
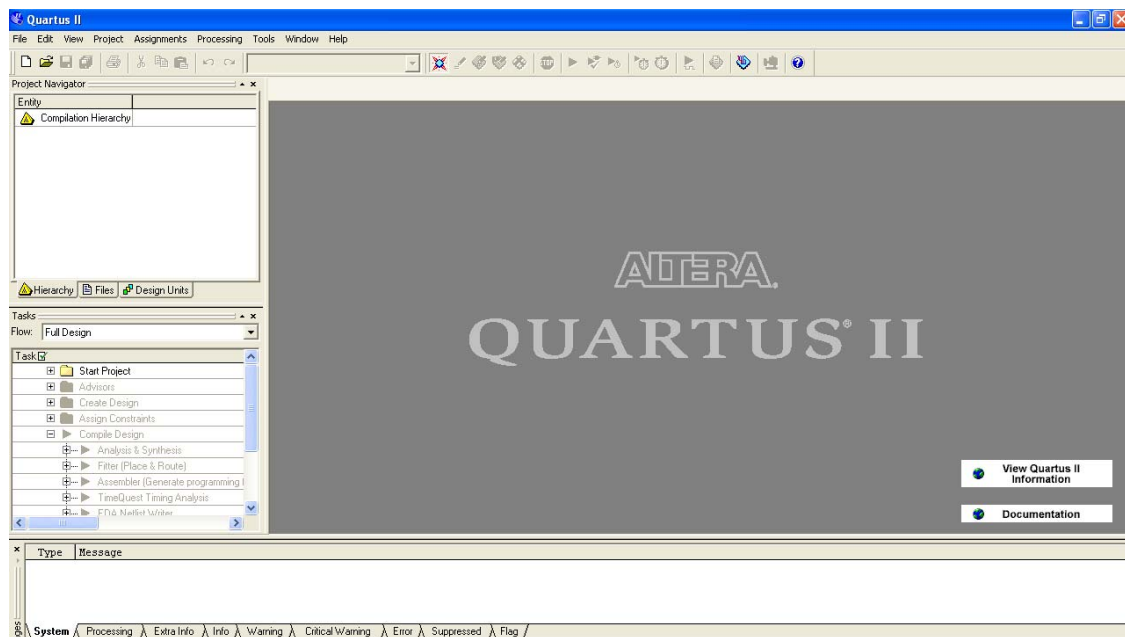


BÀI 1 : HƯỚNG DẪN SỬ DỤNG QUARTUS II

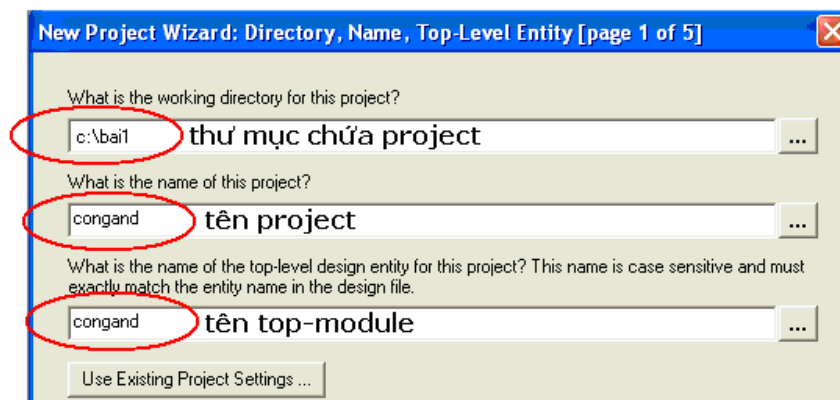
1/- Tạo project :

1. Sau khi cài đặt xong phần mềm QuartusII, bắt đầu chạy chương trình bằng cách double-click vào biểu tượng  trên desktop.

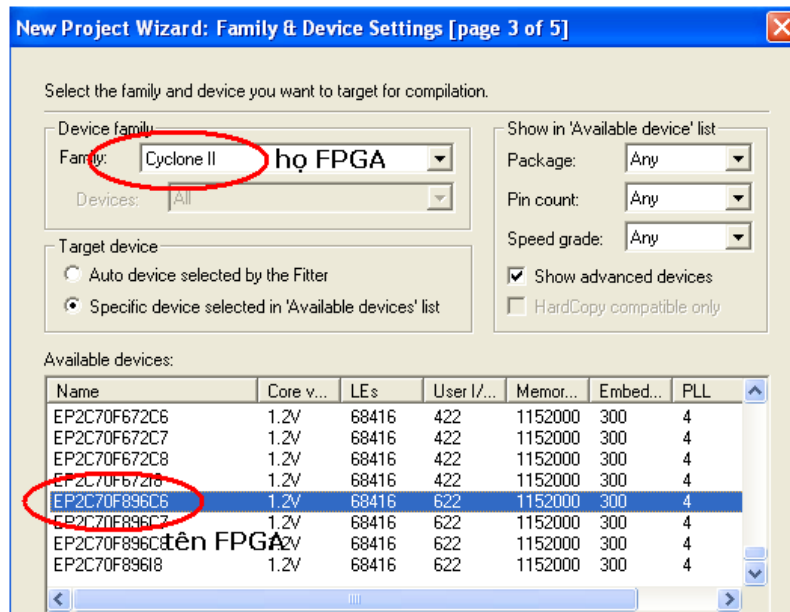
Giao diện QuartusII sẽ xuất hiện :



2. Đầu tiên, cần tạo một project mới : File → New Project Wizard. Ở cửa sổ đầu tiên điền vào thông tin về thư mục chứa project, tên project và tên top-module (tên top-module thường trùng tên project). Click Next 2 lần.

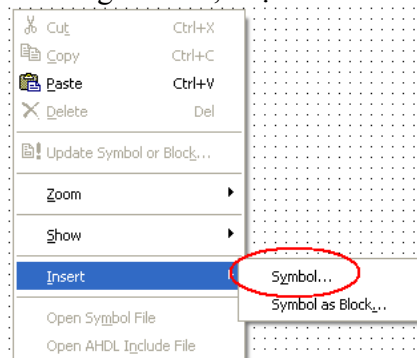


3. Cửa sổ Family & Device Settings dùng để chọn họ và tên linh kiện FPGA để cấu hình. Chọn họ linh kiện **CycloneII**, tên **EP2C70F896C6** (board DE2-70). Chọn Finish.

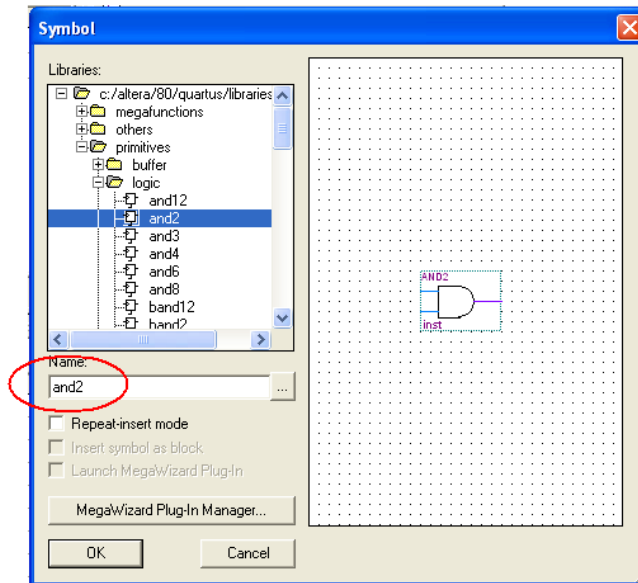


4. Vào File → New → Block Diagram/Schematic File.

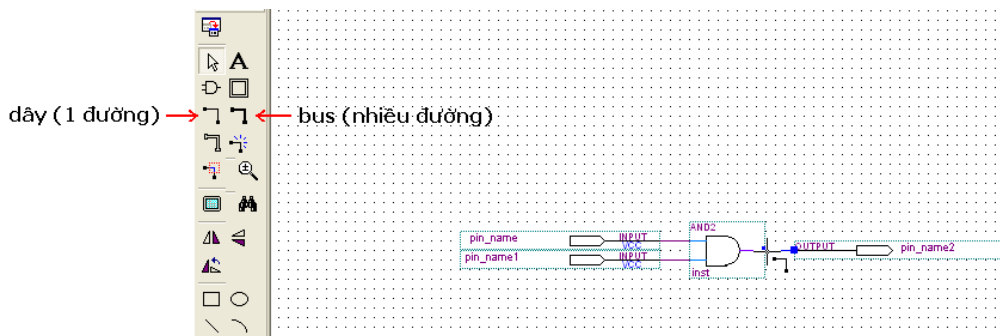
5. Click chuột phải vào trong thiết kế, chọn Insert → Symbol.



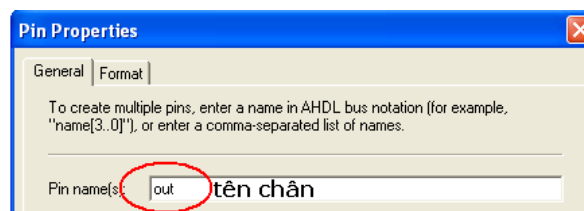
Chọn cổng AND bằng cách gõ vào “and2”. Bấm OK. Gắn vào trong thiết kế.



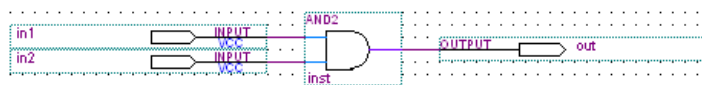
6. Làm tương tự bước 5 để gắn input (ngõ vào) và output (ngõ ra) cho thiết kế (có thể dùng phím Ctrl để copy). Đưa chuột vào chân của linh kiện và thực hiện nối dây.



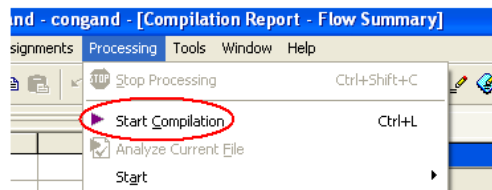
7. Đặt tên cho input và output (input : in1, in2; output : out) bằng cách double-click vào symbol.



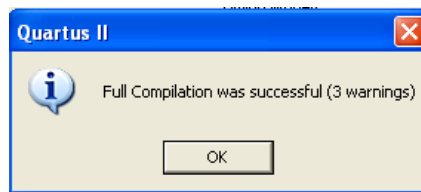
8. Cuối cùng ta được hình cổng AND với input và output, chọn File → Save, tên file : congand.



9. Biên dịch thiết kế chọn Processing → Start Compilation

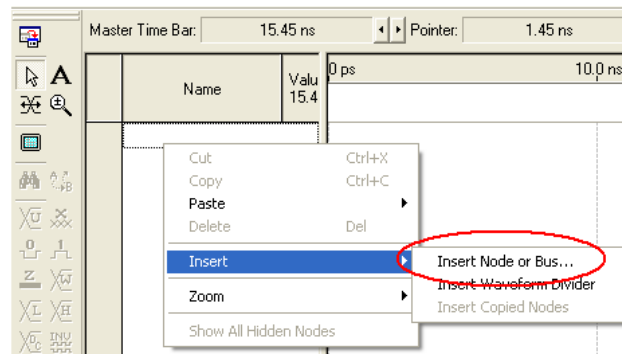


10. Nếu không có lỗi, sẽ xuất hiện cửa sổ báo successful. Bấm OK.

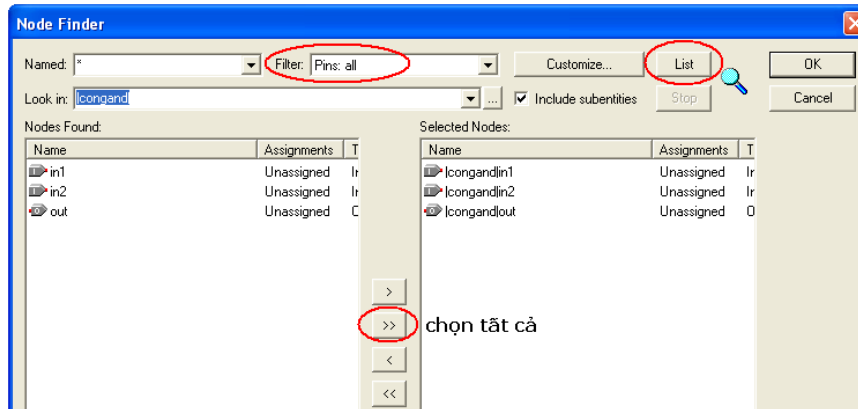
** Mô phỏng thiết kế*

11. Vào File → New → Vector Waveform File.

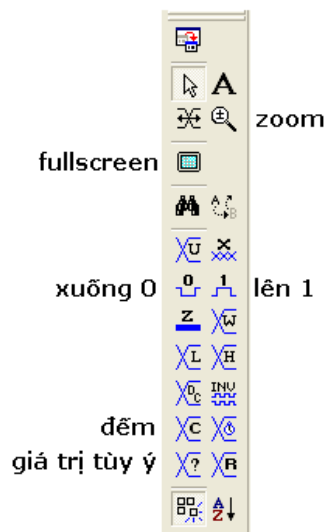
12. Click chuột phải vào cửa sổ “Name”. Chọn Insert → Insert Node or Bus.



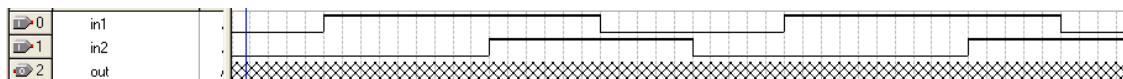
13. Chọn Node Finder. Cửa sổ Node Finder chọn “Pins: all” và bấm List. Chọn tất cả các chân. Bấm OK 2 lần.



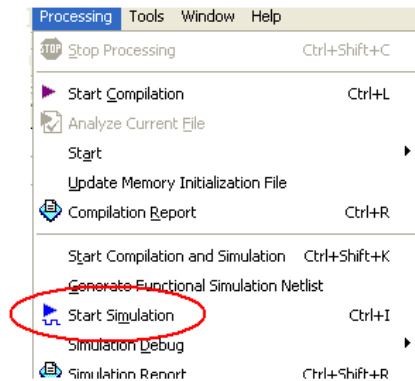
14. Vẽ dạng sóng cho các đường input bằng hộp công cụ bên trái



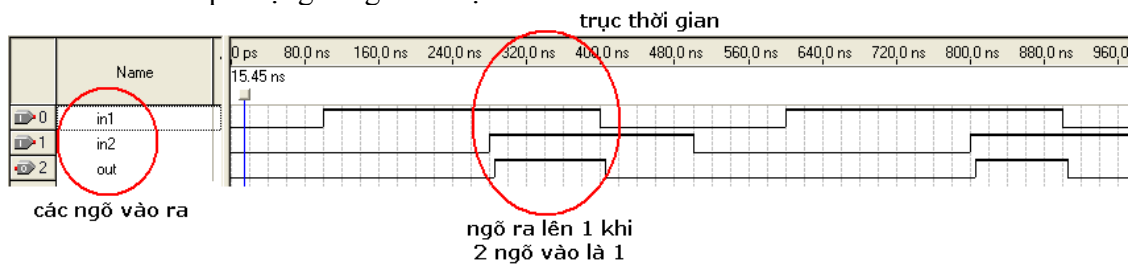
15. Zoom out, dùng các biểu tượng lên 1 và xuống 0 để vẽ các đường tín hiệu ngõ vào. Lưu lại với tên file : conqand.vwf.



16. Vào Processing → Start Simulation để mô phỏng.

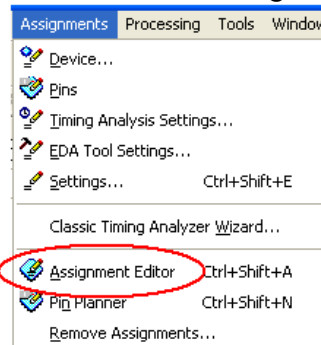


17. Kết quả dạng sóng thu được.



* Cấu hình cho FPGA trên DE2-70

18. Thực hiện map chân cho FPGA : vào Assignments → Assignment Editor

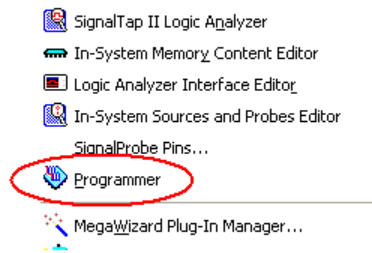


19. Map chân cho 2 ngõ vào của cổng AND với nút gạt SW[0] và SW[1], ngõ ra nối với led đỏ LEDR[0].

	From	To	Assignment Name	Value	Enabled
1			Partition Hierarchy	root_partition	Yes
2		in1	Location	PIN_AA23	Yes
3		in2	Location	PIN_AB26	Yes
4		out	Location	PIN_AJ6	Yes
5	<<new>>	<<new>>	<<new>>		







tên chân FPGA

20. Sau khi map chân xong, Save và Compile lại một lần nữa. Để cấu hình cho FPGA: chọn Tools → Programmer



21. Bấm Start. Sau khi chạy 100%, FPGA đã được cấu hình xong. Kiểm tra lại hoạt động của thiết kế trên kit DE2-70.


Bài tập : Thay đổi các cổng logic OR, XOR, NAND, NOR, XNOR và kiểm tra bảng chân trị của chúng trên DE2-70.

<p>AND</p>  <p>$Y = AB$</p> <table border="1" style="border-collapse: collapse; margin-left: auto; margin-right: auto;"> <thead> <tr> <th>A</th> <th>B</th> <th>Y</th> </tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </tbody> </table>	A	B	Y	0	0	0	0	1	0	1	0	0	1	1	1	<p>OR</p>  <p>$Y = A + B$</p> <table border="1" style="border-collapse: collapse; margin-left: auto; margin-right: auto;"> <thead> <tr> <th>A</th> <th>B</th> <th>Y</th> </tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </tbody> </table>	A	B	Y	0	0	0	0	1	1	1	0	1	1	1	1	<p>XOR</p>  <p>$Y = A \oplus B$</p> <table border="1" style="border-collapse: collapse; margin-left: auto; margin-right: auto;"> <thead> <tr> <th>A</th> <th>B</th> <th>Y</th> </tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </tbody> </table>	A	B	Y	0	0	0	0	1	1	1	0	1	1	1	0
A	B	Y																																													
0	0	0																																													
0	1	0																																													
1	0	0																																													
1	1	1																																													
A	B	Y																																													
0	0	0																																													
0	1	1																																													
1	0	1																																													
1	1	1																																													
A	B	Y																																													
0	0	0																																													
0	1	1																																													
1	0	1																																													
1	1	0																																													
<p>NAND</p>  <p>$Y = \overline{AB}$</p> <table border="1" style="border-collapse: collapse; margin-left: auto; margin-right: auto;"> <thead> <tr> <th>A</th> <th>B</th> <th>Y</th> </tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </tbody> </table>	A	B	Y	0	0	1	0	1	1	1	0	1	1	1	0	<p>NOR</p>  <p>$Y = \overline{A + B}$</p> <table border="1" style="border-collapse: collapse; margin-left: auto; margin-right: auto;"> <thead> <tr> <th>A</th> <th>B</th> <th>Y</th> </tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </tbody> </table>	A	B	Y	0	0	1	0	1	0	1	0	0	1	1	0	<p>XNOR</p>  <p>$Y = \overline{A \oplus B}$</p> <table border="1" style="border-collapse: collapse; margin-left: auto; margin-right: auto;"> <thead> <tr> <th>A</th> <th>B</th> <th>Y</th> </tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </tbody> </table>	A	B	Y	0	0	1	0	1	0	1	0	0	1	1	1
A	B	Y																																													
0	0	1																																													
0	1	1																																													
1	0	1																																													
1	1	0																																													
A	B	Y																																													
0	0	1																																													
0	1	0																																													
1	0	0																																													
1	1	0																																													
A	B	Y																																													
0	0	1																																													
0	1	0																																													
1	0	0																																													
1	1	1																																													

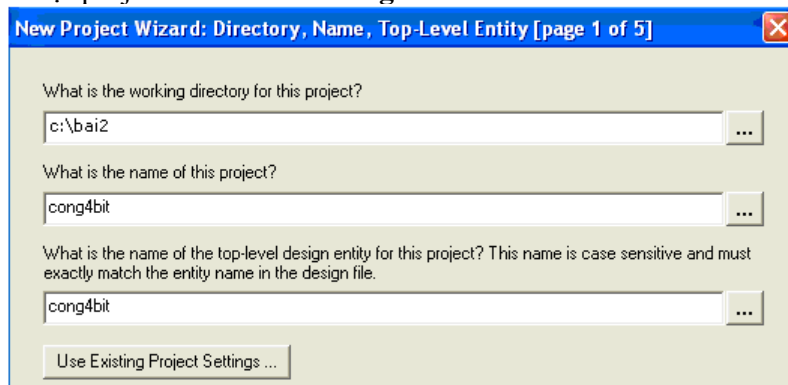
* Tham khảo :
www.altera.com
www.terasic.com

BÀI 2 : THIẾT KẾ MẠCH CỘNG, TRỪ 4 BIT

Hầu hết các thiết kế đều được thực hiện theo mô hình phân cấp. Mô hình phân cấp sử dụng các sub-module kết hợp với nhau trong một top-module để tạo thành thiết kế hoàn chỉnh.

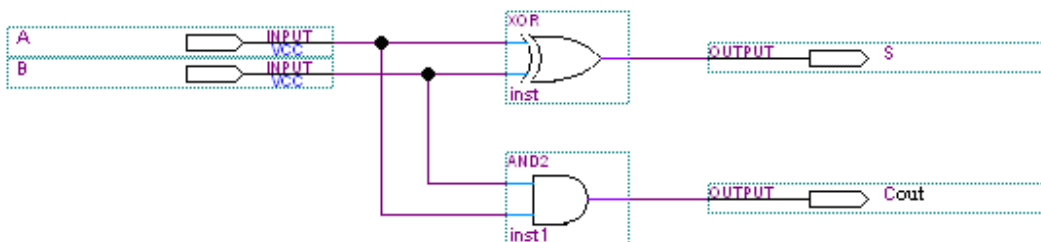
1. Chạy chương trình bằng cách double-click vào biểu tượng  trên desktop.

2. Tạo một project mới có tên : **cong4bit**.



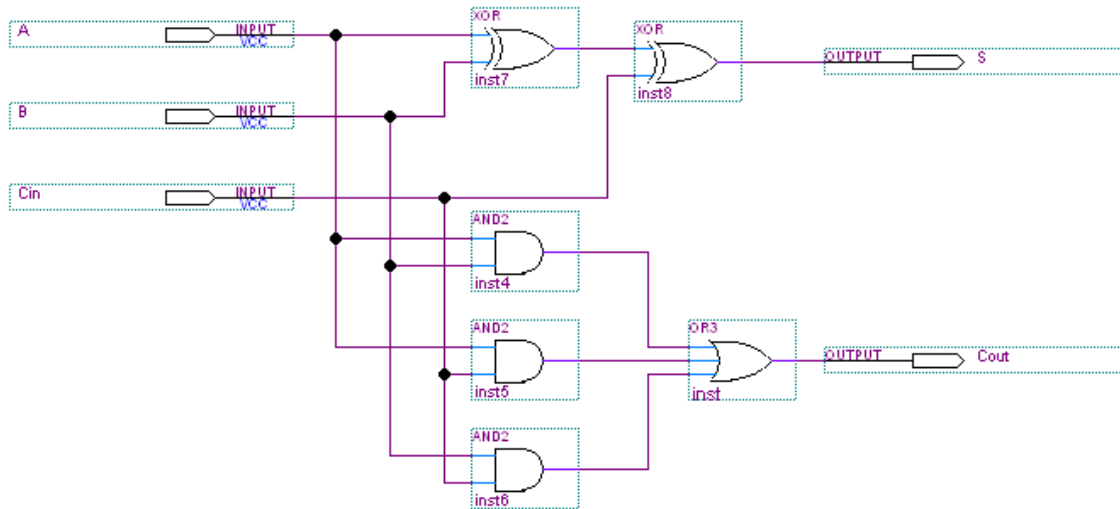
3. Đầu tiên cần tạo mạch cộng 1 bit gồm Half Adder và Full Adder : File → New → Block Diagram/Schematic File.

4. Thực hiện thiết kế mạch cộng 1 bit HA như trong hình :



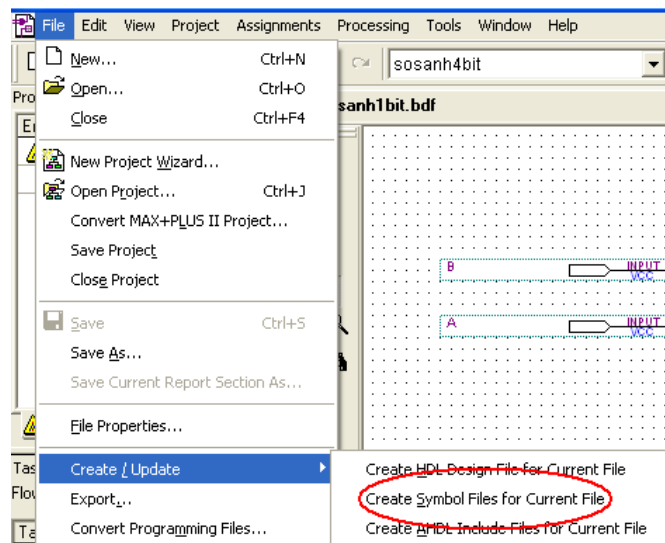
Lưu lại với tên : **HA.bdf**.

5. Tiếp tục, thực hiện thiết kế mạch cộng 1 bit FA.



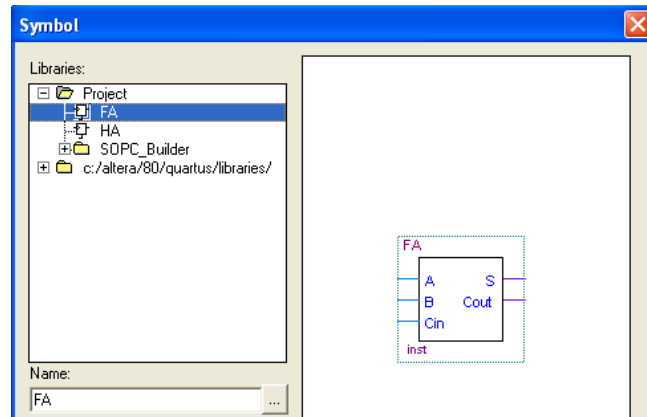
Lưu lại với tên file : **FA.bdf**.

6. Tạo symbol (đóng gói thiết kế) cho file FA.bdf và HA.bdf bằng cách vào File → Create/Update → Create Symbol File for Current File.

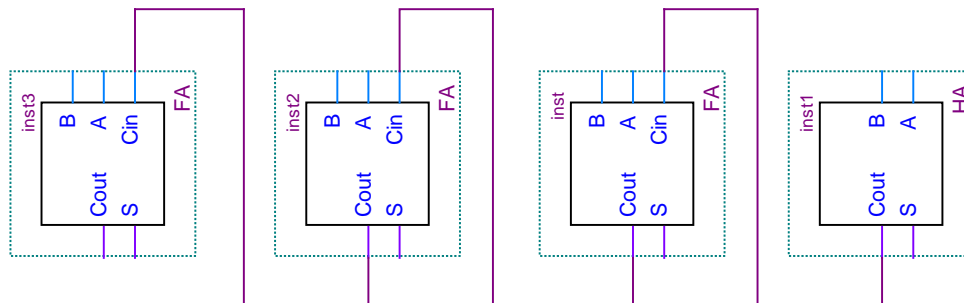


7. Thực hiện thiết kế mạch cộng 4 bit bằng cách ghép 4 module mạch cộng 1 bit lại với nhau. Vào File → New → Block Diagram/Schematic File.

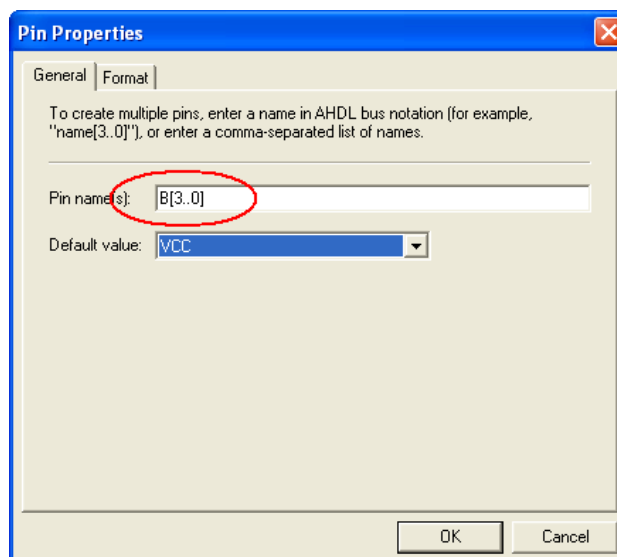
8. Thêm module mạch cộng 1 bit vào : Insert → Symbol → FA (hoặc HA).



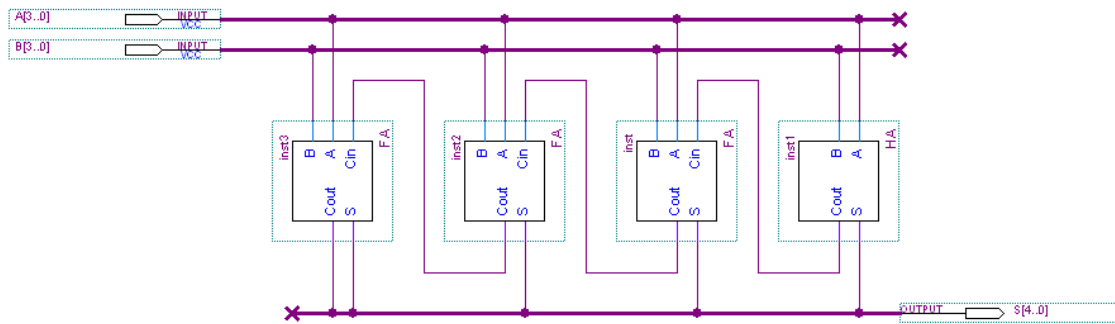
9. Ghép 4 module cộng 1 bit lại để tạo thành mạch cộng 4 bit.



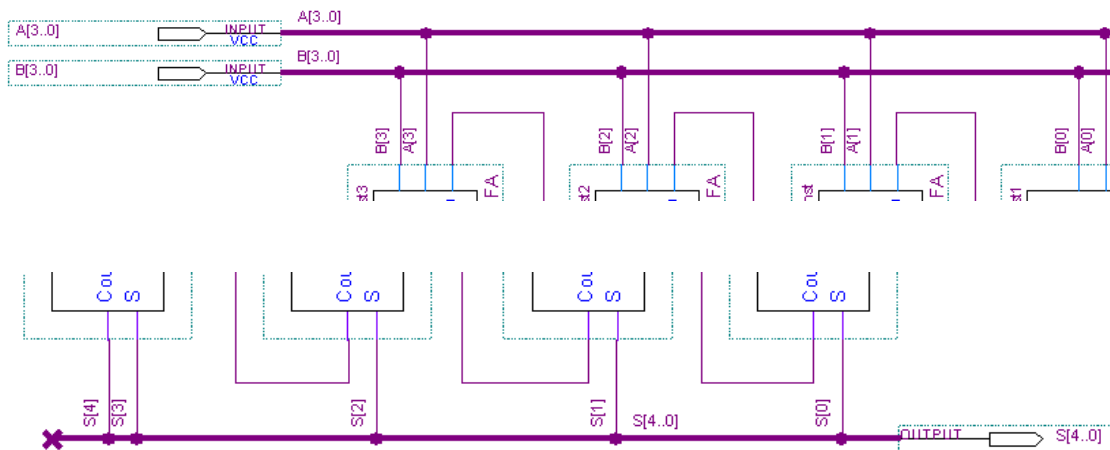
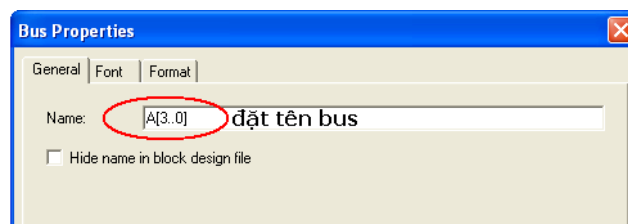
10. Thêm vào các input và output. Các input là A, B dạng bus (4 đường). Các output là S dạng bus (5 đường). Đặt tên cho các input A, B bằng cách double-click vào input, phần "Pin name" gõ vào : **A[3..0]** và **B[3..0]**. Tương tự cho output **S[4..0]**.



11. Vẽ các đường bus () và dây nối () cho mạch.



12. Click chuột phải vào đường bus và dây nối, chọn Properties để đặt tên cho chúng theo hình.

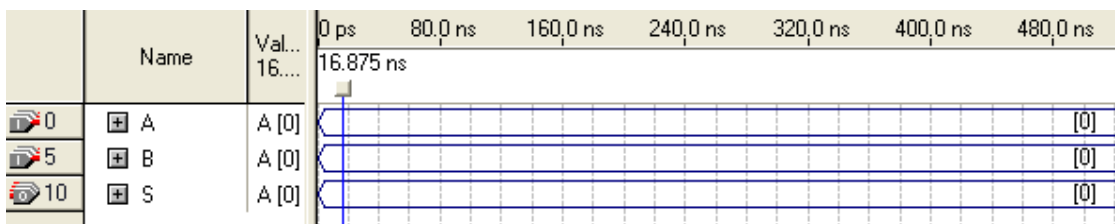


13. Lưu lại với tên : **cong4bit.bdf**.

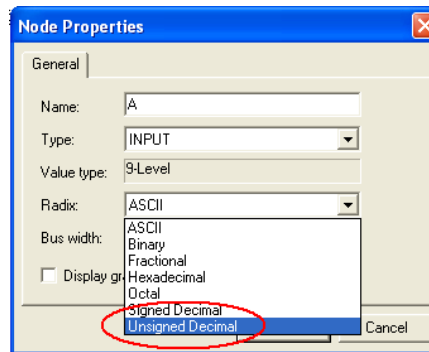
* **Mô phỏng thiết kế**

14. Biên dịch thiết kế chọn Processing → Start Compilation.


15. Tạo ra Vector Waveform File như sau :

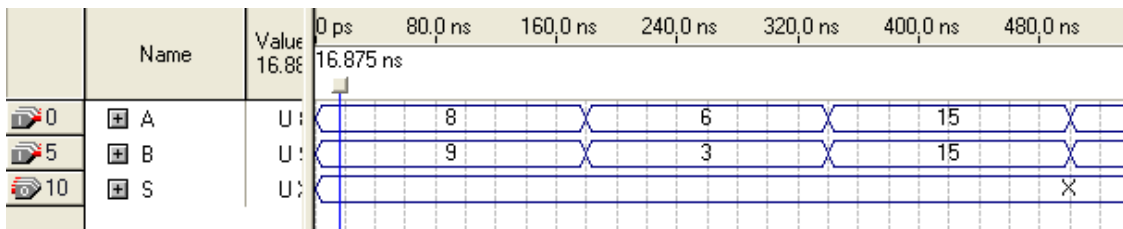


16. Thay đổi hệ cơ số của A, B và S bằng cách click chuột phải vào A, B hoặc S. Chọn Properties. Trong Radix chọn Unsigned Decimal (thập phân không dấu).



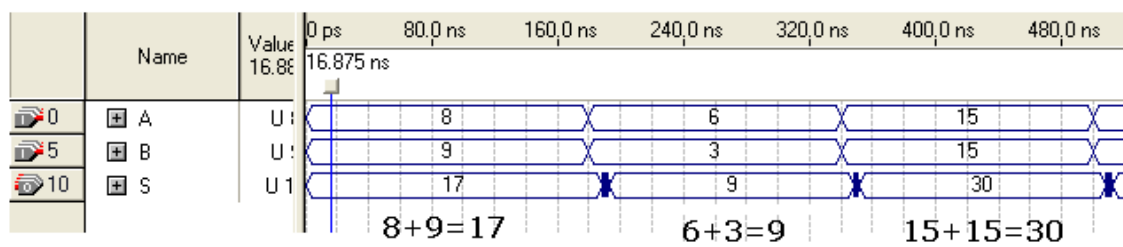
hệ cơ số thập phân
không dấu

17. Vẽ dạng sóng cho A và B bằng công cụ thiết lập giá trị tùy ý .



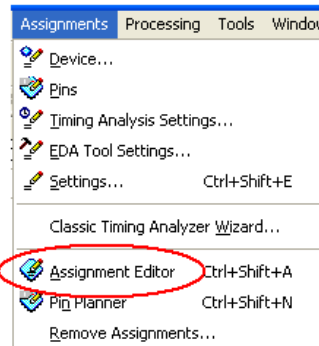
18. Vào Processing → Start Simulation để mô phỏng.

19. Kết quả dạng sóng thu được.



* Cấu hình cho FPGA trên DE2-70

20. Thực hiện map chân cho FPGA : vào Assignments → Assignment Editor



21. Map chân cho 2 ngõ vào A, B với 8 nút gạt và ngõ ra S với 5 led đỏ.

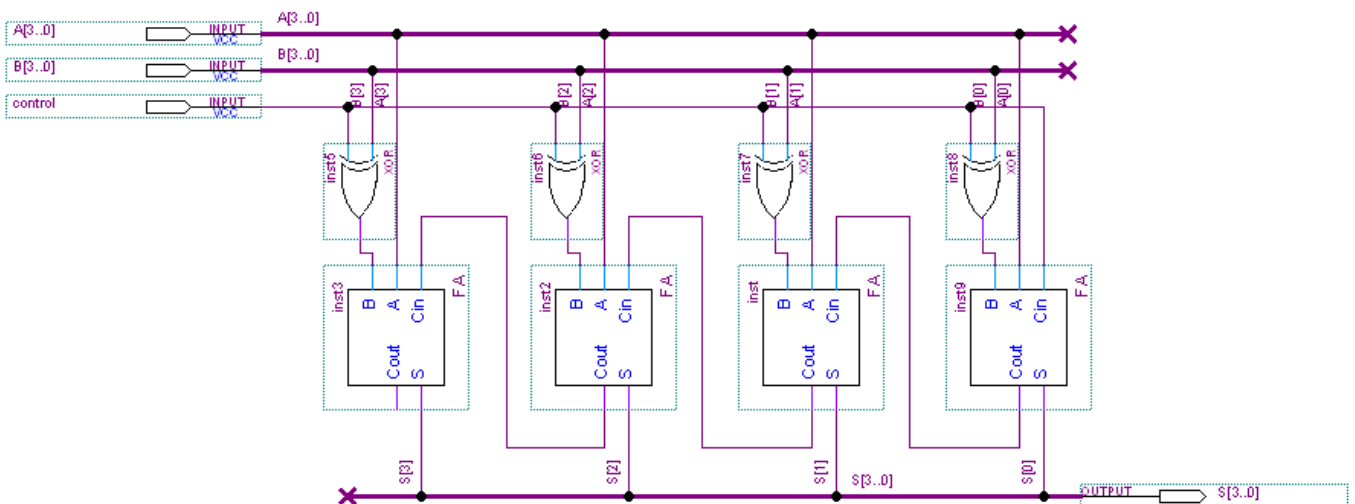
iSW[0]	PIN_AA23
iSW[1]	PIN_AB26
iSW[2]	PIN_AB25
iSW[3]	PIN_AC27
iSW[4]	PIN_AC26
iSW[5]	PIN_AC24
iSW[6]	PIN_AC23
iSW[7]	PIN_AD25

oLEDR[0]	PIN_AJ6
oLEDR[1]	PIN_AK5
oLEDR[2]	PIN_AJ5
oLEDR[3]	PIN_AJ4
oLEDR[4]	PIN_AK3
oLEDR[5]	PIN_AH4
oLEDR[6]	PIN_AJ3
oLEDR[7]	PIN_AJ2

22. Sau khi map chân xong, Save và Compile lại một lần nữa. Để cấu hình cho FPGA: chọn Tools → Programmer.

23. Bấm Start. Sau khi chạy 100%, FPGA đã được cấu hình xong. Kiểm tra lại hoạt động của thiết kế trên kit DE2-70.

Bài tập : Thiết kế mạch cộng/trừ 4 bit và cấu hình trên DE2-70.




* Tham khảo :
www.altera.com
www.terasic.com

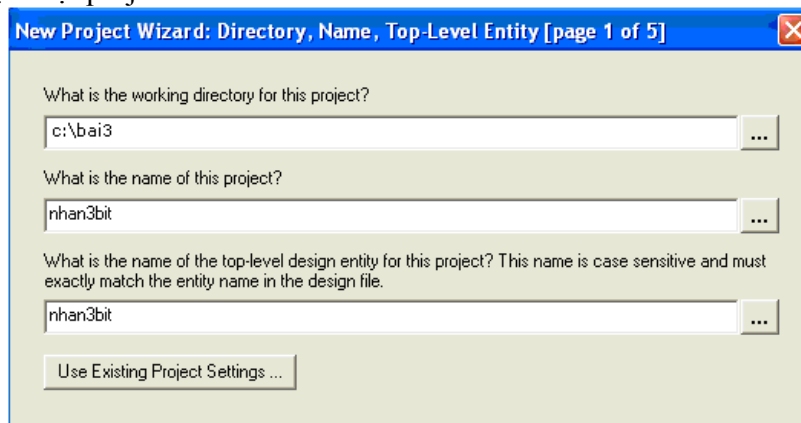
BÀI 3 : THIẾT KẾ MẠCH NHÂN

Cách thực hiện phép nhân 3 bit cho 2 số A và B, kết quả là S :

$$\begin{array}{r}
 \times \quad A_2 \quad A_1 \quad A_0 \\
 \quad B_2 \quad B_1 \quad B_0 \\
 \hline
 A_2B_0 \quad A_1B_0 \quad A_0B_0 \\
 A_2B_1 \quad A_1B_1 \quad A_0B_1 \\
 A_2B_2 \quad A_1B_2 \quad A_0B_2 \\
 \hline
 S_5 \quad S_4 \quad S_3 \quad S_2 \quad S_1 \quad S_0
 \end{array}$$

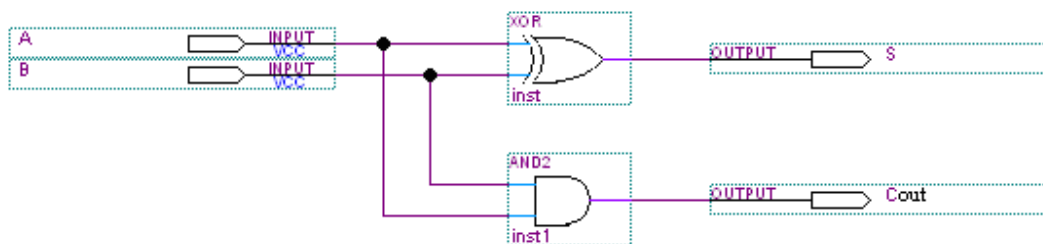
Thiết kế mạch nhân Baugh Wooley 3 bit như sau :

1. Chạy chương trình bằng cách double-click vào biểu tượng  trên desktop.
2. Tạo một project mới có tên : **nhan3bit**.



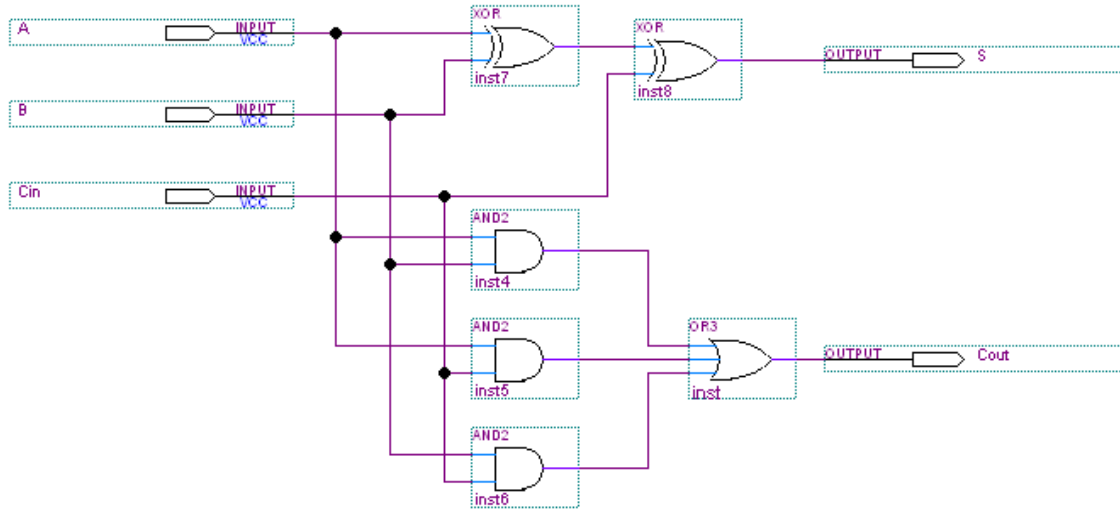
3. Đầu tiên cần tạo mạch cộng 1 bit gồm Half Adder và Full Adder : File → New → Block Diagram/Schematic File.

4. Thực hiện thiết kế mạch cộng 1 bit HA như trong hình :



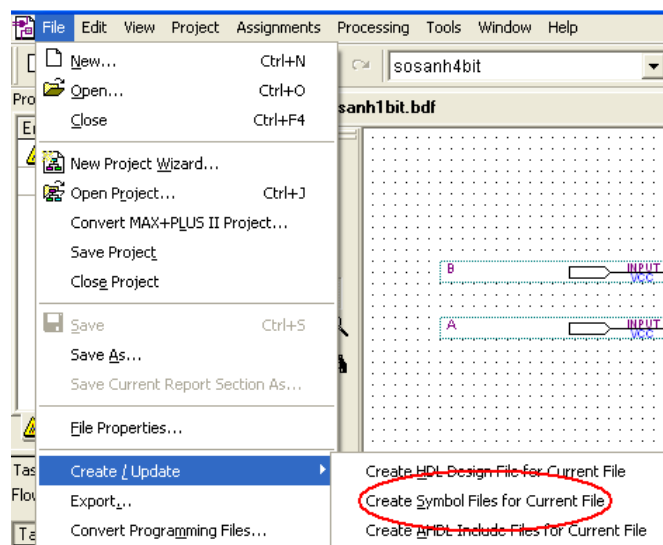
Lưu lại với tên : **HA.bdf**

5. Tiếp tục, thực hiện thiết kế mạch cộng 1 bit FA.



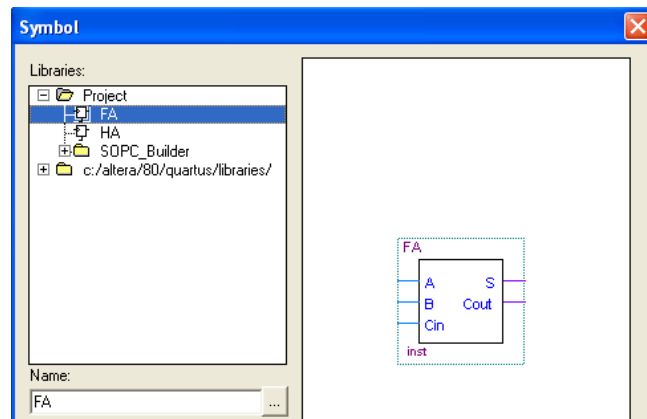
Lưu lại với tên file : **FA.bdf**

6. Tạo symbol (đóng gói thiết kế) cho file FA.bdf và HA.bdf bằng cách vào File → Create/Update → Create Symbol File for Current File.

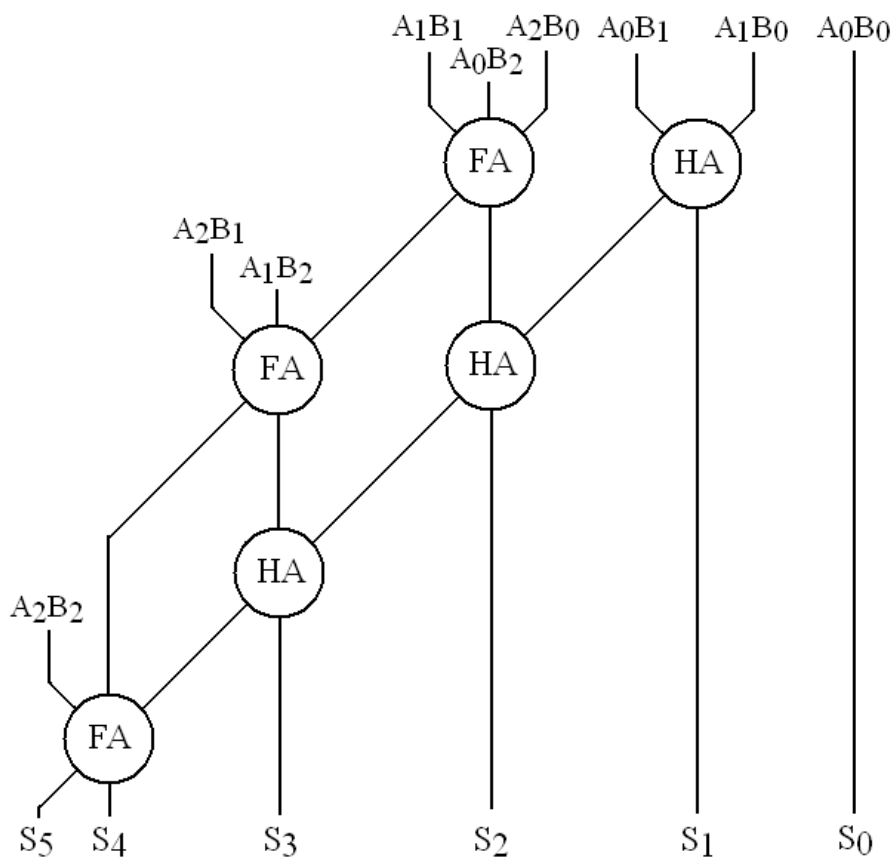


7. Thực hiện thiết kế mạch nhân 3 bit bằng cách ghép các module mạch cộng 1 bit cùng với cổng AND lại với nhau. Vào File → New → Block Diagram/Schematic File.

8. Thêm module mạch cộng 1 bit vào : Insert → Symbol → FA (hoặc HA).



9. Ghép 3 module FA và 3 module HA lại để tạo thành mạch nhân 3 bit.



10. Thêm vào các input và output. Các input là A, B dạng bus (3 đường). Các output là S dạng bus (6 đường). Đặt tên cho các input A, B bằng cách double-click vào input, phần “Pin name” gõ vào : **A[2..0]** và **B[2..0]**. Tương tự cho output **S[5..0]**.

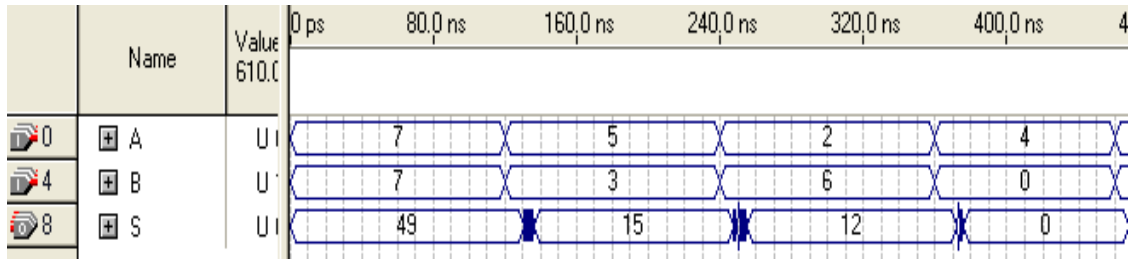
11. Vẽ các đường bus () và dây nối () cho mạch.

12. Click chuột phải vào đường bus và dây nối, chọn Properties để đặt tên cho chúng.

13. Lưu lại với tên : **nhan3bit.bdf**

*** Mô phỏng thiết kế**

14. Kết quả dạng sóng thu được.



*** Cấu hình cho FPGA trên DE2-70**

15. Map chân cho 2 ngõ vào A, B với 6 nút gạt và ngõ ra S với 6 led đỏ.

iSW[0]	PIN_AA23
iSW[1]	PIN_AB26
iSW[2]	PIN_AB25
iSW[3]	PIN_AC27
iSW[4]	PIN_AC26
iSW[5]	PIN_AC24
iSW[6]	PIN_AC23
iSW[7]	PIN_AD25

oLEDR[0]	PIN_AJ6
oLEDR[1]	PIN_AK5
oLEDR[2]	PIN_AJ5
oLEDR[3]	PIN_AJ4
oLEDR[4]	PIN_AK3
oLEDR[5]	PIN_AH4
oLEDR[6]	PIN_AJ3
oLEDR[7]	PIN_AJ2

16. Sau khi FPGA đã được cấu hình xong. Kiểm tra lại hoạt động của thiết kế trên kit DE2-70.

Bài tập : Thiết kế mạch bình phương 3 bit và cấu hình trên DE2-70.


* Tham khảo :
www.altera.com
www.terasic.com

BÀI 4 : THIẾT KẾ MẠCH SO SÁNH 4 BIT

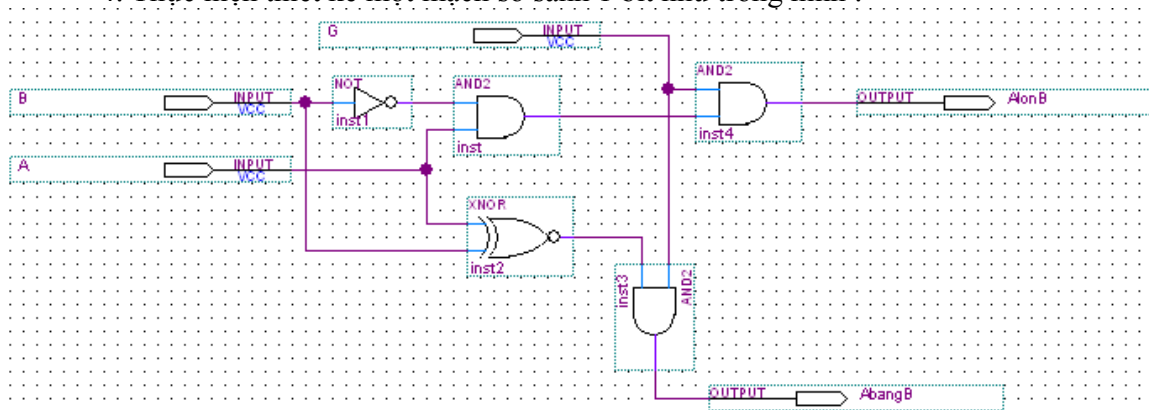
Mạch so sánh 2 số 4 bit được thực hiện theo biểu thức logic sau :

$$(A=B) \Leftrightarrow (A_3=B_3) (A_2=B_2) (A_1=B_1) (A_0=B_0)$$

$$(A>B) \Leftrightarrow (A_3>B_3) + (A_3=B_3) (A_2>B_2) + (A_3=B_3) (A_2=B_2) (A_1>B_1) + (A_3=B_3) (A_2=B_2) (A_1=B_1) (A_0>B_0)$$

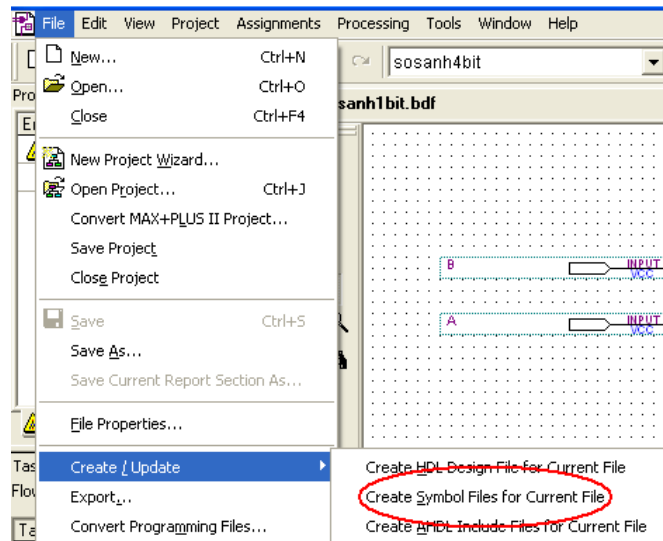
1. Chạy chương trình bằng cách double-click vào biểu tượng  trên desktop.
2. Tạo một project mới có tên : **sosanh4bit**.
3. Đầu tiên cần tạo mạch so sánh 1 bit : File → New → Block Diagram/Schematic File.

4. Thực hiện thiết kế một mạch so sánh 1 bit như trong hình :



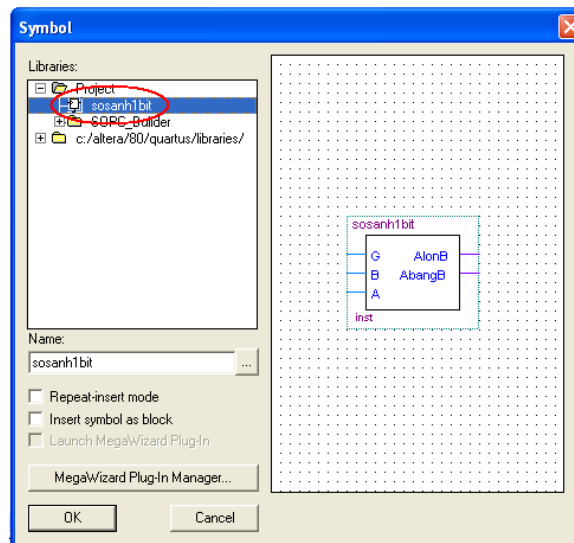
Các input là : **A, B, G**; output là : **AlonB, AbangB**.

5. Lưu lại với tên file : **sosanh1bit.bdf**
6. Tạo symbol (đóng gói thiết kế) cho file sosanh1bit.bdf bằng cách vào File → Create/Update → Create Symbol File for Current File.

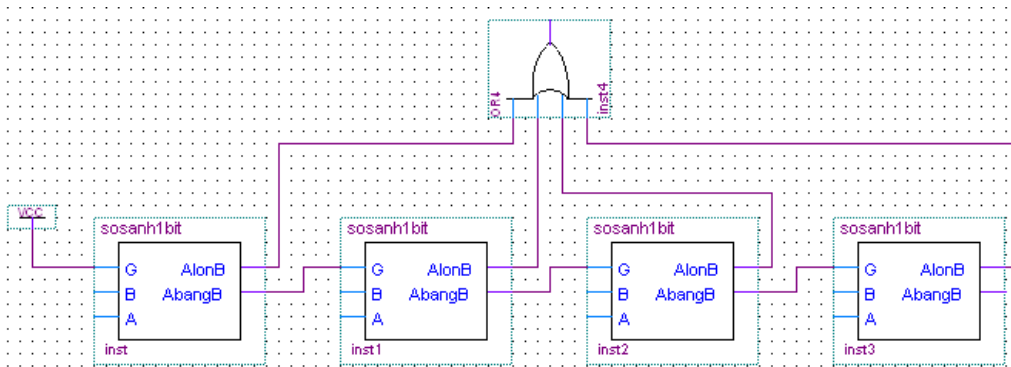


7. Thực hiện thiết kế mạch so sánh 4 bit bằng cách ghép 4 module mạch so sánh 1 bit lại với nhau. Vào File → New → Block Diagram/Schematic File.

8. Thêm module mạch so sánh 1 bit vào : Insert → Symbol → sosanh1bit.



9. Ghép 4 module so sánh 1 bit lại để tạo thành mạch so sánh 4 bit.



10. Thêm vào các input và output. Các input là A, B dạng bus (4 đường). Các output là ABangB và AlonB. Đặt tên cho các input A, B bằng cách double-click vào input, phần “Pin name” gõ vào : **A[3..0]** và **B[3..0]**.

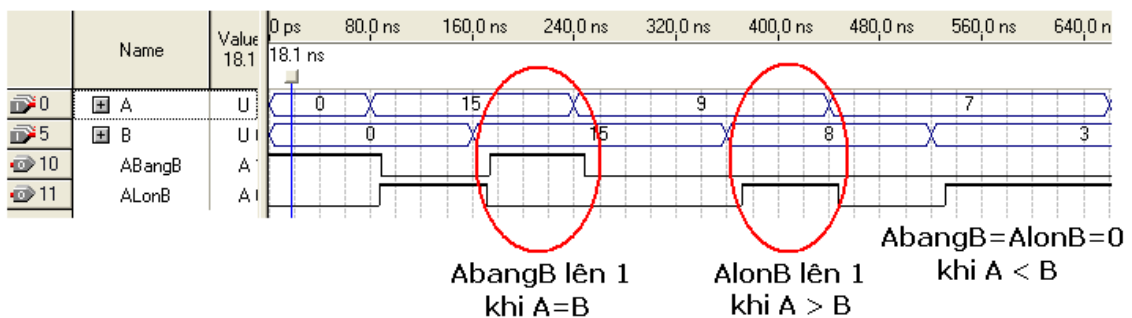
11. Vẽ các đường bus () và dây nối () cho mạch.

12. Click chuột phải vào đường bus và dây nối, chọn Properties để đặt tên cho chúng.

13. Lưu lại với tên : **sosanh4bit.bdf**

*** Mô phỏng thiết kế**

14. Kết quả dạng sóng thu được.



*** Cấu hình cho FPGA trên DE2-70**

15. Map chân cho 2 ngõ vào A, B với 8 nút gạt và 2 ngõ ra AbangB, AlonB với 2 led đỏ.

iSW[0]	PIN_AA23
iSW[1]	PIN_AB26
iSW[2]	PIN_AB25
iSW[3]	PIN_AC27
iSW[4]	PIN_AC26

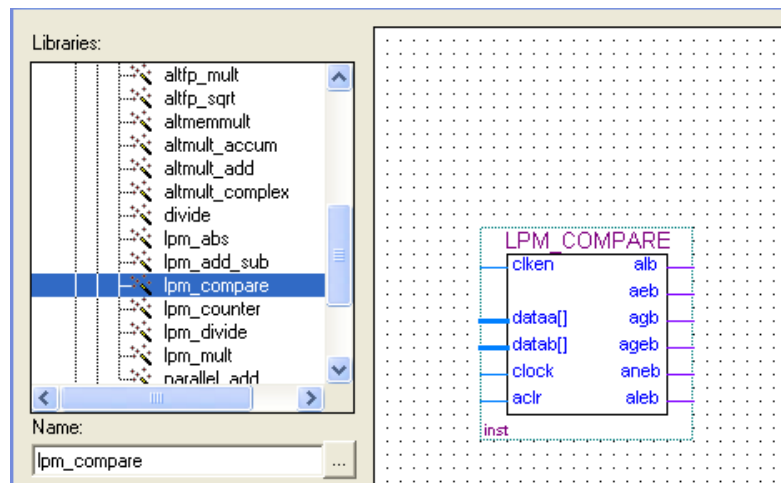
oLEDR[0]	PIN_AJ6
oLEDR[1]	PIN_AK5
oLEDR[2]	PIN_AJ5
oLEDR[3]	PIN_AJ4
oLEDR[4]	PIN_AK3

iSW[5]	PIN_AC24
iSW[6]	PIN_AC23
iSW[7]	PIN_AD25

oLEDR[5]	PIN_AH4
oLEDR[6]	PIN_AJ3
oLEDR[7]	PIN_AJ2

16. Sau khi FPGA đã được cấu hình xong. Kiểm tra lại hoạt động của thiết kế trên kit DE2-70.

Bài tập : Khảo sát mạch so sánh trong thư viện của QuartusII (vào Megafunction → arithmetic → lpm_compare)



* Tham khảo :

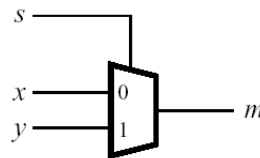
www.altera.com

www.terasic.com


BÀI 5 : THIẾT KẾ MẠCH ĐA HỢP

Biểu thức logic cho bộ đa hợp 2-1 1 bit :

$$m = x\bar{s} + ys$$



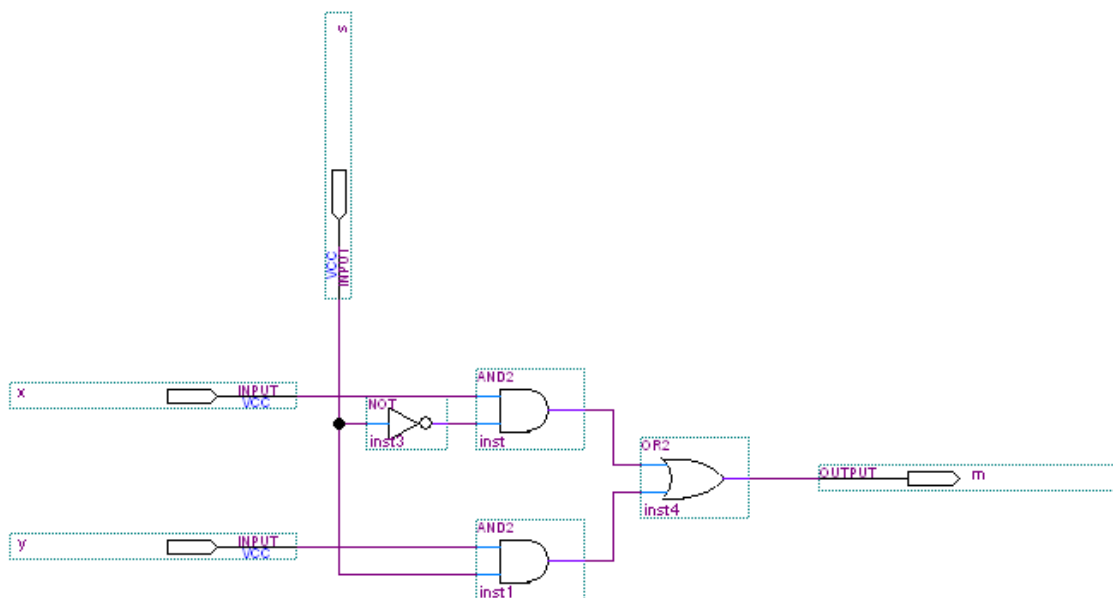
Đầu tiên, ta sẽ thiết kế một bộ đa hợp 2-1 8 bit

1. Chạy chương trình bằng cách double-click vào biểu tượng  trên desktop.

2. Tạo một project mới có tên : **machdahop**

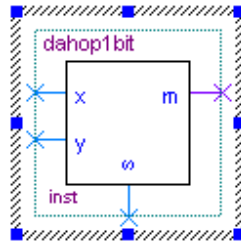
3. Đầu tiên cần tạo mạch đa hợp 2-1 1 bit : File → New → Block Diagram/Schematic File.

4. Thực hiện thiết kế mạch đa hợp 2-1 1 bit như trong hình :

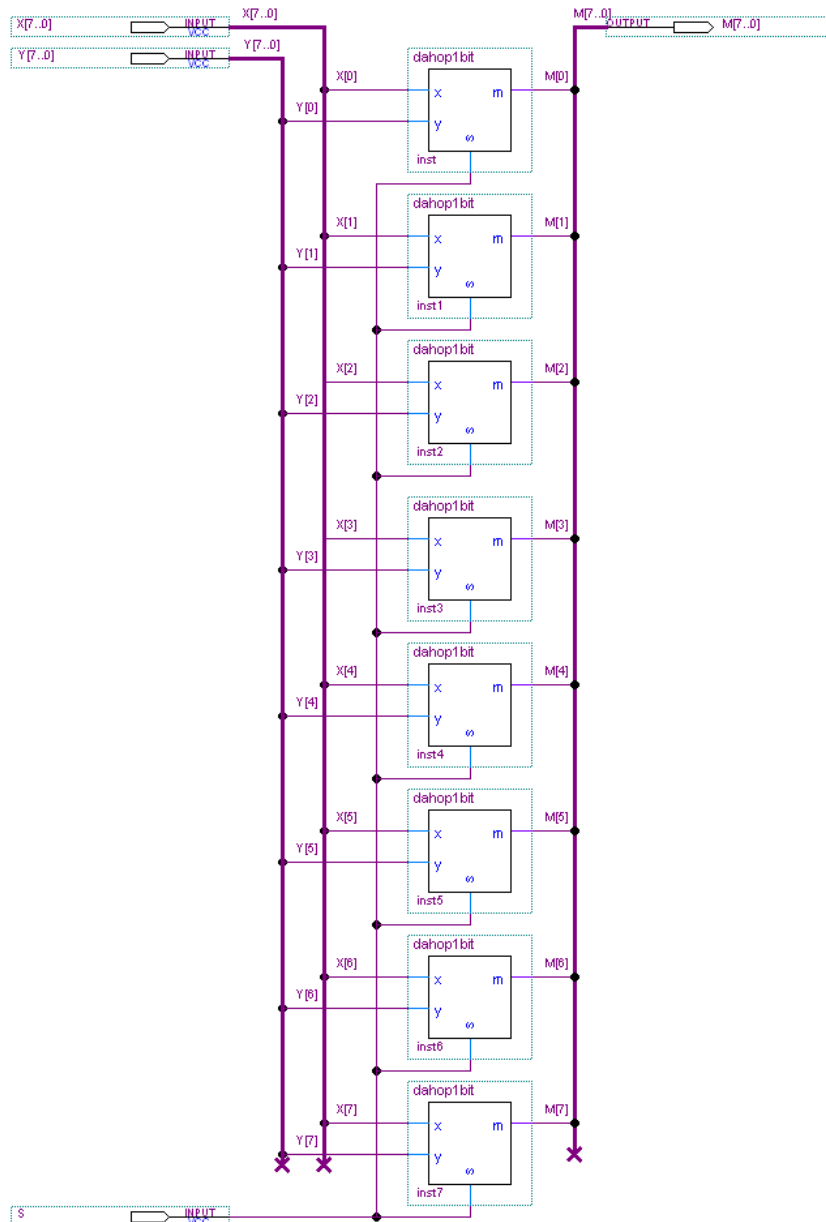


Lưu lại với tên : **dahop1bit.bdf**

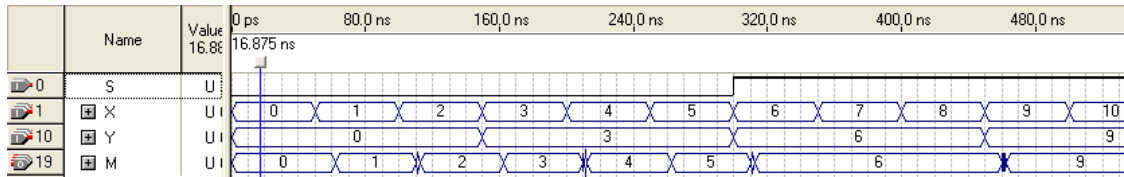
5. Tạo symbol (đóng gói thiết kế) cho file dahop1bit.bdf bằng cách vào File → Create/Update → Create Symbol File for Current File.



6. Thực hiện thiết kế mạch đa hợp 2-1 8 bit bằng cách ghép các module mạch đa hợp 2-1 1 bit lại với nhau.



7. Kết quả dạng sóng thu được :



*** Cấu hình cho FPGA trên DE2-70**

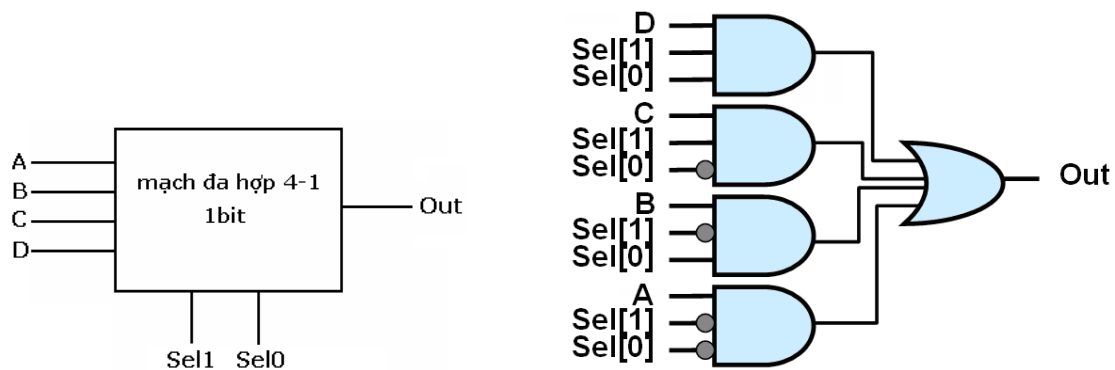
8. Map chân cho 2 ngõ vào X, Y với 16 nút gạt, ngõ vào S với 1 nút gạt, và ngõ ra M với 8 led đồ.

iSW[0]	PIN_AA23
iSW[1]	PIN_AB26
iSW[2]	PIN_AB25
iSW[3]	PIN_AC27
iSW[4]	PIN_AC26
iSW[5]	PIN_AC24
iSW[6]	PIN_AC23
iSW[7]	PIN_AD25
iSW[8]	PIN_AD24
iSW[9]	PIN_AE27
iSW[10]	PIN_W5
iSW[11]	PIN_V10
iSW[12]	PIN_U9
iSW[13]	PIN_T9
iSW[14]	PIN_L5
iSW[15]	PIN_L4

iSW[16]	PIN_L7
iSW[17]	PIN_L8
oLEDR[0]	PIN_AJ6
oLEDR[1]	PIN_AK5
oLEDR[2]	PIN_AJ5
oLEDR[3]	PIN_AJ4
oLEDR[4]	PIN_AK3
oLEDR[5]	PIN_AH4
oLEDR[6]	PIN_AJ3
oLEDR[7]	PIN_AJ2

9. Sau khi FPGA đã được cấu hình xong. Kiểm tra lại hoạt động của thiết kế trên kit DE2-70.

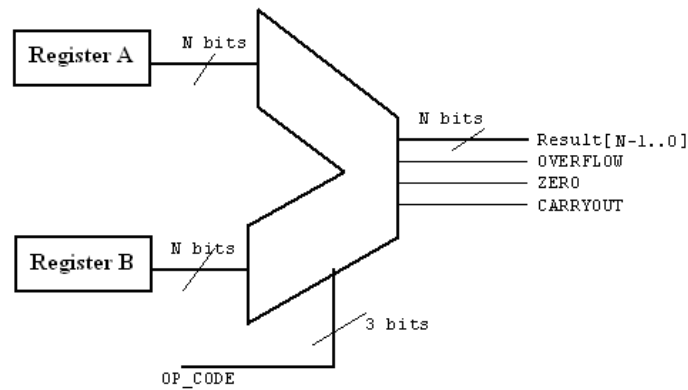
Bài tập : Thiết kế mạch đa hợp 4-1 3 bit



$$m = A.\overline{Sel1}.\overline{Sel0} + B.Sel1.Sel0 + C.Sel1.\overline{Sel0} + D.Sel1.Sel0$$

* Tham khảo :
www.altera.com
www.terasic.com

BÀI 6 : THIẾT KẾ ALU

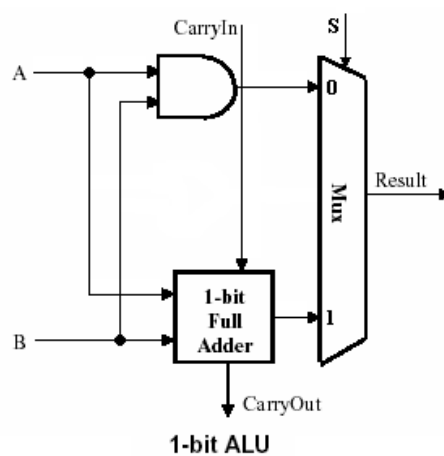


Bộ ALU cơ bản

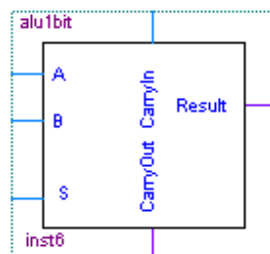
Đầu tiên, ta sẽ thiết kế một bộ ALU gồm 2 chức năng : AND và cộng.

1. Tạo một project mới có tên : **alu8bit**

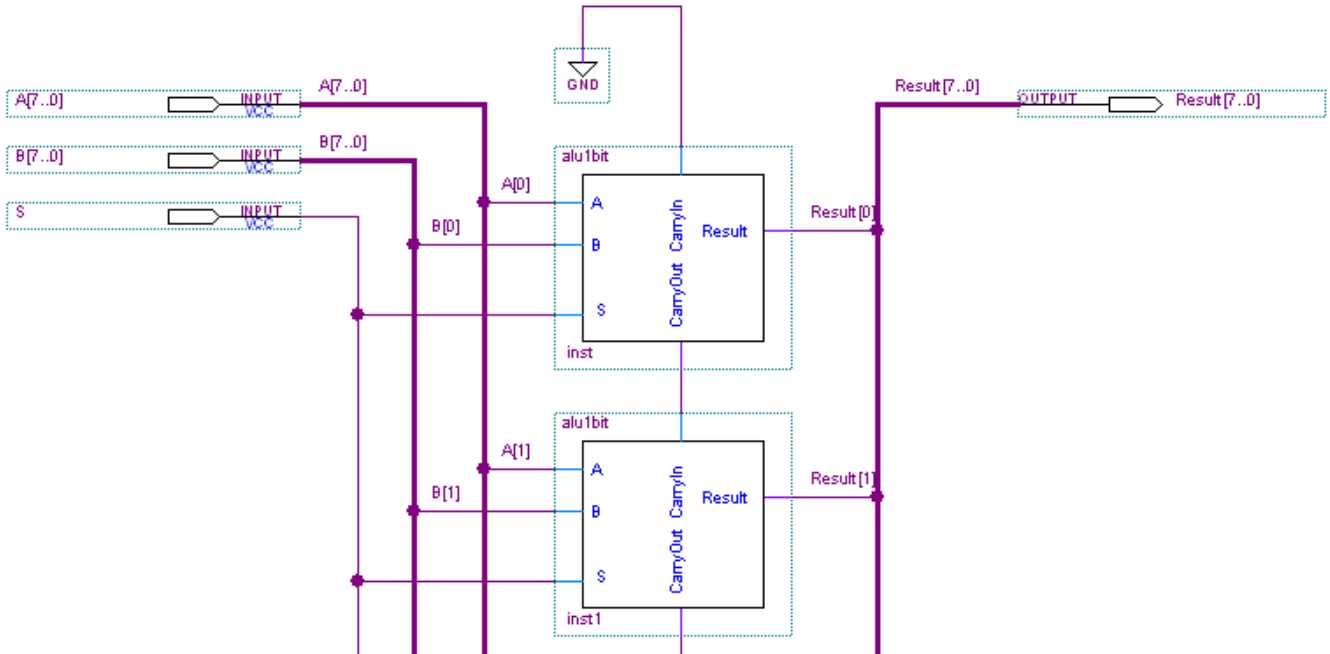
2. Tạo bộ ALU 1 bit như hình dưới (gồm 1 cổng AND, 1 bộ cộng FA, 1 bộ đa hợp 2-1 1 bit). Lưu lại với tên file : **alu1bit.bdf**



1-bit ALU

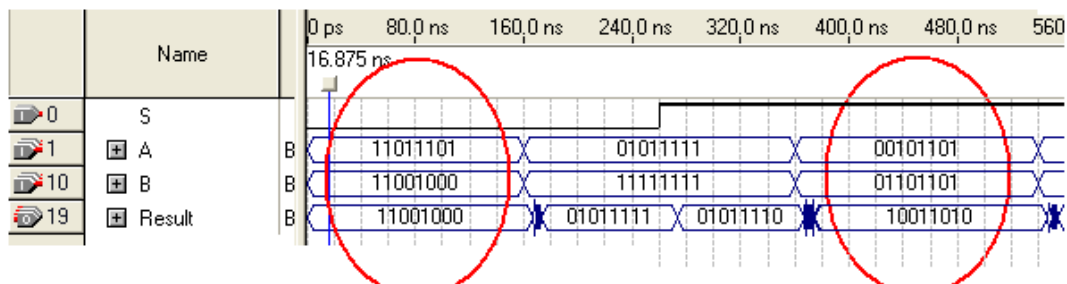


3. Thực hiện thiết kế bộ ALU 8 bit bằng cách ghép 8 bộ ALU 1 bit lại với nhau.



Lưu lại với tên : **alu8bit.bdf**

4. Kết quả mô phỏng :



11011101 and 11001000 = 11001000
S=0 : AND

00101101+01101101=10011010
S=1 : cộng

** Cấu hình cho FPGA trên DE2-70*

5. Map chân cho 2 ngõ vào A, B với 16 nút gạt, ngõ vào S với 1 nút gạt, và ngõ ra Result với 8 led đỏ.

iSW[0]	PIN_AA23
iSW[1]	PIN_AB26
iSW[2]	PIN_AB25
iSW[3]	PIN_AC27
iSW[4]	PIN_AC26
iSW[5]	PIN_AC24
iSW[6]	PIN_AC23
iSW[7]	PIN_AD25
iSW[8]	PIN_AD24
iSW[9]	PIN_AE27
iSW[10]	PIN_W5
iSW[11]	PIN_V10
iSW[12]	PIN_U9
iSW[13]	PIN_T9
iSW[14]	PIN_L5
iSW[15]	PIN_L4

iSW[16]	PIN_L7
iSW[17]	PIN_L8
oLEDR[0]	PIN_AJ6
oLEDR[1]	PIN_AK5
oLEDR[2]	PIN_AJ5
oLEDR[3]	PIN_AJ4
oLEDR[4]	PIN_AK3
oLEDR[5]	PIN_AH4
oLEDR[6]	PIN_AJ3
oLEDR[7]	PIN_AJ2

6. Sau khi FPGA đã được cấu hình xong. Kiểm tra lại hoạt động của thiết kế trên kit DE2-70.

Bài tập : Thiết kế bộ ALU 4 bit gồm 5 chức năng : cộng, trừ, NAND, OR, XOR

* *Tham khảo* :

www.altera.com
www.terasic.com

BÀI 7 : THANH GHI, BỘ ĐẾM

* Flip-flop T

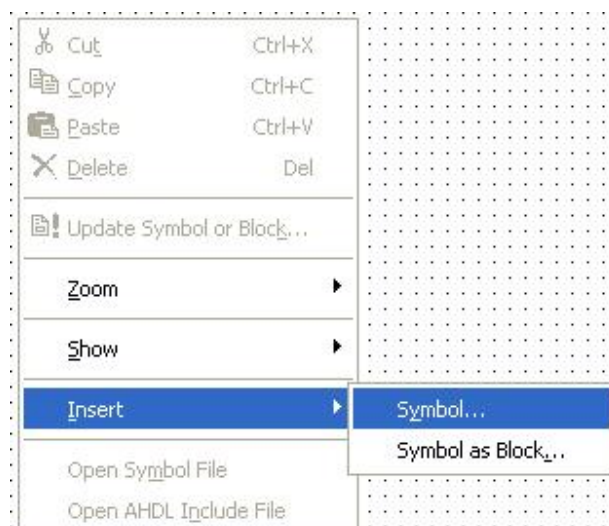
1. Chạy chương trình bằng cách double-click vào biểu tượng QuartusII trên Desktop.

2. Tạo project có tên FlipFlopT.

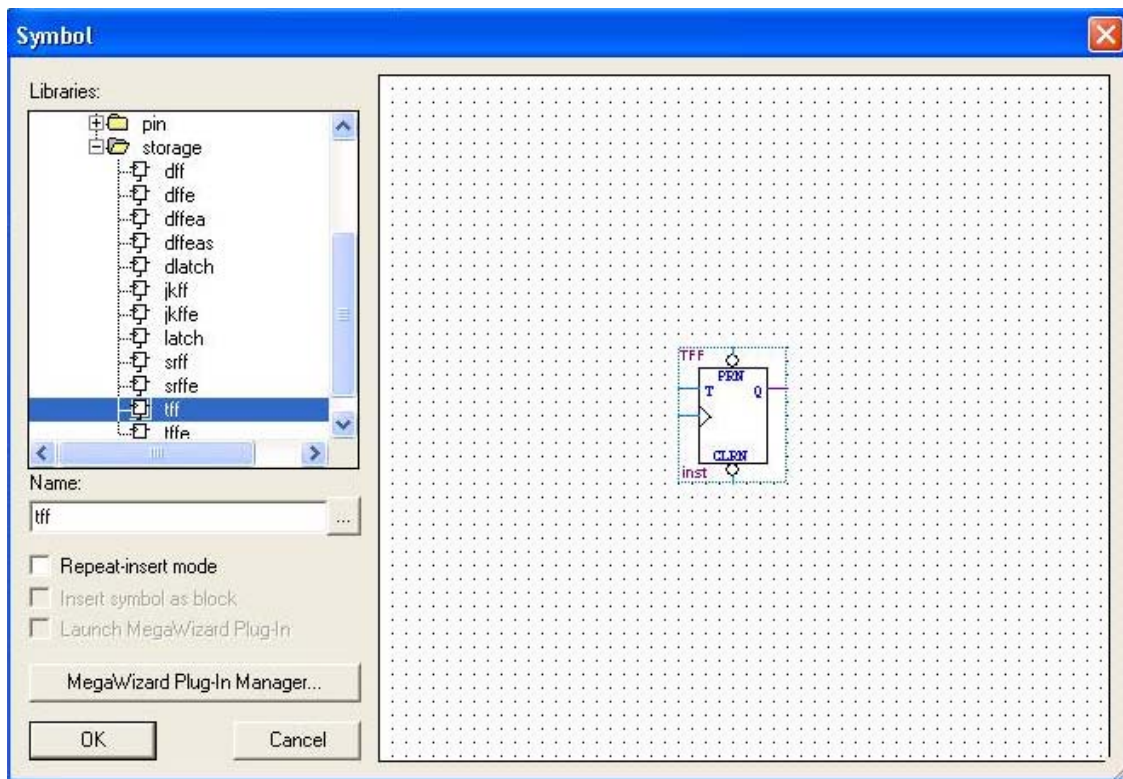


3. Vào File → New → Block Diagram/Schematic File.

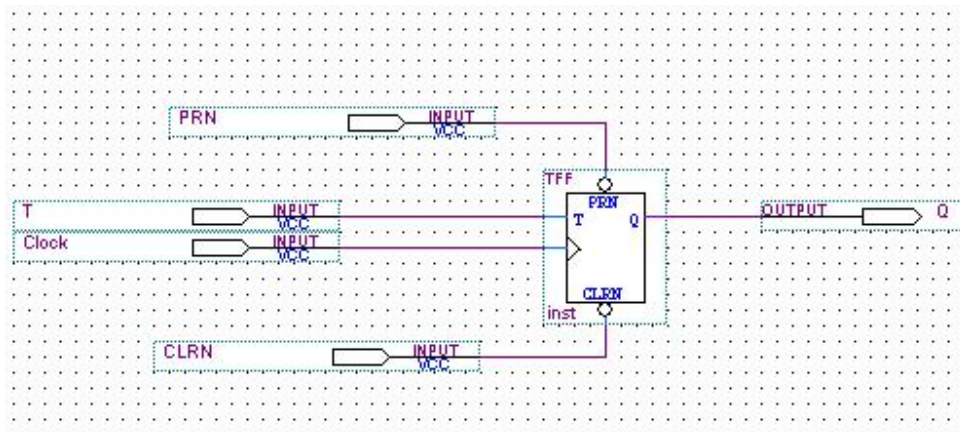
4. Click chuột phải vào trong thiết kế, chọn Insert → Symbol.



5. Gõ tff vào ô name → OK



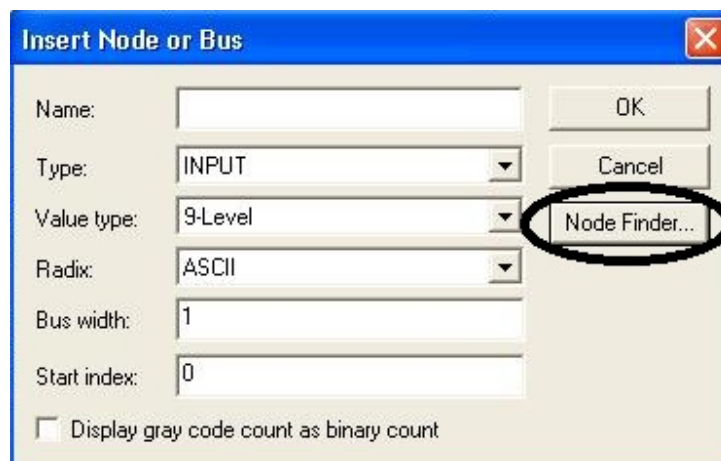
6. Thêm các input, output và đặt tên tín hiệu cho schematic như trong hình.



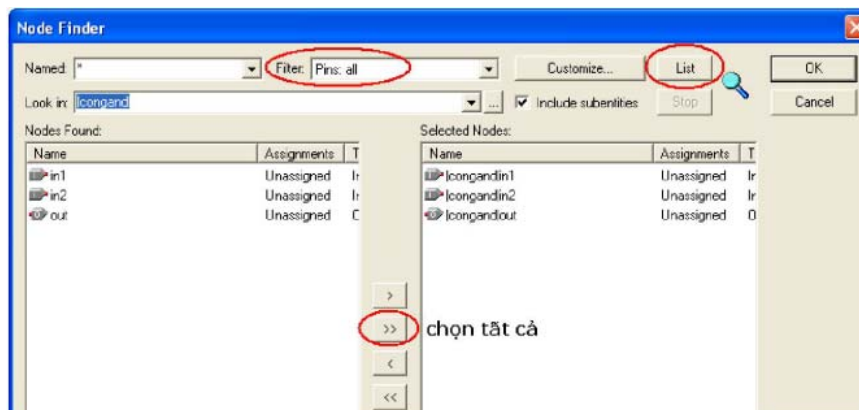
7. Lưu lại với tên **FlipFlopT**

8. Biên dịch thiết kế: Processing → Start Compilation

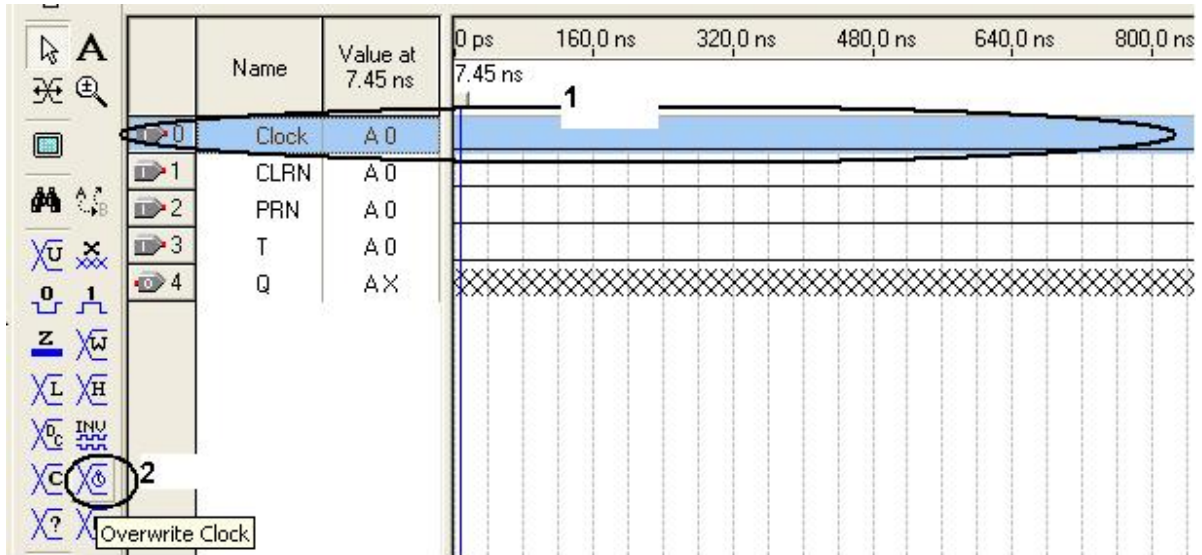
9. Nếu không có lỗi, sẽ xuất hiện cửa sổ báo successful. Bấm OK.
10. Vào File → New → Vector Waveform File.
11. Click chuột phải vào cửa sổ “Name”. Chọn Insert → Insert Node or Bus.
12. Chọn Node Finder.



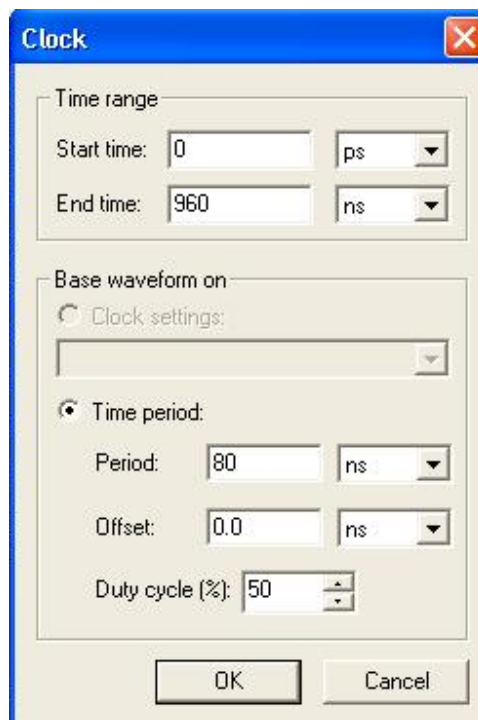
13. Cửa sổ Node Finder chọn “Pins: all” và bấm List. Chọn tất cả các chân. Bấm OK 2 lần.



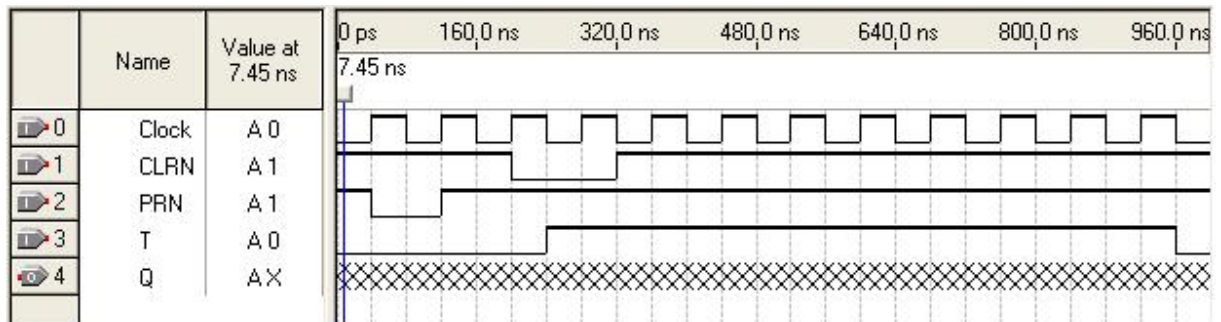
14. Vẽ dạng sóng cho chân Clock: lựa chọn tín hiệu Clock và sử dụng công cụ tạo sóng.



15. Quy định chu kỳ cho xung Clock → OK



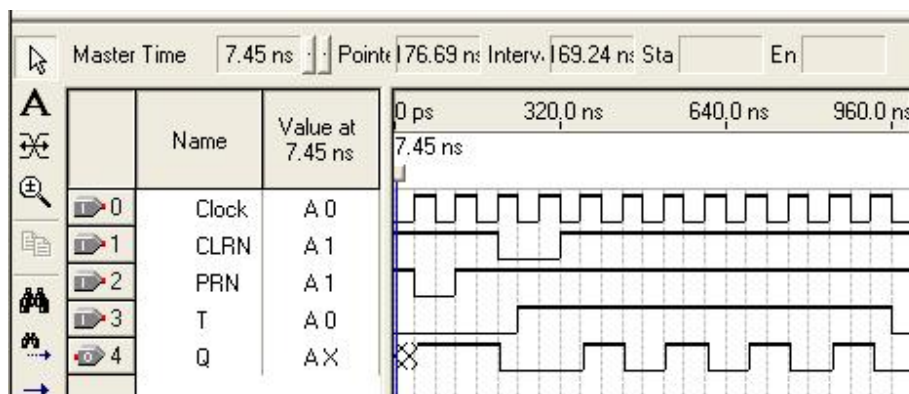
16. Vẽ dạng sóng cho các tín hiệu khác theo hình sau.



17. Lưu file dạng sóng.

18. Mô phỏng (Processing → Start Simulation)

19. Quan sát dạng sóng và nhận xét liên hệ giữa Q và các tín hiệu khác



20. Cho biết mối liên hệ giữa Q và Clock khi các tín hiệu khác ở mức cao.

21. Map chân cho Clock với KEY (nút bấm), CLRN, PRN, T với 3 switch (nút gạt) và ngõ ra Q với đèn led. Biên dịch và cấu hình xuống board DE2-70.

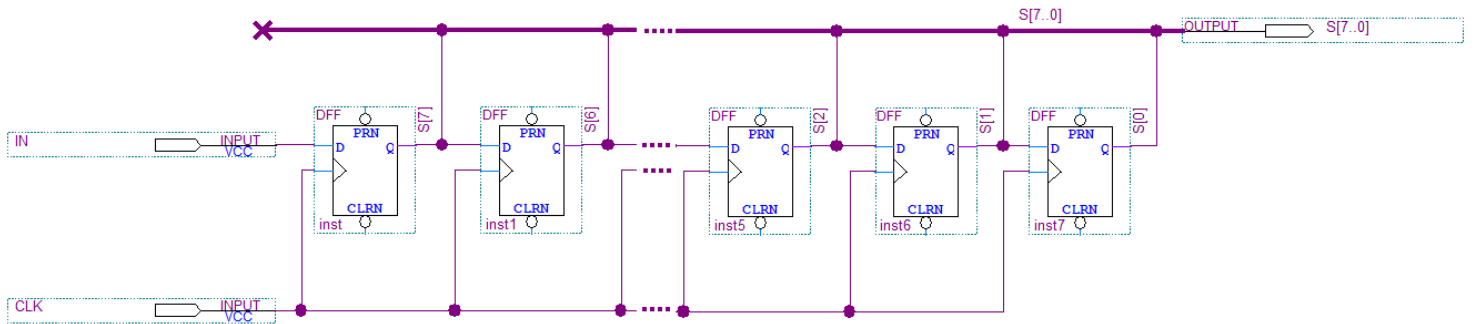
Bài tập : Khảo sát các flip flop khác: flip flop D (tên trong thư viện là dff), chốt (tên trong thư viện là latch).

* Thanh ghi/ Thanh ghi dịch

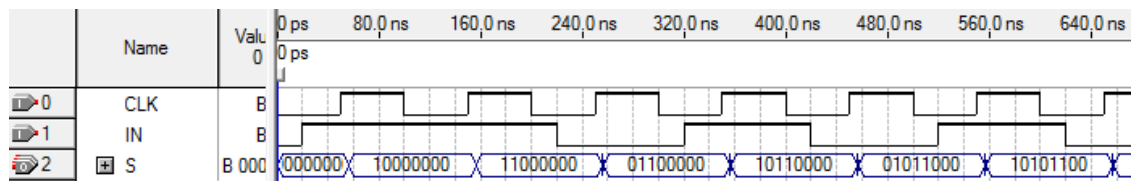
Trong một CPU, ngoài ALU còn có các thanh ghi (register), đơn vị điều khiển (control unit) và bộ nhớ cache.

1. Tạo một project mới có tên : **thanhgidich8bit**

2. Thanh ghi dịch được cấu tạo từ các flip flop D (DFF), do đó để thiết kế một thanh ghi dịch 8 bit ta sẽ dùng 8 flip flop D ghép lại. Lưu lại với tên file : **thanhgidich8bit.bdf**

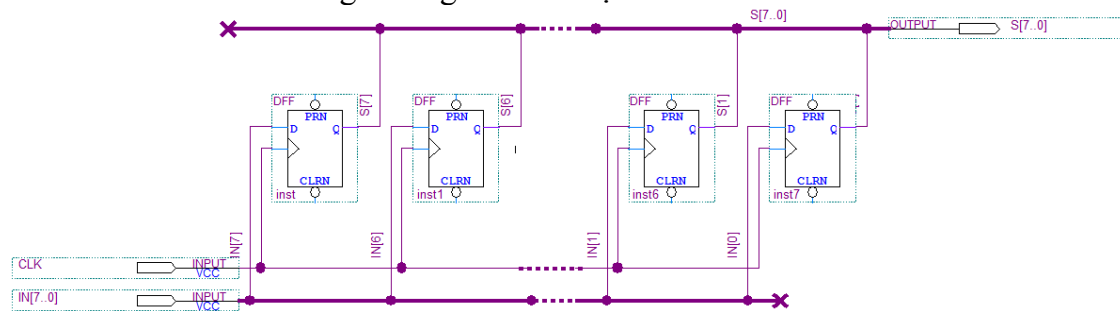


3. Kết quả mô phỏng



4. Map chân cho CLK là nút bấm (KEY), IN là nút gạt (SWITCH), S là 8 đèn led. Biên dịch và cấu hình xuống board DE2-70.

5. Thiết kế thanh ghi bằng cách sửa lại thiết kế

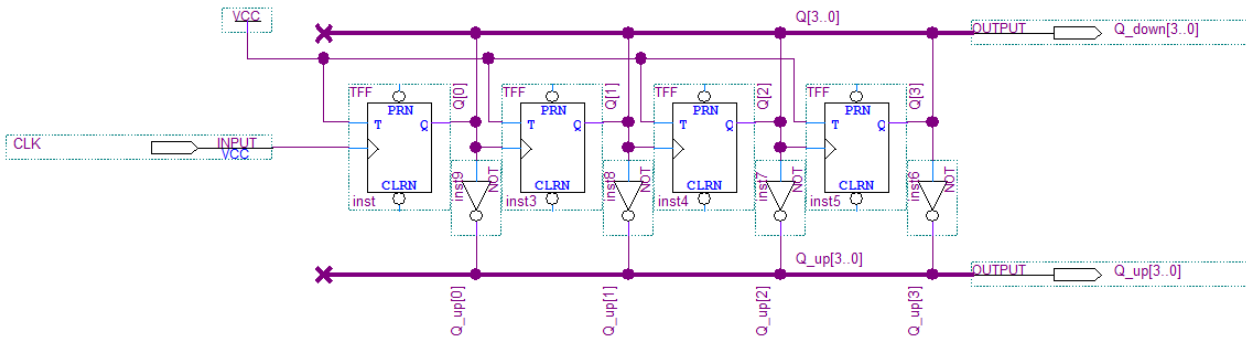


6. Mô phỏng, map chân và cấu hình xuống board DE2-70.

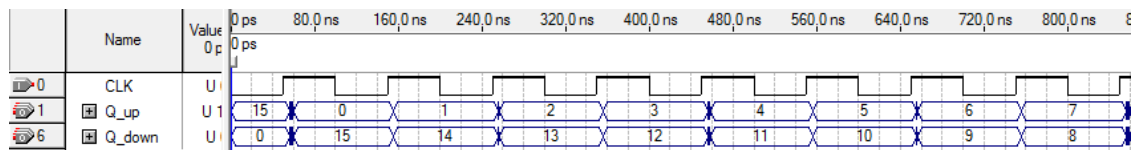
*** Bộ đếm**

1. Tạo một project mới có tên : **dem4bit**

2. Bộ đếm được cấu tạo từ các flip flop T (**TFF**), do đó để thiết kế một thanh ghi 4 bit ta sẽ dùng 4 flip flop T ghép lại. Lưu lại với tên file : **dem4bit.bdf**



3. Mô phỏng, map chân và cấu hình xuống board DE2-70.



* *Tham khảo* :
www.altera.com
www.terasic.com

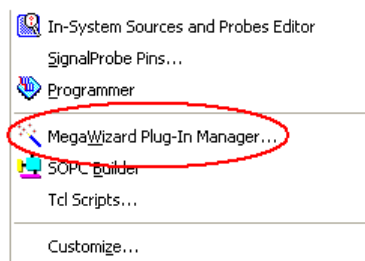
BÀI 8 : BỘ NHỚ

Trong một hệ thống máy tính, bộ nhớ là thành phần đóng vai trò rất quan trọng và không thể thiếu. Bộ nhớ được sử dụng để chứa mã lệnh (vùng nhớ lệnh) và dữ liệu (vùng nhớ dữ liệu) nhằm phục vụ cho CPU trong quá trình xử lý.

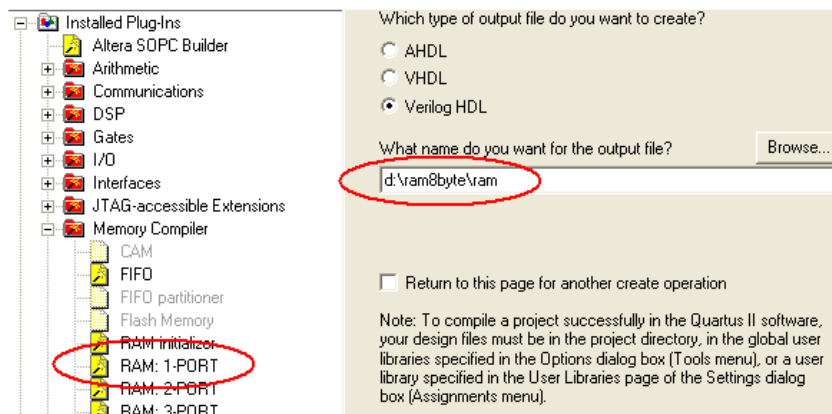
Có 2 loại bộ nhớ chính : ROM và RAM (đều được hỗ trợ trong thư viện của Quartus II)

* RAM

1. Tạo một project mới có tên : **ram8byte**
2. Chọn New → Block Diagram. Vào Tools → MegaWizard Plug-In Manager. Click Next.



3. Trong Memory Compiler, chọn RAM 1-PORT. Đặt tên file : ram. Bấm Next.



Do bộ nhớ của RAM là 8 byte nên độ rộng bus dữ liệu là 8 bit, bus địa chỉ là 3 bit. Lần lượt thiết lập các tùy chọn theo các hình sau.

How wide should the 'q' output bus be? bits

How many 8-bit words of memory? words

What should the memory block type be?

Auto M512 M4K

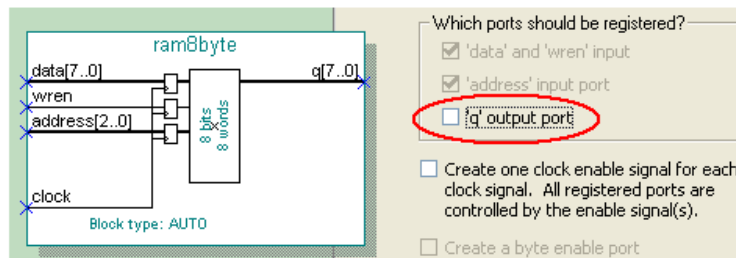
M-RAM LCs

Set the maximum block depth to words

What clocking method would you like to use?

Single clock

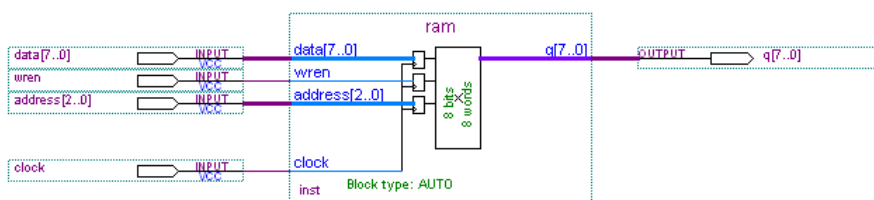
Dual clock: use separate 'input' and 'output' clocks



File	Description
<input checked="" type="checkbox"/> ram.v	Variation file
<input type="checkbox"/> ram.inc	AHDL Include file
<input type="checkbox"/> ram.cmp	VHDL component declaration file
<input checked="" type="checkbox"/> ram.bsf	Quartus II symbol file
<input type="checkbox"/> ram_inst.v	Instantiation template file
<input type="checkbox"/> ram_bb.v	Verilog HDL black-box file
<input type="checkbox"/> ram_waveforms.html	Sample waveforms in summary
... ram_wave*.jpg	Sample waveform file(s)

Bấm Finish.

4. Lấy bộ nhớ RAM vừa tạo ra và gắn các input và output vào. Lưu lại với tên file : **ram8byte.bdf**



5. Mô phỏng

