – 9
- Mô phỏng trong ISim
- Gán chân, quan sát kết quả.

5.2.3 Đồng hồ đếm lên xuống 00 – FF

- Chương trình gợi ý

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity Lab3 is
   Port ( CLK : in  STD_LOGIC;
          RST : in  STD_LOGIC;
          DIR : in  STD_LOGIC;
          HEX : out  STD_LOGIC_VECTOR (6 downto 0);
          AN : out  STD_LOGIC_VECTOR (3 downto 0));
end Lab3;

architecture Behavioral of Lab3 is
          signal tmp     : STD_LOGIC_VECTOR (17 downto 0);
          signal count : STD_LOGIC_VECTOR (22 downto 0);
          signal hex1 : STD_LOGIC_VECTOR (3 downto 0);
          signal hex2 : STD_LOGIC_VECTOR (3 downto 0);
          signal number1, number2 : STD_LOGIC_VECTOR (6 downto 0);

begin
          process (CLK)
          begin
                    if(CLK'event and CLK = '1') then
                    if (RST = '1') then
                              tmp <= "000000000000000000";
                              count <= "0000000000000000000000";
                              hex1 <= "0000";
                              hex2 <= "1010";
                    else
                              count <= count + '1';
                              tmp <= tmp + '1';
                    end if;
```

```
        if (count = "11111111111111111111") then
                if (DIR = '1') then
                        hex1 <= hex1 + '1';
                else
                        hex1 <= hex1 - '1';
          end if;
        end if;
        if ((hex1 = "1111") and (count = "11111111111111111111")) then
                if (DIR = '1') then
                        hex2 <= hex2 + '1';
                else
                        hex2 <= hex2 - '1';
                end if;
        end if;
        if (tmp = "10000000000000000") then
                AN <= "0111";
                HEX <= number2;
        else if (tmp = "11111111111111111") then
                AN <= "1011";
                HEX <= number1;
        end if;
        end if;
        end if;
end process;


process (hex1)
begin
                case hex1 is
                        when "0000" => number1 <= "1000000";
                        when "0001" => number1 <= "1111001";
                        when "0010" => number1 <= "0100100";
                        when "0011" => number1 <= "0110000";
                        when "0100" => number1 <= "0011001";
                        when "0101" => number1 <= "0010010";
                        when "0110" => number1 <= "0000010";
                        when "0111" => number1 <= "1111000";
                        when "1000" => number1 <= "0000000";
                        when "1001" => number1 <= "0010000";
                        when "1010" => number1 <= "0001000";
                        when "1011" => number1 <= "0000011";
                        when "1100" => number1 <= "1000110";
                        when "1101" => number1 <= "0100001";
                        when "1110" => number1 <= "0000110";
                        when "1111" => number1 <= "0001110";
```

```
                when others => null;
            end case;
    end process;

    process (hex2)
    begin
            case hex2 is
                when "0000" => number2 <= "1000000";
                when "0001" => number2 <= "1111001";
                when "0010" => number2 <= "0100100";
                when "0011" => number2 <= "0110000";
                when "0100" => number2 <= "0011001";
                when "0101" => number2 <= "0010010";
                when "0110" => number2 <= "0000010";
                when "0111" => number2 <= "1111000";
                when "1000" => number2 <= "0000000";
                when "1001" => number2 <= "0010000";
                when "1010" => number2 <= "0001000";
                when "1011" => number2 <= "0000011";
                when "1100" => number2 <= "1000110";
                when "1101" => number2 <= "0100001";
                when "1110" => number2 <= "0000110";
                when "1111" => number2 <= "0001110";
                when others => null;
            end case;
        end process;
end Behavioral;
```

- Mô phỏng trong ISim
- Gán chân, quan sát kết quả.

5.2.4 Đồng hồ đếm lên xuống 00 – 99

- Viết chương trình
- Mô phỏng trong ISim
- Gán chân, quan sát kết quả

Bài 6
TỰ CHỌN

6.1 Giới thiệu

Trong bài này, sinh viên sẽ tự viết một chương trình tự chọn. Thiết kế được khuyến khích tận dụng tài nguyên có sẵn của board Xilinx S3.

6.2 Ví dụ

Chương trình ví dụ sử dụng cổng VGA

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity VGA is
port(CLK  : in std_logic;
        RED   : out std_logic;
        GREEN : out std_logic;
        BLUE  : out std_logic;
        HS    : out std_logic;
        VS    : out std_logic);
end VGA;

architecture Behavioral of VGA is

signal CLK25            : std_logic;
signal horizontal_counter : std_logic_vector (9 downto 0);
signal vertical_counter   : std_logic_vector (9 downto 0);

begin
process (CLK)
begin
 if (CLK'event and CLK='1') then
         if (CLK25 = '0') then
        CLK25 <= '1';
else
        CLK25 <= '0';
        end if;
end if;
end process;
```

```
        process (CLK25)
        begin
                if (CLK25'event and CLK25 = '1') then
                if (horizontal_counter >= "0010010000" )                -- 144
               and (horizontal_counter < "1100010000" )                 -- 784
               and (vertical_counter >= "0000100000" )                  -- 32
               and (vertical_counter < "0111111101" )                   -- 509
        then
               RED <= horizontal_counter(9)   XOR vertical_counter(8);
               GREEN <= horizontal_counter(7)  XOR vertical_counter(6);
               BLUE <=  horizontal_counter(5)  XOR vertical_counter(4);
        else
               RED <= '0';
               GREEN <= '0';
               BLUE <= '0';
        end if;

         horizontal_counter <= horizontal_counter + "0000000001";

         if (horizontal_counter > "0000000000" )
               and (horizontal_counter < "0001100001" )                 --96+1
         then
               HS <= '0';
        else
               HS <= '1';
        end if;
        if (vertical_counter > "0000000000" )
               and (vertical_counter < "0000000011" )                   --2+1
        then
                VS <= '0';
        else
               VS <= '1';
        end if;

        if (horizontal_counter = "1100100000") then                     --800
               vertical_counter <= vertical_counter + "0000000001";
               horizontal_counter <= "0000000000";
        end if;
        if (vertical_counter = "1000001001") then                       --521
               vertical_counter <= "0000000000";
        end if;
        end if;
        end process;
end Behavioral;
```

Bài 7
CPU MỀM - PICOBLAZE SOFT CPU

7.1 Giới thiệu

CPU mềm (soft CPU) là một lõi vi xử lý có thể được thực hiện hoàn toàn bằng kỹ thuật tổng hợp logic. Nó có thể được thực hiện bởi các linh kiện logic lập trình được như FPGA, CPLD. Thông thường, người ta chỉ sử dụng một FPGA cho một lõi CPU mềm; tuy nhiên, vẫn có khả năng tổng hợp nhiều lõi CPU mềm trên một FPGA phụ thuộc độ lớn của linh kiện. Việc này tạo nên cấu trúc đa lõi với khả năng tận dụng tài nguyên tốt hơn.

Hiện nay có rất nhiều lõi CPU mềm với các tính năng và đặc trưng khác nhau. Điển hình với FPGA của Xilinx có MicroBlaze và PicoBlaze, với lõi PicoBlaze là mã nguồn mở. Đối với Altera là lõi Nios và NiosII; hãng Lattice có LatticeMico32…

MicroBlaze có cấu trúc tập lệnh gần giống cấu trúc DLX nền RISC với một vài thay đổi. MicroBlaze là lõi CPU 32bit, hệ Endian lớn. Sử dụng MicroBlaze cần có gói phần mềm EDK hoặc IDS với giấy phép sử dụng từ Xilinx.

Trong khi đó, PicoBlaze là chuỗi bộ ba lõi CPU mềm miễn phí. Nó cũng dựa trên nền tảng cấu trúc 8bit RISC và có thể hoạt động với họ Virtex4 tại 100MIPS. PicoBlaze ban đầu được lấy tên KCPSM do Ken Chapman tại Xilinx thiết kế.

7.2 Thực hành

Để có thể làm việc với PicoBlaze, người dùng cần có phần mềm hỗ trợ *pBlazeIDE.exe*, tập tin *KCPSM3.vhd* cho lõi phiên bản PicoBlaze3, tập tin *KCPSM3.exe* để chuyển đổi tập tin viết bằng hợp ngữ, tập tin *ROM_form.vhd* và *ROM_form.coe*. Tất cả cần nằm trong cùng thư mục làm việc với tập tin chương trình đang viết. Các thông tin chi tiết về PicoBlaze và mô phỏng ModelSim có thể tìm thấy trong "*PicoBlaze 8bit Embedded Microcontroller User Guide*" và "*KCPSM3 Manual*".

Trong bài thực hành này, pBlazeIDE phiên bản 3.6b được sử dụng.

Để viết chương trình mới: ***File → New***
Chọn lõi CPU: ***Settings → PicoBlaze3***

Để sử dụng chương trình đã viết sẵn: ***File → Import***
Có thể sử dụng **Notepad** để viết chương trình bằng hợp ngữ với tập lệnh và thông số của lõi KCPSM3, chương trình lưu dưới dạng *vidu.psm*

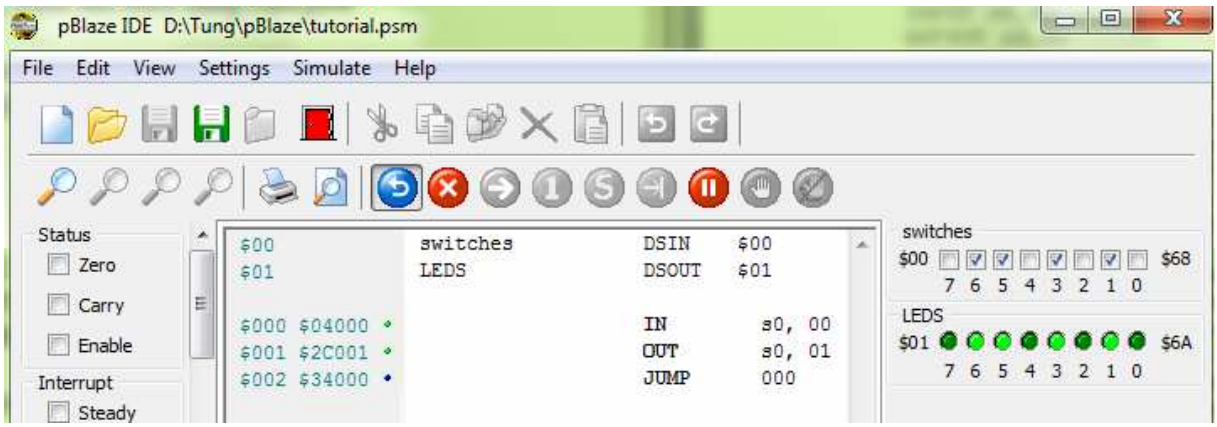
Ví dụ chương trình sử dụng PicoBlaze để đọc đầu vào 8bit, lưu vào thanh ghi rồi đặt trạng thái lối ra tương ứng; lặp lại chương trình liên tục:

```
; switches       DSIN      $00
; LEDS           DSOUT     $01

        INPUT     s0, 00
        OUTPUT    s0, 01
        JUMP      000
```

Mô phỏng chương trình: ***Simulate → Simulate***

Các nút công cụ:

Khi mô phỏng cần bỏ dấu chấm phẩy "**;**" trước hai dòng định nghĩa trên cùng.



Sau khi mô phỏng thành công, cần chú thích những dòng đặc biệt bằng cách đặt dấu chấm phẩy "**;**" trước chúng.
Sử dụng công cụ gõ lệnh: ***Start → Run → cmd***
Vào thư mục chương trình làm việc và nhập lệnh ***KCPSM3 vidu.psm***

Chương trình *KCPSM3.exe* sẽ tạo ra tập tin *vidu.vhd* với đầy đủ thông số cần thiết. Sau khi có tập tin chương trình, cần có tập tin liên kết lõi CPU với chương trình đã có. Tập tin liên kết *top.vhd* có dạng:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity picotest is
        Port (      switches : in std_logic_vector(7 downto 0);
                    LEDS : out std_logic_vector(7 downto 0);
```

```
                clk : in std_logic);
        end picotest;
    architecture Behavioral of picotest is
    component kcpsm3
        Port (      address : out std_logic_vector(9 downto 0);
                instruction : in std_logic_vector(17 downto 0);
                port_id : out std_logic_vector(7 downto 0);
                write_strobe : out std_logic;
                out_port : out std_logic_vector(7 downto 0);
                read_strobe : out std_logic;
                in_port : in std_logic_vector(7 downto 0);
                interrupt : in std_logic;
                interrupt_ack : out std_logic;
                reset : in std_logic;
                clk : in std_logic);
    end component;
-- declaration of program memory
     component vidu
        Port (      address : in std_logic_vector(9 downto 0);
                instruction : out std_logic_vector(17 downto 0);
                clk : in std_logic);
    end component;
-- Signals used to connect PicoBlaze core to program memory and I/O logic
    signal address       : std_logic_vector(9 downto 0);
    signal instruction   : std_logic_vector(17 downto 0);
    signal port_id       : std_logic_vector(7 downto 0);
    signal out_port      : std_logic_vector(7 downto 0);
    signal in_port       : std_logic_vector(7 downto 0);
    signal write_strobe  : std_logic;
    signal read_strobe   : std_logic;
    signal interrupt_ack : std_logic;
    signal reset         : std_logic :='0';
    signal interrupt     : std_logic :='0';
-- Start of circuit description
    begin
  -- Instantiating the PicoBlaze core
     processor: kcpsm3
        port map(      address => address,
                instruction => instruction,
                port_id => port_id,
                write_strobe => write_strobe,
                out_port => out_port,
                read_strobe => read_strobe,
                in_port => in_port,
                interrupt => interrupt,
```

```
                    interrupt_ack => interrupt_ack,
                    reset => reset,
                    clk => clk);
 -- Instantiating the program memory
      program: vidu
            port map(      address => address,
                    instruction => instruction,
                    clk => clk);
 -- Connect I/O of PicoBlaze
                    in_port <= switches;
                    LEDS <= out_port;
      end Behavioral;
```

Sử dụng Xilinx ISE tạo chương trình mới sử dụng ba tập *tin top.vhd, kcmsm3.vhd* và *vidu.vhd*.Tiến hành gán chân, tạo tập tin nạp FPGA như những bài trước. Kiểm tra kết quả trên bo mạch.

Bài 1
MICROWIND 2

1.1 Giới thiệu

Trên thế giới hiện nay, có rất nhiều công cụ hỗ trợ thiết kế vi mạch. Tuy nhiên chúng đều rất đắt và chỉ phù hợp cho mục đích công nghiệp, ví dụ như Cadence, Synopsys…

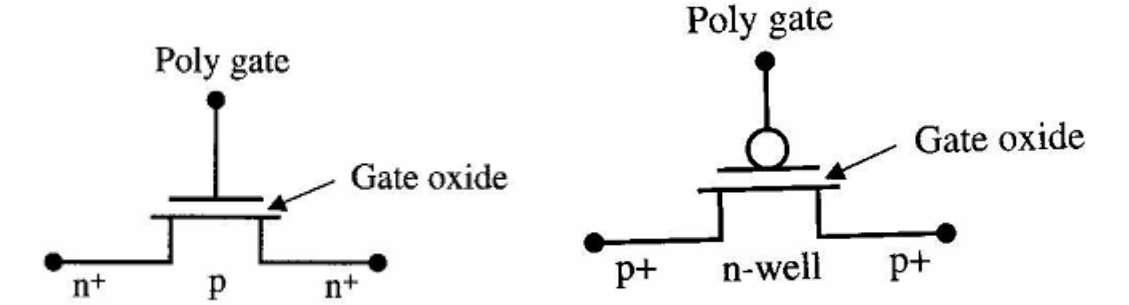
Microwind là phần mềm thiết kế vi mạch miễn phí trong hệ điều hành Windows, song gần đây phiên bản mới của Microwind đã bắt đầu thu phí sử dụng. Trong tài liệu này, phiên bản Microwind 2 được sử dụng như là công cụ hỗ trợ thực tập.

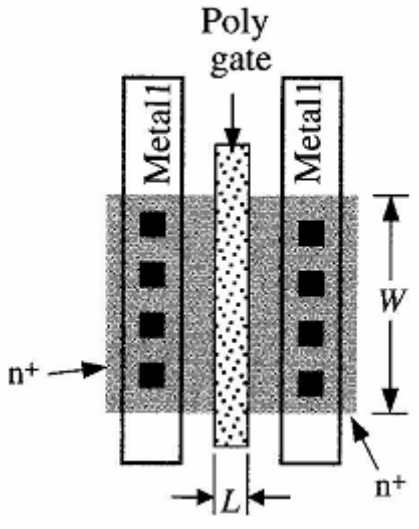
Trong phần trước của tài liệu, sinh viên đã biết cách làm việc với FPGA. Trong phần này, sinh viên sẽ được làm quen với kĩ thuật thiết kế vi mạch. Kiến thức được cung cấp chỉ mang tính giới thiệu, việc thiết kế vi mạch trong thực tế rất phức tạp, đòi hỏi người thiết kế phải có kiến thức về nhiều lĩnh vực đặc biệt là vật lý.

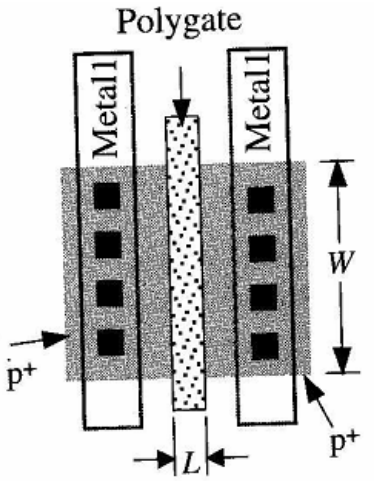
1.2 Công nghệ CMOS

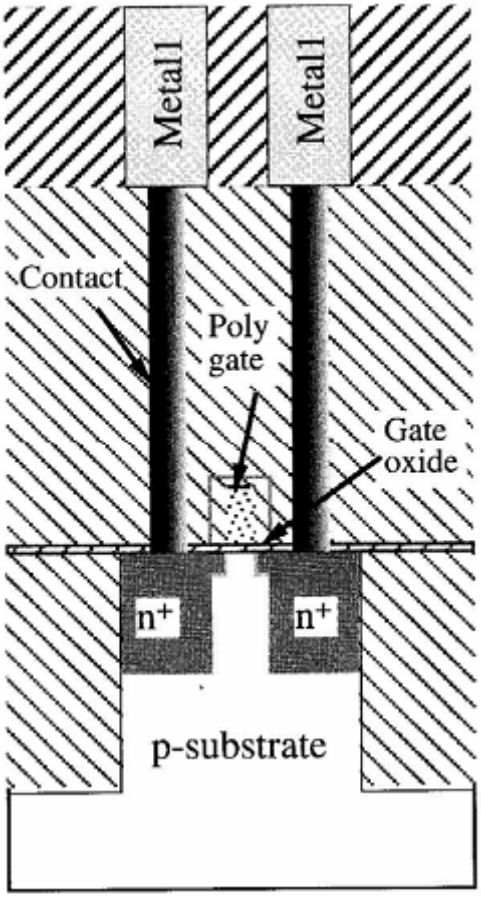
Kiến thức cơ bản đã trình bày trong giáo trình "Công Nghệ Vi Mạch Điện Tử", trong phần này của tài liệu thực tập chỉ nhắc lại một số ý chính. Sinh viên cần xem lại lý thuyết trước khi thực tập.

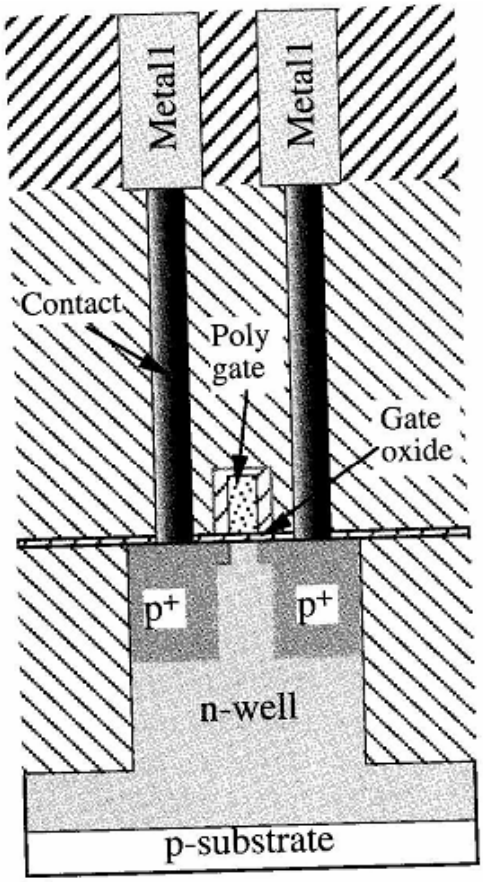
Trước kia, vi mạch thường sử dụng NMOS transistor nhưng gần đây người ta sử dụng kĩ thuật CMOS để thực hiện những thiết kế. CMOS là viết tắt của Complementary MOS; nó có 2 thành phần là NMOS và PMOS. Biểu tượng, hình chiếu trên và mặt cắt ngang lần lượt của NMOS và PMOS:











NMOS **PMOS**

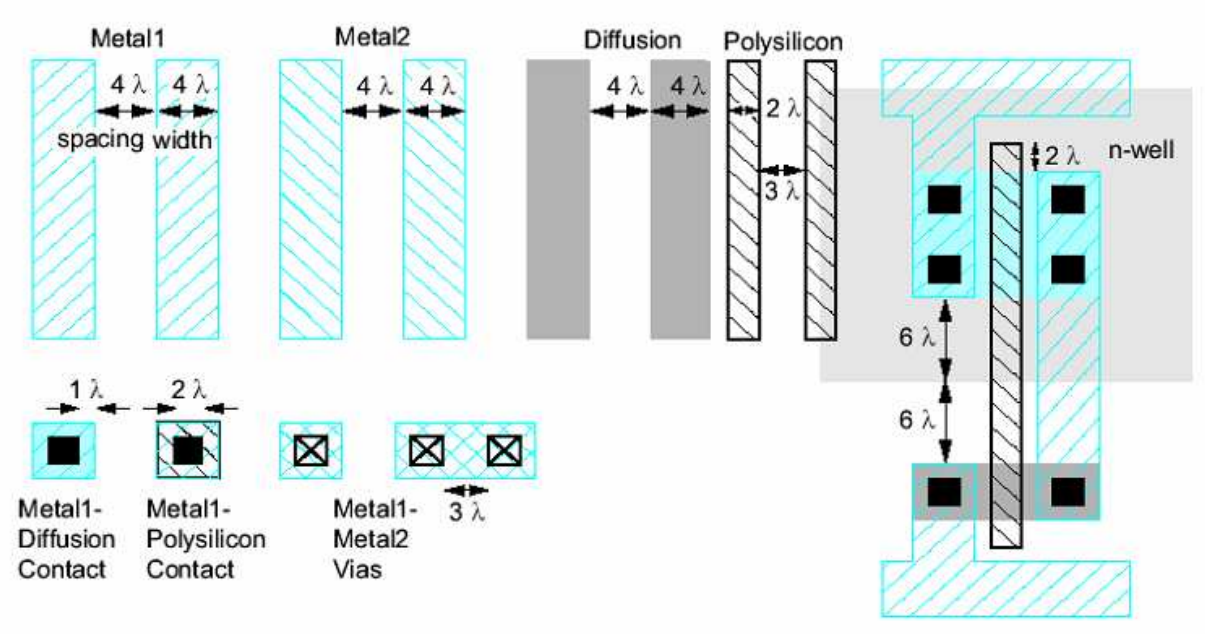
Do đặc trưng công nghệ và quy định từ thời kì đầu của sản xuất vi mạch nên W > L trong đó L là chiều dài kênh dẫn, W là chiều rộng giếng pha tạp.

Một số khái niệm:

- Kích thước đặc trưng (Feature size) f là kích thước nhỏ nhất của đường polysilicon, hay là khoảng cách giữa cực máng và cực cổng.
- Lamda λ = f/2
- Kích thước transistor: xác định bằng Rộng / Dài
- Kích thước nhỏ nhất là 4 λ / 2 λ (kích thước đơn vị)

Ví dụ: trong công nghệ f = 90nm, Rộng W = 180nm và Dài L = 90nm

- Kiểm tra luật thiết kế DRC: thiết kế phải đảm bảo khoảng cách giữa các đường tuân theo quy định của công nghệ sản xuất.

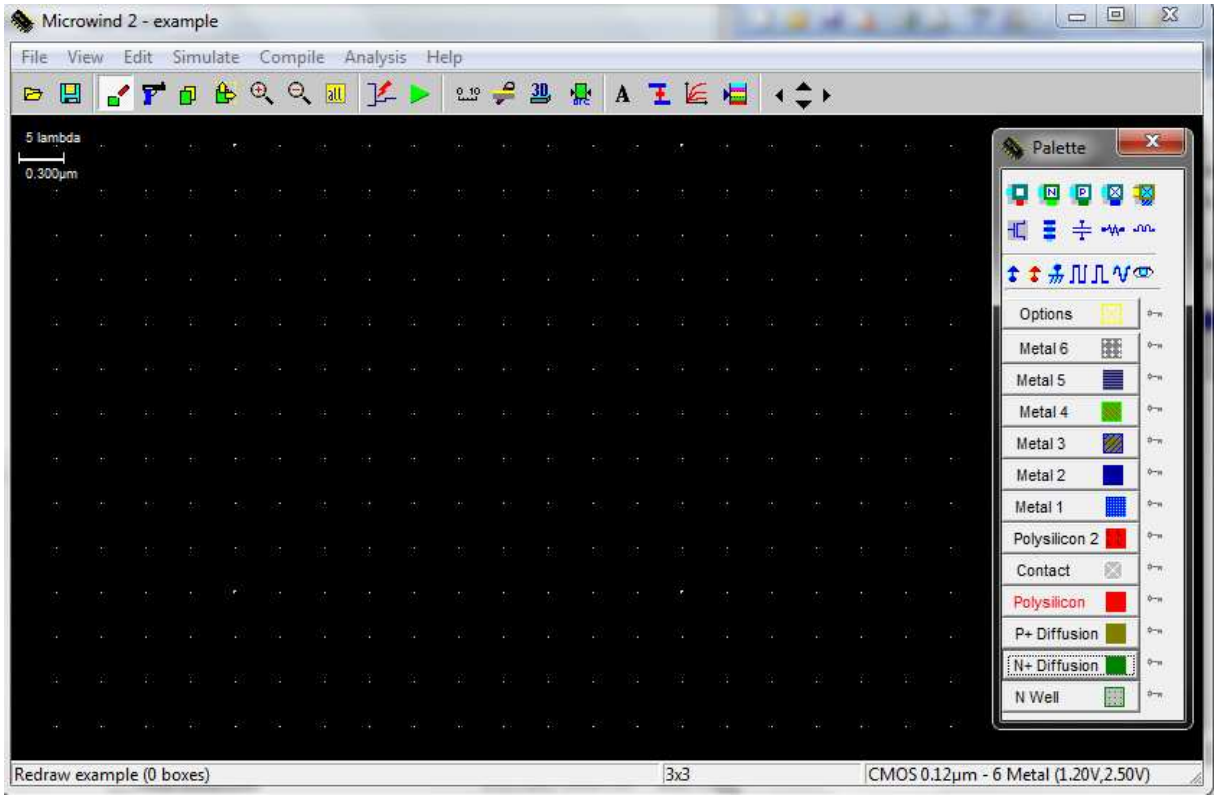


- Kiểm tra lỗi LVS: so sánh kết quả vẽ mạch với sơ đồ nguyên lý.

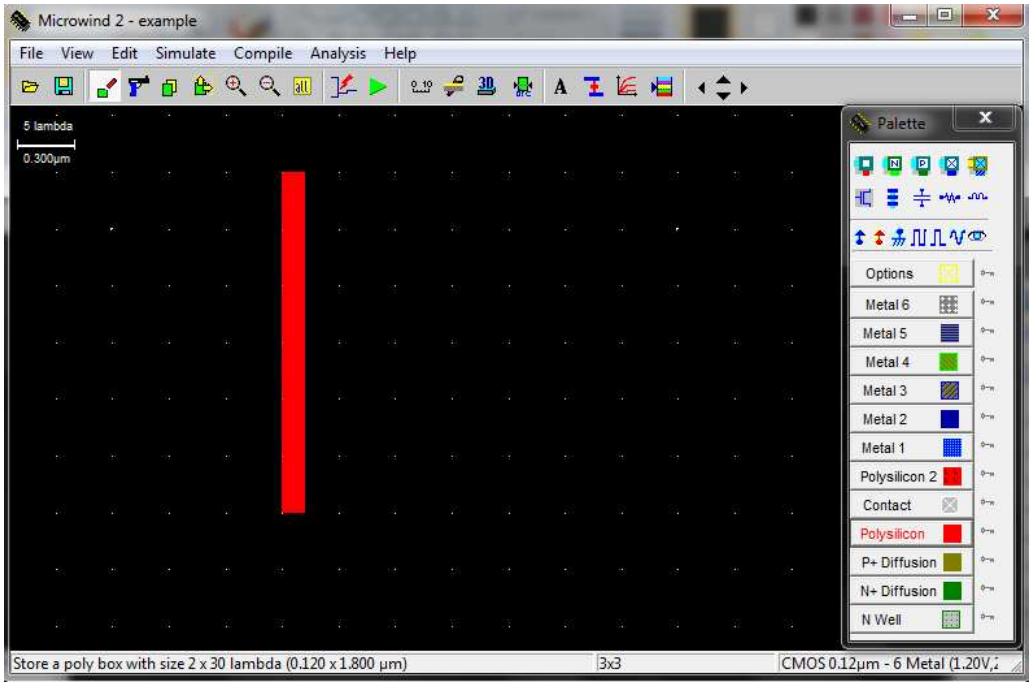
1.3 Thực tập

Khởi động chương trình **Microwind 2**. Công nghệ được sử dụng là 120nm, thực hiện tối đa 6 lớp kim loại và sử dụng điện áp thấp.

Phía trên cửa sổ thiết kế là thanh công cụ chính; bên cạnh chương trình chính là cửa sổ với công cụ để vẽ mạch và lấy linh kiện có sẵn – ***Palette***.

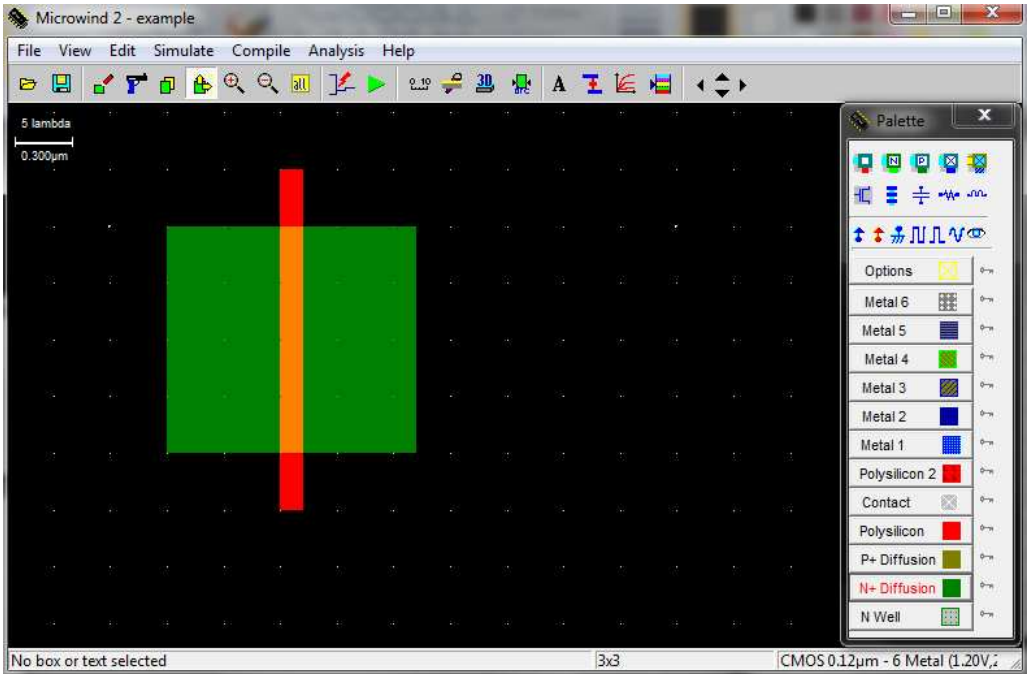


-  Thiết kế transistor nMOS
- Chọn đường vẽ là Polysilicon  . Vẽ một hình chữ nhật tạo lớp polysilicon, chiều rộng tối thiểu 2 λ (trường hợp này là 120nm).



- Chuyển sang chọn lớp N+ diffusion 

Vẽ một hình vuông thể hiện vùng pha tạp n+



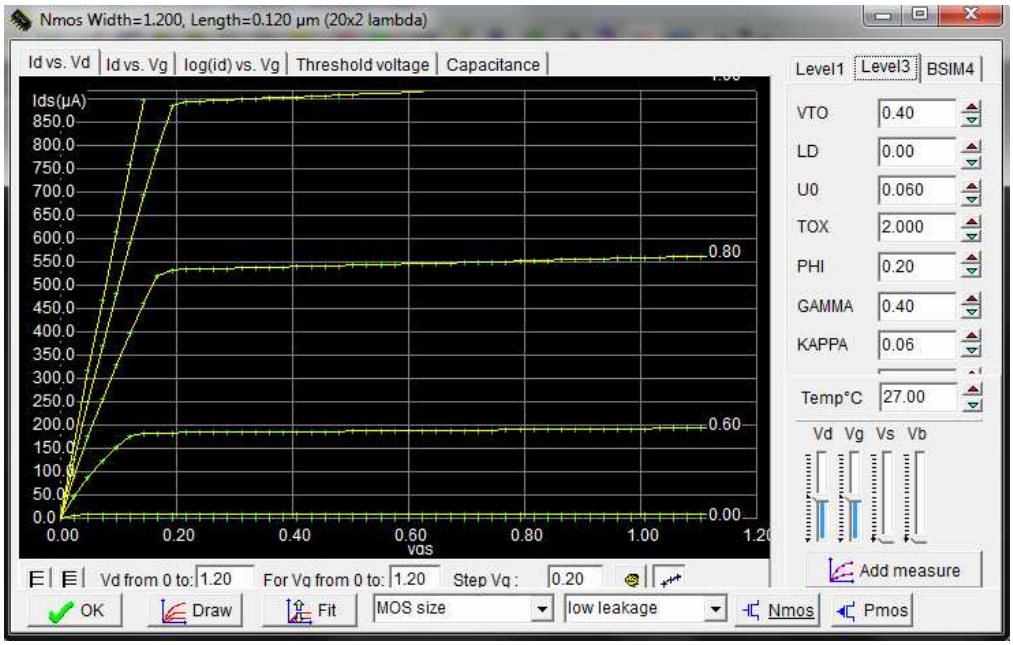
Vùng giữa lớp polysilicon và pha tạp n+ tạo thành kênh dẫn của nMOS.

- Mô phỏng quá trình bằng cách chọn thể hiện 2D hoặc 3D 

Với 2D sẽ cung cấp hình mặt cắt ngang nMOS, trong đó vùng cực cổng poly sẽ có màu đỏ, vùng cực máng/nguồn có màu xanh lá cây trên vùng nền wafer màu xám. Cực cổng được cách ly bởi lớp oxit mỏng gọi là oxit cổng.

Tính chất vật lý của 2 vùng máng và nguồn là như nhau. Về mặt lý thuyết, cực nguồn là nơi cung cấp hạt tải. Trong trường hợp của nMOS, hạt tải chính là electron. Do vậy, cực nguồn là vùng có điện áp thấp nhất. Cực cổng polysilicon nằm trên kênh dẫn, nó chia đôi vùng pha tạp thành vùng máng và vùng nguồn. Cực cổng điều khiển dòng điện chảy qua 2 vùng. Điện áp cao đặt trên cực cổng thu hút electron, tạo ra vùng kênh electron và cho phép dòng điện chảy qua. Điện áp thấp sẽ cấm vùng kênh.

- Kích chọn biểu tượng thể hiện đặc tính MOS 



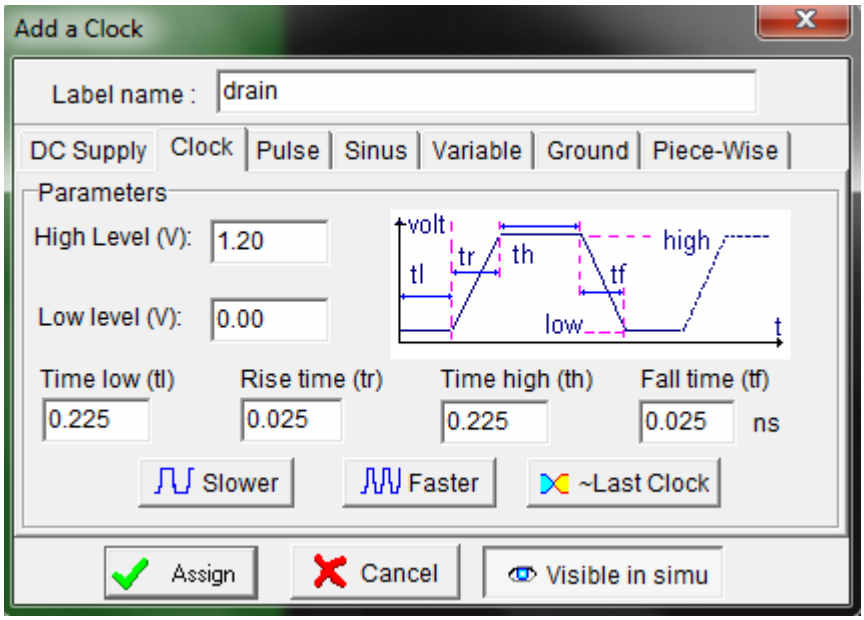
Kích thước của MOS có ảnh hưởng rất lớn đến giá trị của đặc tính dòng điện. Quan sát trong cửa sổ đặc tính, các thuộc tính của MOS được thể hiện cụ thể. Tham số của MOS tương ứng SPICE mức 3, có thể chọn quan sát mức 1 và BSIM4.

Ngoài việc thể hiện nMOS transistor, cửa sổ cho phép thay đổi nhiều tham số và chuyển sang xem đặc tính của pMOS transistor.

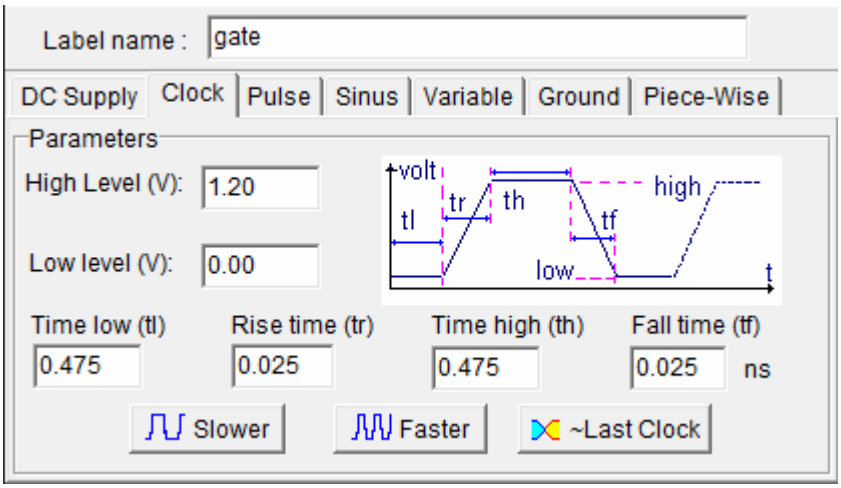
- Mô phỏng

Trước khi mô phỏng, phải đặt thuộc tính cho các mạch điện. Trong ví dụ này, cách đơn giản nhất là đặt lên cực cổng và nguồn một xung nhịp, kết quả thể hiện trên cực máng.

- Cấp nguồn dao động lên vùng máng. Kích chọn biểu tượng là **Clock** (Bộ dao động), đổi tên thành "drain" rồi nhấn **Assign**. Chu kì mặc định là 0,5ns được đặt lên cực máng.



Đặt tín hiệu dao động thứ 2 lên cực cổng. Thao tác như trên. Lần này chu kì là 1ns.

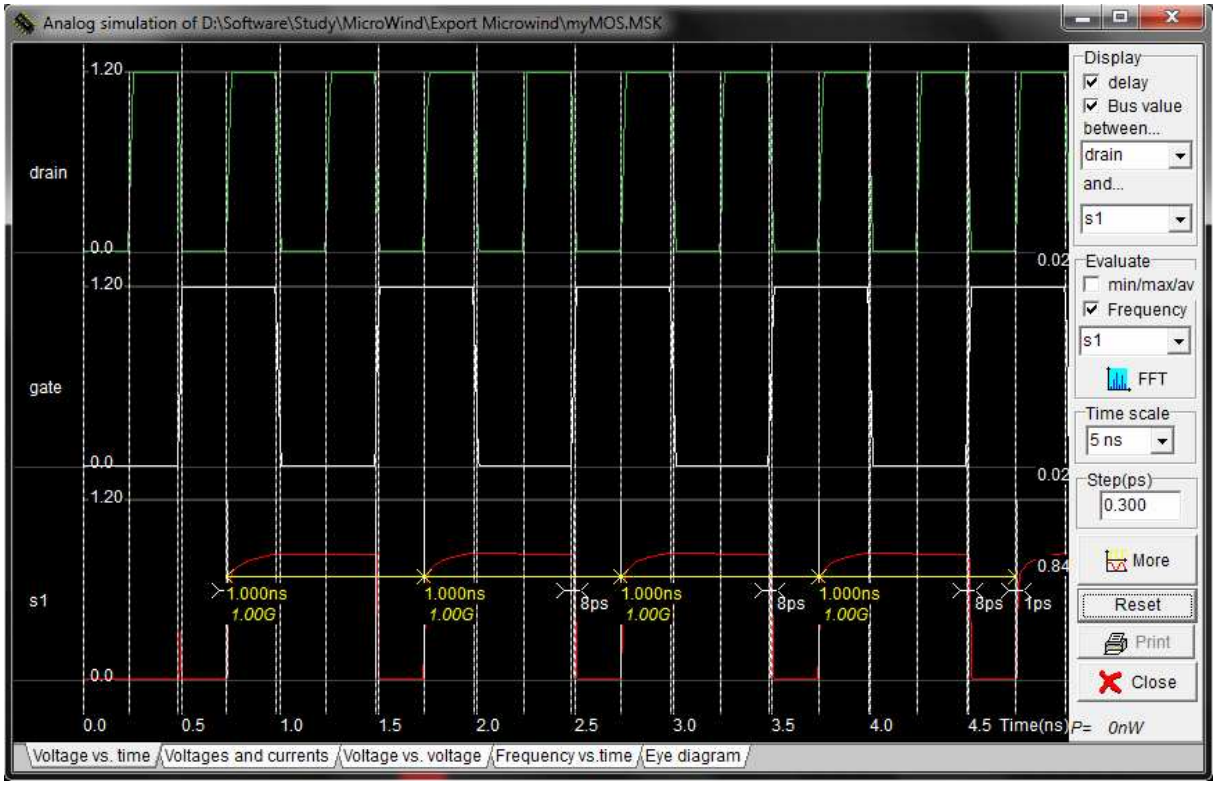


- Kiểm tra lối ra:

Kích chọn biểu tượng là **Visible** (Hiển thị), rồi kích lên vùng nguồn; Chọn **Assign**.

- ***Chú ý***: Luôn luôn lưu trước khi mô phỏng. Chọn **File → Save as**… → gõ tên rồi nhấn OK để lưu lại.

- Khởi động mô phỏng: kích chọn biểu tượng mô phỏng trên thanh công cụ.



Khi điện áp cực cổng bằng 0, không có kênh dẫn nên nút s1 (nguồn) bị ngắt khỏi máng. Khi cực cổng mở, tín hiệu tại nguồn có dạng như máng. Có thể nhận thấy rằng nMOS hoạt động tốt tại mức thế thấp chứ không phải thế cao. Điện áp cuối cùng là 1,2V; đây là giá trị của Vdd trừ đi thế ngưỡng.

1.4 Báo cáo thực tập

Sau thực tập, báo cáo được viết như phần I; bao gồm những nội dung chính sau:

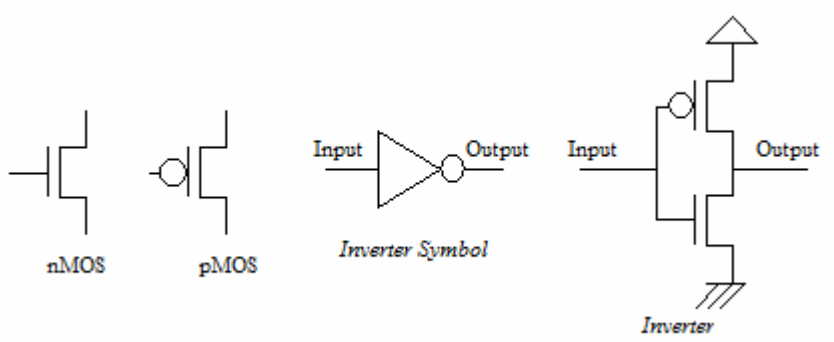
- Quy trình thực tập
- Kết quả thực tập từng phần
- Nhận xét và kết luận

Bài 2
CMOS Inverter

2.1 Giới thiệu

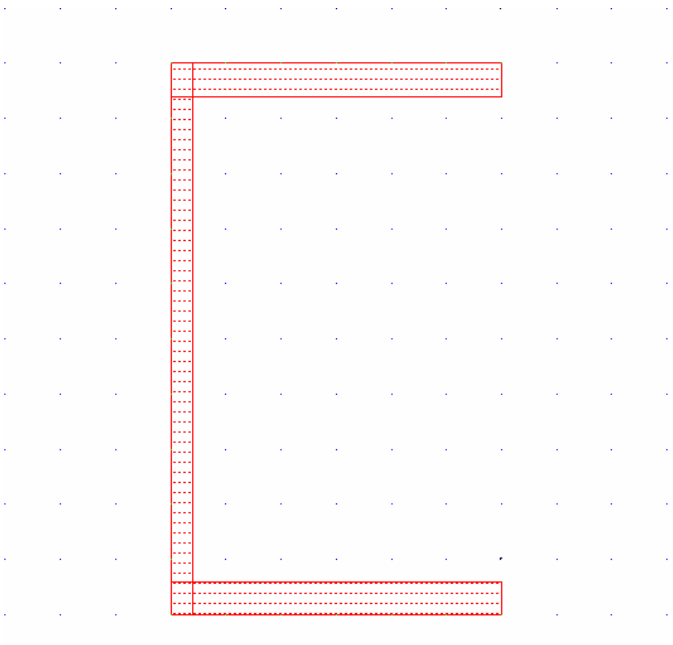
Trong bài thực tập này, cổng đảo sử dụng công nghệ CMOS được sử dụng để minh họa kĩ thuật thiết kế hoàn toàn chủ động. Tất cả các thành phần của linh kiện được vẽ thủ công.

Cổng đảo CMOS chỉ chứa 2 transistor là loại n và loại p.

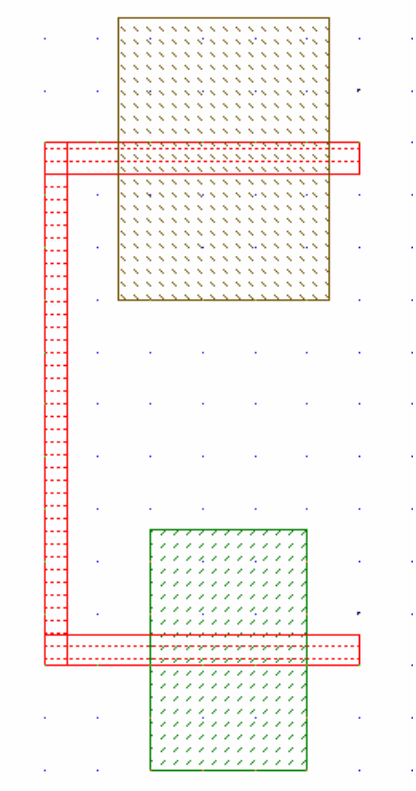


2.2 Thực hành

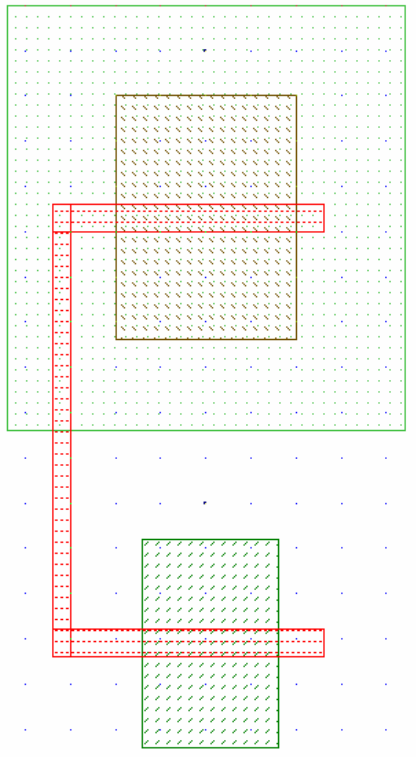
Sử dụng lớp *polysilicon* vẽ hộp chữ nhật có bề rộng nhỏ nhất là 2 λ. Vẽ tiếp 2 hình để tạo hình như sau:



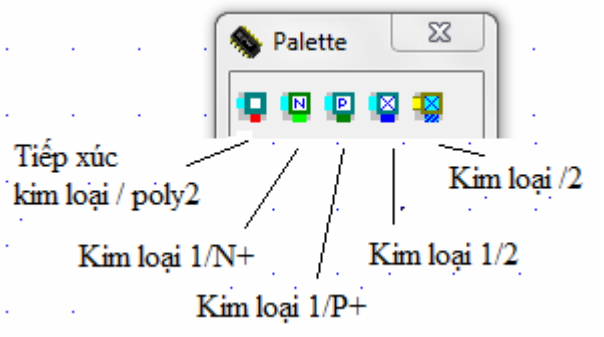
- Chọn lớp vẽ là *N+ diffusion*, vẽ một hình hộp ở phía dưới; lớp N+ được thể hiện bằng hình màu xanh lá cây. Vùng dưới lớp polysilicon là kênh dẫn của nMOS.



- Chuyển sang lớp *P+ diffusion*, vẽ hình hộp ở phía trên. Lớp p+ có màu nâu. Chuyển sang lớp vẽ *N well*. Vẽ giếng pha tạp N bao quanh vùng p+.

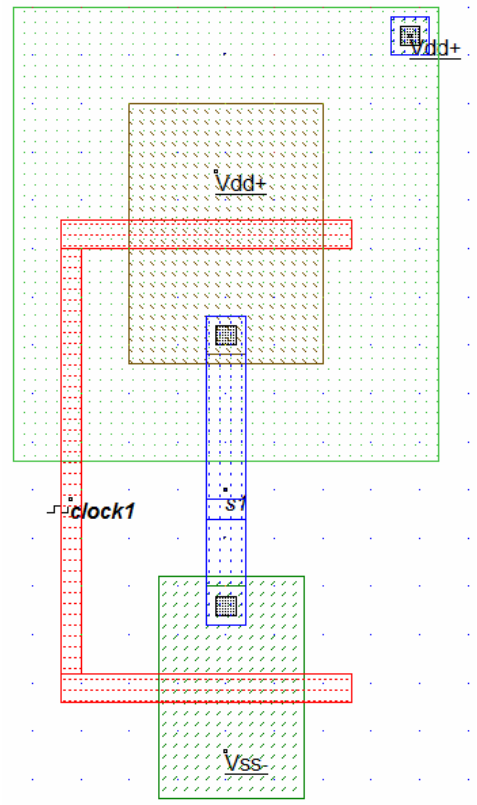


Những vùng pha tạp phải được kết nối bằng lớp kim loại. Khu vực kim loại được cách ly khỏi vùng pha tạp bởi lớp oxit SiO_2. Điểm tiếp xúc được sử dụng để khoan lỗ kết nối kim loại với vùng pha tạp. Điểm tiếp xúc có thể được tự vẽ hoặc lấy sẵn trong bảng công cụ.



- Kích chọn điểm tiếp xúc *Kim loại/n+*. Kích chuột lên vùng pha tạp n+ để gắn.
- Điểm tiếp xúc *Kim loại/n+* nằm trên góc của giếng n+, nó sẽ được áp thế Vdd.
- Kích chọn tiếp xúc *Kim loại/p+* và gắn nó lên vùng pha tạp p+

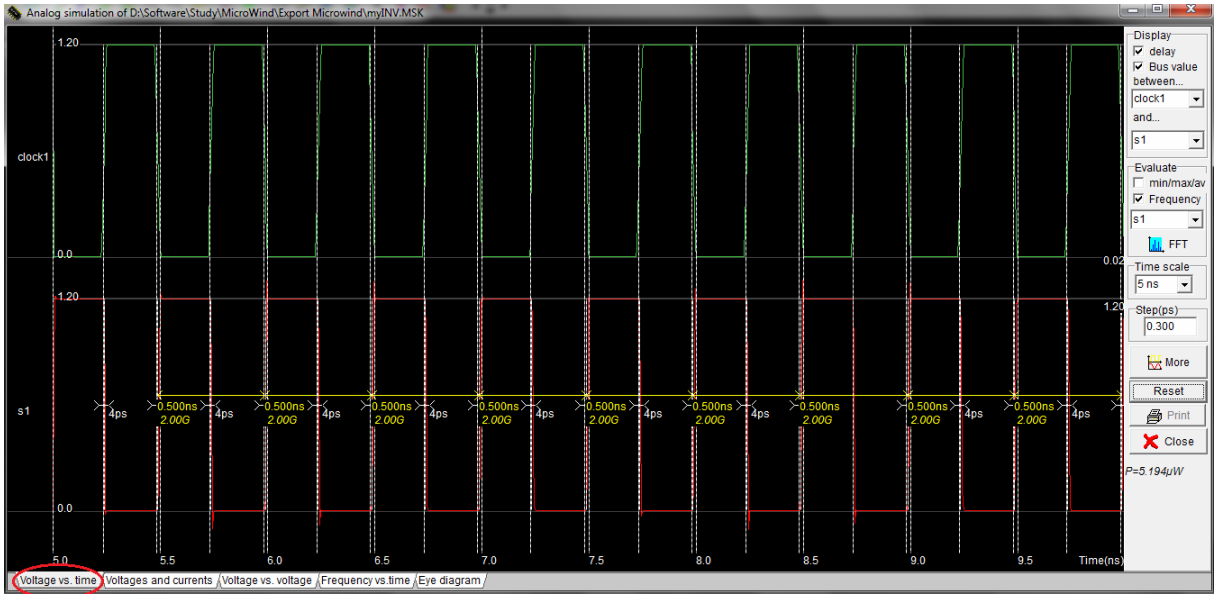
- Đặt thuộc tính để mô phỏng



- Chọn Vdd và kích lên vùng pha tạp p
- Vùng giếng n cũng cần được áp thế Vdd
- Chọn Vss và kích lên vùng pha tạp n
- Lối vào cần áp xung: kích chọn Clock và đặt lên cổng polysilicon.
- Kích chọn Visible và đặt lên dây kim loại

**** Chú ý***: Luôn luôn lưu trước khi mô phỏng. Chọn **File → Save as**… → gõ tên rồi nhấn OK để lưu lại.

- Khởi động chương trình mô phỏng 



- Chọn thẻ Voltage & Currents để quan sát cả thông số điện áp và dòng điện.
- Chọn thẻ Voltage vs. Voltage để quan sát đặc tính hoạt động của cổng đảo.

- Kiểm tra Luật thiết kế DRC 

Lỗi thiết kế được thể hiện trong cửa sổ thiết kế, bên cạnh là thông tin báo lỗi. Chỉ khi thiết kế không chứa lỗi mới đủ điều kiện gửi đi chế tạo.

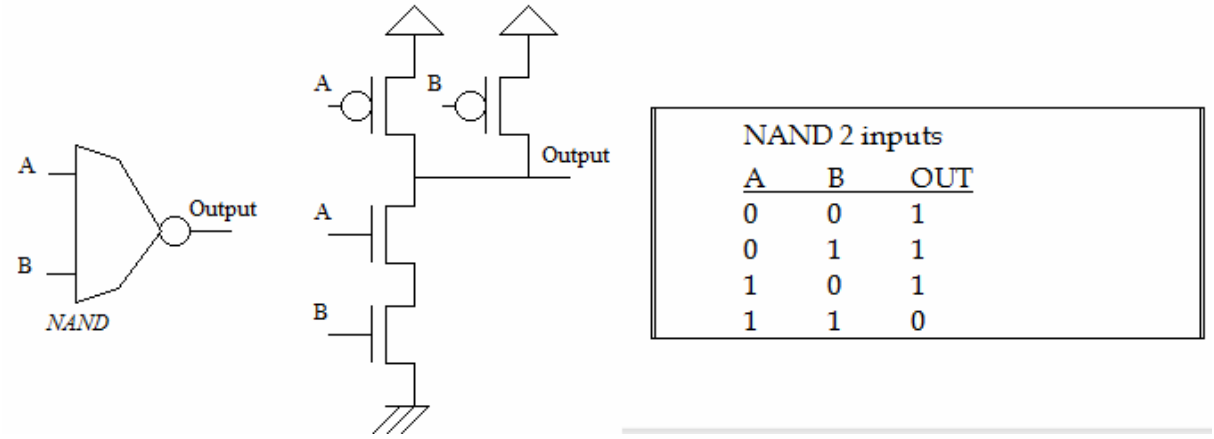
Bài 3
CÁC CỔNG CỬA CƠ BẢN

3.1 Giới thiệu

Trong bài trước đã trình bày cách thiết kế một cổng đảo theo phương pháp hoàn toàn chủ động. Để tiết kiệm thời gian thiết kế, người ta xây dựng thư viện linh kiện và tiếp cận phương pháp thiết kế bán chủ động.

Một vi mạch phức tạp có thể được thiết kế bằng ngôn ngữ mô tả hệ thống như Verilog, hoặc biên dịch từ một sơ đồ nguyên lý mạch. Trong bài thực tập này, một cổng NAND được tạo ra từ cách biên dịch một sơ đồ mạch sử dụng phần mềm DSCH.

Cổng NAND 2 lối vào bao gồm 2 nMOS mắc nối tiếp với 2 pMOS mắc song song. Bảng sự thật và sơ đồ như sau:



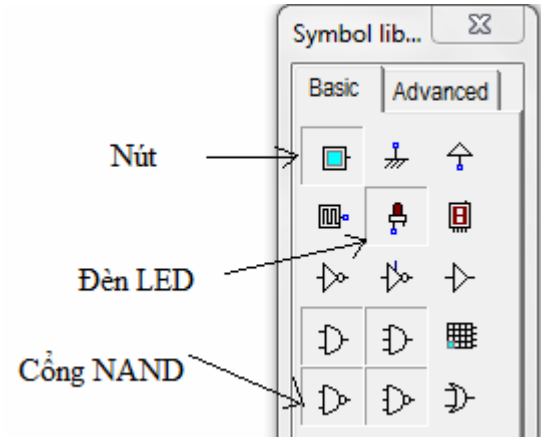
NAND 2 inputs

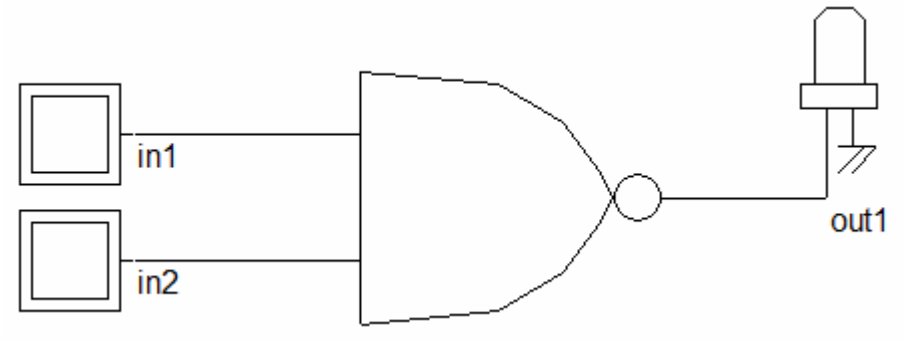
A	B	OUT
0	0	1
0	1	1
1	0	1
1	1	0

3.2 Thực tập

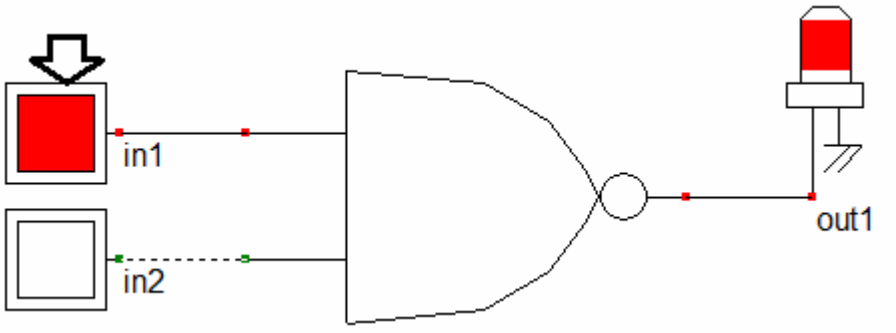
- Khởi động chương trình DSCH.
- Bên cạnh cửa sổ chính là bảng linh kiện cho sẵn
- Chọn biểu tượng NAND 2 lối vào, kéo thả vào cửa sổ thiết kế.
- Chọn Button (Nút), kéo thả 2 Nút gần cổng NAND thể hiện trạng thái lối vào
- Chọn đèn LED, để tại lối ra.

- Nối dây bằng  phía trên thanh công cụ

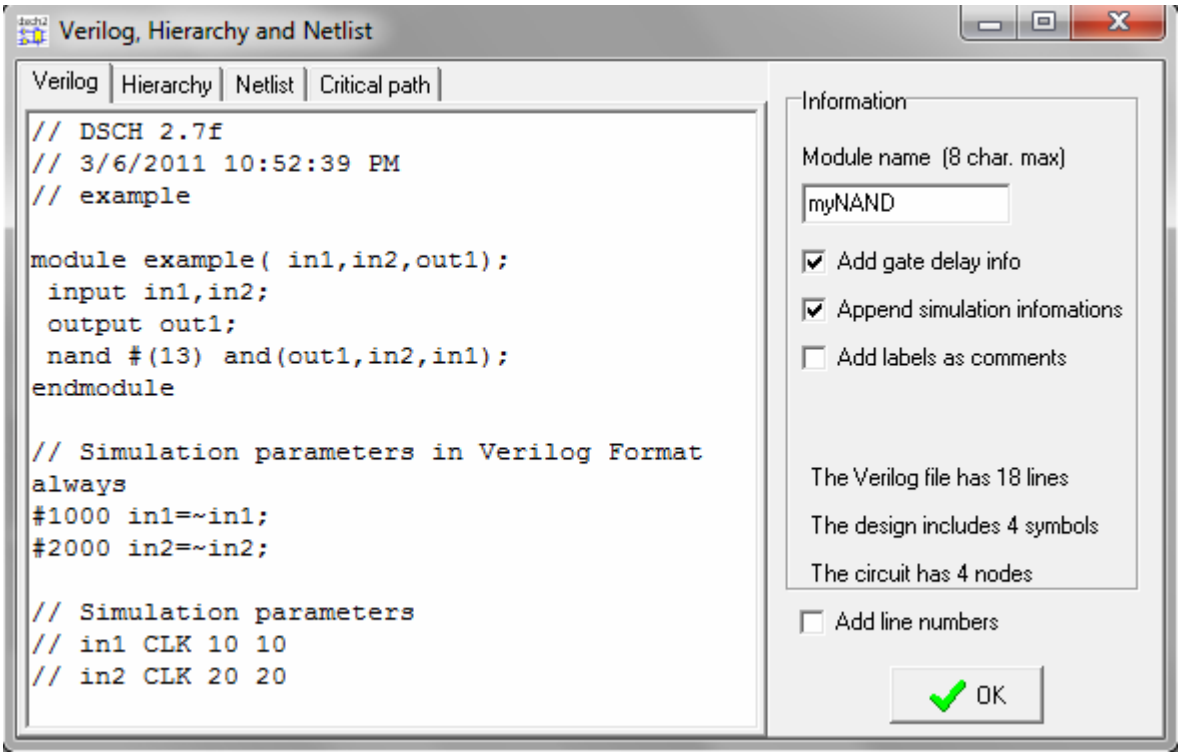




Mô phỏng mạch bằng nút  phía trên thanh công cụ; nhấn nút 1 hoặc 2 để thay đổi trạng thái lối vào và quan sát đèn LED để nhận biết trạng thái lối ra.

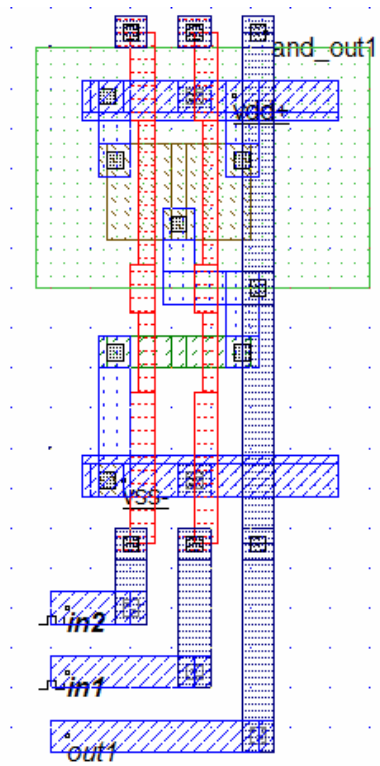


Bước tiếp theo chọn **File → Make Verilog File → OK**



Khởi động chương trình **Microwind**, chọn **Compile → Compile Verilog File (.txt)**

Chọn tập tin vừa tạo ra ở trên → **Compile → Back to editor**



Mô phỏng cổng NAND

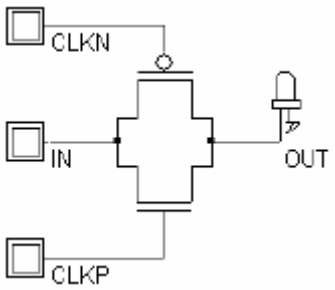
- Chọn Clock, cấp cho lối vào in1 và in2
- Chọn Visible và kích lên lối ra out1.
- Mô phỏng 

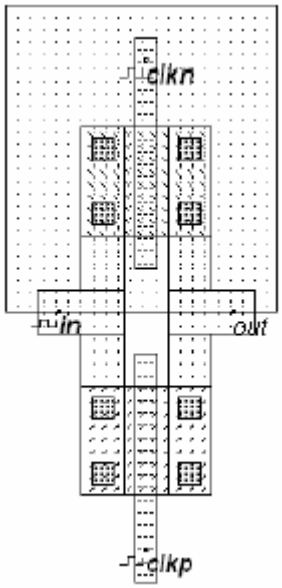
Độ dốc xung lối ra khi lên cao nhanh hơn vì mắc pMOS song song.

3.3 Các cổng cửa cơ bản

3.3.1 Cổng truyền (Transmission gate)

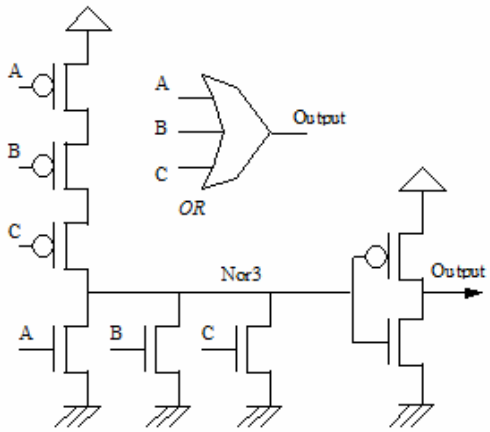
Dựa trên những thao tác đã làm ở phần trước, khảo sát cổng truyền





3.3.2 Cổng OR 3 lối vào

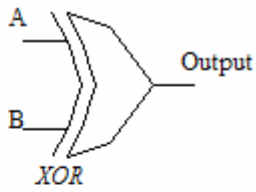
Dựa trên những thao tác đã làm ở phần trước, khảo sát cổng OR 3 lối vào.

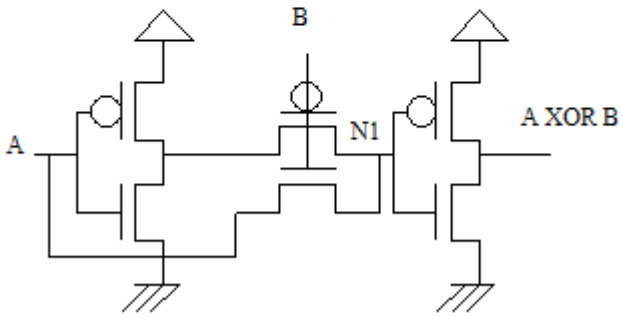


OR 3 Inputs			
A	B	C	Or3
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

3.3.3 Khảo sát cổng XOR

XOR 2 inputs		
A	B	OUT
0	0	0
0	1	1
1	0	1
1	1	0



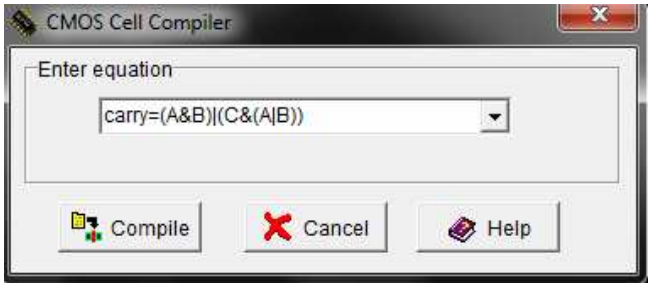


XOR gate

3.4 Cổng kết hợp

Bộ biên dịch có sẵn trong Microwind cho phép tạo ra nhiều cổng phức tạp bằng cách kết hợp phép toán OR và AND.

Chọn **Compile → Compile one Line**



Khảo sát một số cổng theo bảng cho sẵn sau

COMPILER SYNTAX	
CELL	FORMULA
Inverter	out=/in
NAND gate	n=/(a.b)
AND gate	s=(a.b)
3 Input OR	s=a+b+c
3 Input NAND	out=/(a.b.c)
AND-OR Gate	cgate=a.(b+c)
CARRY Cell	cout=(a.b)+(cin.(a+b))

Bài 4
CỔNG SỐ HỌC

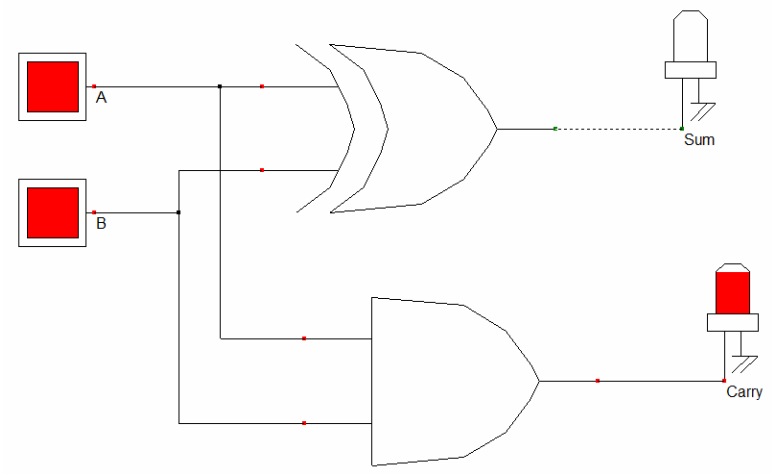
4.1 Bộ cộng bán phần

Trong phần I của tài liệu, các cổng số học được viết bởi ngôn ngữ VHDL và lập trình cho FPGA. Trong bài này, chúng được khảo sát ở góc độ thiết kế ASIC.

- Bảng sự thật và sơ đồ

HALF ADDER

A	B	SUM	CARRY
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



4.1.1 Thiết kế bằng phương pháp hoàn toàn chủ động

- Khởi động chương trình DSCH
- Mô phỏng bộ cộng bán phần
- Khởi động chương trình Microwind
- Thiết kế cổng AND 2 lối vào
- Thiết kế cổng XOR 2 lối vào
- Đi dây kim loại để liên kết 2 cổng trên để tạo bộ cộng bán phần.

4.1.2 Biên dịch từ Verilog

- Khởi động chương trình Notepad
- Viết đoạn chương trình mô tả bộ cộng bán phần bằng ngôn ngữ Verilog

```
module halfadd (a, b, sum, carry);
        input a, b;
        output sum, carry;

        XOR xor1(sum, a, b);
        AND and1(carry, a, b);
endmodule
```

- Lưu tập tin dưới dạng "*.txt", ví dụ halfadd.txt
- Trong Microwind, chọn **Compile → Compile Verilog File…**

4.2 Bộ cộng toàn phần

Giống như trên, bộ cộng toàn phần được thực hiện từ những cổng cửa cơ bản.

FULL ADDER

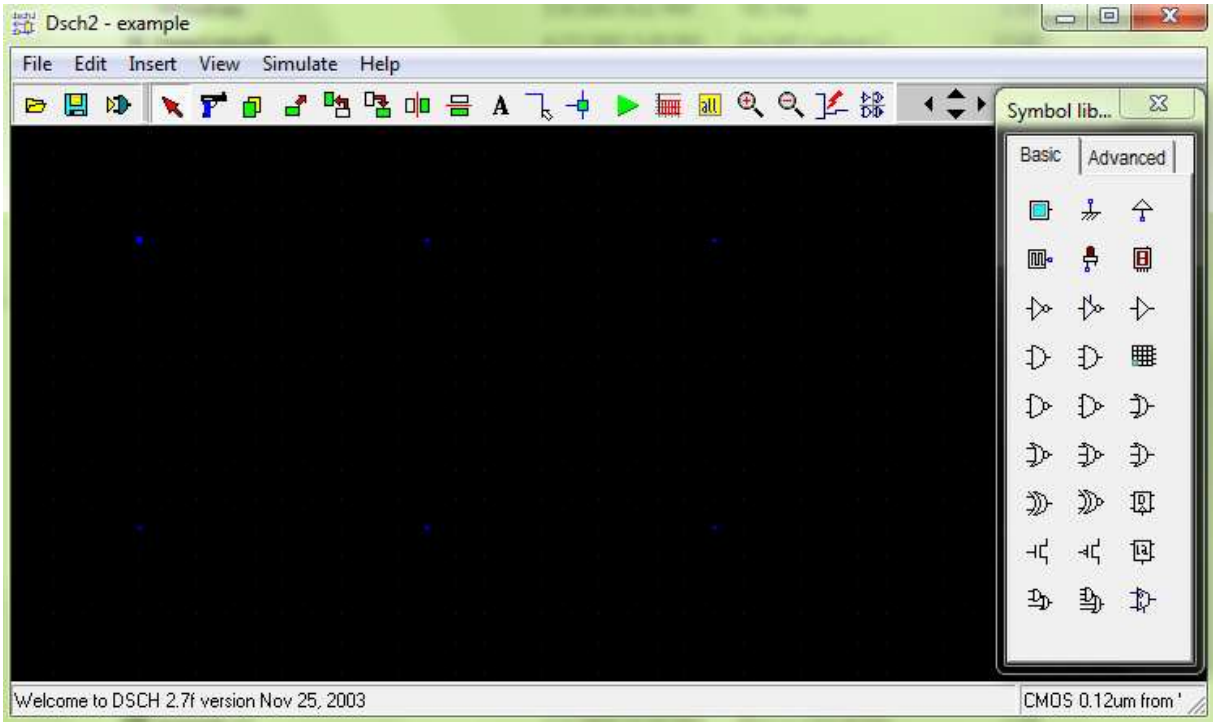
A	B	C	SUM	CARRY
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

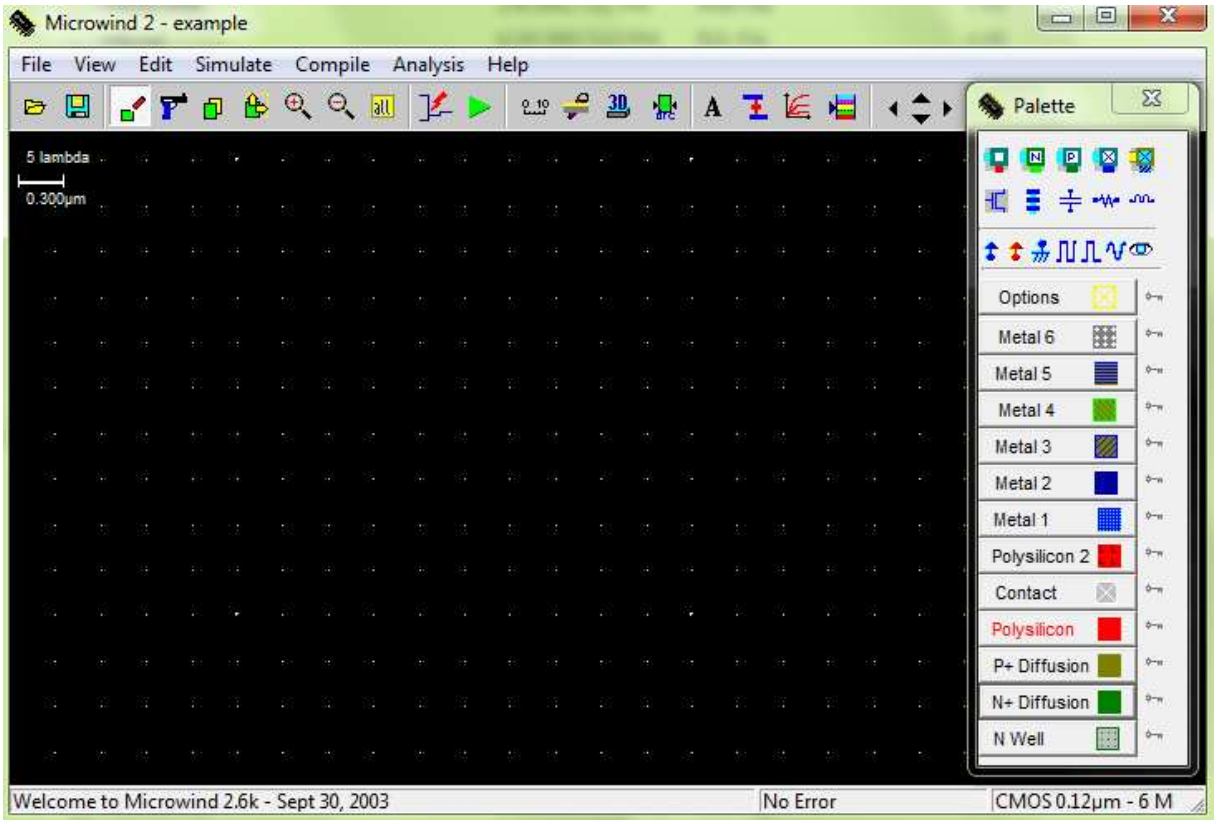
- Khởi động DSCH, vẽ sơ đồ mạch cộng toàn phần.
- Mô phỏng
- Biên dịch ra tập tin Verilog
- Khởi động Microwind, biên dịch từ tập tin Verilog ra bản vẽ thiết kế

Bài 5

TỰ CHỌN

Sử dụng DSCH2 & Microwind2 thiết kế một mạch số tự chọn.





PHỤ LỤC A

BẢNG GÁN CHÂN FPGA XC3S200 – FT256 BOARD XILINX S3

A.1 Switch, Buttons & LED

Switch	SW7	SW6	SW5	SW4	SW3	SW2	SW1	SW0
FPGA Pin	K13	K14	J13	J14	H13	H14	G12	F12

Push Button	BTN3 (User Reset)	BTN2	BTN1	BTN0
FPGA Pin	L14	L13	M14	M13

LED	LD7	LD6	LD5	LD4	LD3	LD2	LD1	LD0
FPGA Pin	P11	P12	N12	P13	N14	L12	P14	K12

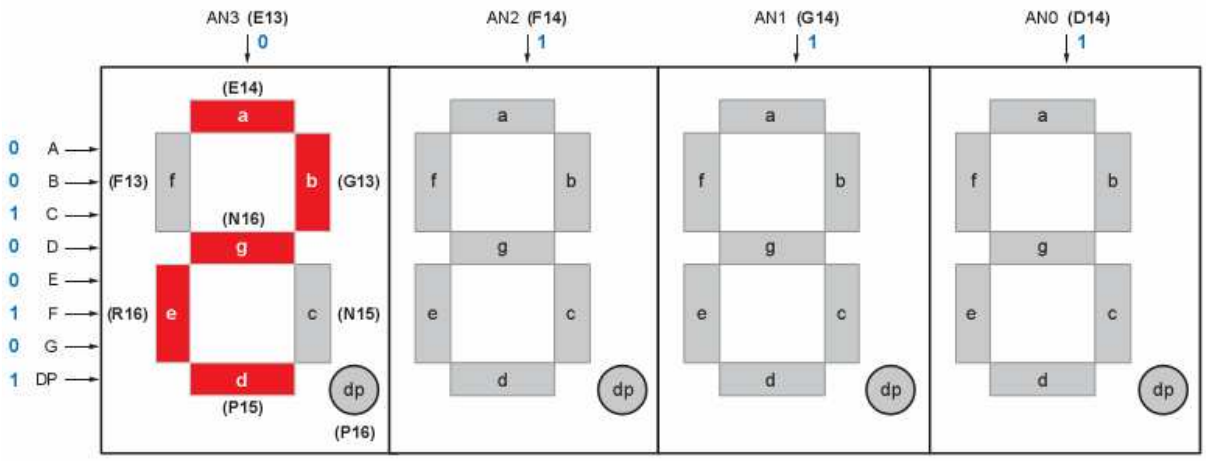
A.2 Seven Segments LEDs

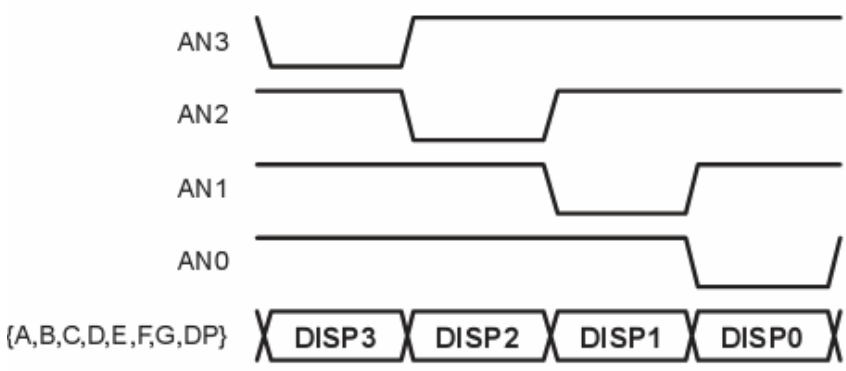
FPGA Connections to Seven-Segment Display (Active Low)

Segment	FPGA Pin
A	E14
B	G13
C	N15
D	P15
E	R16
F	F13
G	N16
DP	P16

Digit Enable (Anode Control) Signals (Active Low)

Anode Control	AN3	AN2	AN1	AN0
FPGA Pin	E13	F14	G14	D14





Display Characters and Resulting LED Segment Control Values

Character	a	b	c	d	e	f	g
0	0	0	0	0	0	0	1
1	1	0	0	1	1	1	1
2	0	0	1	0	0	1	0
3	0	0	0	0	1	1	0
4	1	0	0	1	1	0	0
5	0	1	0	0	1	0	0
6	0	1	0	0	0	0	0
7	0	0	0	1	1	1	1
8	0	0	0	0	0	0	0
9	0	0	0	0	1	0	0
A	0	0	0	1	0	0	0
b	1	1	0	0	0	0	0
C	0	1	1	0	0	0	1
d	1	0	0	0	0	1	0
E	0	1	1	0	0	0	0
F	0	1	1	1	0	0	0

A.3 Clock Sources

Clock Oscillator Sources

Oscillator Source	FPGA Pin
50 MHz (IC4)	T9
Socket (IC8)	D9

A.4 VGA

VGA Port Connections to the Spartan-3 FPGA

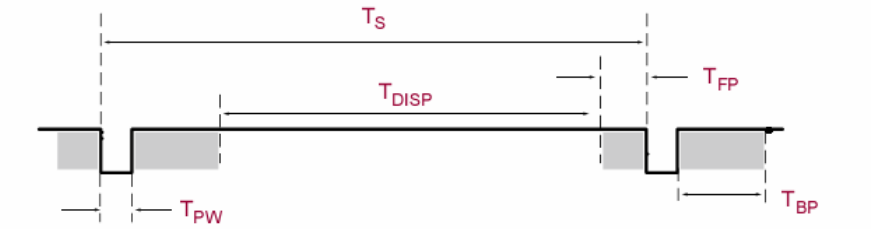
Signal	FPGA Pin
Red (R)	R12
Green (G)	T12
Blue (B)	R11
Horizontal Sync (HS)	R9
Vertical Sync (VS)	T10

3-Bit Display Color Codes

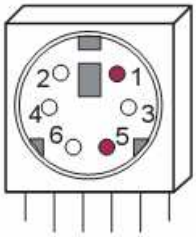
Red (R)	Green (G)	Blue (B)	Resulting Color
0	0	0	Black
0	0	1	Blue
0	1	0	Green
0	1	1	Cyan
1	0	0	Red
1	0	1	Magenta
1	1	0	Yellow
1	1	1	White

640x480 Mode VGA Timing

Symbol	Parameter	Vertical Sync			Horizontal Sync	
		Time	Clocks	Lines	Time	Clocks
T_S	Sync pulse time	16.7 ms	416,800	521	32 µs	800
T_{DISP}	Display time	15.36 ms	384,000	480	25.6 µs	640
T_{PW}	Pulse width	64 µs	1,600	2	3.84 µs	96
T_{FP}	Front porch	320 µs	8,000	10	640 ns	16
T_{BP}	Back porch	928 µs	23,200	29	1.92 µs	48



A.5 RS232 & PS2



PS/2 DIN Connector

PS/2 Connections to the Spartan-3 FPGA

PS/2 DIN Pin	Signal	FPGA Pin
1	DATA (PS2D)	M15
2	Reserved	—
3	GND	GND
4	Voltage Supply	—
5	CLK (PS2C)	M16
6	Reserved	—

Accessory Port Connections to the Spartan-3 FPGA

Signal	FPGA Pin
RXD	T13
TXD	R13
RXD-A	N10
TXD-A	T14

A.6 SRAM

External SRAM Control Signal Connections to Spartan-3 FPGA

Signal	FPGA Pin	A1 Expansion Connector Pin
OE#	K4	16
WE#	G3	18

External SRAM Address Bus Connections to Spartan-3 FPGA

Address Bit	FPGA Pin	A1 Expansion Connector Pin
A17	L3	35
A16	K5	33
A15	K3	34
A14	J3	31
A13	J4	32
A12	H4	29
A11	H3	30
A10	G5	27
A9	E4	28
A8	E3	25
A7	F4	26
A6	F3	23
A5	G4	24
A4	L4	14
A3	M3	12
A2	M4	10
A1	N3	8
A0	L5	6

SRAM IC10 Connections

Signal	FPGA Pin	A1 Expansion Connector Pin
IO15	R1	
IO14	P1	
IO13	L2	
IO12	J2	
IO11	H1	
IO10	F2	
IO9	P8	
IO8	D3	
IO7	B1	19
IO6	C1	17
IO5	C2	15
IO4	R5	13
IO3	T5	11
IO2	R6	9
IO1	T8	7
IO0	N7	5
CE1 (chip enable IC10)	P7	
UB1 (upper byte enable IC10)	T4	
LB1 (lower byte enable IC10)	P6	

SRAM IC11 Connections

Signal	FPGA Pin
IO15	N1
IO14	M1
IO13	K2
IO12	C3
IO11	F5
IO10	G1
IO9	E2
IO8	D2
IO7	D1
IO6	E1
IO5	G2
IO4	J1
IO3	K1
IO2	M2
IO1	N2
IO0	P2
CE2 (chip enable IC11)	N5
UB2 (upper byte enable IC11)	R4
LB2 (lower byte enable IC11)	P5