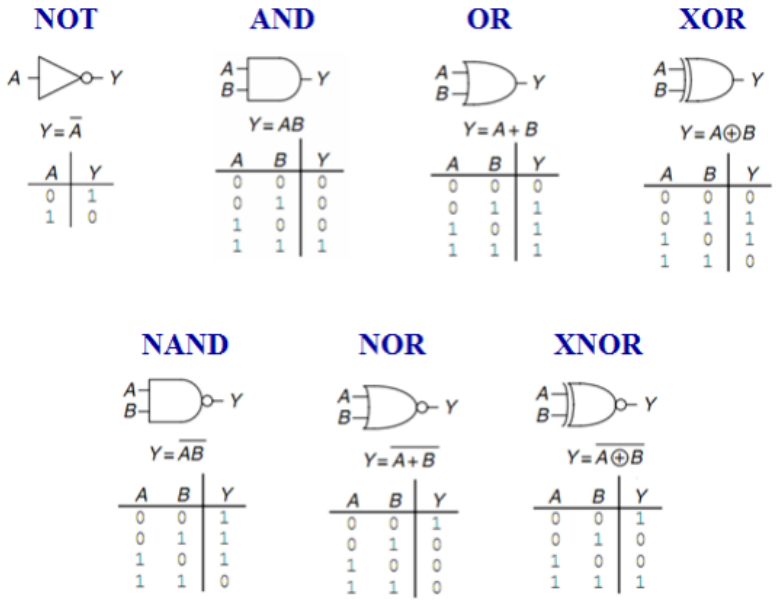
PHẦN 1 - CÁC MẠCH SỐ CƠ BẢN

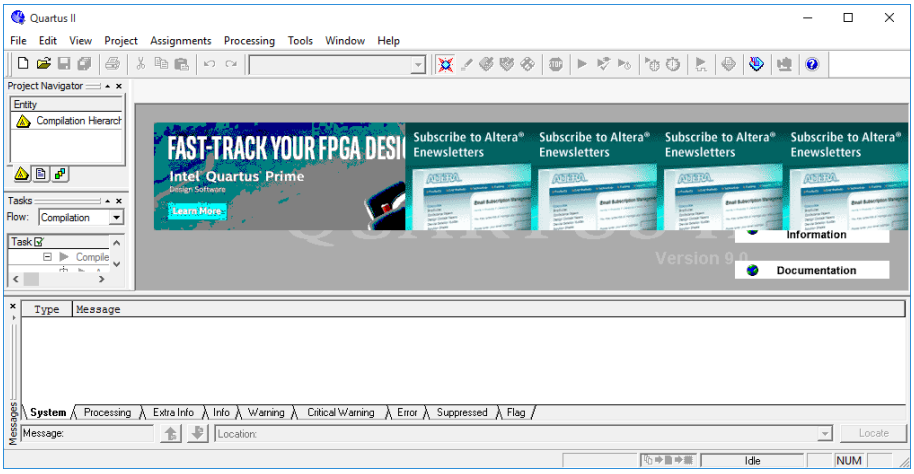
CỔNG LOGIC

Toàn bộ hệ thống số đều dựa trên 7 cổng logic cơ bản: **NOT, AND, OR, XOR, NAND, NOR, XNOR**. Dưới đây là ký hiệu và bảng chân trị của 7 cổng này:

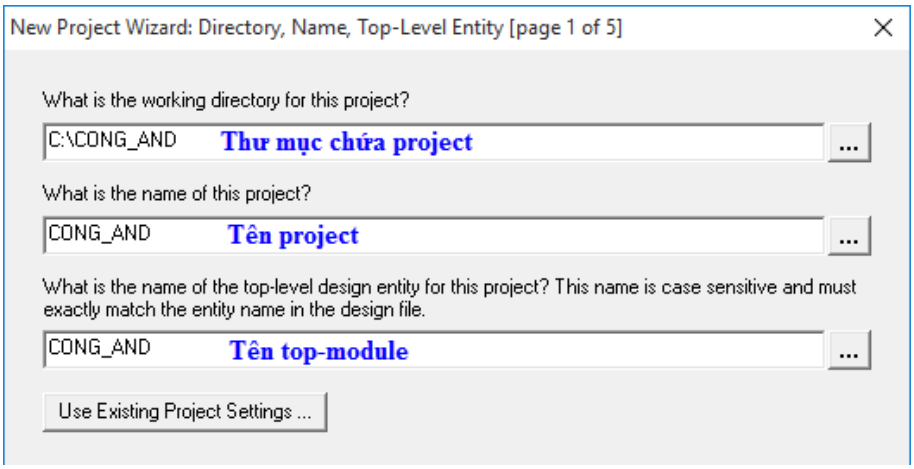


Trong phần này, chúng ta sử dụng phần mềm Quatus II để để tạo thiết kế mạch chứa cổng logic và mô phỏng theo bảng chân trị. Các bước thực hiện như sau:

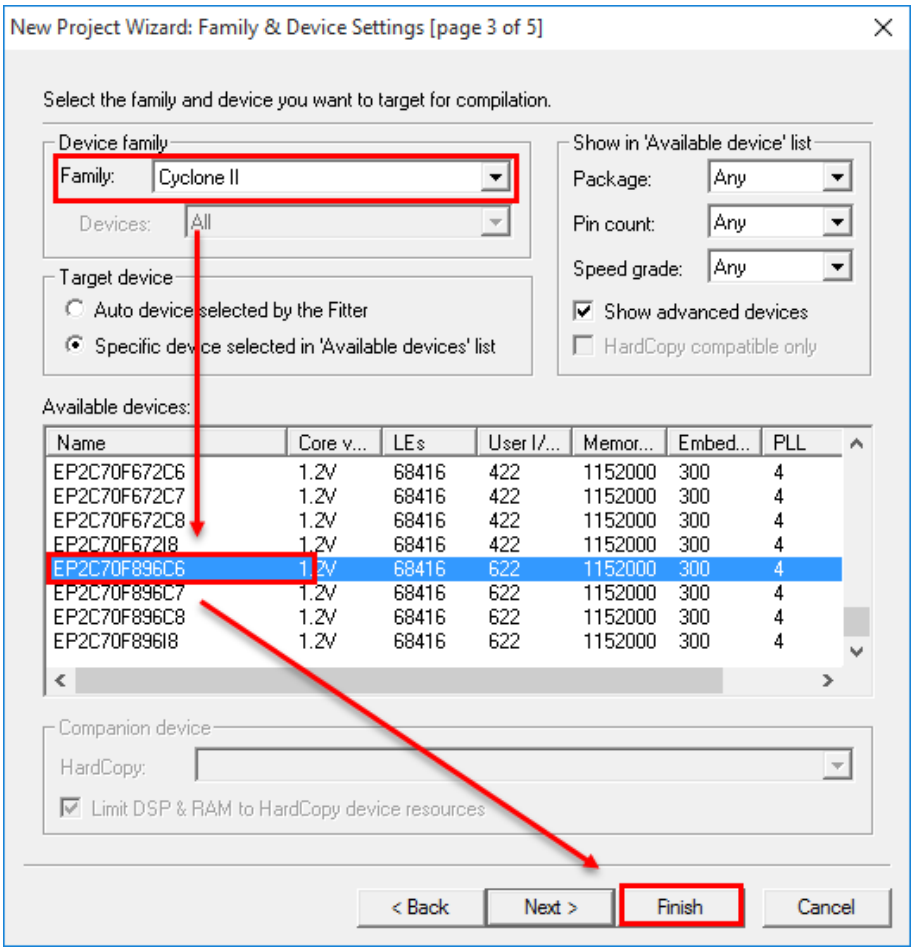
1. Khởi động phần mềm Quatus II từ Desktop hoặc vào menu Start → Program → Quatus II. Giao diện Quatus II xuất hiện:



2. Trước khi tiến hành thiết kế mạch, chúng ta tạo project mới: File → New Project Wizard. Trong cửa sổ đầu tiên: điền thông tin về thư mục chứa project, tên project và tên top-module (tên top-module trùng với tên project). Click Next 2 lần

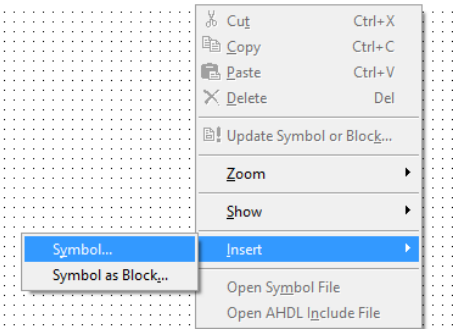


3. Cửa sổ Family & Device Settings dùng để chọn họ và tên linh kiện FPGA. Chọn họ linh kiện **Cyclone II**, tên linh kiện **EP2C70F896C6**. Chọn Finish

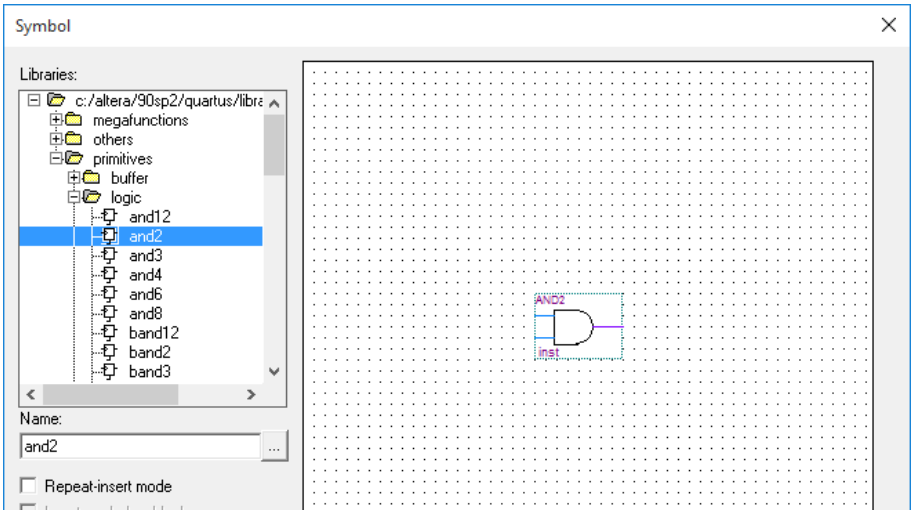


4. Tạo tập tin thiết kế chứa cổng **AND**. Vào File → New → Block Diagram/Schematic File.

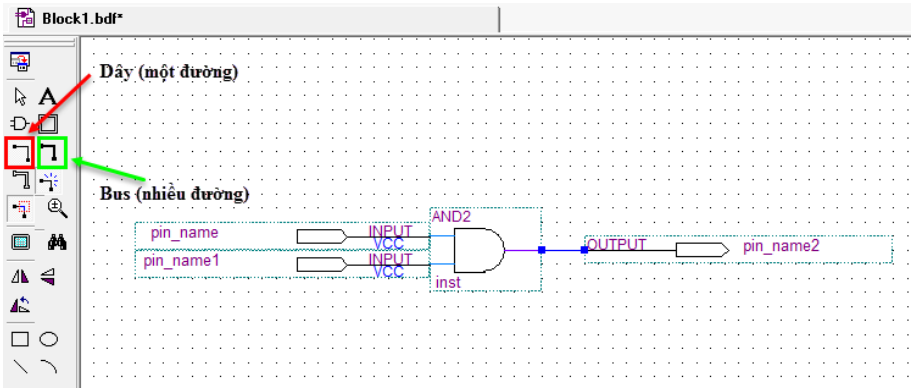
5. Click chuột phải vào trong thiết kế, chọn Insert → Symbol. Hoặc double-click vào vùng thiết kế



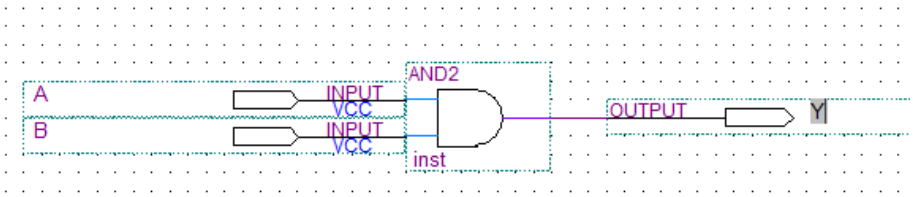
Chọn cổng AND bằng cách gõ "AND2". Bấm OK và gắn vào trong thiết kế



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

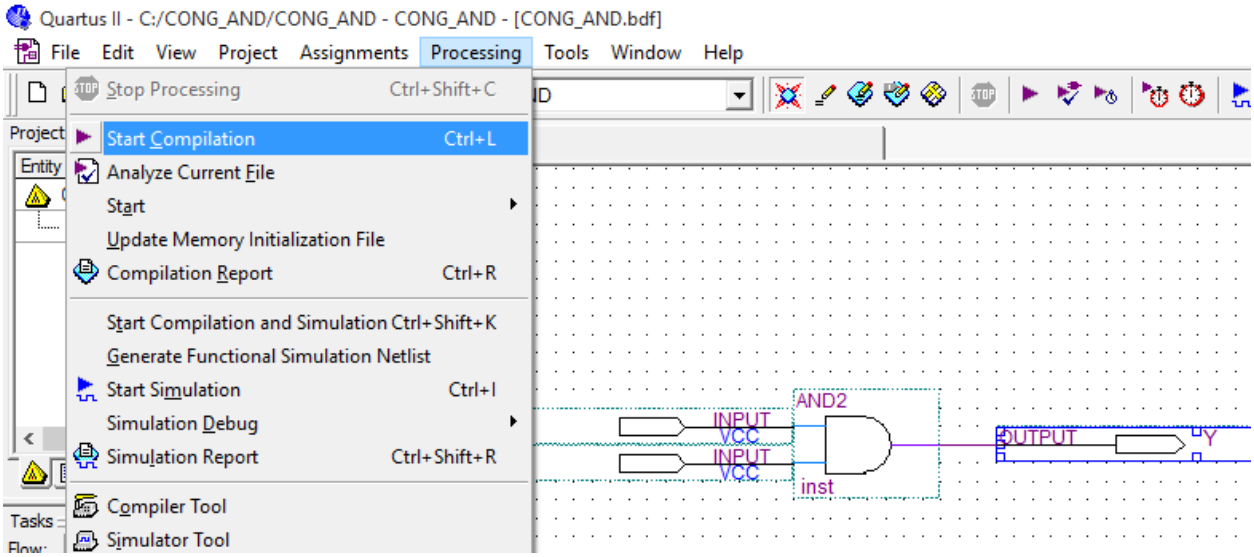


7. Đặt tên cho input lần lượt là **A, B** và output là **Y** bằng cách click chọn symbol → double-click để đổi tên.

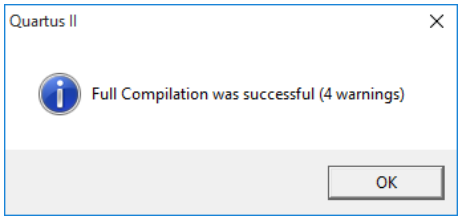


8. Lưu thiết kế và đặt tên là **CONG_AND.bdf** (giữ nguyên tên gợi ý của chương trình). Chọn File → Save

9. Biên dịch thiết kế. Chọn Processing → Start Compilation

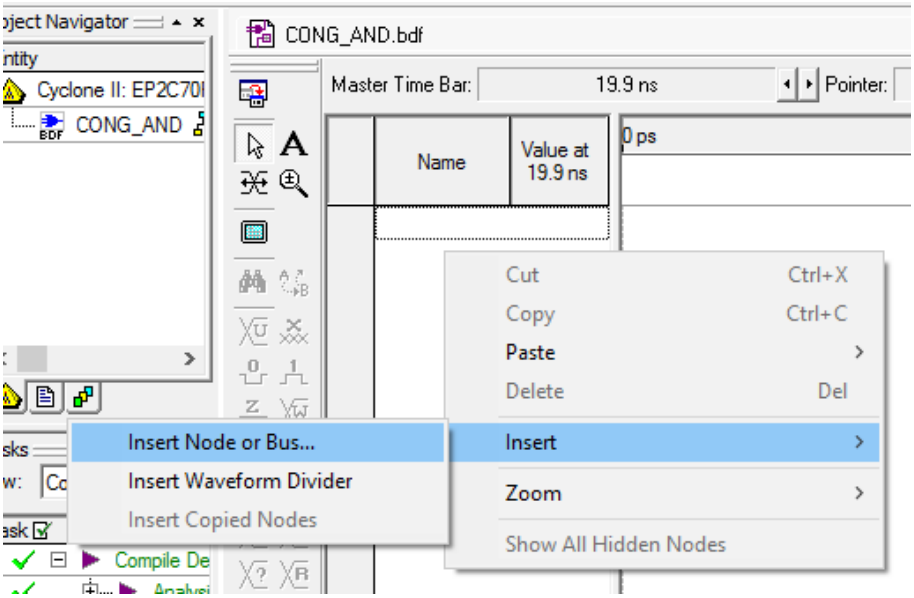


10. Nếu không có lỗi sẽ xuất hiện cửa sổ báo successful. Bấm OK

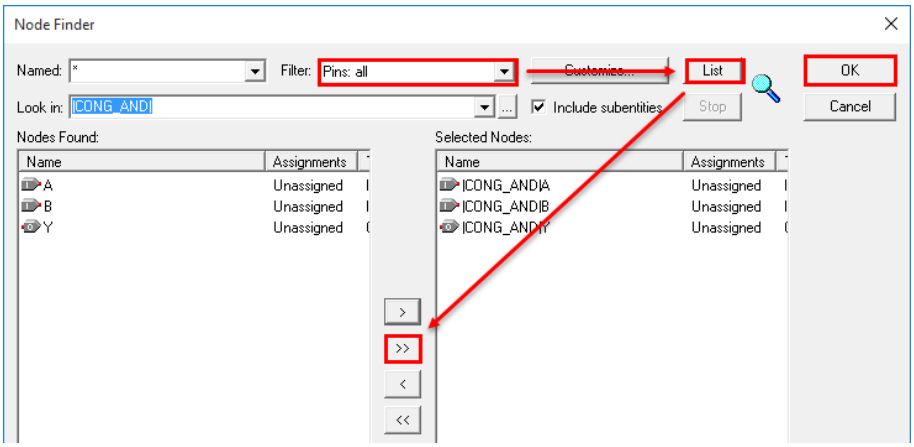


11. Tạo tập tin mô phỏng thiết kế. Vào File → New → Vector Waveform File

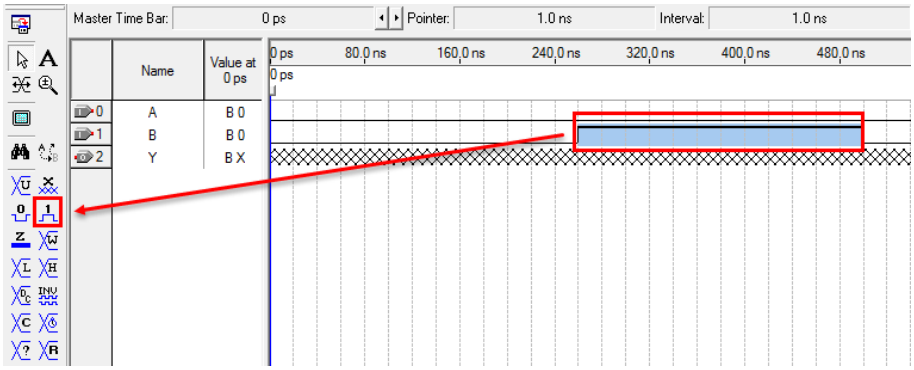
12. Click chuột phải vào cửa sổ "Name". Chọn Insert → Insert Node or Bus



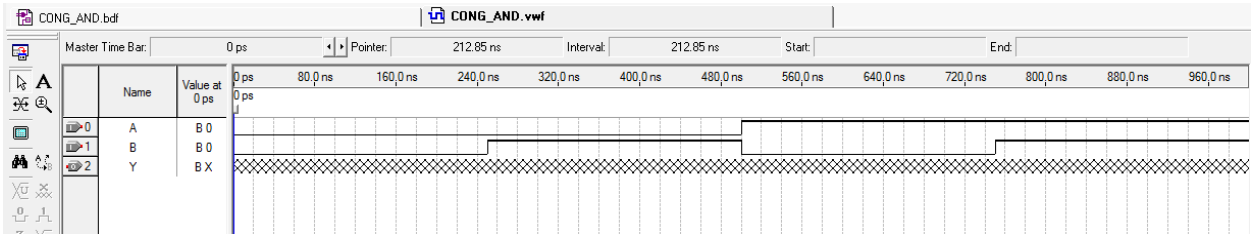
13. Chọn Node Finder. Cửa sổ Node Finder chọn: Pins: all → List → chọn tất cả các chân → Bấm OK 2 lần



14. Giữ Ctrl và lăn chuột để thu nhỏ cửa sổ mô phỏng. Gán giá trị cho các input bằng hộp công cụ bên trái. Chọn đoạn tín hiệu cần gán giá trị bằng cách đặt trỏ chuột ở đầu đoạn → giữ và kéo đến cuối đoạn → nhả chuột để xác định. Sau đó chọn giá trị 1 hoặc 0 (giá trị mặc định là 0).

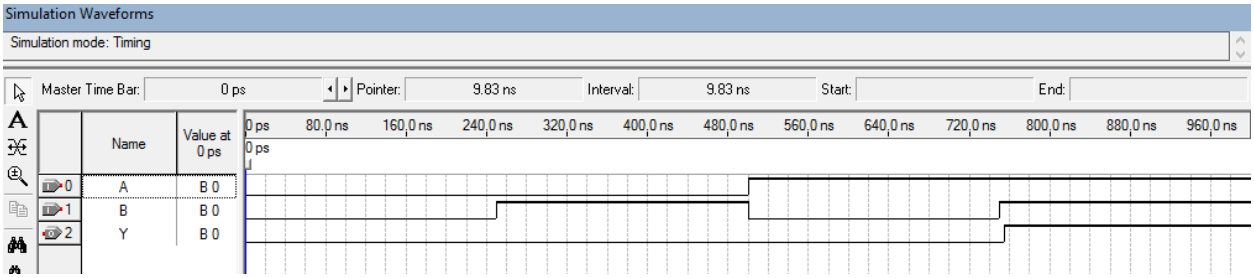


15. Thiết lập giá trị cho các đoạn tín hiệu còn lại. Lưu lại với tên tập tin: **CONG_AND.vwf**



16. Vào menu Processing → Start Simulation để mô phỏng

17. Kết quả dạng sóng thu được



18. Đối chiếu với bảng chân trị để kiểm tra kết quả

BÀI TẬP: Tạo các project khác tương ứng với các cổng logic: **NOT, OR, XOR, NAND, NOR, XNOR**. Tạo các file thiết kế và mô phỏng tương ứng.

MẠCH CỘNG HA (Half Adder – mạch cộng 2 bit)

1. Bảng trạng thái

INPUT		OUTPUT	
A	**B**	**S**	**Cout**
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

2. Phương trình trạng thái

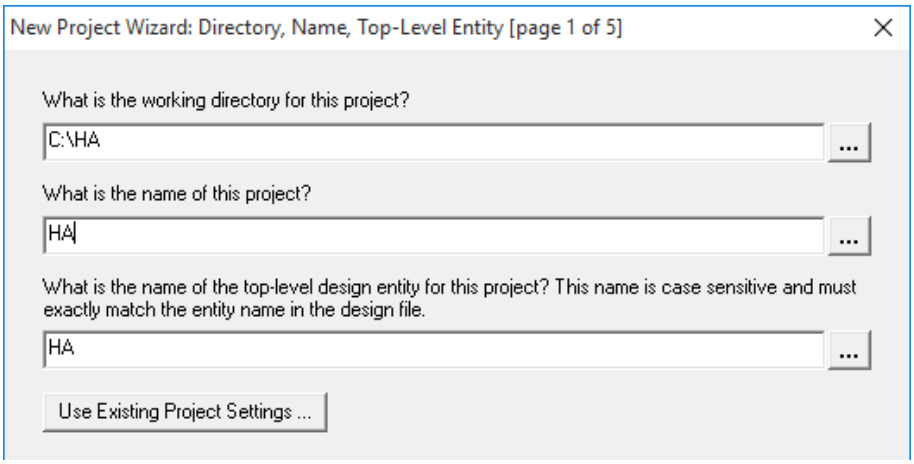
$$S = A \oplus B$$

$$Cout = A \,.\, B$$

Các bước thực hiện trên phần mềm **Quatus II:**

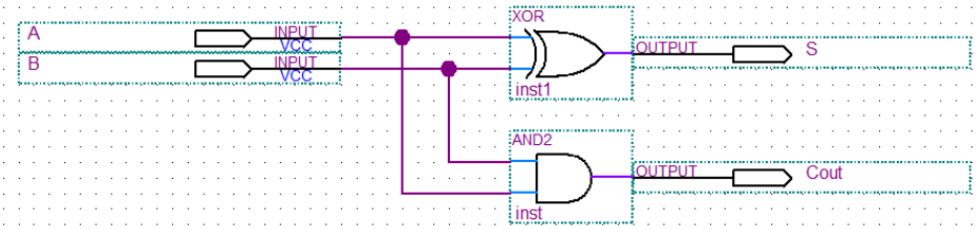
3. Khởi động phần mềm **Quatus II**

4. Tạo một project mới, đặt tên: **HA**



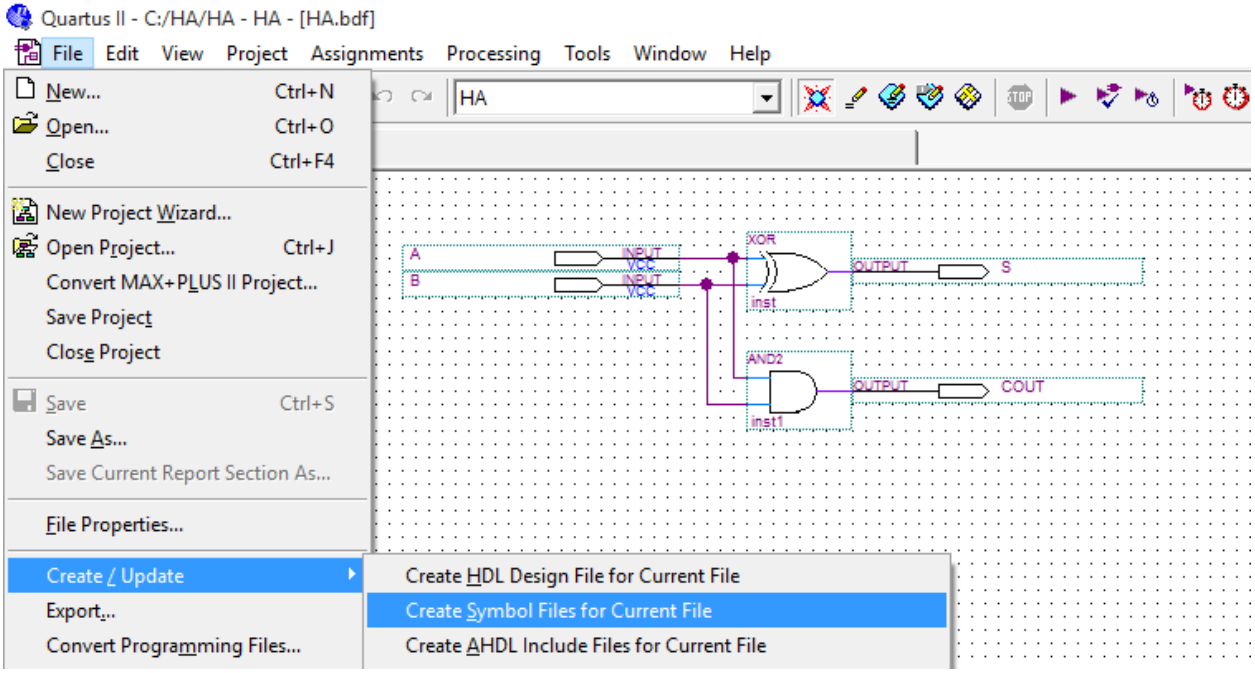
5. Tạo file thiết kế: File → New → Block Diagram/Schematic File

6. Thực hiện thiết kế mạch cộng HA như hình bên dưới. Lưu lại với tên **HA.bdf**

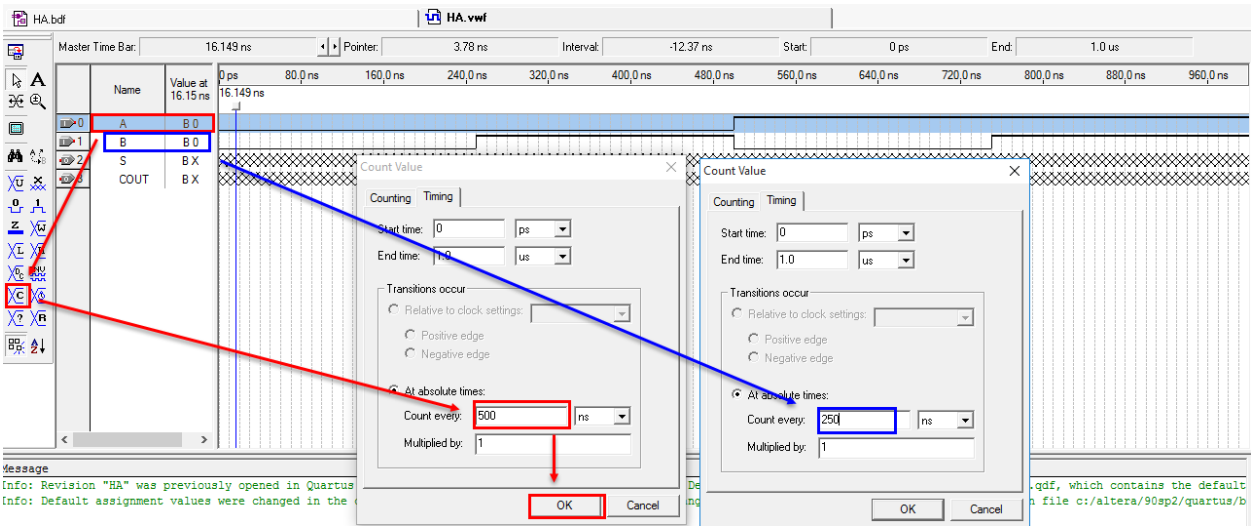


7. Biên dịch thiết kế: Processing → Start Compilation

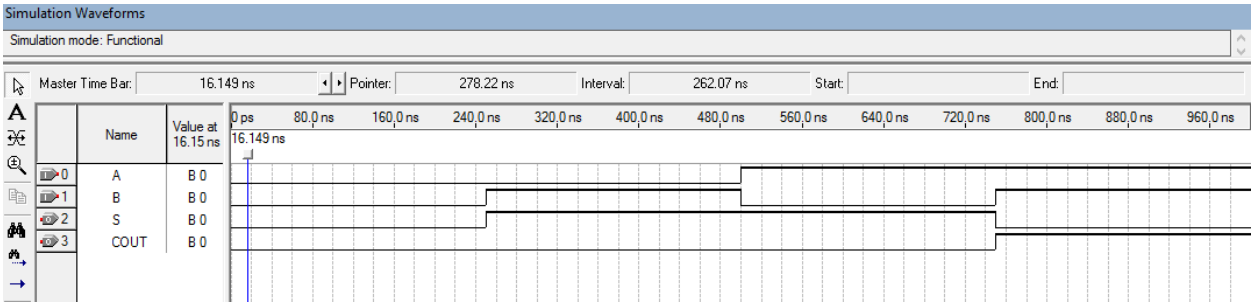
8. Mạch HA được sử dụng rất nhiều trong các thiết kế sau này. Để thuận tiện, chúng ta thực hiện đóng gói thiết kế (tạo symbol file) cho file **HA.bdf** bằng cách vào menu File → Create/Update → Create Symbol File for Current File. Lưu lại với tên là **HA.bsf**



9. Tạo tập tin mô phỏng: File → New → Vector Waveform File. Cài đặt các tham số như hình bên dưới. Lưu lại với tên: **HA.vwf**



10. Mô phỏng thiết kế **HA**



MẠCH CỘNG FA (Full Adder – mạch cộng 3 bit)

1. Bảng trạng thái

INPUT			OUTPUT	
A	**B**	**Cin**	**S**	**Cout**
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

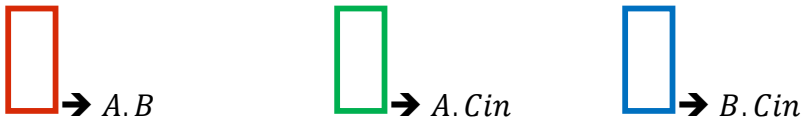
2. Phương trình trạng thái

$$S = A \oplus B \oplus Cin$$

$$Cout = \bar{A}.B.Cin + A.\bar{B}.Cin + A.B.\overline{Cin} + A.B.Cin$$

Rút gọn Cout dùng giản đồ Kanaugh

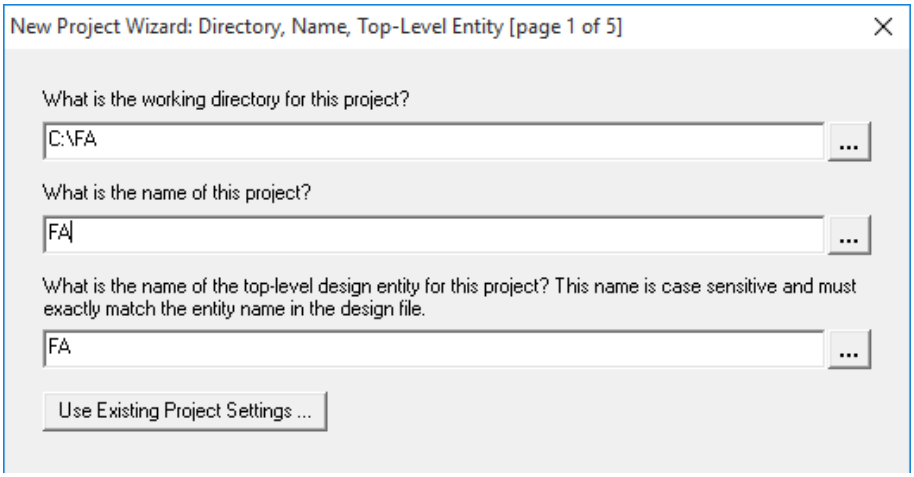
Cin \ AB	00	01	11	10
0			1	
1		1	1	1



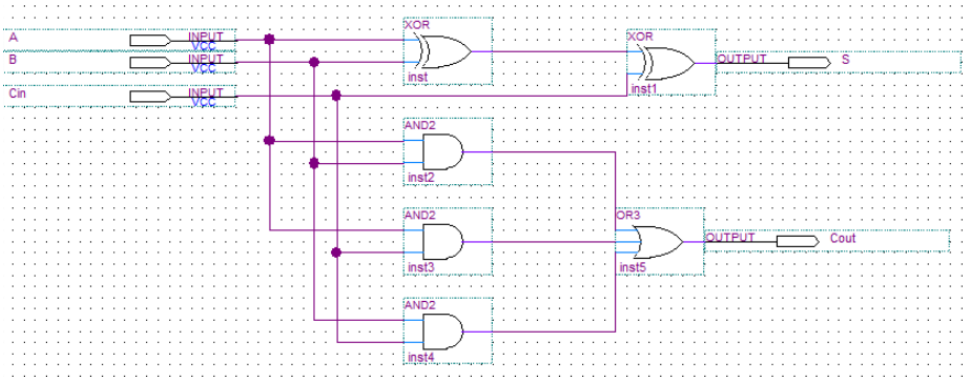
⇨ $Cout = A.B + A.Cin + B.Cin$

Các bước thực hiện trên phần mềm **Quatus II:**

3. Tạo project mới, đặt tên: **FA**



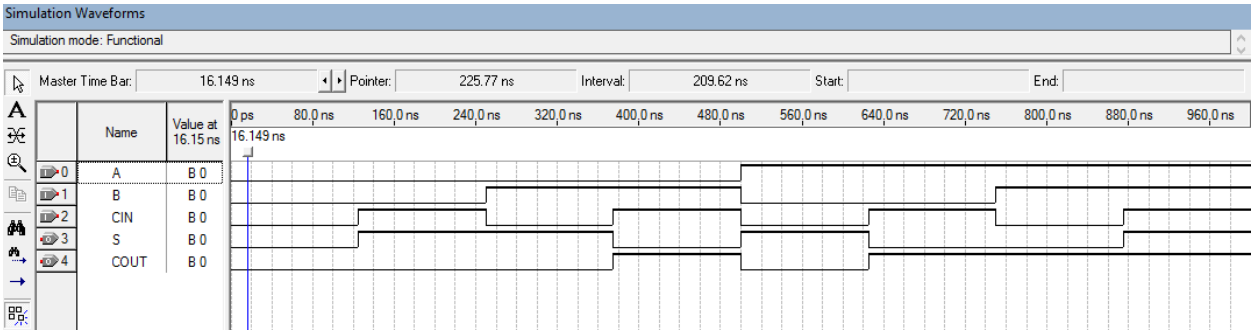
4. Tạo tập tin thiết kế mạch **FA**. Lưu lại với tên: **FA.bdf**



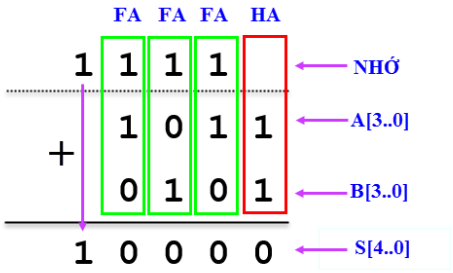
5. Biên dịch thiết kế: Processing → Start Compilation

6. Đóng gói thiết kế. Đặt tên là: **FA.bsf**

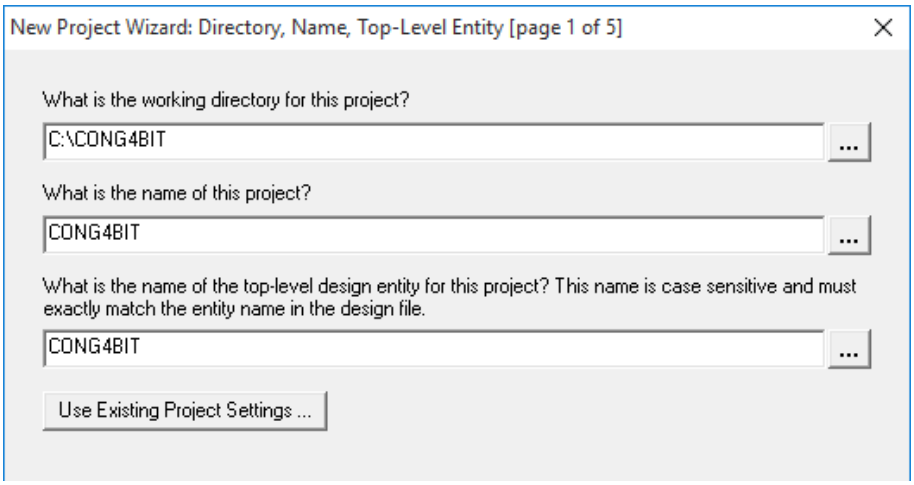
5. Tạo tập tin mô phỏng. Lưu lại với tên **FA.vwf**



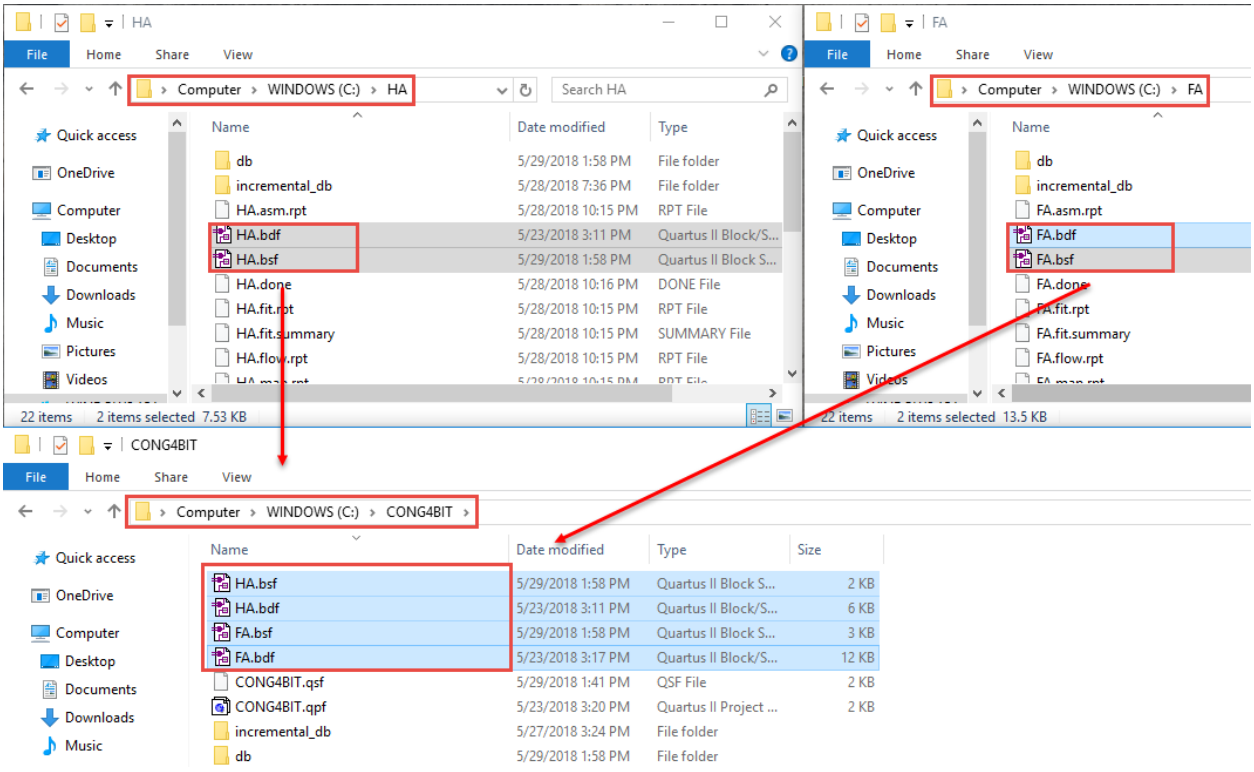
MẠCH CỘNG 4 BIT KHÔNG DẤU



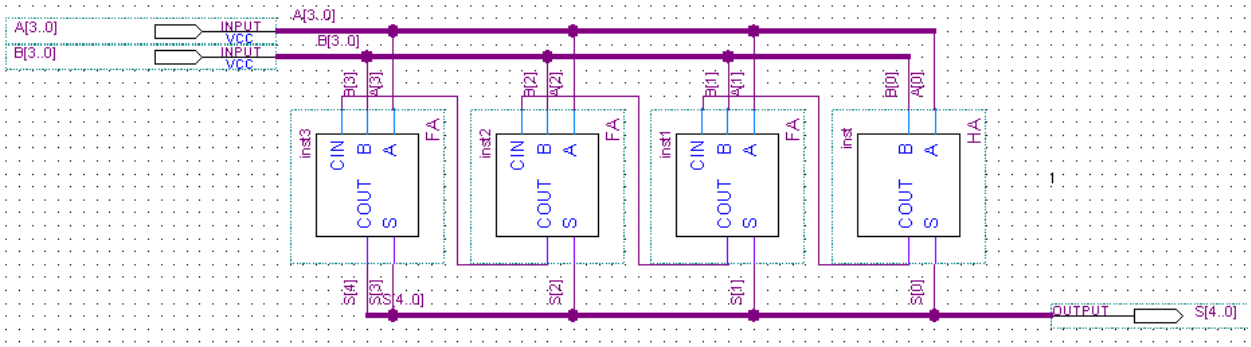
1. Tạo project mới, đặt tên: **CONG4BIT**



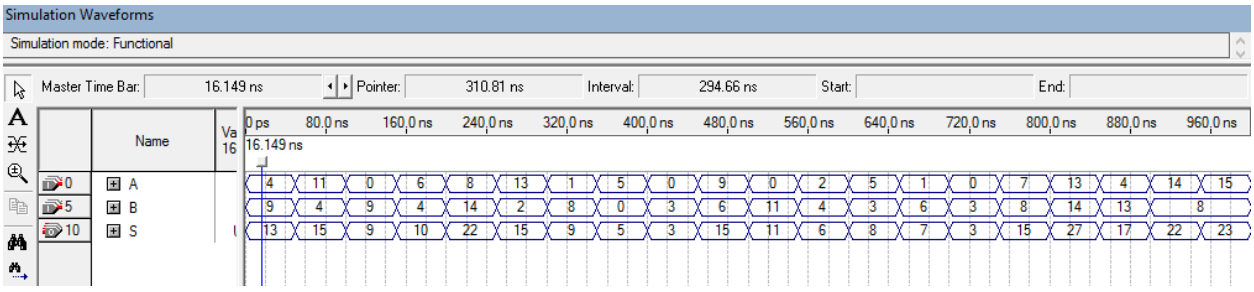
2. Mạch cộng 4 bit được xây dựng từ 1 mạch cộng **HA** và 3 mạch cộng **FA**. Do đó, trước khi tiến hành thiết kế mạch cộng 4 bit, chúng ta sao chép các tập tin **HA.bdf, HA.bsf**, **FA.bdf** và **FA.bsf** từ các project **HA** và **FA** vào project vừa tạo.



2. Tạo tập tin thiết kế mạch cộng 4 bit. Lưu lại với tên: **CONG4BIT.bdf**



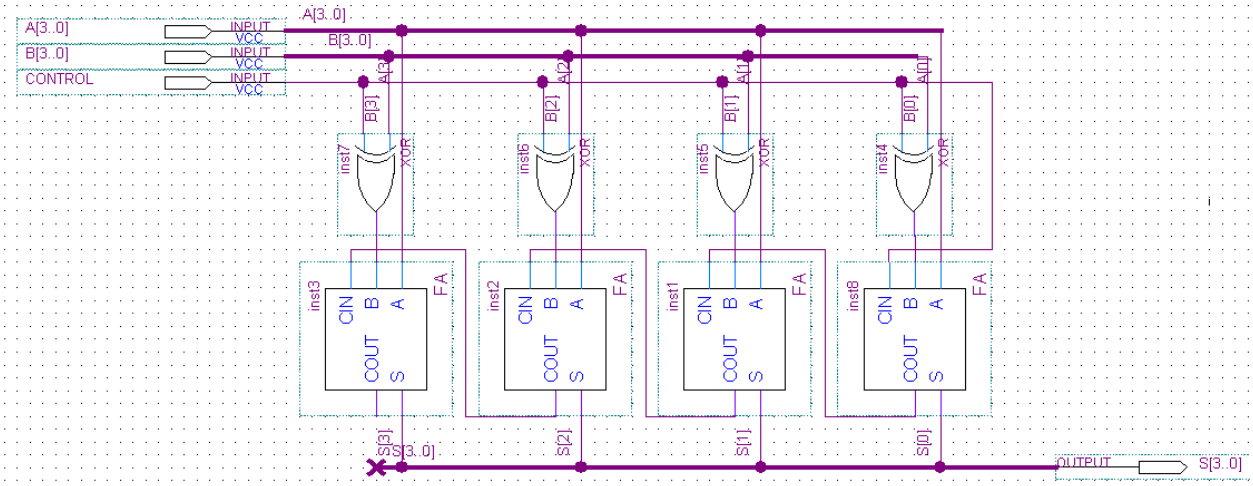
3. Tạo tập tin mô phỏng. Lưu ý: chọn hệ cơ số khi mô phỏng là **Unsigned Decimal**. Lưu lại với tên: **CONG4BIT.vwf**



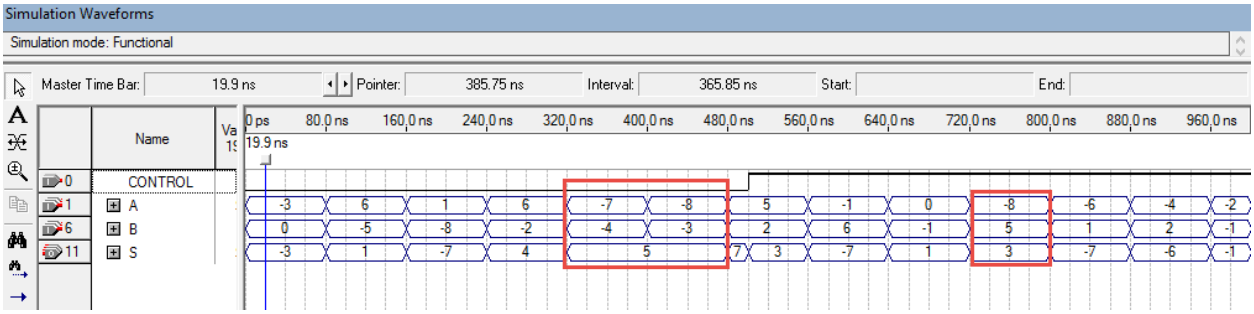
MẠCH CỘNG/TRỪ 4 BIT CÓ DẤU

1. Tạo project mới, đặt tên: **CONGTRU4BIT**

2. Tạo tập tin thiết kế mạch cộng/trừ 4 bit. Lưu lại với tên: **CONGTRU4BIT.bdf**. (Sao chép các tập tin cần thiết **HA, FA** vào project mới)



3. Tạo tập tin mô phỏng. Lưu ý: chọn hệ cơ số khi mô phỏng là **Signed Decimal**. Lưu lại với tên: **CONGTRU4BIT.vwf**

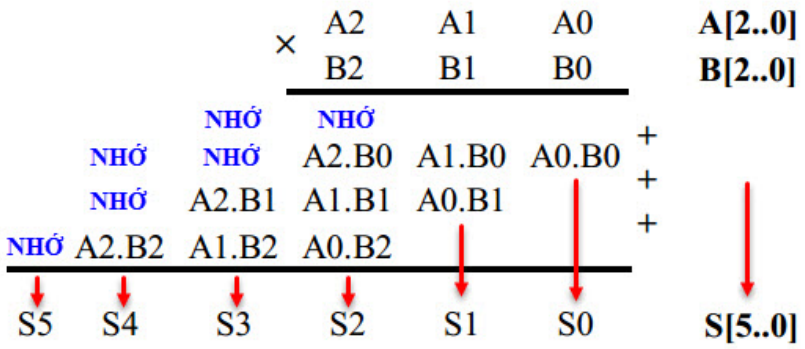


Giải thích:

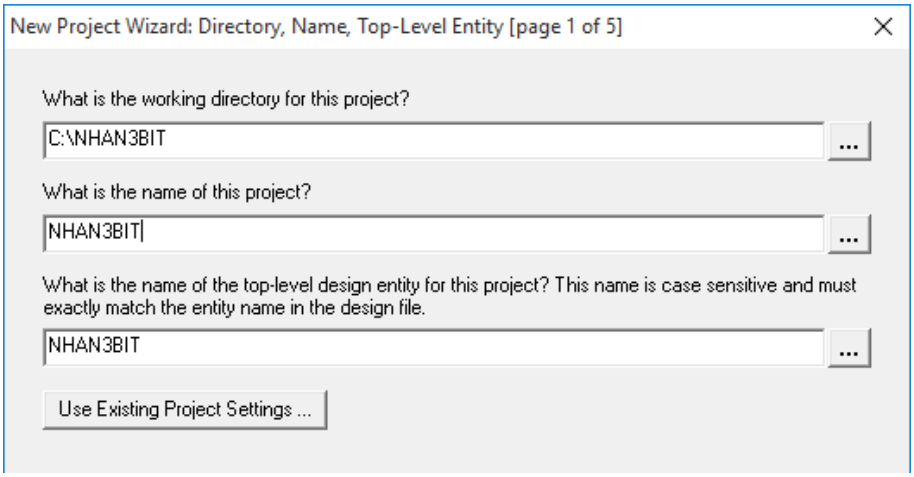
- Khi tín hiệu **CONTROL** = 0 → mạch thực hiện chức năng cộng (S=A+B), khi **CONTROL** = 1 → mạch thực hiện chức năng trừ (S=A-B)
- Mạch thực hiện chức năng cộng/trừ trên số có dấu 4 bit nên kết quả chỉ hiện thị đúng khi tổng/hiệu thuộc miền giá trị [-8,+7]

MẠCH NHÂN 3 BIT KHÔNG DẤU

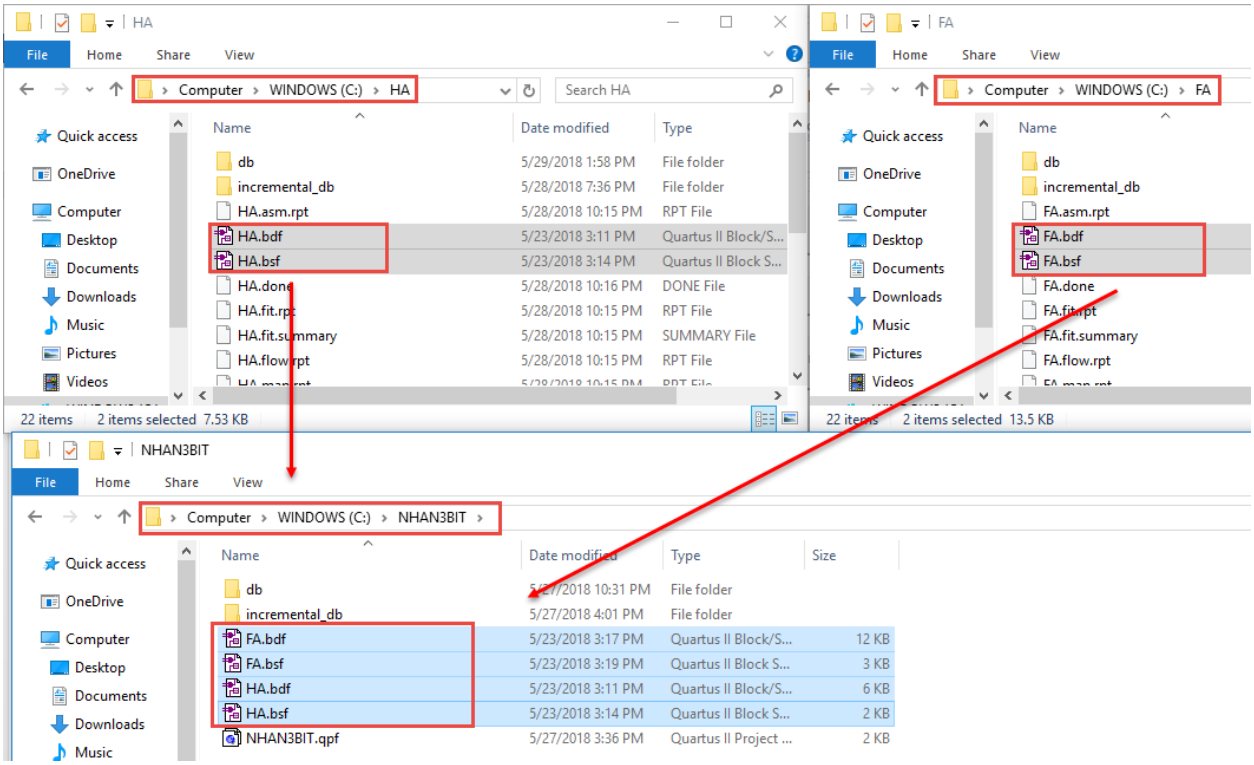
Cách thực hiện phép nhân 2 số A, B (3 bit) kết quả là S (6 bit)



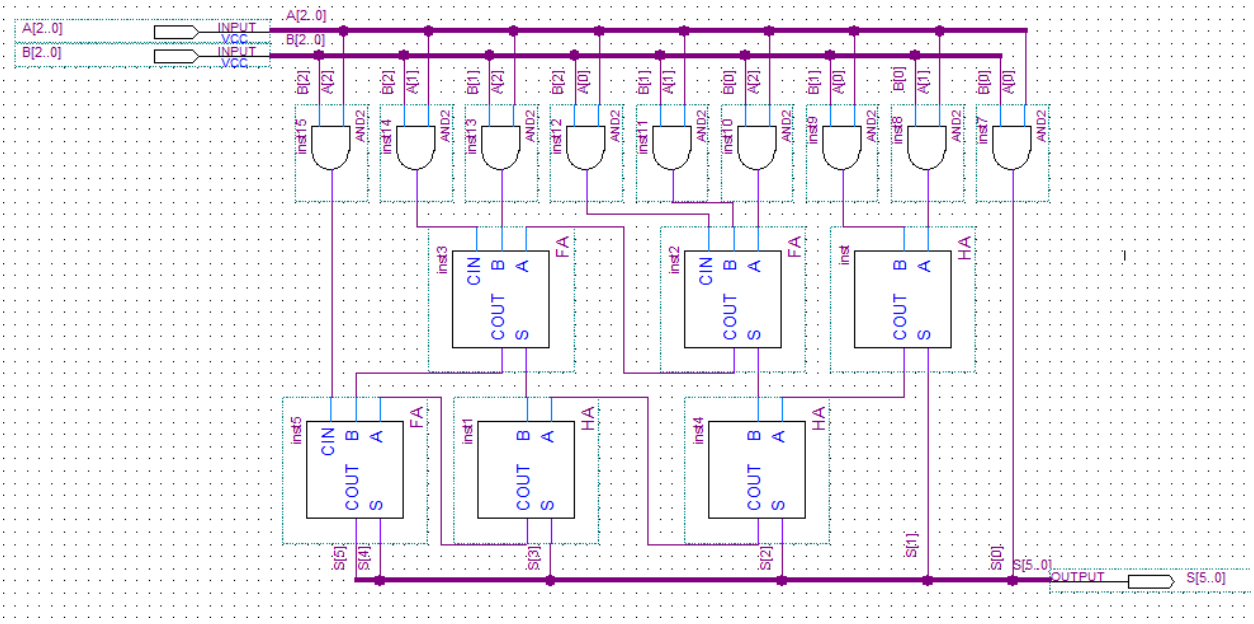
1. Tạo project mới đặt tên là: **NHAN3BIT**



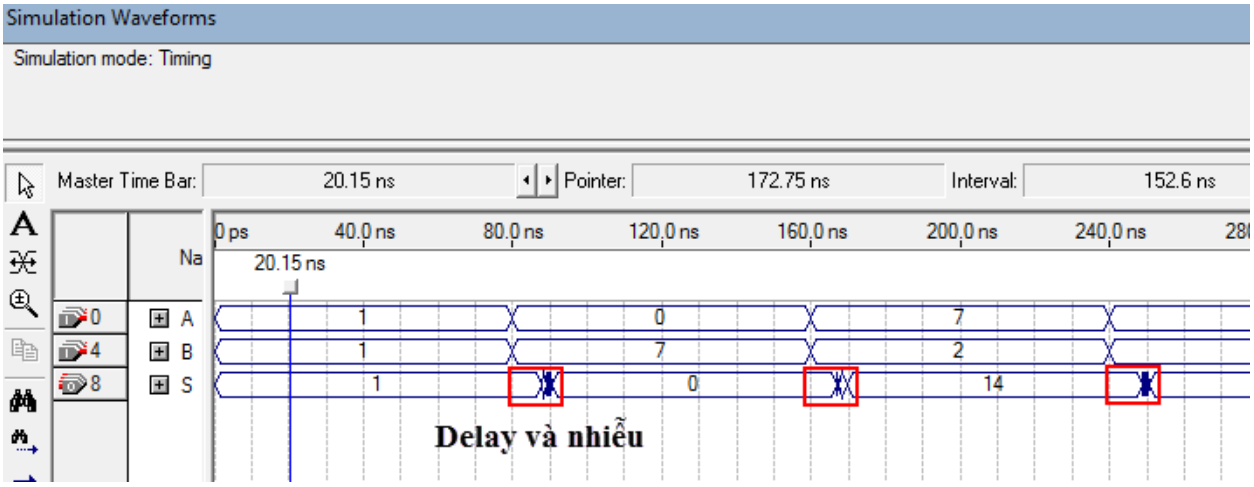
2. Mạch nhân 3 bit được ghép các module mạch cộng 1 bit (**FA, HA**)và các cổng **AND**. Chúng ta tiến hành sao chép các tập tin cần thiết vào project mới.



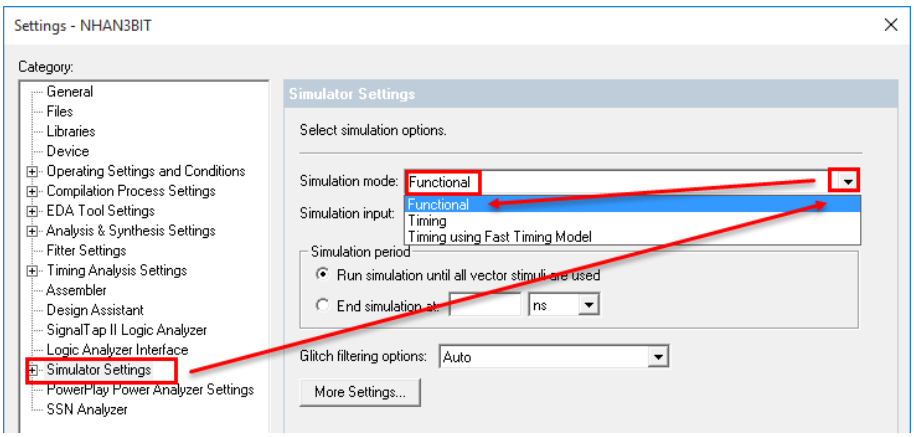
3. Tạo tập tin thiết kế mạch nhân 3 bit. Lưu lại với tên: **NHAN3BIT.bdf**



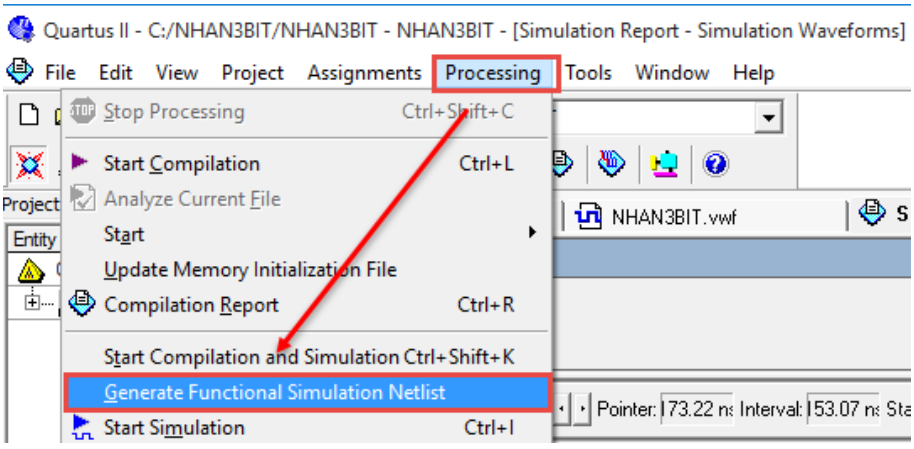
4. Tạo tập tin mô phỏng. Lưu ý: chọn hệ cơ số khi mô phỏng là **Unsigned Decimal**. Lưu lại với tên: **NHAN3BIT.vwf**



Muốn loại bỏ delay và nhiễu hiển thị trên kết quả ta chuyển chế độ mô phỏng từ Timing sang Functional. Từ menu Assignments → Settings → Simulator Settings → Simulation mode: chọn Functional

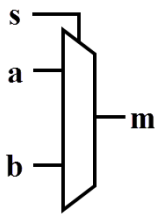


Từ menu Processing → Generate Functional Simulation Netlist. Chạy lại mô phỏng và quan sát kết quả.



MẠCH ĐA HỢP

I. Mạch đa hợp 2-1 1 bit (MUX2_1_1BIT)



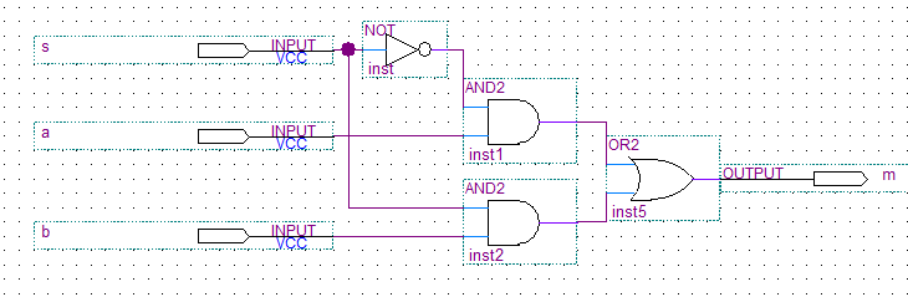
1. Bảng trạng thái

s	Ngõ ra tại m
0	a
1	b

2. Phương trình trạng thái

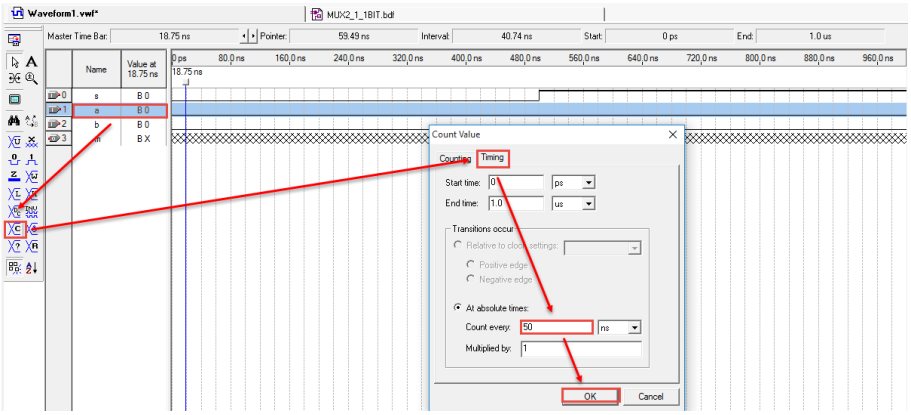
$$m = \bar{s}.a + s.b$$

3. Mạch **MUX2_1_1BIT**

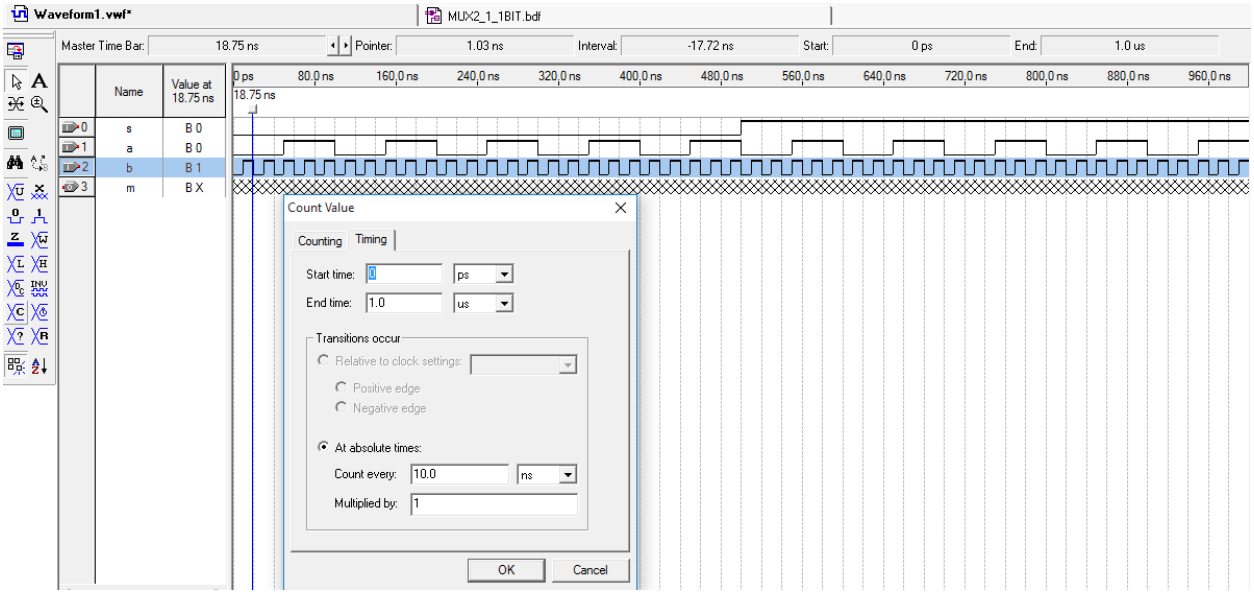


4. Mô phỏng

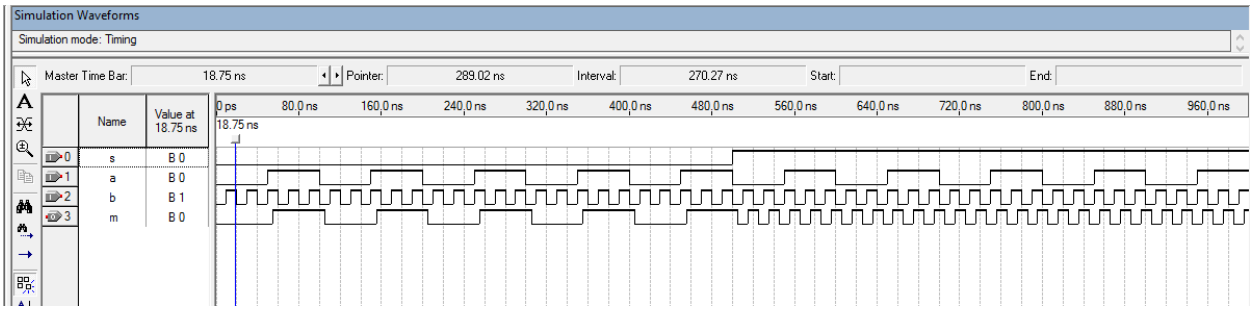
- Chọn ngõ vào a, chọn Count Value → Timing → Mục count every: đặt giá trị là 50



- Tương tự, tại ngõ vào b, đặt giá trị là 10. Mục đích của việc này là tạo ra 2 tín hiệu có thể hiện khác nhau.



- Lưu và chạy mô phỏng, kết quả thể hiện như hình bên dưới



Giải thích: khi s bằng 0 → ngõ ra m bằng a, khi s bằng 1 → ngõ ra m bằng b

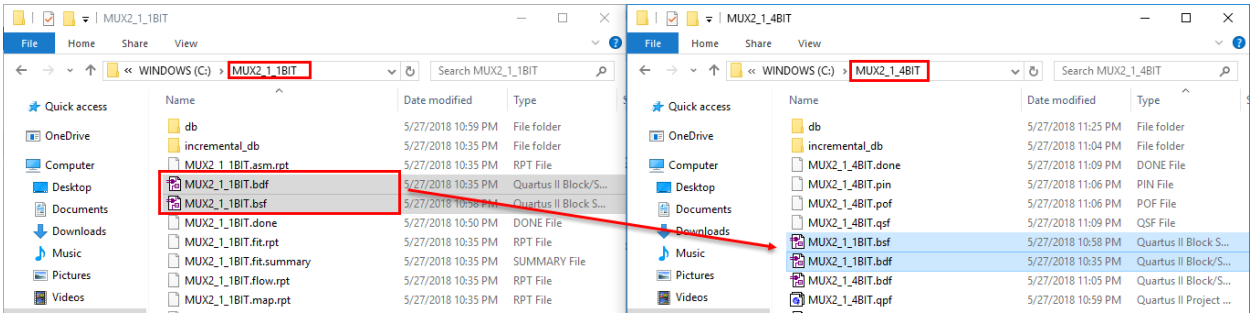
- Thực hiện đóng gói thiết kế cho mạch đa hợp 2-1 1bit. Lưu và đặt tên là: **MUX2_1_1BIT.bsf**

II. Mạch đa hợp 2-1 4 bit (MUX2_1_4BIT)

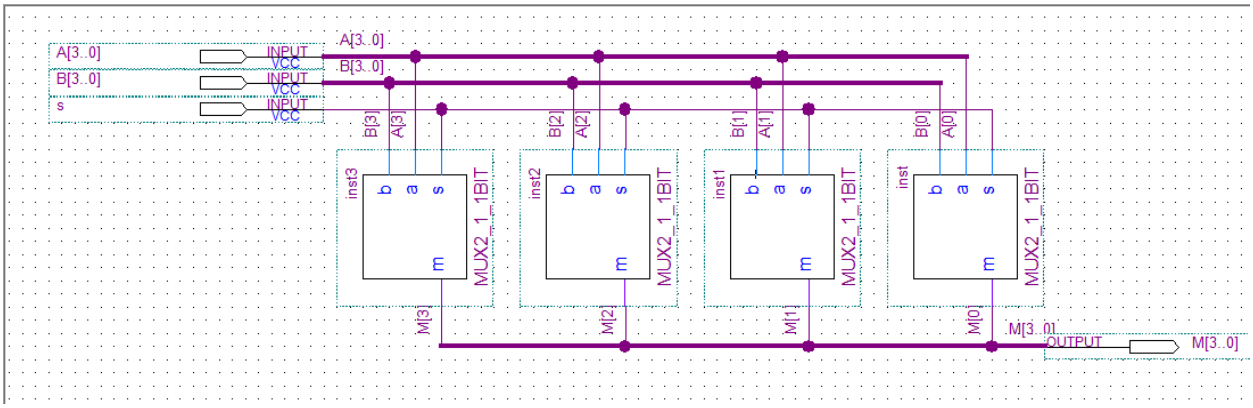
Thực hiện ghép 4 mạch **MUX2_1_1BIT** để tạo thành mạch đa hợp 2-1 4 bit

1. Tạo project mới đặt tên là **MUX2_1_4BIT**

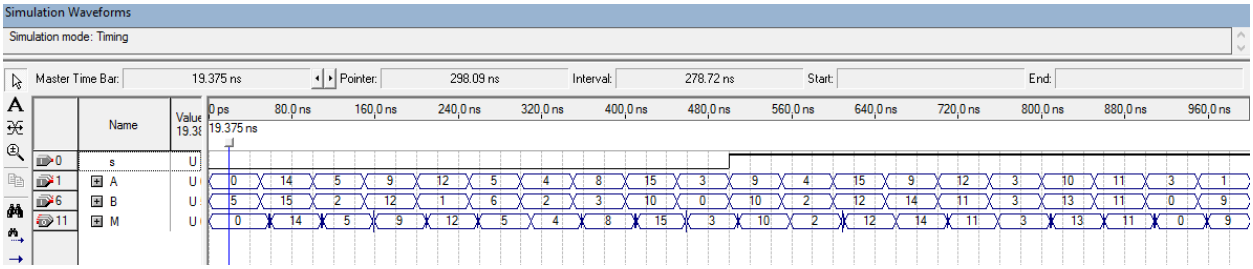
2. Sao chép các file cần thiết vào project mới



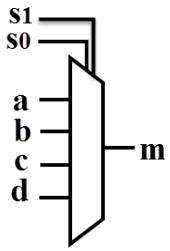
3. Mạch **MUX2_1_4BIT**



4. Mô phỏng



II. Mạch đa hợp 4-1 1 bit (MUX4_1_1BIT)



1. Bảng trạng thái

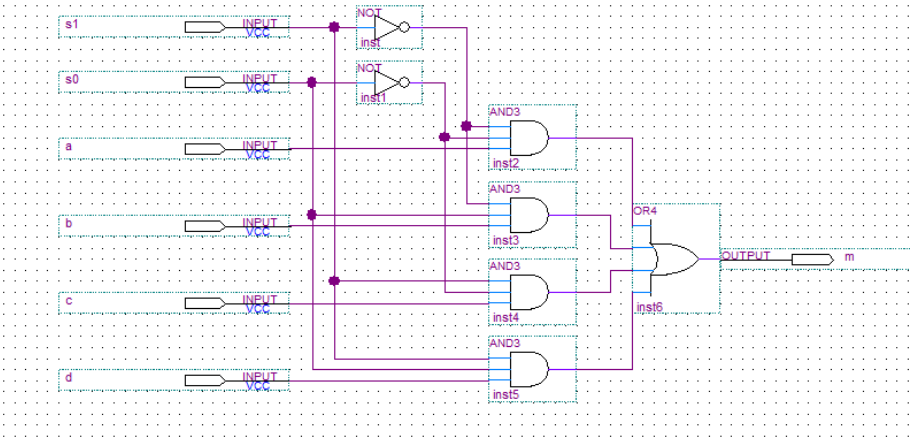
s1	s0	Ngõ ra tại m
0	0	a
0	1	b
1	0	c

1	1	d

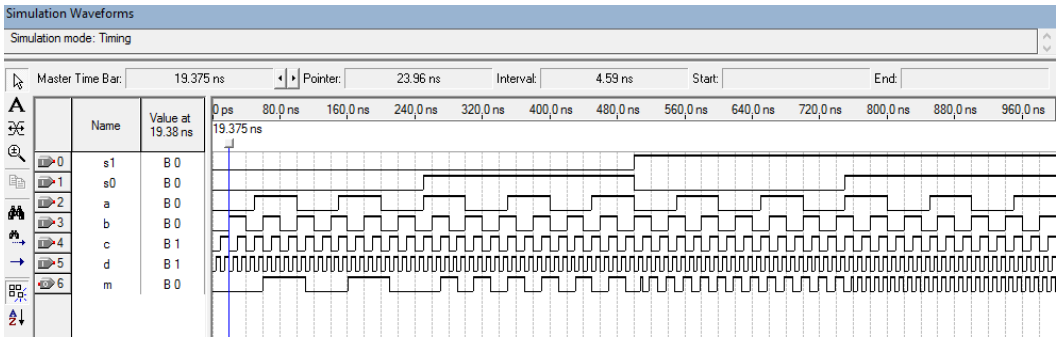
2. Phương trình trạng thái

$$m = \overline{s1}.\overline{s0}.a + \overline{s1}.s0.b + s1.\overline{s0}.c + s1.s0.d$$

3. Mạch **MUX4_1_1BIT**



4. Mô phỏng



- Thực hiện đóng gói thiết kế cho mạch đa hợp 4-1 1bit. Lưu và đặt tên là: **MUX4_1_1BIT.bsf**

IV. Mạch đa hợp 4-1 4 bit (MUX4_1_4BIT)

Thực hiện ghép 4 mạch **MUX4_1_1BIT** để tạo thành mạch đa hợp 4-1 4 bit. *(Sinh viên tự làm)*

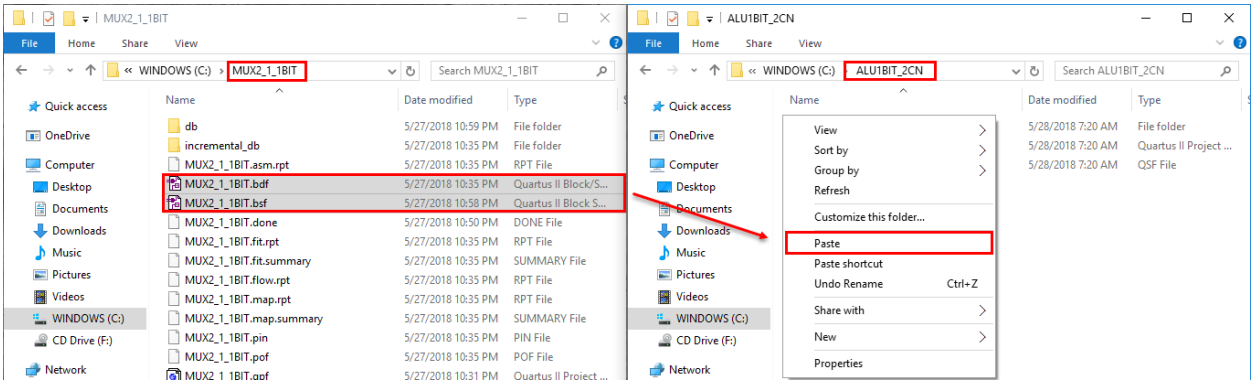
ĐƠN VỊ XỬ LÝ SỐ HỌC (ALU)

I. ALU 1 bit 2 chức năng AND, OR (ALU1BIT_2CN)

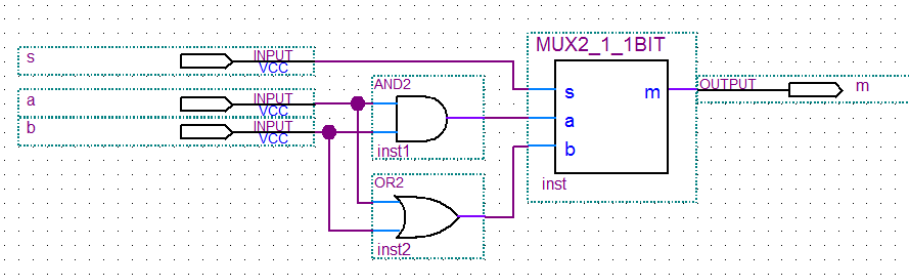
Xây dựng mạch ALU 1 bit 2 chức năng từ mạch đa hợp 2-1 1 bit kết hợp với cổng AND, OR

1. Tạo project mới đặt tên là **ALU1BIT_2CN**

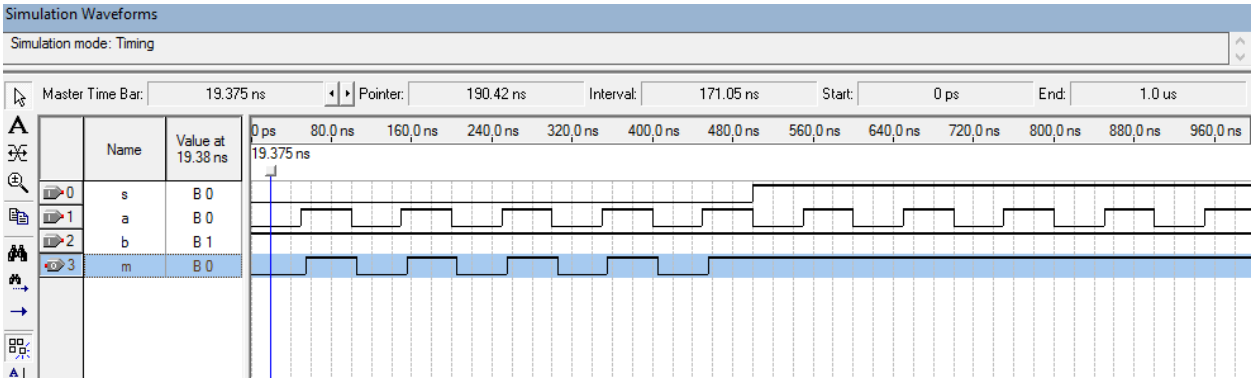
2. Sao chép các file cần thiết vào project mới



3. Mạch **ALU1BIT_2CN**



4. Mô phỏng



Giải thích: Khi s bằng 0 → mạch thực hiện chức năng **AND**, khi s bằng 1 → mạch thực hiện chức năng **OR**

II. ALU 4 bit 2 chức năng AND, OR (ALU4BIT_2CN)

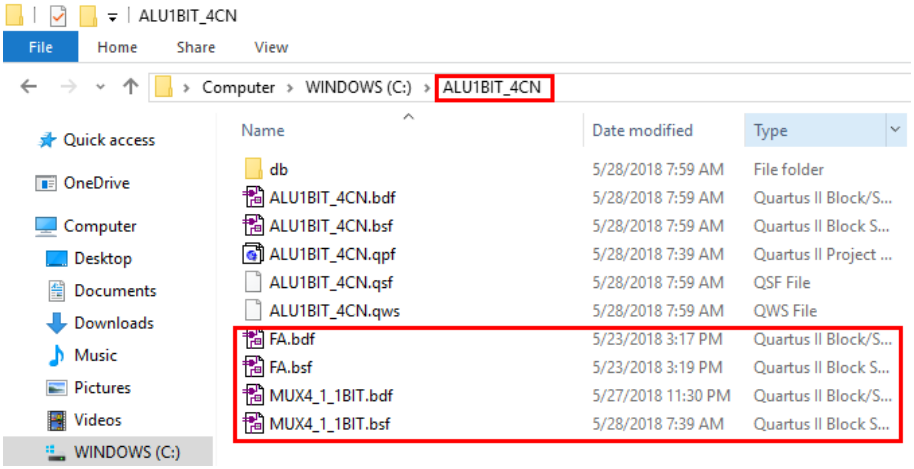
Thực hiện ghép 4 mạch **ALU1BIT_2CN** để tạo thành mạch **ALU4BIT_2CN** *(Sinh viên tự làm)*

III. ALU 1 bit 4 chức năng AND, OR, XOR, CỘNG/TRỪ (ALU1BIT_4CN)

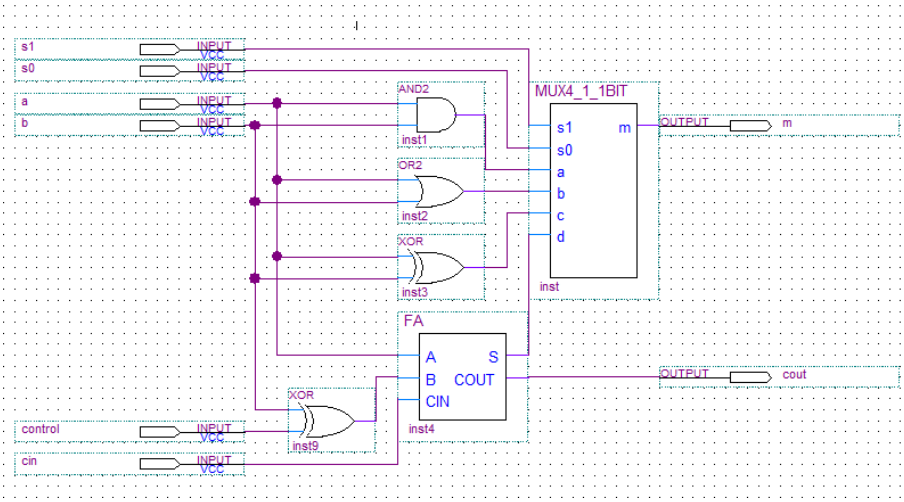
Xây dựng mạch ALU 1 bit 4 chức năng từ mạch đa hợp 4-1 1 bit kết hợp với cổng AND, OR, XOR và mạch CỘNG/TRỪ

1. Tạo project mới đặt tên là **ALU1BIT_4CN**

2. Sao chép các file cần thiết vào project **ALU1BIT_4CN**



3. Mạch **ALU1BIT_4CN**



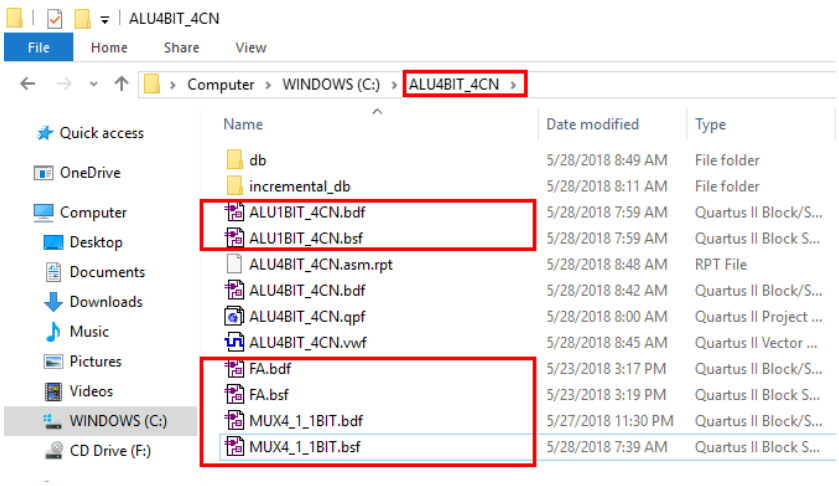
4. Thực hiện đóng gói thiết kế cho mạch **ALU1BIT_4CN**. Lưu và đặt tên: **ALU1BIT_4CN.bsf**

IV. ALU 4 bit 4 chức năng AND, OR, XOR, CỘNG/TRỪ (ALU4BIT_4CN)

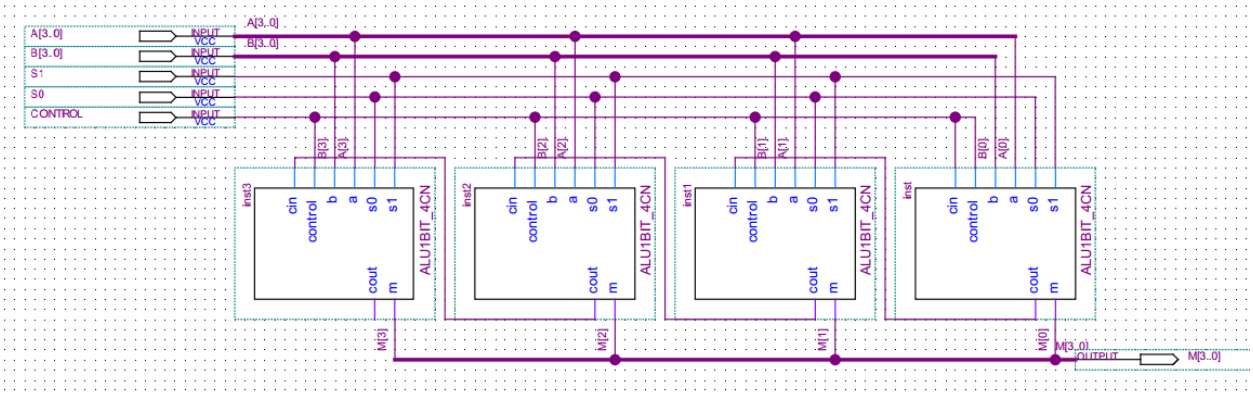
Thực hiện ghép 4 mạch **ALU1BIT_4CN** để tạo thành mạch **ALU4BIT_4CN**

1. Tạo project mới đặt tên là **ALU4BIT_4CN**

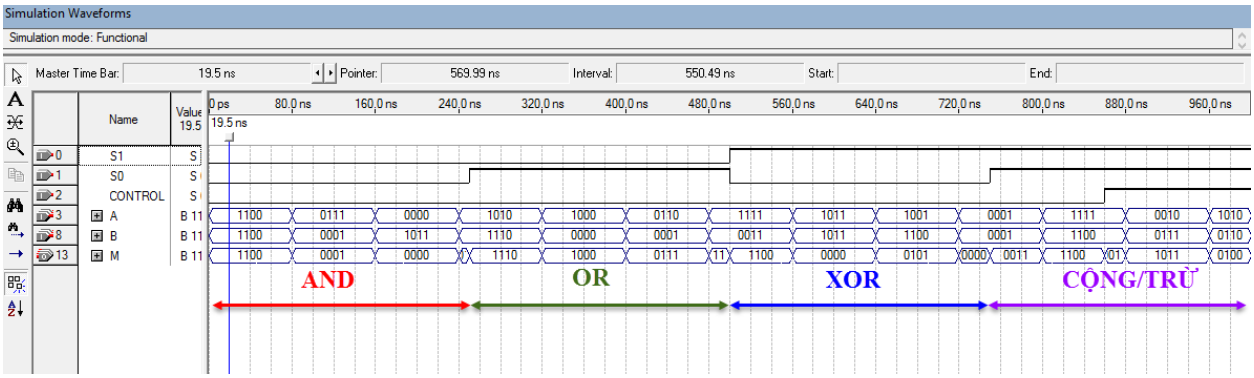
2. Sao chép các file cần thiết vào project **ALU4BIT_4CN**



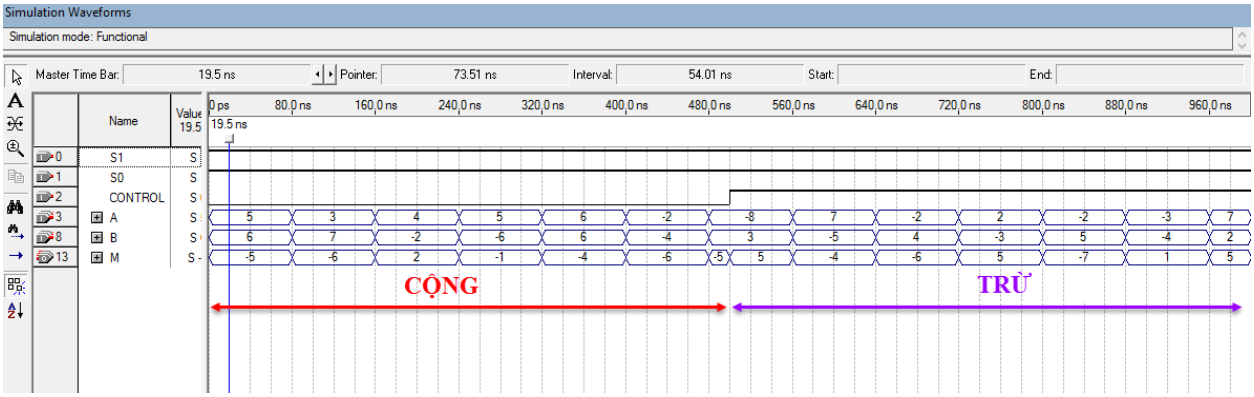
3. Mạch **ALU4BIT_4CN**



4. Mô phỏng



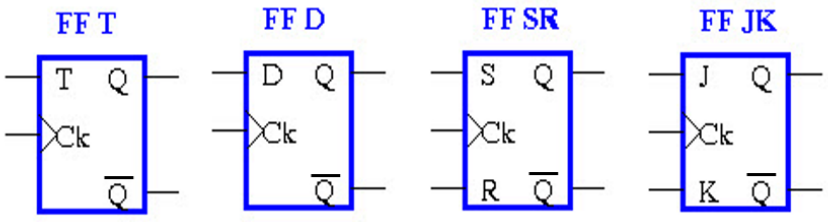
Để dễ kiểm tra chức năng CỘNG/TRỪ của mạch, ta chuyển hệ cơ số khi mô phỏng là **signed decimal**


Simulation Waveforms
Simulation mode: Functional
Master Time Bar: 19.5 ns
Pointer: 73.51 ns
Interval: 54.01 ns
Start:
End:
Name
Value 19.5
S1
S0
CONTROL
A
B
M
CỘNG
TRỪ

BỘ ĐẾM, THANH GHI

I. Flip-flop (FF)

FF là mạch có khả năng lật lại trạng thái ngõ ra tuỳ theo sự tác động thích hợp của ngõ vào, điều này có ý nghĩa quan trọng trong việc lưu trữ dữ liệu trong mạch và xuất dữ liệu ra khi cần. Có nhiều loại flip-flop khác nhau, chúng được sử dụng rộng rãi trong nhiều ứng dụng.



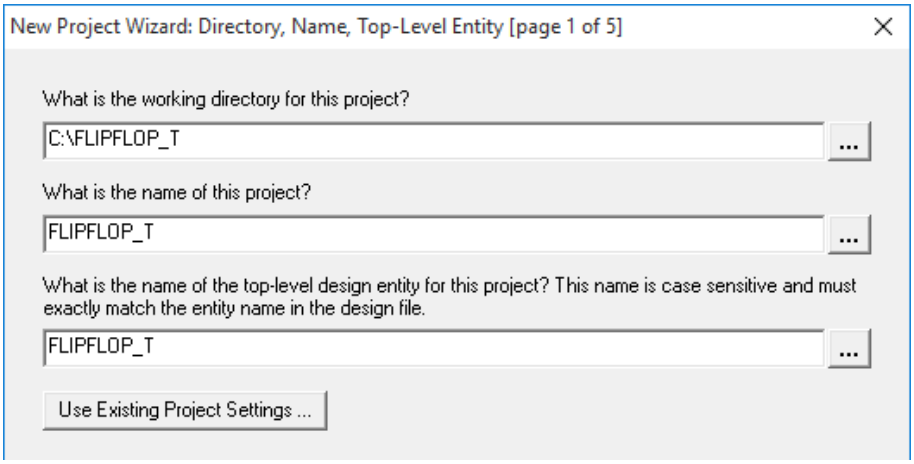
Trong phần này, chúng ta khảo sát flip-flop T dùng trong thiết kế bộ đếm và flip-flop D trong thiết kế thanh ghi. Hai loại flip-flop này hoạt động tại vị trí cạnh lên của xung clock (↑) (mức điện áp chuyển từ thấp lên cao)

1. Flip-flop T

Bảng trạng thái hoạt động

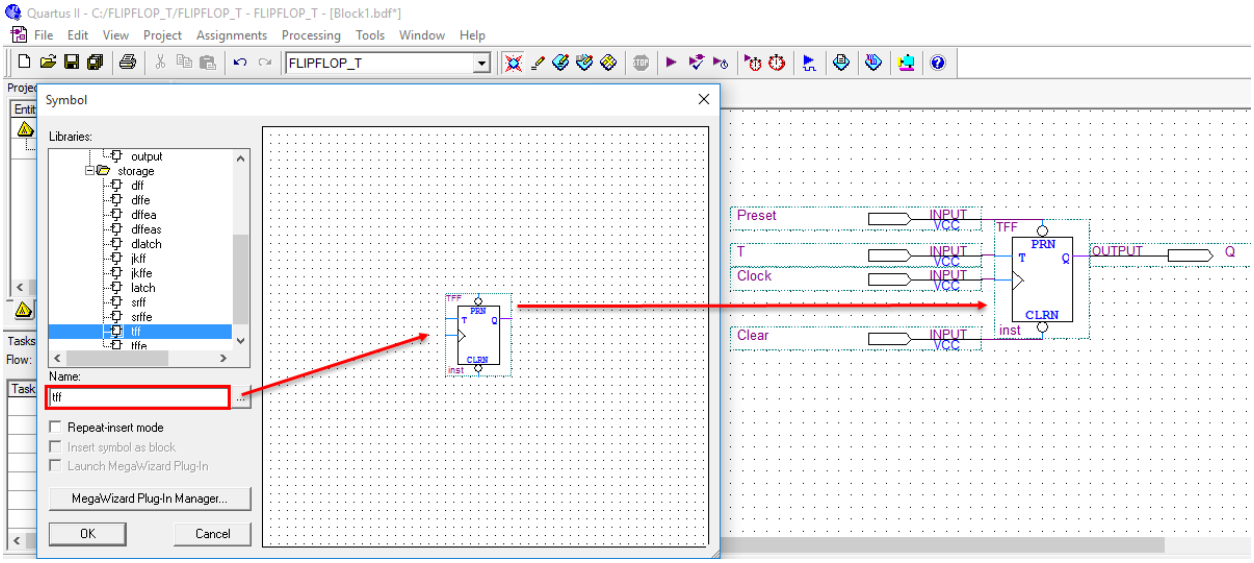
INPUTS				OUTPUTS
Preset	Clear	Clock	T	Q
0	0	x	x	0
0	1	x	x	1
1	0	x	x	0
1	1	↑	1	Đảo trạng thái
1	1	↑	0	Giữ trạng thái

1.1. Tạo project mới có tên: **FLIPFLOP_T**



1.2. Tạo tập tin thiết kế: File → New → Block Diagram/Schematic File

1.3. Thêm khối **tff**, các input và output vào thiết kế, đặt tên như hình bên dưới

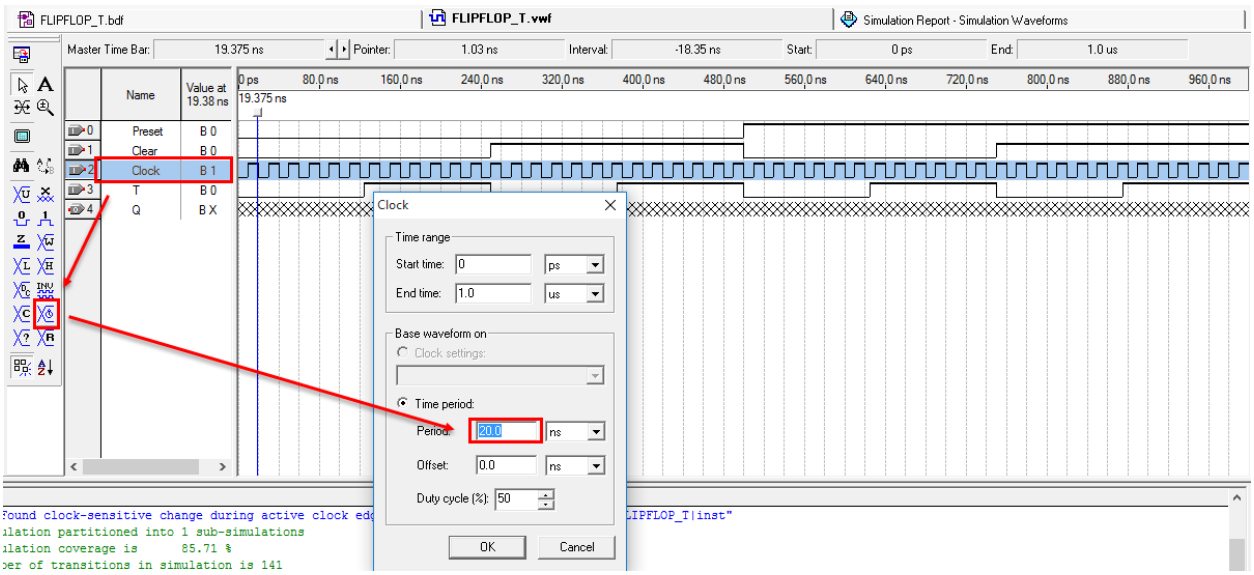


1.4. Lưu lại với tên **FLIPFLOP_T.bdf**

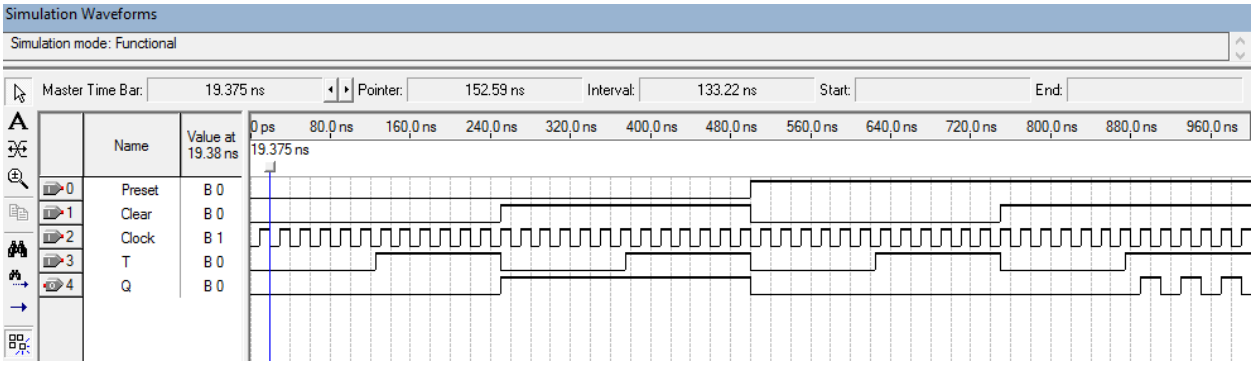
1,5. Biên dịch thiết kế: Processing → Start Compilation

1.6. Tạo tập tin mô phỏng: File → New → Vector Waveform File

1.7. Thêm các ngõ tín hiệu và cài đặt các tham số như hình bên dưới. Lưu lại với tên **FLIPFLOP_T.vwf**



1.8. Chạy mô phỏng, kết quả thể hiện như hình bên dưới

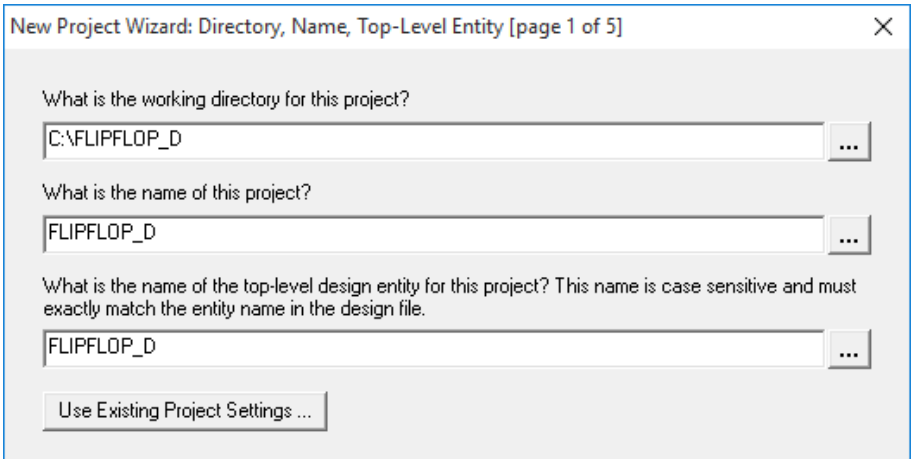


2. Flip-flop D

Bảng trạng thái hoạt động

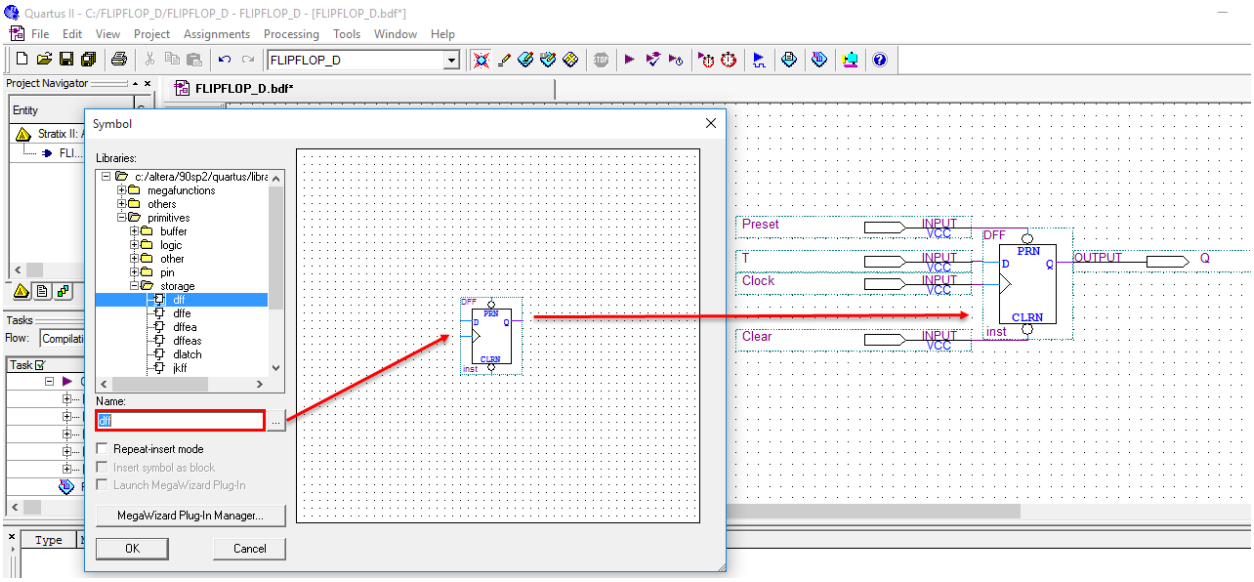
INPUTS				OUTPUTS
Preset	**Clear**	**Clock**	**D**	**Q**
0	0	x	x	0
0	1	x	x	1
1	0	x	x	0
1	1	↑	1	1
1	1	↑	0	0

2.1. Tạo project mới có tên: **FLIPFLOP_D**



2.2. Tạo tập tin thiết kế: File → New → Block Diagram/Schematic File

2.3. Thêm khối **dff**, các input và output vào thiết kế, đặt tên như hình bên dưới

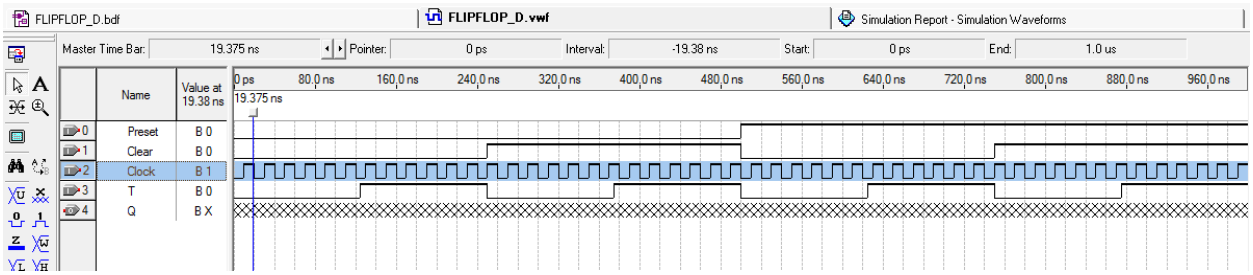


2.4. Lưu lại với tên **FLIPFLOP_D.bdf**

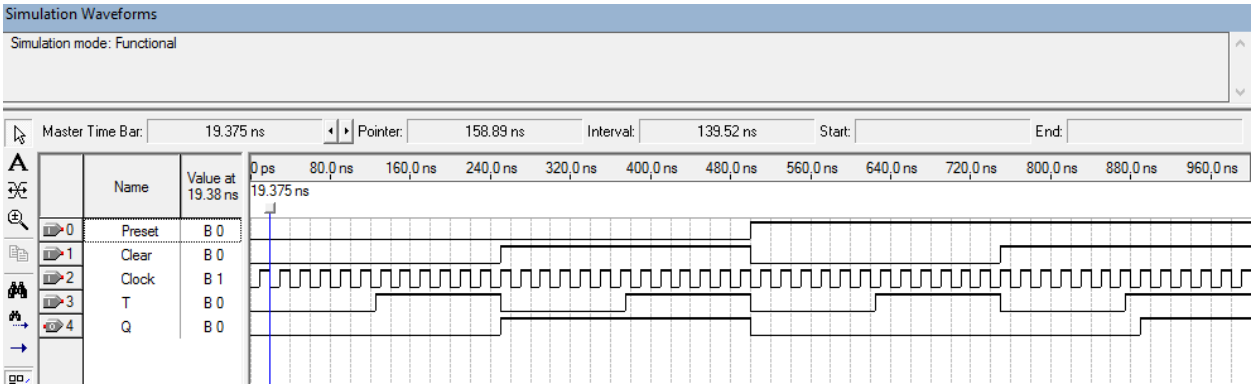
2.5. Biên dịch thiết kế: Processing → Start Compilation

2.6. Tạo tập tin mô phỏng: File → New → Vector Waveform File

2.7. Thêm các ngõ tín hiệu và cài đặt các tham số như hình bên dưới. Lưu lại với tên **FLIPFLOP_D.vwf**

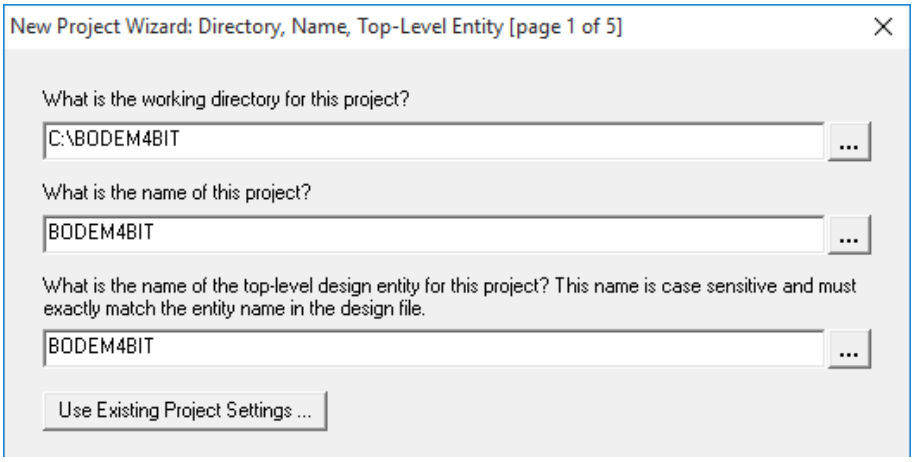


2.8. Chạy mô phỏng, kết quả thể hiện như hình bên dưới

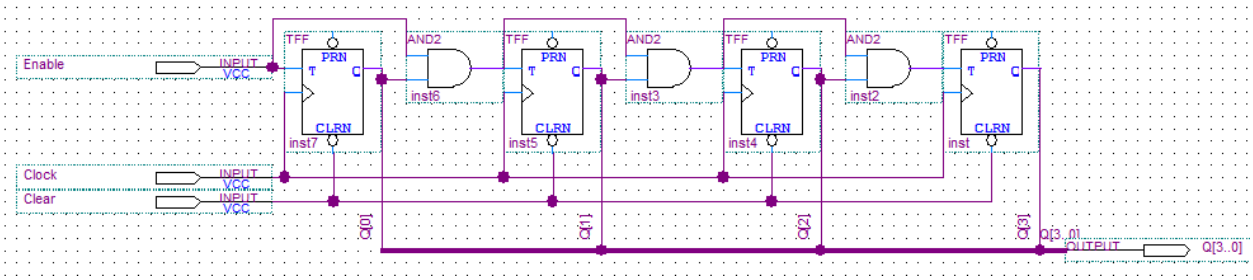


II. Bộ đếm

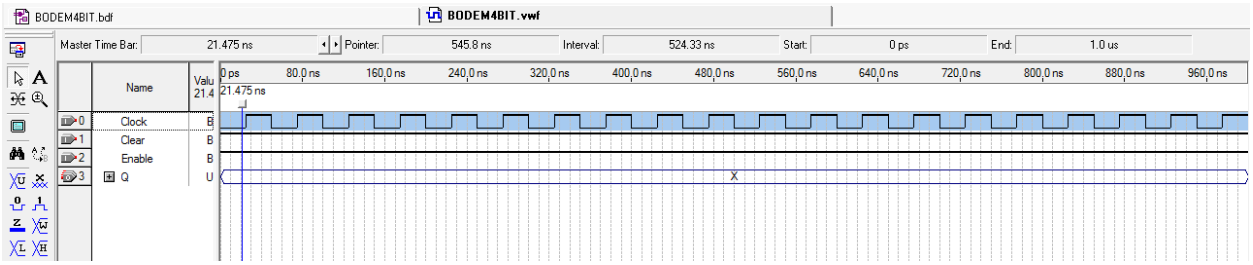
1. Tạo project mới đặt tên: **BODEM4BIT**



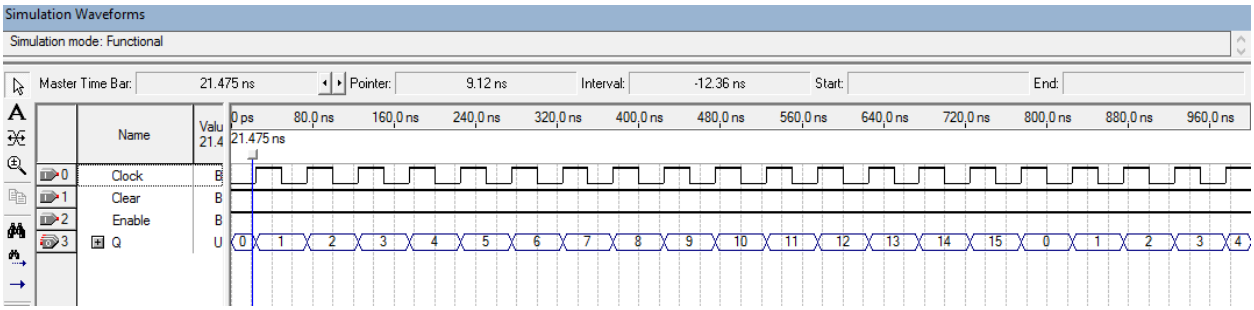
2. Bộ đếm được cấu tạo từ các flip-flop T (**TFF**), do đó để thiết kế một thanh ghi 4 bit ta sẽ dùng 4 flip-flop T ghép lại. Lưu lại với tên: **BODEM4BIT.bdf**



3. Tạo tập tin mô phỏng: File → New → Vector Waveform File. Cài đặt các giá trị tương ứng như hình bên dưới. Lưu lại với tên: **BODEM4BIT.vwf**



5. Chạy mô phỏng

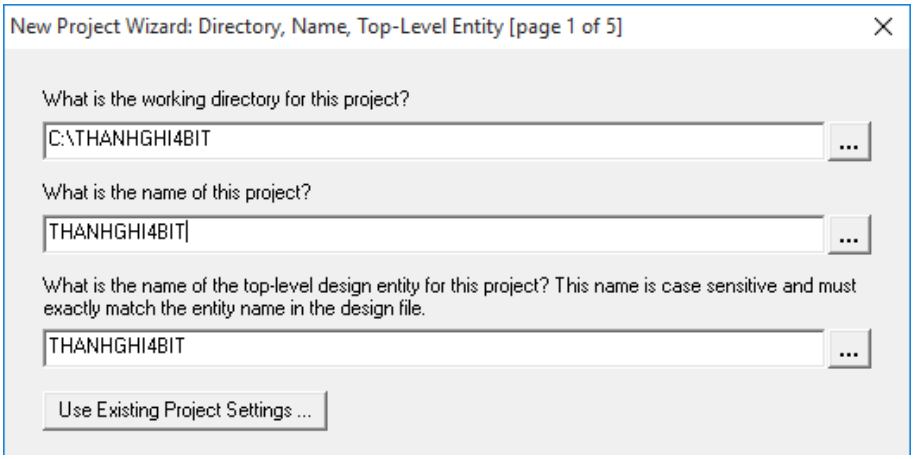


Giải thích: Tại vị trí cạnh lên của xung clock (chuyển từ mức thấp lên mức cao), tăng giá trị của Q lên 1 đơn vị.

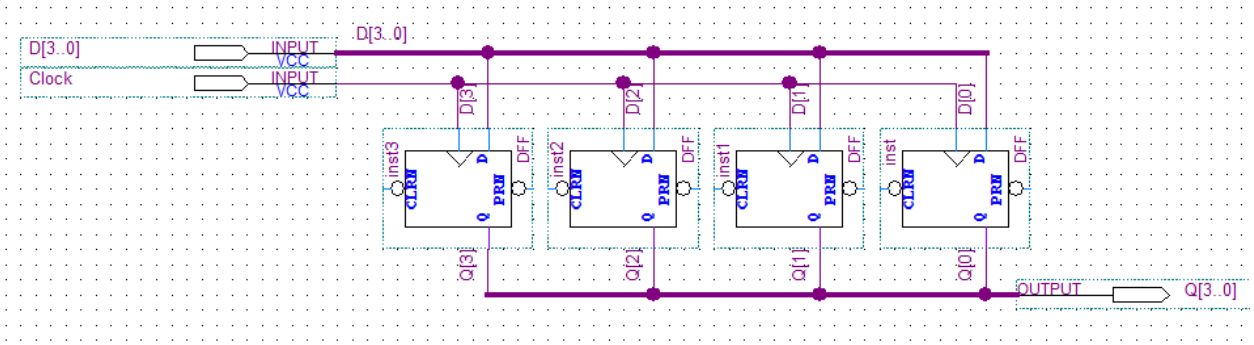
II. Thanh ghi

Trong CPU, ngoài đơn vị xử lý số học (**ALU**) còn có các thanh ghi (register), đơn vị điều khiển (control unit – CU) và bộ nhớ cache

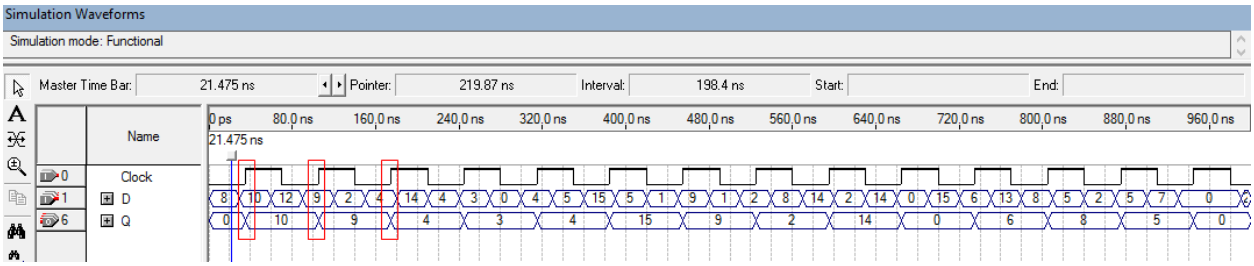
1. Tạo project mới có tên: **THANHGHI4BIT**



2. Thanh ghi được cấu tạo từ các flip-flop D (**DFF**), do đó để thiết kế một thanh ghi 4 bit ta sẽ dùng 4 flip-flop D ghép lại. Lưu lại với tên: **THANHGHI4BIT.bdf**



3. Mô phỏng

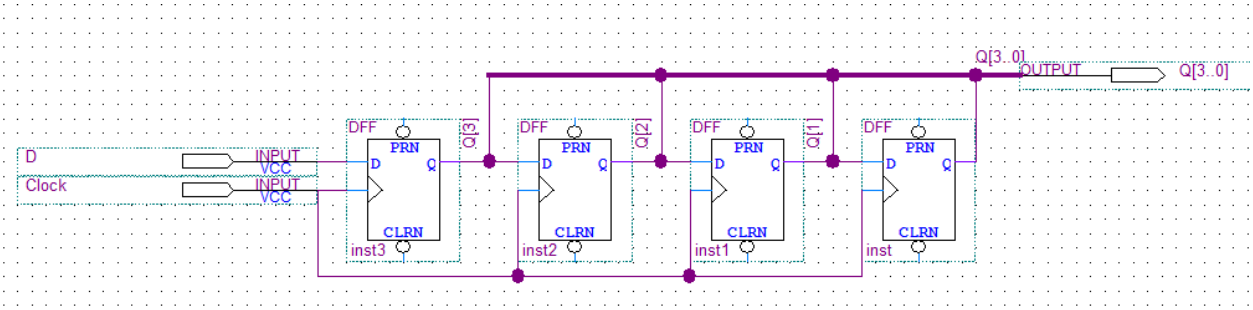


Giải thích: Tại vị trí cạnh lên của xung clock (chuyển từ mức thấp lên mức cao), lấy giá trị của D ghi vào Q

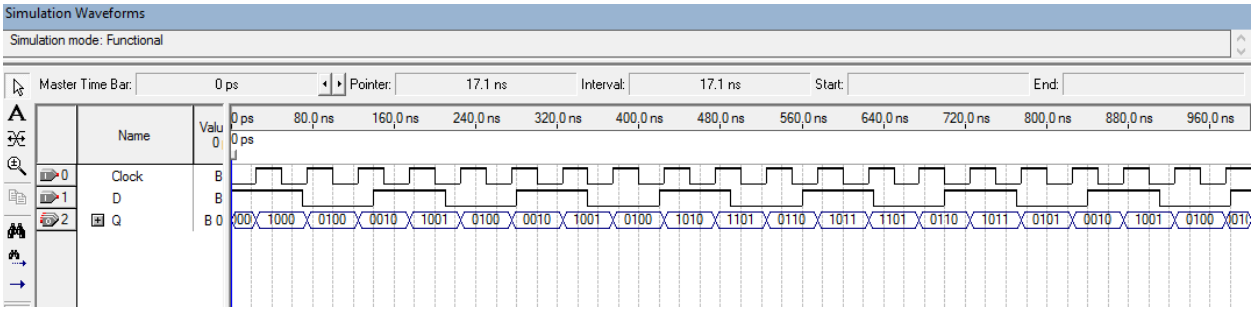
III. Thanh ghi dịch 4 bit

Thiết kế thanh ghi dịch 4 bit dùng flip-flop D ghép lại

1. Mạch **THANHGHIDICH4BIT**



2. Mô phỏng

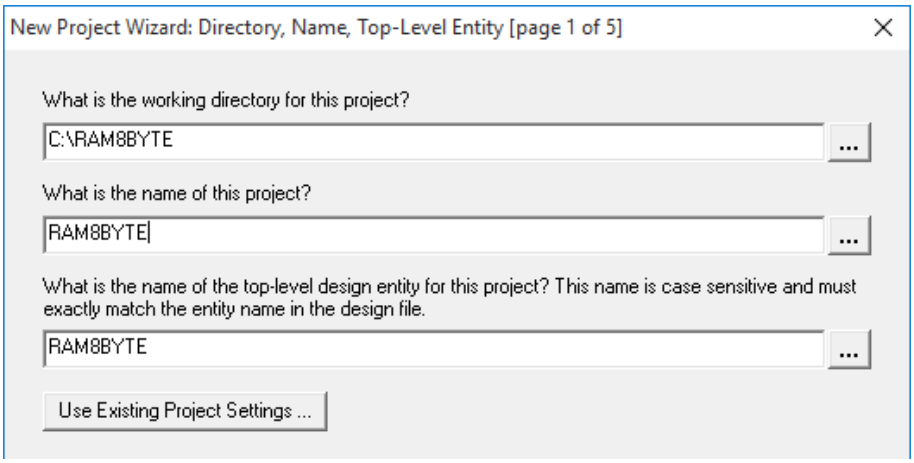


Giải thích: Tại vị trí cạnh lên của xung clock, dịch giá trị của Q qua phải 1 bit và ghi giá trị của D vào bit cao nhất (MSB) của Q

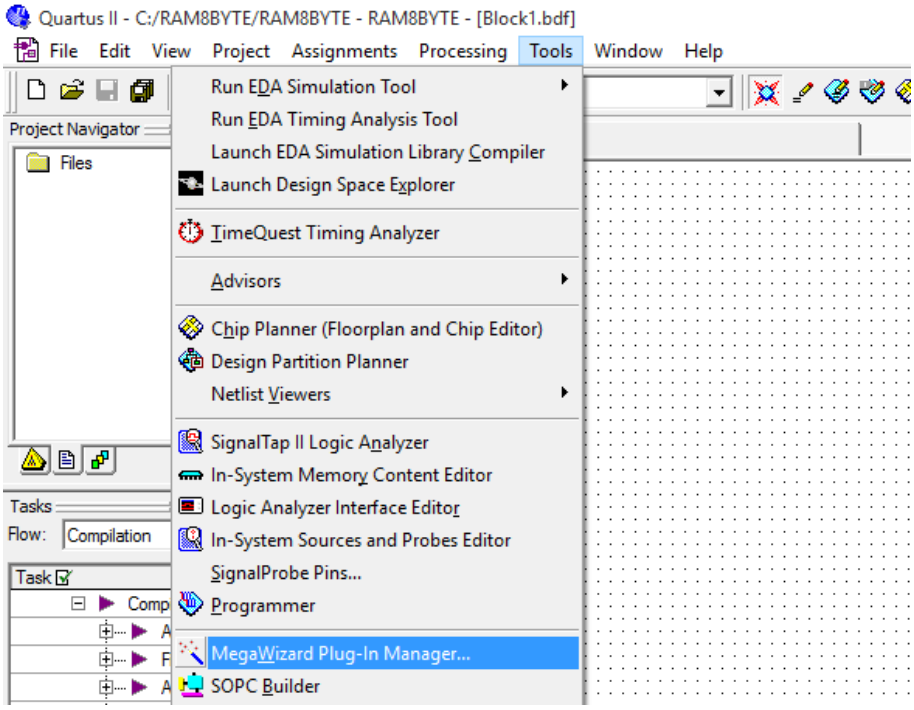
BỘ NHỚ RAM

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.

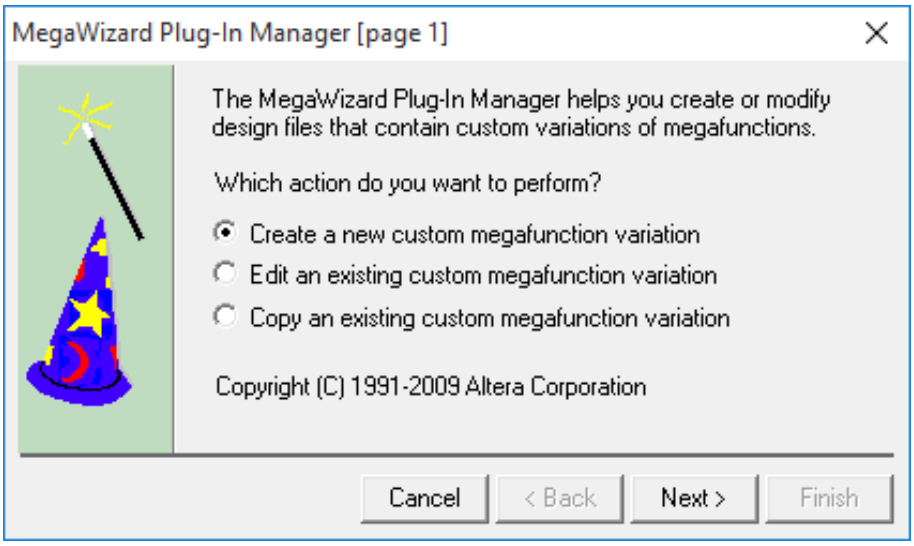
1. Tạo một project mới, đặt tên **RAM8BYTE**



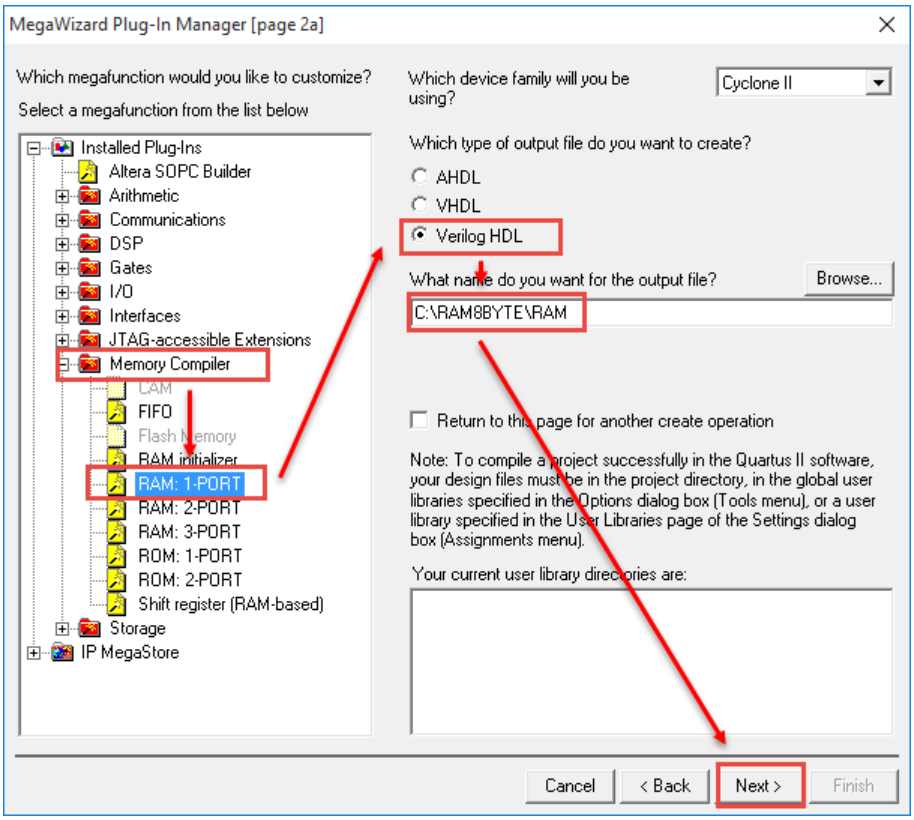
2. Chọn New → Block Diagram/Schematic File để tạo thiết kế mới. Vào Tools → MegaWizard Plug-In Manager.



3. Chọn Next

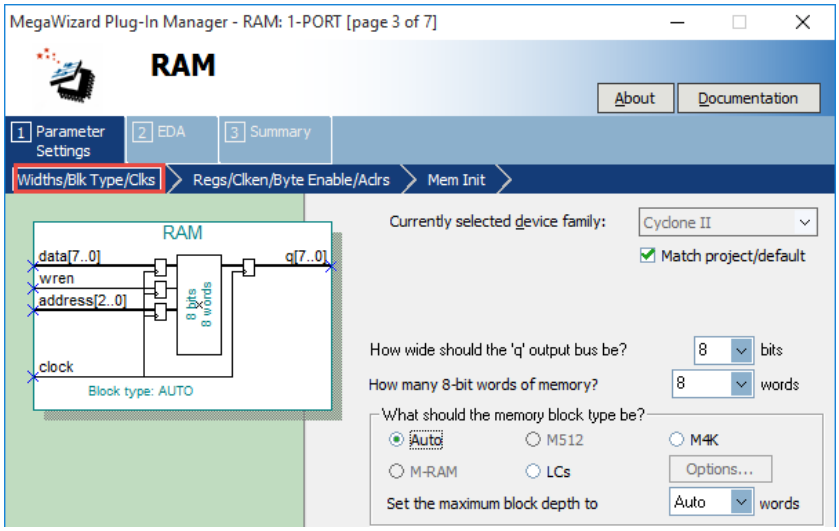


4. Trong Memory Compiler, chọn **RAM 1-PORT**. Đặt tên tập tin: **RAM**. Chọn Next

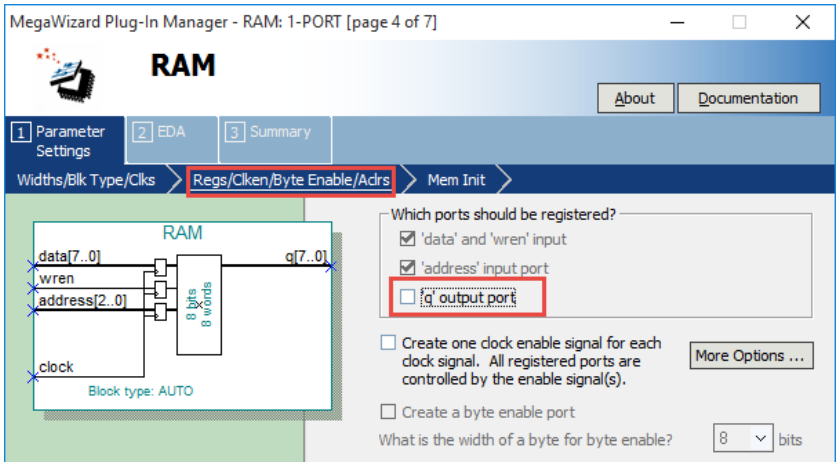


Do bộ nhớ của **RAM** là 8 byte nên độ rộng của bus dữ liệu là 8 bit, bus địa chỉ là 3 bit. Lần lượt thiết lập các tùy chọn như hình bên dưới.

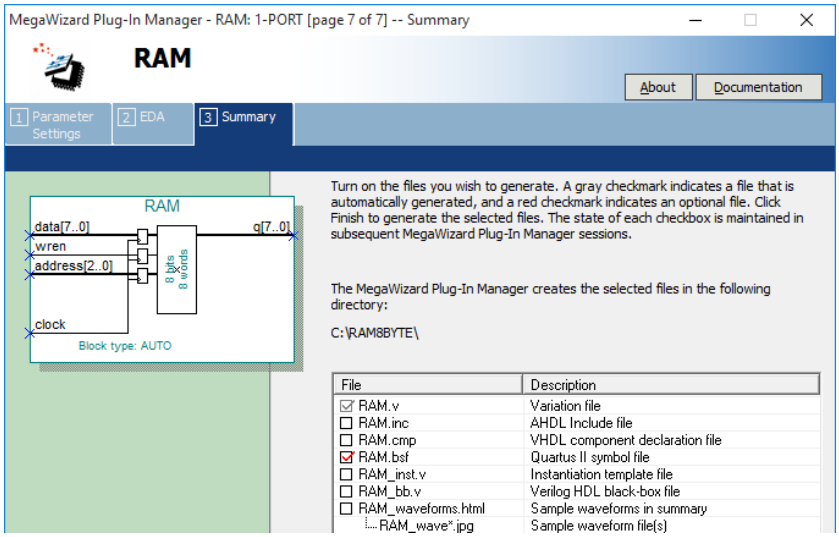
Widths/Blk Type/Clks



Regs/Clken/Byte Enable/Aclrs

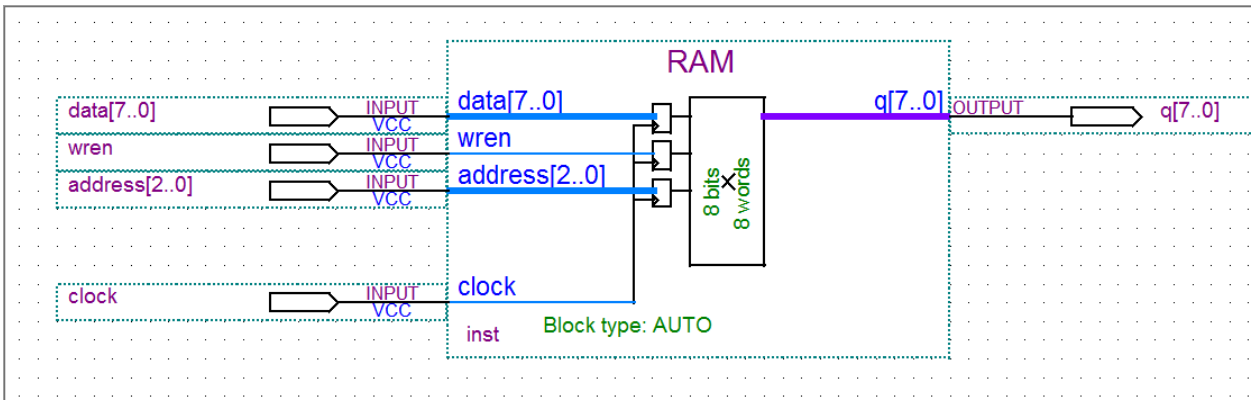


Sumary

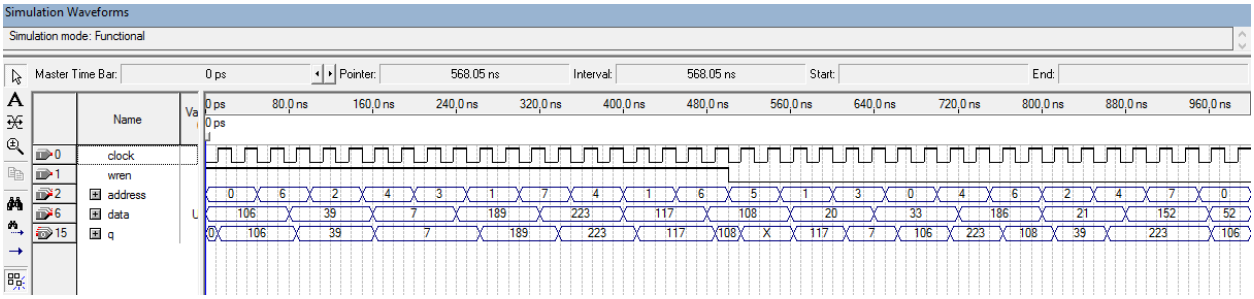


Chọn **Finish**

5. Lấy bộ nhớ RAM vừa tạo ra và gắn các input và output vào. Đặt tên như hình bên dưới. Lưu lại với tên **RAM8BYTE.bdf**



6. Mô phỏng



Giải thích:

- Khi chân tín hiệu wren = 1 → quá trình ghi dữ liệu từ *data* tại địa chỉ *address* vào *q*, khi wren = 0 → quá trình đọc dữ liệu tại địa chỉ *address* vào *q*
- Quá trình đọc/ghi dữ liệu được thực hiện tại vị trí cạnh lên của xung clock (xung clock chuyển từ mức thấp sang mức cao)