TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KỸ THUẬT MÁY TÍNH

LỚP **CE118.N21.2**

BÁO CÁO LAB 4

THIẾT KẾ VI XỬ LÝ ĐƠN GIẢN

MSSV: 21521810

HỌ TÊN: NGUYỄN QUỐC TRƯỜNG AN

I. LÝ THUYẾT

Một vi xử lý đơn giản sẽ bao gồm 2 khối chính đó là Controller và Datapath.

- Controller có nhiệm vụ điều khiển đường đi của dữ liệu (điều khiển việc đọc ghi của Register File, …) và điều kiển việc thực hiện tính toán (điều khiển Opcode của khối ALU, …) trong khối Datapath.
- Datapath chứa các khối cần thiết để thực hiện việc tính toán (Register File, ALU, Bộ dịch, …) được điều khiển bởi Controller.

II. THỰC HÀNH

Sinh viên thực hiện thiết kế một vi xử lý đơn giản dùng để tính toán biểu thức sau:

$\mathbf{D_3I_3 + D_2I_2 – D_1I_1 + D_0I_0}$

Trong đó:

- D_x là 4 ký số cuối của MSSV (Ví dụ 4 ký số cuối MSSV là 6789 thì $D_3 = 6$, $D_2 = 7$, $D_1 = 8$, $D_0 = 9$)
- I_x là 4 ký số được nhập lần lượt tại ngõ vào (I_x có 4 bit, ***số I_x sẽ được sinh ngẫu nhiên lúc báo cáo***, sinh viên có thể sử dụng chức năng sinh số ngẫu nhiên trong phần mềm mô phỏng quartus để kiểm tra thiết kế)

HIỆN THỰC THIẾT KẾ

I. Tổng quan yêu cầu và các khối cần thiết kế

1) Các bước giải quyết bài toán

-**Mã số sinh viên:** 21521810 => Biểu thức cần thực hiện: $\mathbf{1I_3 + 8I_2 – 1I_1 + 0I_0}$

a) Các bước thực hiện tính toán:

Bước	Công việc	Trạng thái
1	Data0 ← I0	S0
2	Data1 ← I1	S1
3	Data2 ← I2	S2
4	Data3 ← I3	S3
5	Data0 ← Data0 * 0	S4
6	Data1 ← Data1	S5
7	Data2 ← Data2 << 3	S6
8	Data3 ← Data3	S7
9	Sum ← Data3 + Data2	S8
10	Temp ← Sum – Data1	S9
11	Sum ← Temp + Data0	S10
12	Output ← Sum	S11

Bảng 1 – Các bước thực hiện để tính $\mathbf{1I_3 + 8I_2 – 1I_1 + 0I_0}$

b) Xác định các khối cần thiết để thực hiện tính toán

-Từ bảng 1 ta thấy:

+Ta cần lưu trữ các biến Data0, Data1, Data2, Data3, Temp, Sum. Theo như lý thuyết ta cần Register-File có 8 thanh ghi 8-bit, tuy nhiên, ta có thể thấy các bước thực hiện bài toán phía trên, các biến không đồng thời được lưu và sử dụng tới nên khi biến nào không còn dùng nữa ta có thể ghi đè giá trị của biến khác lên nó. Vì vậy thực chất ta chỉ cần **Register-File có 4 thanh ghi 8-bit**.

+Để tính toán ta cần có **ALU** thực hiện các chức năng: x0, << 0 (x1), << 3 (x8), + (cộng), - (trừ)

+Có 2 nguồn gán giá trị cho biến (một là từ đầu vào, hai là từ kết quả của các phép tính) nên ta cần có bộ **MUX21 8-bit**.

+Để lưu trữ kết quả ta cần có 1 **Register 8-bit** và một **cổng Tri-state** để cho phép xuất kết quả output khi đã thực hiện xong việc tính toán biểu thức.

+Ngoài các bộ phận trên của khối DATAPATH, ta cần thiết kế thêm khối **CONTROLLER** để điều khiển DATAPATH thực hiện đúng yêu cầu tính toán.

II. Thiết kế DATAPATH

a) Thiết kế MUX21 8-bit

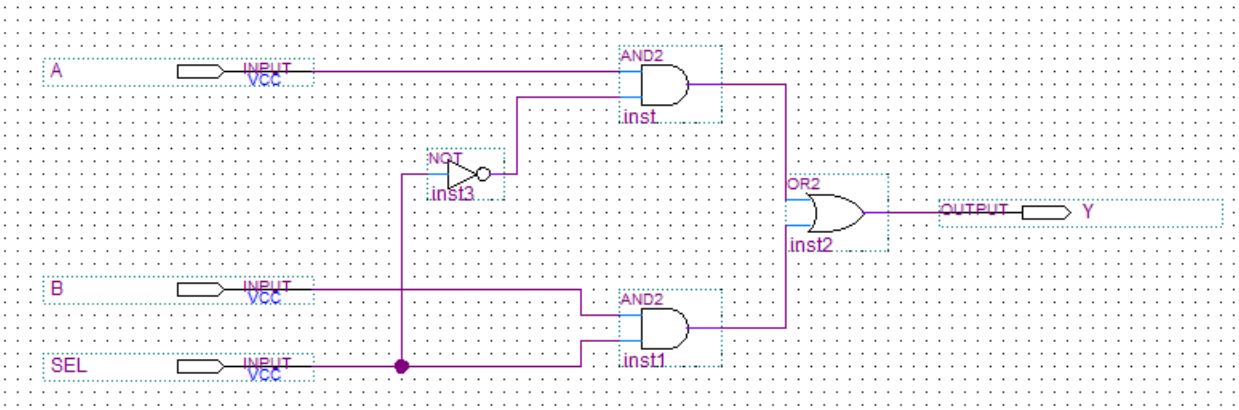
-Để tạo ra MUX21 8-bit trước tiên ta cần thiết kế MUX21 1-bit:

+Bảng sự thật MUX21 1-bit:

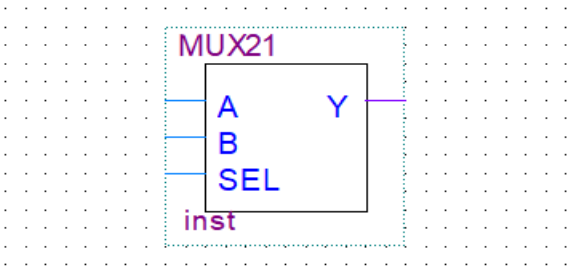
Select	Output
0	I0
1	I1

Bảng 2 – Bảng sự thật cổng MUX21

+Schematic MUX21 1-bit:

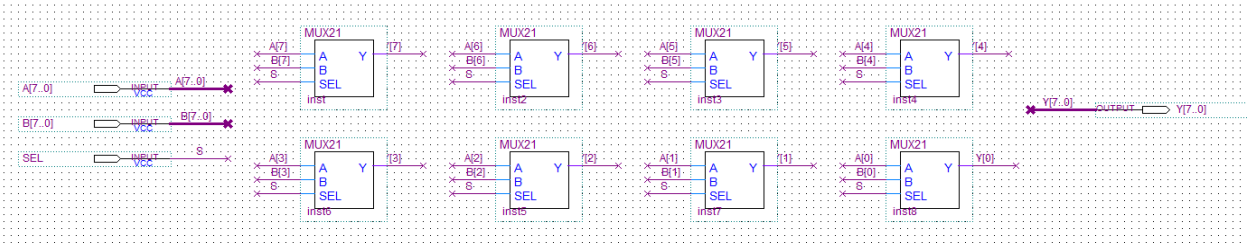


+Đóng gói MUX21 1-bit:

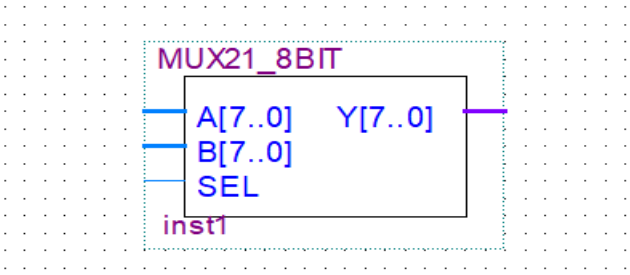


-Ta tiến hành ghép nối 8 MUX21 1-bit sẽ tạo thành MUX21 8-bit:

+Schematic MUX21 8-bit:



+Đóng gói MUX21 8-bit:



-Giải thích thiết kế MUX21 8-bit:

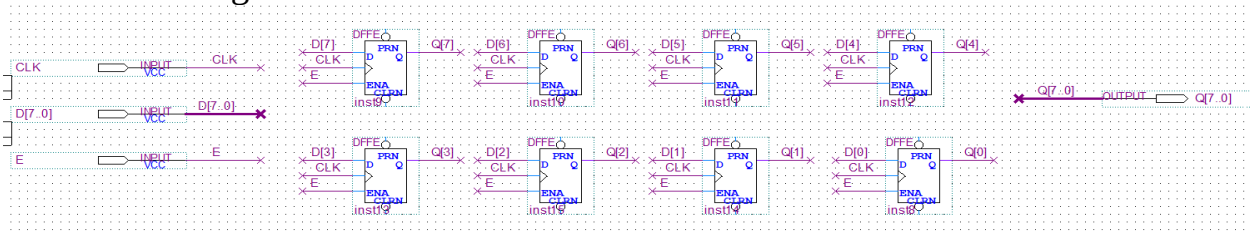
+Tín hiệu SEL dùng để lựa chọn một trong hai ngõ vào A hoặc B

+Nếu SEL = 0, Y = A, ngược lại SEL = 1 thì Y = B.

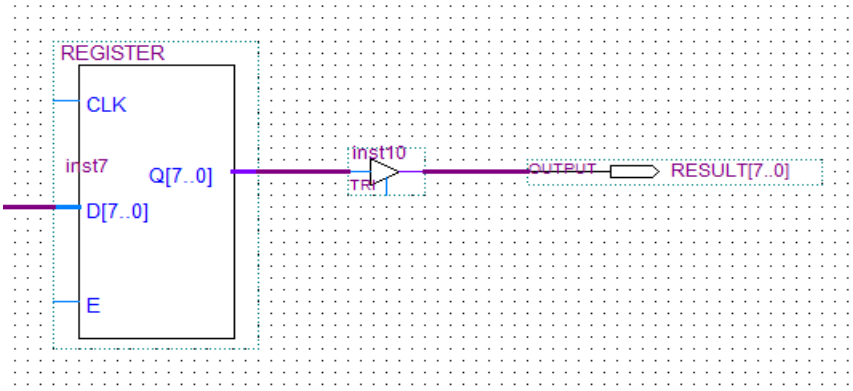
b) Thiết kế Register 8-bit cùng với cổng Tri-state

-Ta thấy để lưu trữ 8-bit cần có 8 D-Flipflop kết nối với 8-bit của dữ liệu cần lưu. Để cho phép xuất ngõ ra khi có tín hiệu cho phép cần cổng Tri-state nối với ngõ ra của Register 8-bit.

+Schematic Register 8-bit:



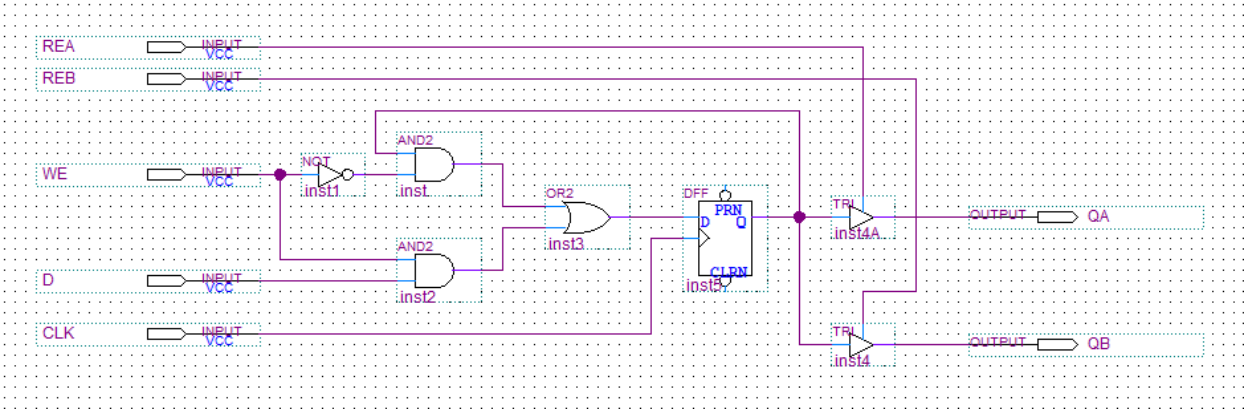
+Đóng gói Register 8-bit và kết nối với Tri-state:



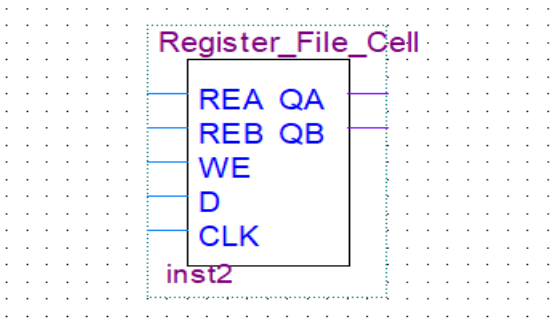
-Giải thích thiết kế Register 8-bit:
+Tín hiệu CLK là tín hiệu xung clock cấp cho thanh ghi hoạt động.
+D là dữ liệu ngõ vào để thực hiện ghi.
+E là tính hiệu enable của Register.
+Ngõ ra của Register được nối tiếp với cổng Tri-state có chức năng cho phép xuất kết quả, nếu tín hiệu enable Tri-state = 0, ngõ ra là Z, ngược lại tín hiệu enable Tri-state = 1 thì tín hiệu ngõ ra được phép đi qua xuất ra ngoài.

c) Thiết kế Register-File có 4 thanh ghi 8-bit

-Ta có thể xem cấu tạo của Register-File là một mảng 1 chiều của các thanh ghi, các thanh ghi được truy cập thông qua địa chỉ.
-Ở đây ta cần thiết kế Register-File với 4 thanh ghi 8-bit với 2 port đọc và 1 port ghi. Vậy ta cần 4 thanh ghi Register 8-bit, 2 bộ giải mã địa chỉ 2-to-4 dành cho 2 port đọc, một bộ giải mã địa chỉ 2-to4 dành cho 1 port ghi.
-Ta có thiết kế của một Register-File Cell như sau:
+Schematic Register-File Cell:

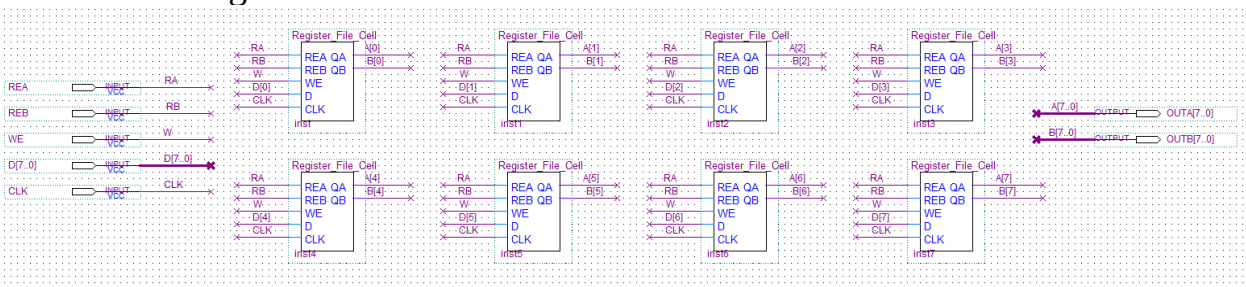


+Đóng gói Register-File Cell:

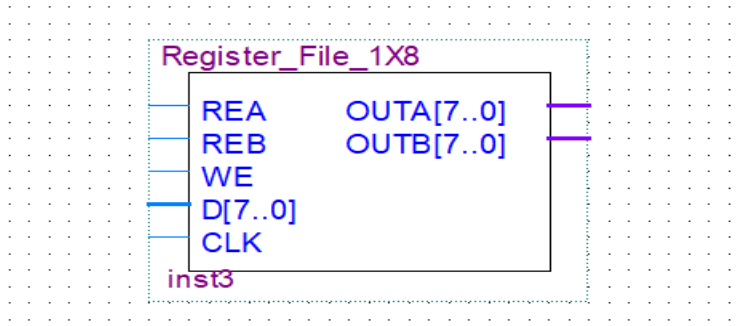


-Từ 8 Register-File Cell ta có thể tạo ra một thanh ghi 8-bit trong Register-File 4 thanh ghi 8-bit:

+Schematic Register-File 1x8:

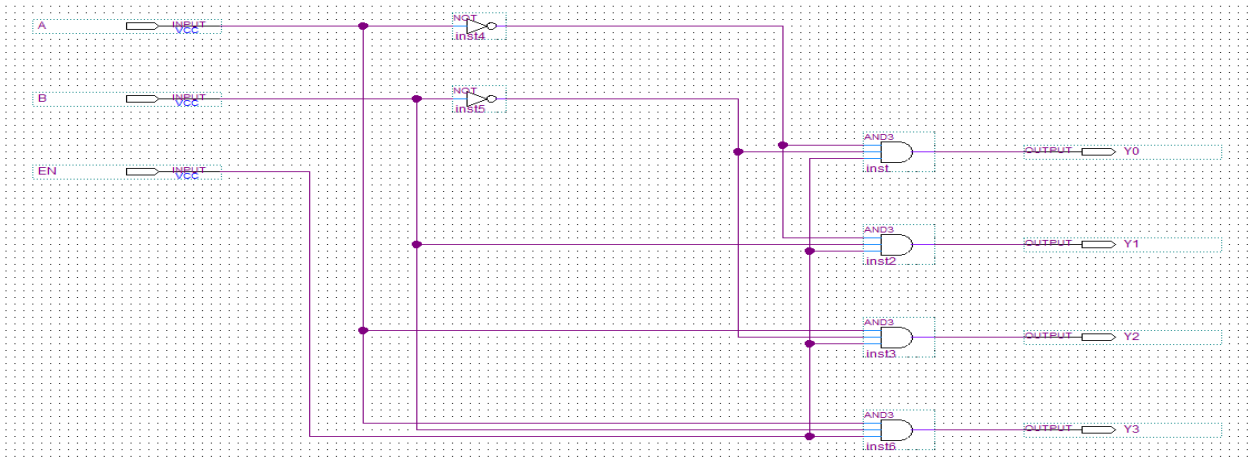


+Đóng gói Register-File 1x8:

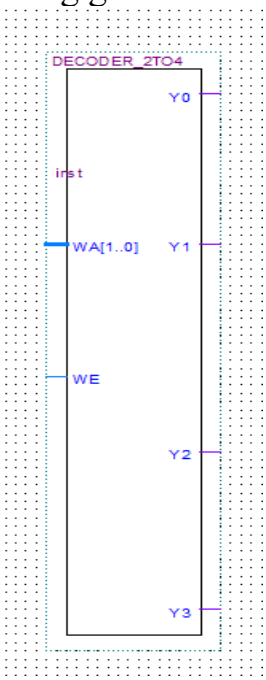


-Ta có thiết kế của bộ giải mã địa chỉ Decoder 2-to-4 như sau:

+Schematic Decoder 2-to-4:

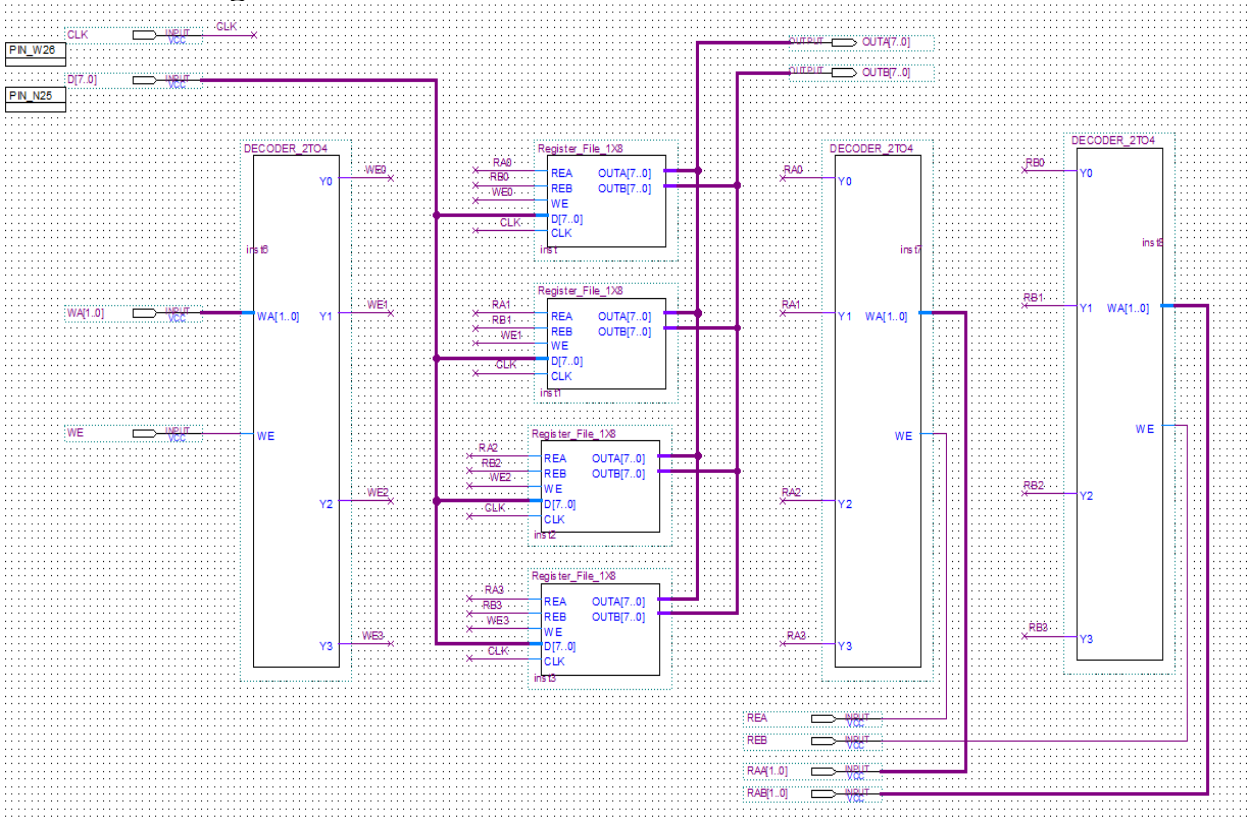


+Đóng gói Decoder 2-to-4:

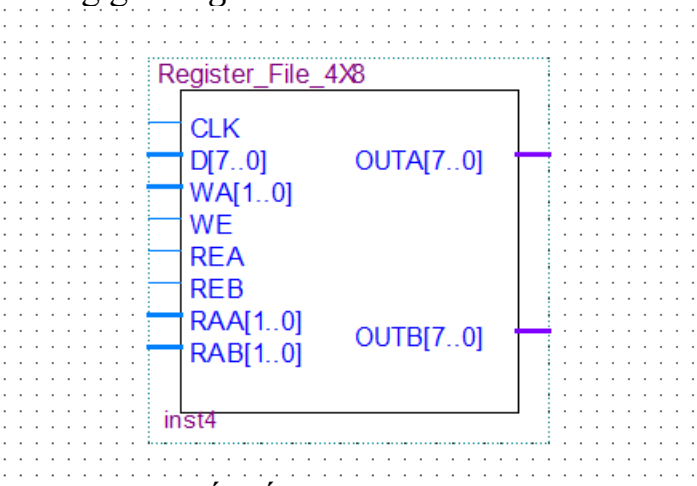


-Từ 4 khối Register-File 1 thanh ghi 8-bit kết hợp với 3 khối giải mã địa chỉ Decoder 2-to-4 ta có thể tạo ra 1 khối Register-File 4 thanh ghi 8-bit:

+Schematic Register-File 4x8:



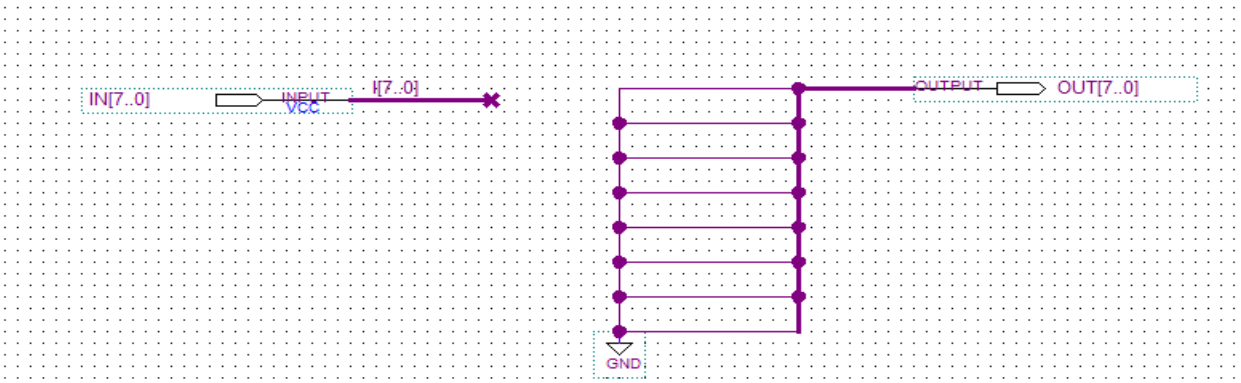
+Đóng gói Register-File 4x8:



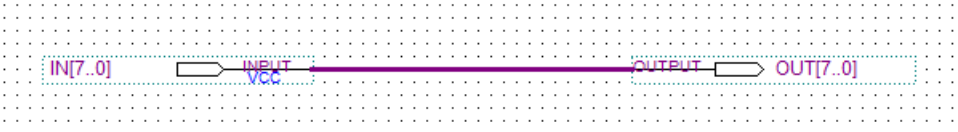
-Giải thích thiết kế Register-File 4x8:
+Ta có CLK được cung cấp để Register-File hoạt động.
+WE/WA: cho phép ghi/địa chỉ ghi
+REA/RAA: cho phép đọc port A/địa chỉ đọc port A
+REB/RAB: cho phép đọc port B/địa chỉ đọc port B
+D/OUTA/OUTB: dữ liệu vào/dữ liệu ra port A/dữ liệu ra port B

d) Thiết kế ALU

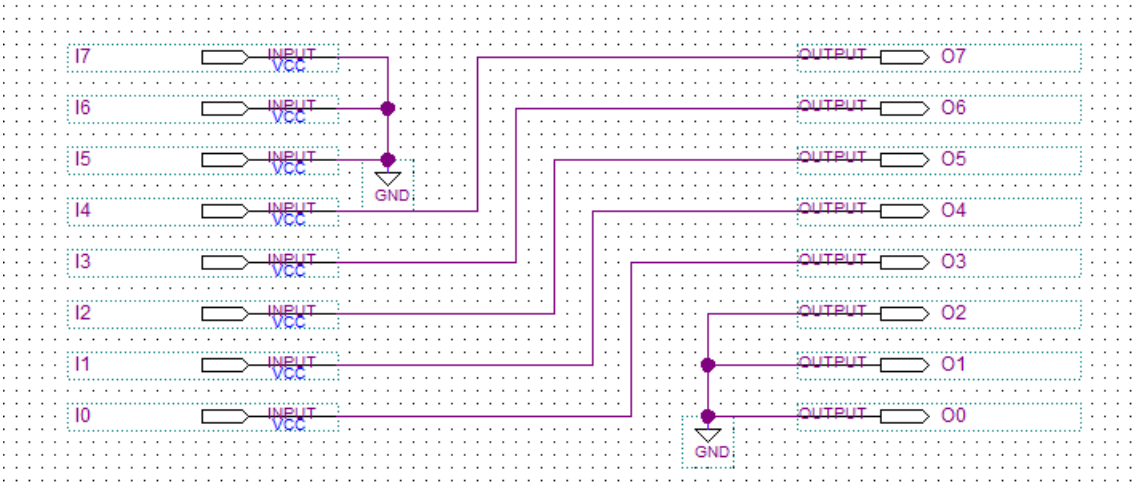
-Khối ALU phục vụ tính toán trong bài này có chức năng tính cộng, trừ, nhân 0, << 0 (không dịch), << 3
*Ta đi vào chi tiết thiết kế từng chức năng:
-Bộ hỗ trợ nhân 0, << 0, << 3:
+Schematic nhân 0:



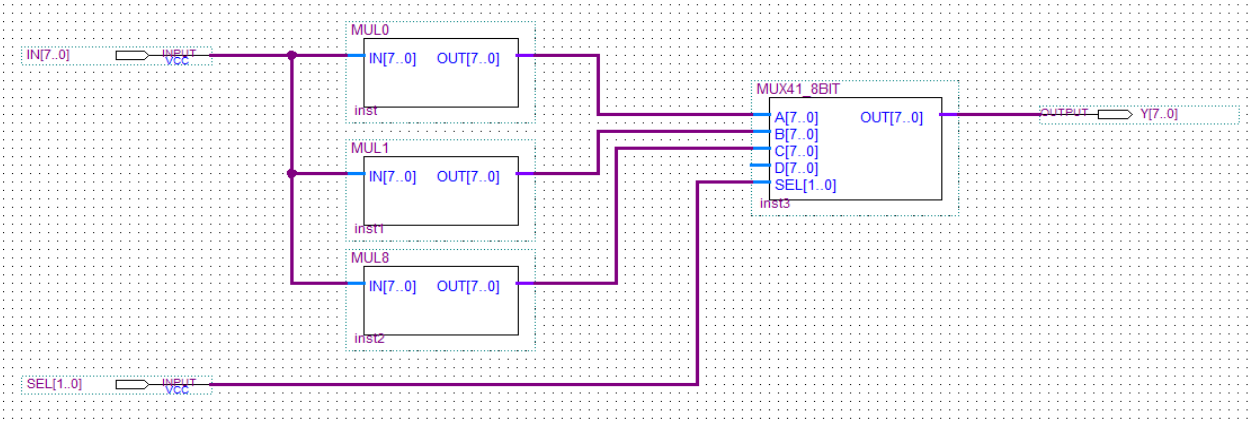
+Schematic << 0:



+Schematic << 3:



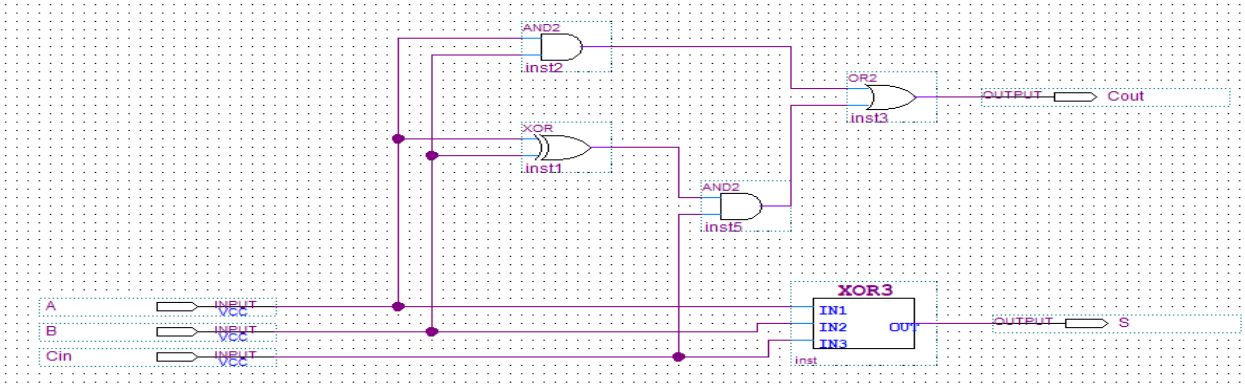
+Schematic bộ nhân 0, << 0, << 3:



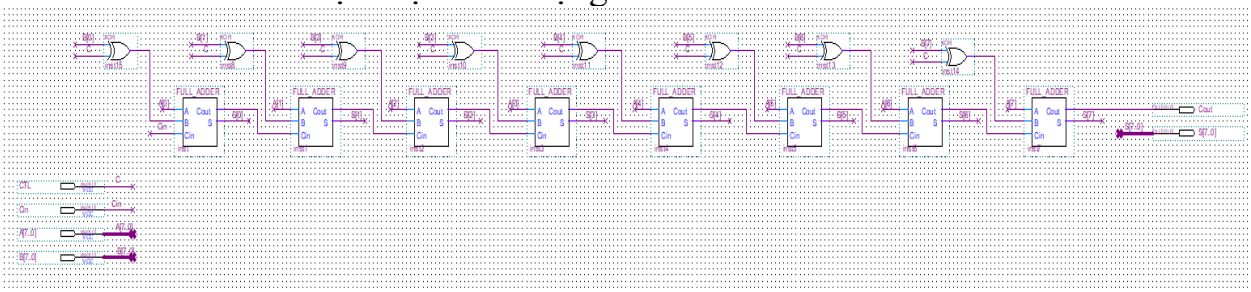
-Bộ hỗ trợ cộng và trừ:

Để thực hiện thiết kế khối chức năng này, trước tiên ta cần thiết kế khối Full-Adder 1-bit.

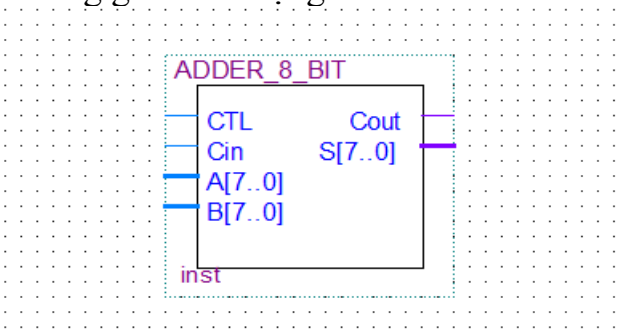
+Schematic Full-Adder 1-bit:



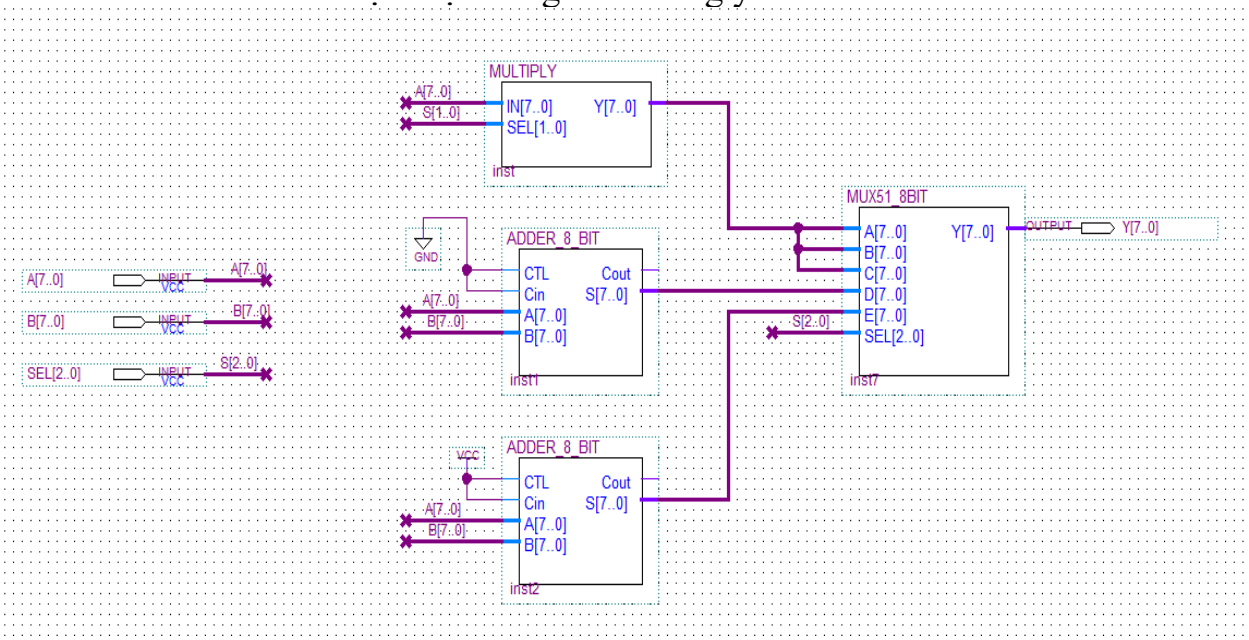
+Để thực hiện khối Adder 8-bit và có thể tái sử dụng cho chức năng của bộ Trừ, ta kết hợp thêm cổng xor ở ngõ vào B của mỗi Full-Adder và dùng 8 Full-Adder 1-bit để thực hiện khối Cộng/Trừ 8-bit:



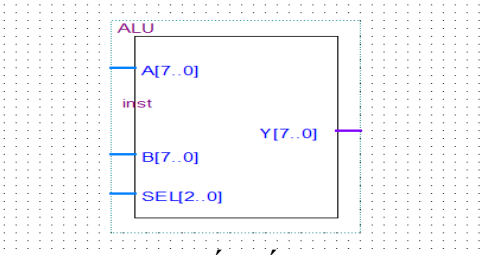
+Đóng gói khối Cộng/Trừ 8-bit:



-Từ thiết kế của các khối thành phần, ta kết nối chúng lại với nhau tạo thành khối ALU hoàn chỉnh thực hiện đúng chức năng yêu cầu của bài toán:



+Đóng gói ALU:



-Giải thích thiết kế ALU:

+A, B: là 2 ngõ vào của ALU

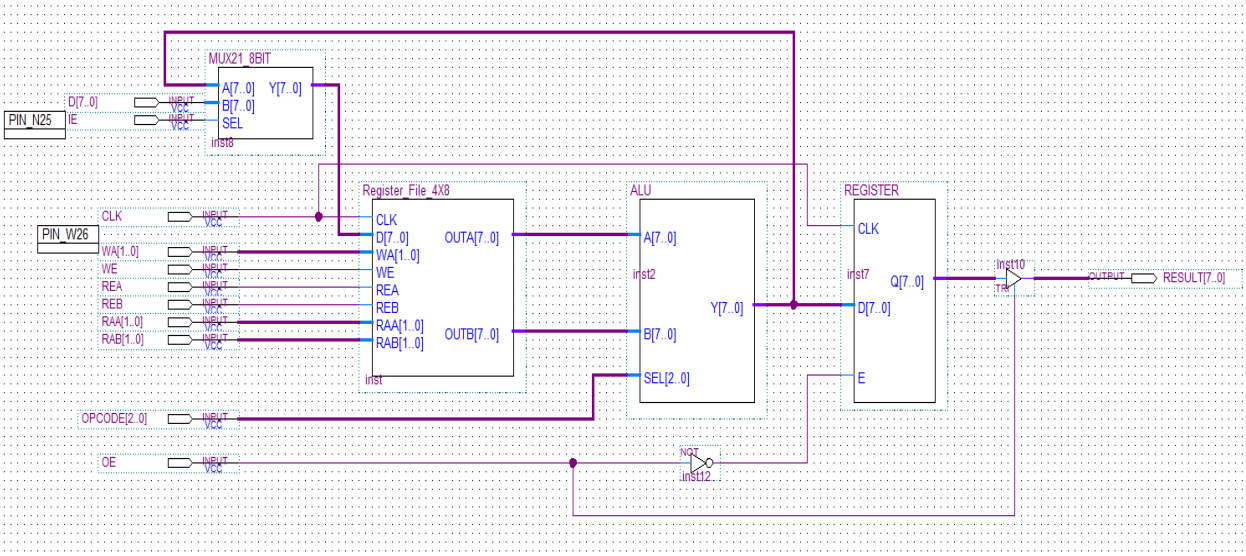
+Y: là kết quả đầu ra

+SEL là tín hiệu lựa chọn chức năng của ALU với 000 ứng với nhân 0, 001 ứng với << 0 (không dịch), 010 ứng với << 3, 011 ứng với + (cộng), 100 ứng với – (trừ).

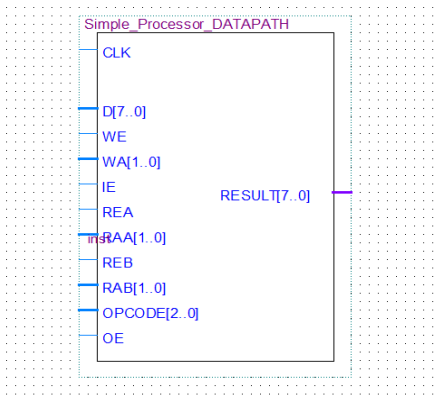
e) Kết nối các khối tạo thành DATAPATH

-Khi đã có đầy đủ các khối thành phần của DATAPATH, ta thực hiện kết nối các khối lại với nhau sao cho phù hợp với yêu cầu tính toán:

+Schematic DATAPATH:



+Đóng gói DATAPATH:

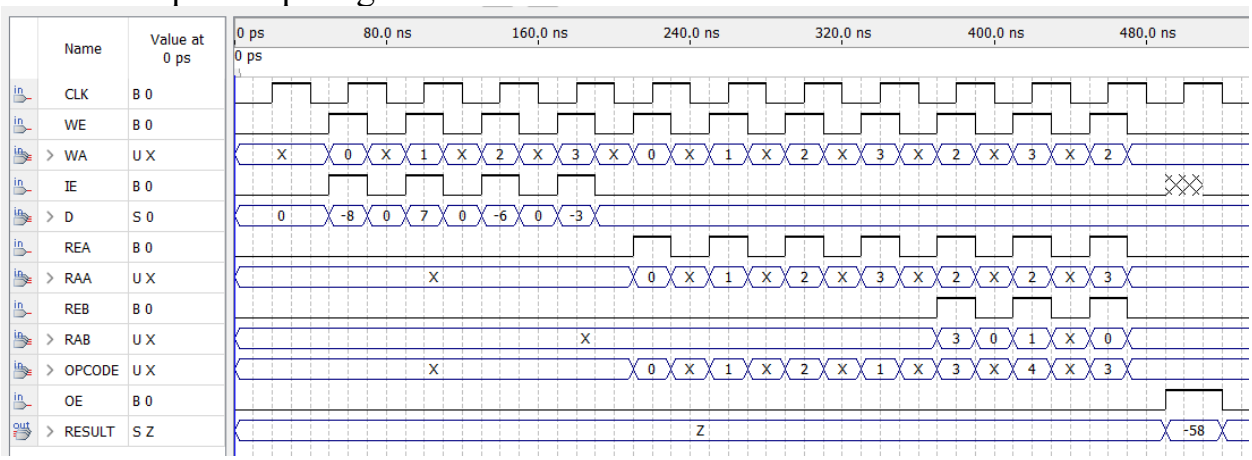


-Giải thích hoạt động DATAPATH:
+CLK: clock cung cấp để hoạt động
+D: dữ liệu vào
+WE/WA: tín hiệu cho phép ghi/địa chỉ ghi
+REA/RAA: cho phép đọc port A/địa chỉ đọc port A
+REB/RAB: cho phép đọc port B/địa chỉ đọc port B
+IE: tín hiệu lựa chọn dữ liệu từ đầu vào hoặc từ kết quả tính toán của ALU
+OPCODE: tín hiệu điều khiển lựa chọn chức năng tính toán
+OE: tính hiệu cho phép xuất output.

f) Mô phỏng và kiểm thử chức năng khối DATAPATH

-Sau khi kết nối và hoàn thiện thiết kế DATAPATH, ta cần mô phỏng kiểm tra xem DATAPATH có thực sự chạy đúng với chức năng ta muốn bằng các set các tín hiệu điều khiển tương ứng với các bước tính toán được trình bày ở trên

-Ta có kết quả mô phỏng kiểm tra:

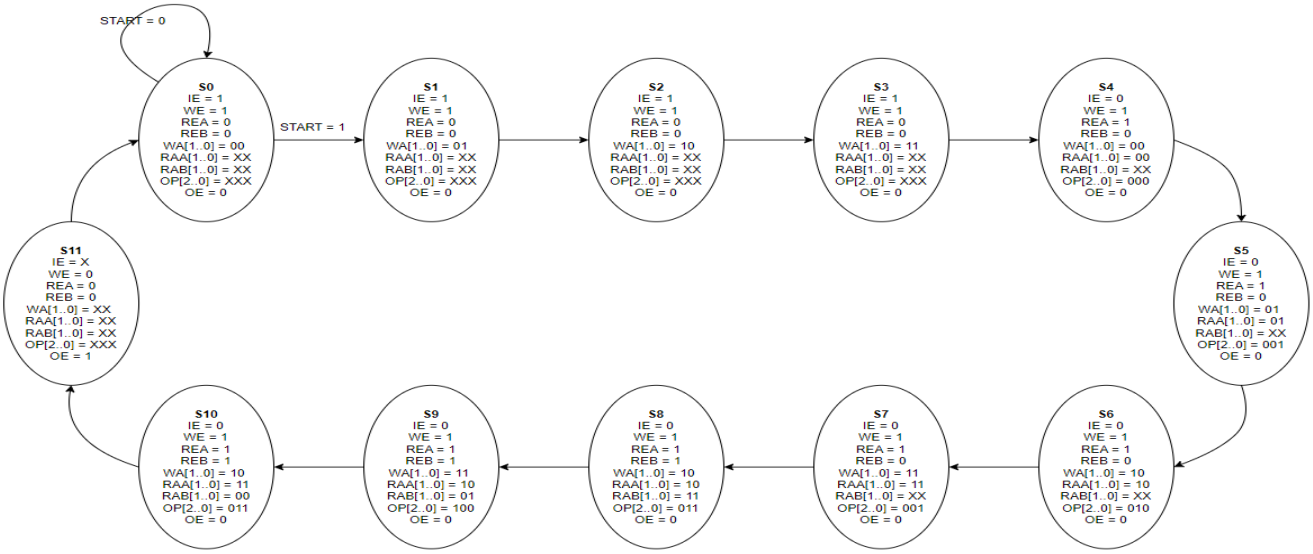


-Phân tích: Ta nhập vào 4 số input lần lượt là **-8, 7, -6, -3** và kết quả mô phỏng nhận được là **-58**, có tín hiệu báo **DONE** khi hoàn thành vào cho phép xuất ra kết quả. Kết quả -58 trên bằng đúng với kết quả thực hiện tính toán theo biểu thức cần tính: $\mathbf{1I_3 + 8I_2 - 1I_1 + 0I_0}$

III. Thiết kết CONTROLLER

a) Sơ đồ chuyển trạng thái

-Từ các bước làm phân tích bài toán ở trên, ta có sơ đồ chuyển trạng thái như sau:



-Để tăng tính tương tác với người dùng và dễ dàng điều khiển hệ thống khi vận hành, ta thêm tín hiệu START cho hệ thống, khi START = 0 và hệ thống đang ở S0 thì không thực hiện tính toán, ngược lại khi START = 1 và hệ thống đang ở S0 thì bắt đầu đi vào việc tính toán.

b) Bảng chuyển trạng thái

-Dựa vào việc phân tích các bước tính toán biểu thức vào sơ đồ chuyển trạng thái, ta có bảng chuyển trạng thái như sau:

Trạng thái hiện tại	Trạng thái kế tiếp	
	START = 0	START = 1
S0	S0	S1
S1	S2	S2
S2	S3	S3
S3	S4	S4
S4	S5	S5
S5	S6	S6
S6	S7	S7
S7	S8	S8
S8	S9	S9
S9	S10	S10
S10	S11	S11
S11	S0	S0

Bảng 3 – Bảng chuyển trạng thái

-Phân tích:
+Khi ở trạng thái S0, nếu tín hiệu START = 0 thì không chuyển trạng thái, ngược lại START = 1 thì thực hiện chuyển trạng thái và tính toán
+Khi ở trạng thái khác S0 thì hệ thống tự chuyển trạng thái không còn phụ thuộc vào tín hiệu START

c) Mã hóa các trạng thái

-Để thực hiện chuyển trạng thái, ta cần mã hóa các trạng thái tương ứng.
-Ta thấy có 12 trạng thái, vậy nên cần 4-bit để mã hóa
-Ta lựa chọn phương pháp mã hóa theo số đếm như bảng bên dưới:

Trạng thái	Mã hóa
S0	0000
S1	0001
S2	0010
S3	0011
S4	0100
S5	0101
S6	0110
S7	0111
S8	1000
S9	1001
S10	1010
S11	1011

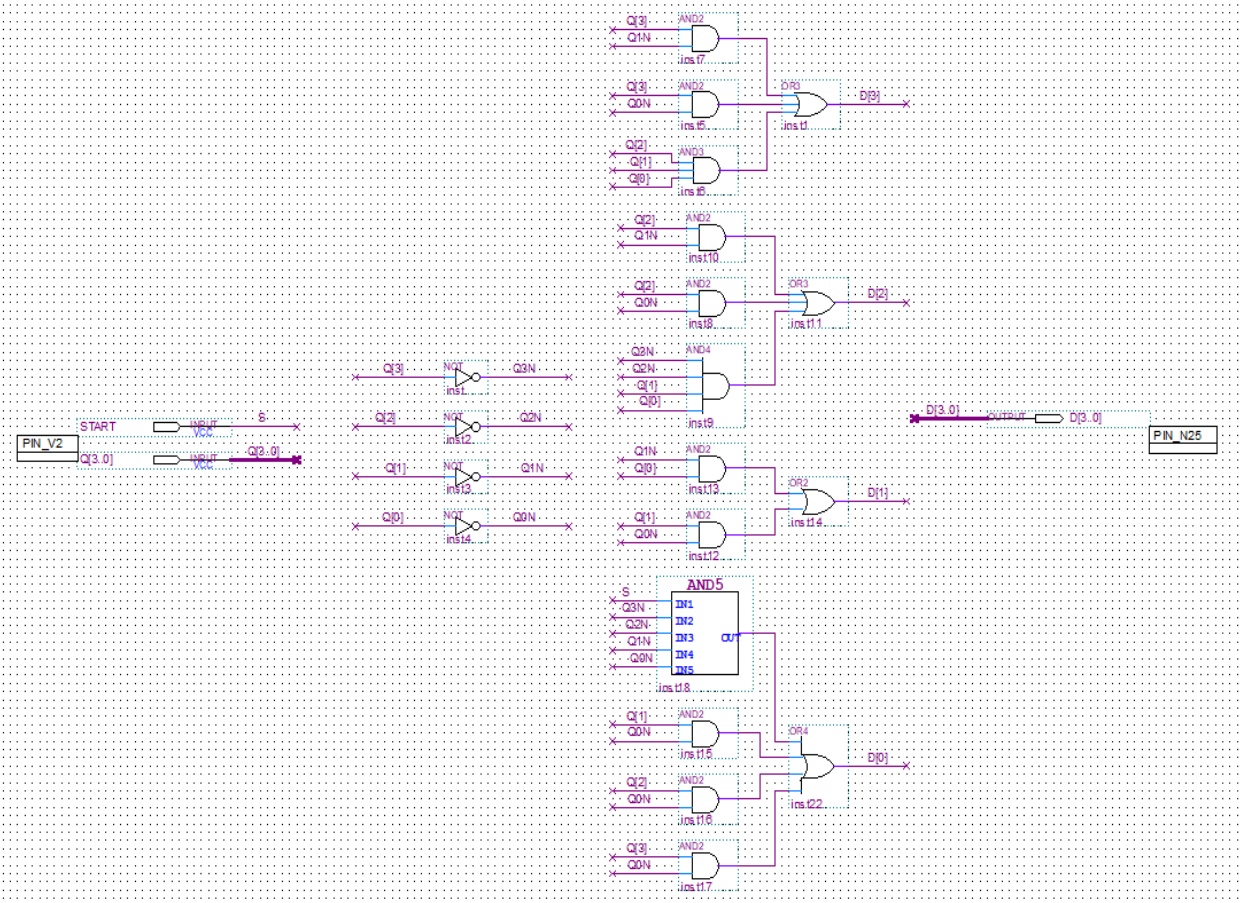
Bảng 4 – Bảng mã hóa trạng thái

d) Khối Next-State

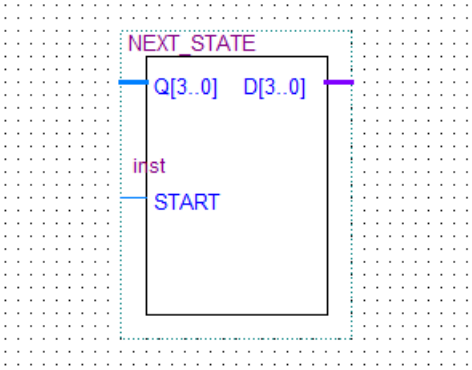
-Khối này có chức năng thực hiện chuyển đổi luận lý từ trạng thái này sang trạng thái khác.
-Từ bảng chuyển trạng thái (Bảng 3) ta rút ra được biểu thức chuyển trạng thái như sau:

- D[3] = Q[3].Q[1]’ + Q[3].Q[0]’ + Q[2].Q[1].Q[0]
- D[2] = Q[2].Q[1]’ + Q[2].Q[0]’ + Q[3]’.Q[2]’.Q[1].Q[0]
- D[1] = Q[1]’.Q[0] + Q[1].Q[0]’
- D[0] = START.Q[3]’.Q[2]’.Q[1]’.Q[0]’ + Q[1].Q[0]’ + Q[2].Q[0]’ + Q[3].Q[0]’

-Schematic khối Next-State:



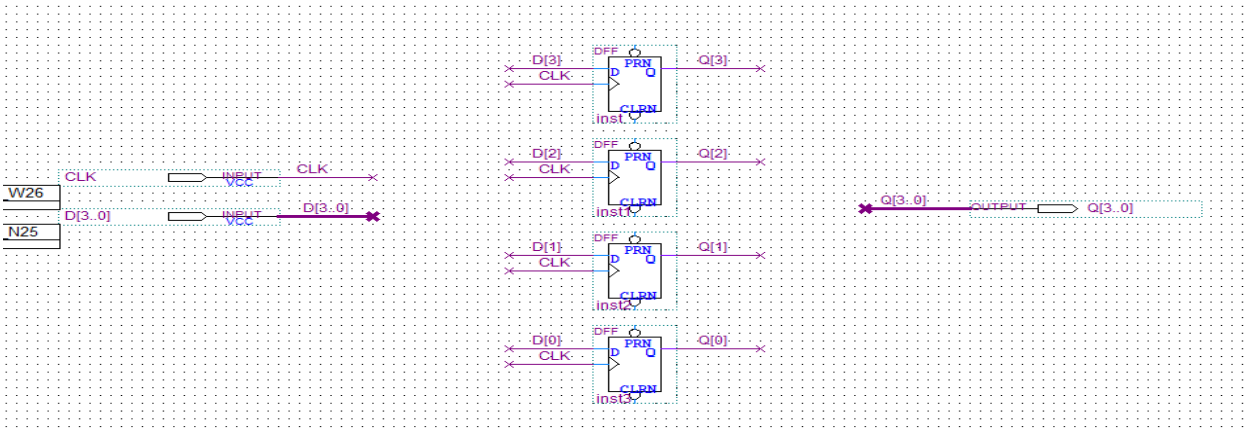
-Đóng gói khối Next-State:



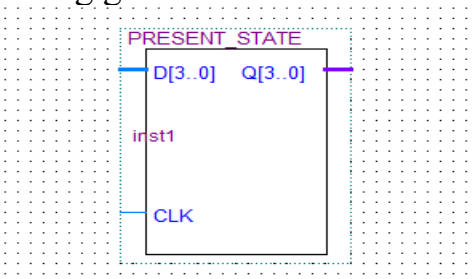
e) Khối Present-State

-Khối này gồm các D-Flipflop có chức năng chính là lưu trữ trạng thái hiện tại của hệ thống.

-Schematic khối Present-State:



-Đóng gói khối Present-State:



f) Khối Output – tín hiệu điều khiển DATAPATH

-Khối này có chức năng chuyển đổi trạng thái hiện tại của CONTROLLER thành các tín hiệu điều khiển DATAPATH thực hiện chức năng mong muốn ứng với trạng thái đó.

-Từ yêu cầu bài toán kết hợp với cấu trúc của DATAPATH và các yêu cầu tính toán tại từng trạng thái cụ thể, ta rút ra được bảng sau:

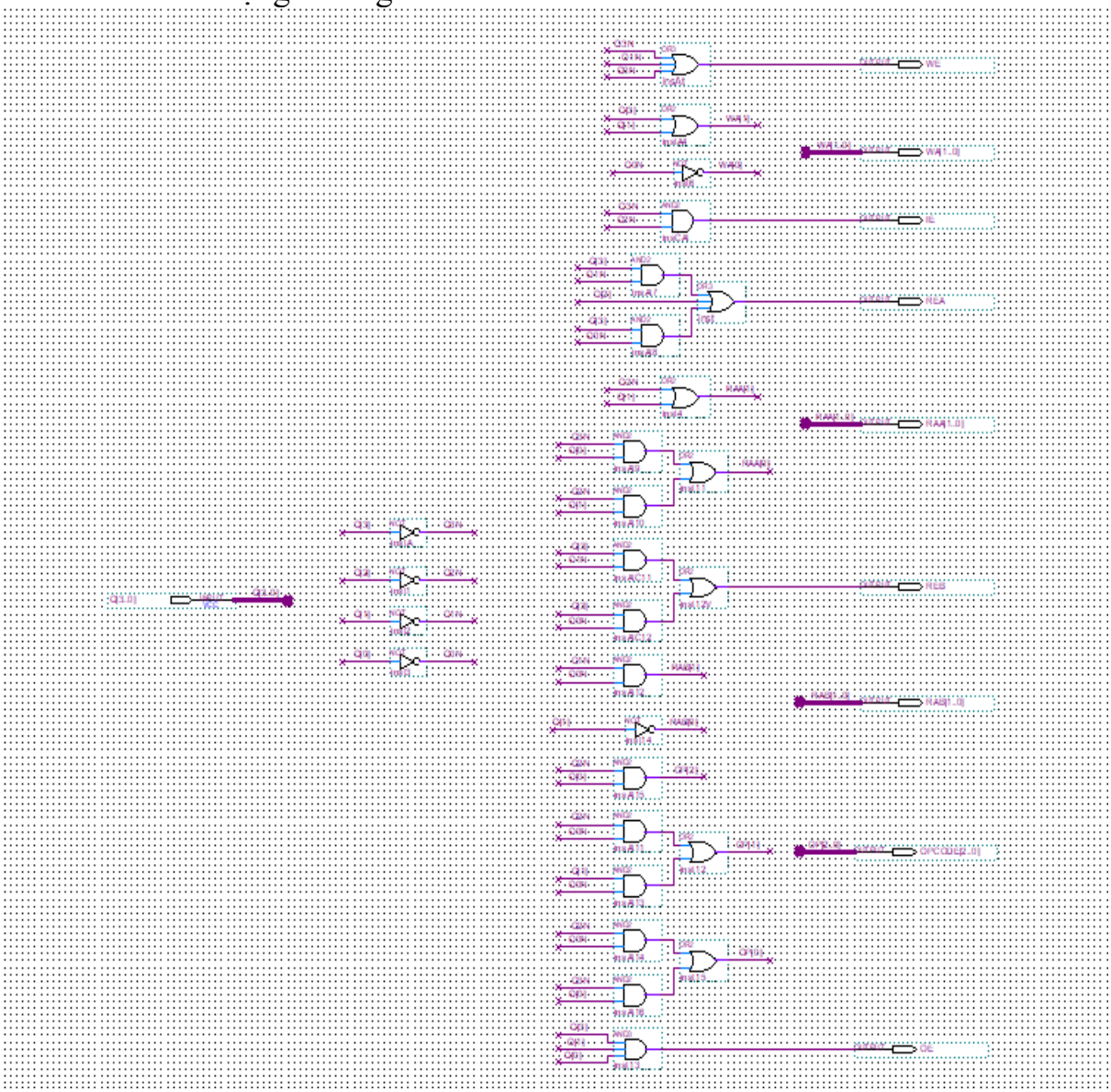
TTHT	MH	WE	WA[1..0]	IE	REA	RAA[1..0]	REB	RAB[1..0]	OP[2..0]	OE
S0	0000	1	00	1	0	XX	0	XX	XXX	0
S1	0001	1	01	1	0	XX	0	XX	XXX	0
S2	0010	1	10	1	0	XX	0	XX	XXX	0
S3	0011	1	11	1	0	XX	0	XX	XXX	0
S4	0100	1	00	0	1	00	0	XX	000	0
S5	0101	1	01	0	1	01	0	XX	001	0
S6	0110	1	10	0	1	10	0	XX	010	0
S7	0111	1	11	0	1	11	0	XX	001	0
S8	1000	1	10	0	1	10	1	11	011	0
S9	1001	1	11	0	1	10	1	01	100	0
S10	1010	1	10	0	1	11	1	00	011	0
S11	1011	0	XX	X	0	XX	0	XX	XXX	1

Bảng 5 – Bảng sự thật chuyển đổi ngõ ra CONTROLLER

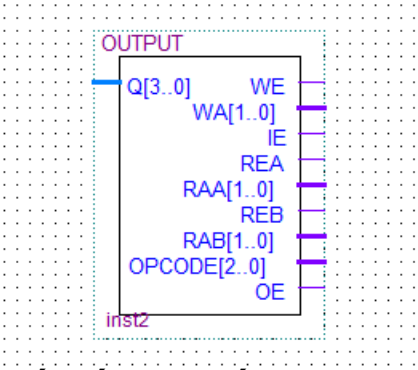
-Từ bảng trên (Bảng 5), ta rút ra được biểu thức các ngõ ra như sau:

- WE = Q[3]’ + Q[1]’ + Q[0]’
- WA[1] = Q[3] + Q[1]
- WA[0] = Q[0]
- IE = Q[3]’.Q[2]’
- REA = Q[2] + Q[3].Q[1]’ + Q[3].Q[0]’
- RAA[1] = Q[2]’ + Q[1]
- RAA[0] = Q[3]’.Q[0] + Q[2]’.Q[1]
- REB = Q[3].Q[1]’ + Q[3].Q[0]’
- RAB[1] = Q[1]’.Q[0]’
- RAB[0] = Q[1]’

-Schematic các trạng thái ngõ ra:



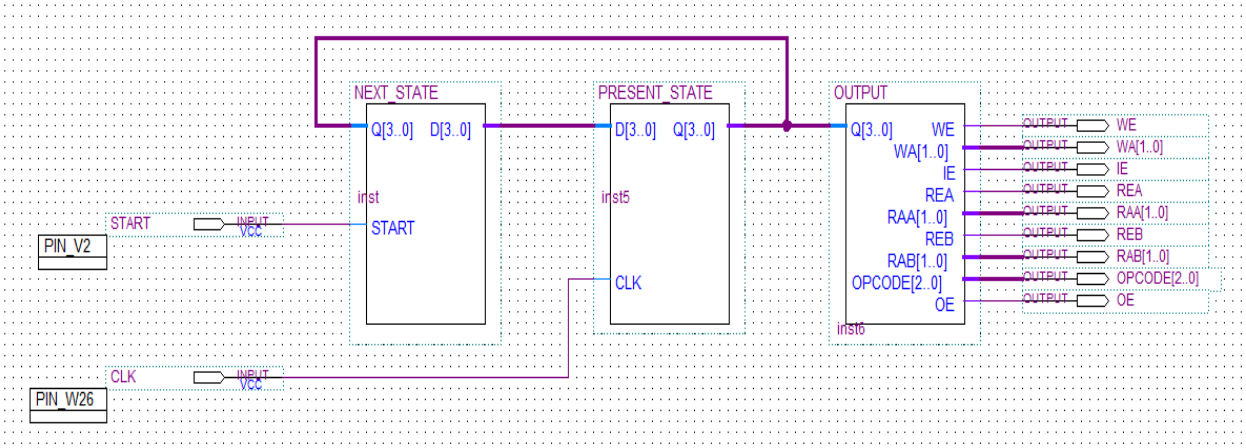
-Đóng gói các trạng thái ngõ ra:



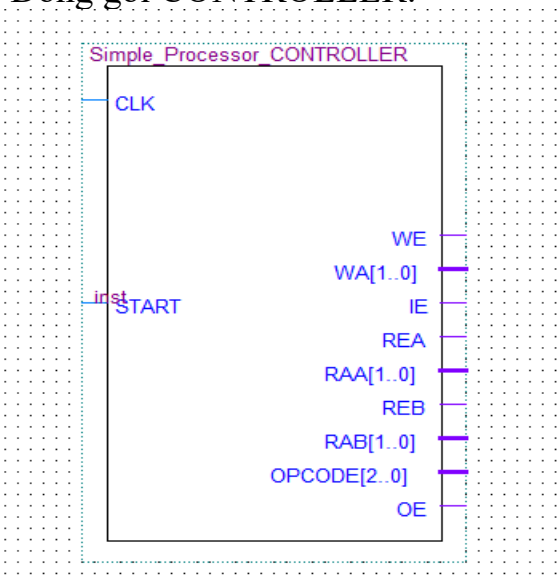
g) Kết nối các khối tạo thành CONTROLLER

-Kết nối 3 khối Next-State, Present-State, Output lại ta được khối CONTROLLER hoàn chỉnh.

-Schematic CONTROLER:



-Đóng gói CONTROLLER:

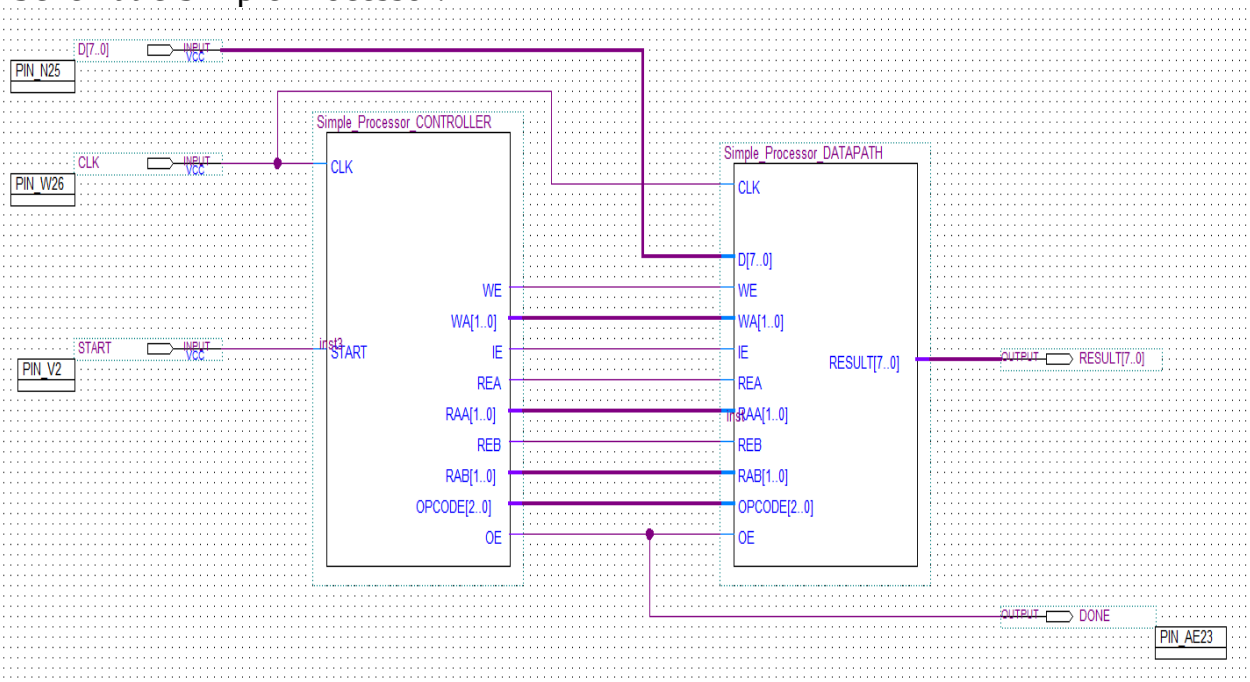


IV. Toàn hệ thống SIMPLE PROCESSOR

a) Kết nối DATAPATH và CONTROLLER

-Ta kết nối CONTROLLER và DATAPATH sẽ được một hệ thống hoàn thiện thực hiện chức năng tính giá trị biểu thức: $\mathbf{1I_3 + 8I_2 - 1I_1 + 0I_0}$

-Schematic Simple Processor:



-Phân tích:

+Processor có chân cấp xung clock

+Chân dữ liệu input 8-bit (thực chất chỉ cần 4-bit nhưng để dễ dàng tính toán khỏi phải mở rộng bit, ta cho input 8-bit)

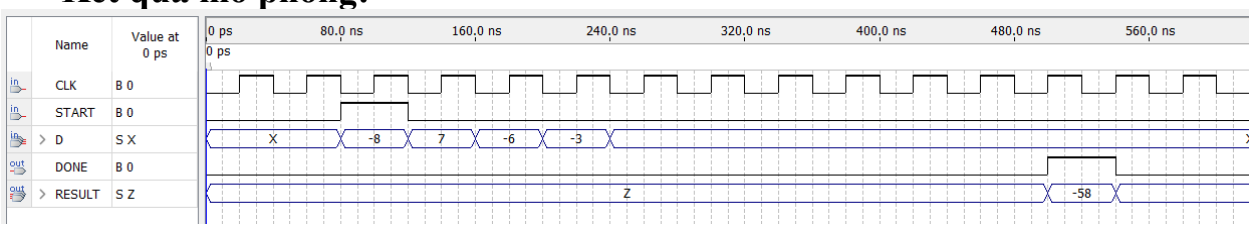
+Chân tín hiệu START cho phép bắt đầu tính toán

+Ngõ ra RESULT 8-bit và tín hiệu báo DONE khi Processor thực hiện tính toán xong giá trị một biểu thức

b) Mô phỏng và kiểm tra chức năng

-Để kiểm tra tính đúng đắn của thiết kế, ta thực hiện chạy mô phỏng lại TestCase đã chạy kiểm tra mô phỏng DATAPATH trước đó: **-8, 7, -6, -3**

-Kết quả mô phỏng:



-Đánh giá:

+Kết quả tính toán trùng khớp với kết quả khi tính toán biểu thức $\mathbf{1I_3 + 8I_2 - 1I_1 + 0I_0}$ với TestCase **-8, 7, -6, -3.**

+Processor thực hiện đúng số chu kì quy định thị cho ra kết quả chính xác
+Tính toán được với input là số có dấu 4-bit (-8 tới 7)
+Có tín hiệu báo DONE khi thực hiện xong kết quả và chỉ cho phép kết quả xuất ra khi thực hiện xong.