

ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA KỸ THUẬT MÁY TÍNH



BÁO CÁO CUỐI KỲ NỘI DUNG BÀI TẬP
THIẾT KẾ LUẬN LÝ SỐ
LỚP: CE118.P13

GIẢNG VIÊN HƯỚNG DẪN: TS.LÂM ĐỨC
KHÁI

TP. HỒ CHÍ MINH – NĂM 2024

THÀNH VIÊN NHÓM THỰC HIỆN

STT	MSSV	Họ và tên	Đóng góp
1	22521198	Đoàn Đăng Quang	30%
2	22521238	Giản Thanh Sang	40%
3	22520469	Nguyễn Ngọc Huy Hoàng	30%

CÁC NỘI DUNG BÀI TẬP ĐÃ THỰC HIỆN

STT	TÊN BÀI
1	Mạch Kiểm tra Parity trong chuỗi input (6.19)
2	Mạch phát hiện đủ 3 bit ‘1’ trong chuỗi input (6.18)
3	Mạch dịch thanh ghi có nạp song song
4	Mạch đếm lên/xuống 4 bit có nạp song song
5	Mạch đếm BCD
6	Tập thanh ghi với 1 cổng ghi và 2 cổng đọc
7	RAM 16x8
8	Stack 16x8
9	Queue 16x8
10	Bộ cộng từ 1 đến n
11	Bộ đếm bit 1 trong chuỗi
12	Thiết kế mạch tính xấp xỉ căn bậc 2 - Register Sharing
13	Thiết kế mạch tính xấp xỉ căn bậc 2 - Bus Sharing
14	Thiết kế mạch tính xấp xỉ căn bậc 2 - Functional Unit Sharing
15	Thiết kế mạch tính xấp xỉ căn bậc 2 - Register Merging
16	Thiết kế mạch tính xấp xỉ căn bậc 2 - Functional Unit Pipelining
17	Thiết kế mạch tính xấp xỉ căn bậc 2 - Datapath Pipelining
18	Thiết kế mạch tính xấp xỉ căn bậc 2 - Functional Unit & Datapath Pipelining

LỜI CẢM ƠN.....	5
Bài 1: Thiết kế mạch kiểm tra parity trong chuỗi input	7
1. Mô tả chức năng:	7
2. Sơ đồ chuyển trạng thái:	7
3. Kết quả mô phỏng:.....	8
Bài 2: Thiết kế Mạch đếm đủ 3 bit 1 trong dây input	9
1. Tổng quan:	9
2. Thiết kế máy trạng thái:	9
3. Kết quả mô phỏng:.....	12
Bài 3: Thiết kế Register đa chức năng SRwPL.....	13
1. Tổng quan:	13
2. Thiết kế chi tiết:	13
3. Kết quả mô phỏng:.....	14
Bài 4: Thiết kế Up-Down Counter with Parallel Load.....	15
1. Tổng quan:	15
2. Thiết kế chi tiết:	15
3. Kết quả mô phỏng:.....	17
Bài 5: Thiết kế BCD counter.....	18
1. Tổng quan:	18
2. Thiết kế chi tiết:	18
3. Mô phỏng.....	20
Bài 6: Thiết kế Register File 2 Read port 1 Write Port.....	22
1. Tổng quan:	22
2. Thiết kế chi tiết	22
3. Kết quả mô phỏng.....	23
Bài 7: Thiết kế RAM.....	24
1. Tổng quan:	24
2. Thiết kế mạch:.....	24
3. Kết quả mô phỏng:.....	27
Bài 8: Thiết kế Stack.....	28
1. Tổng quan:	28
2. Thiết kế mạch:.....	28

3. Kết quả mô phỏng:	31
Bài 9: Thiết kế Queue	32
1. Tổng quan:	32
2. Thiết kế chi tiết	33
Bài 10: Tính tổng dãy số từ 1 đến n	34
1. Tổng quan:	34
2. Thiết kế Datapath:	35
3. Thiết kế Controller:	36
4. Kết quả mô phỏng:	39
Bài 11: Đếm số bit 1 trong input	40
1. Tổng quan:	40
2. Thiết kế Datapath:	40
3. Thiết kế Controller:	41
4. Kết quả mô phỏng:	42
Bài 12: Thiết kế mạch tính xấp xỉ căn bậc 2 Register Sharing - (Graph - partitioning)	43
1. Tổng quan:	43
2. Thiết kế Datapath:	46
3. Thiết kế Controller:	49
4. Mô phỏng:	51
Bài 13: Thiết kế mạch tính xấp xỉ căn bậc 2 Bus Sharing	53
1. Tổng quan:	53
2. Thiết kế Datapath:	53
3. Thiết kế Controller:	54
4. Mô phỏng:	56
Bài 14: Thiết kế mạch tính xấp xỉ căn bậc 2 Functional Unit Sharing	58
1. Tổng quan:	58
2. Thiết kế Datapath:	59
3. Thiết kế khối điều khiển:	62
4. Mô phỏng:	63
Bài 15: Thiết kế mạch tính xấp xỉ căn bậc 2 Register Merging	65
1. Tổng quan:	65
2. Thiết kế Datapath:	66
3. Thiết kế Controller:	68
4. Mô phỏng:	69

Bài 16: Thiết kế mạch tính xấp xỉ căn bậc 2 Functional Unit Pipelining	70
1. Tổng quan:	70
2. Thiết kế datapath.....	71
3. Thiết kế Controller:.....	74
4. Mô phỏng:.....	77
Bài 17: Thiết kế mạch tính xấp xỉ căn bậc 2 Datapath Pipelining.	78
1. Tổng quan:	78
2. Thiết kế Datapath:.....	79
3. Thiết kế Controller:.....	82
4. Kết quả mô phỏng:.....	91
Bài 18: Thiết kế mạch tính xấp xỉ căn bậc 2 Datapath & Functional Unit Pipelining	92
1. Tổng quan:	92
2. Thiết kế Datapath:.....	92
3. Thiết kế Controller:.....	96
4. Kết quả mô phỏng:.....	101

LỜI CẢM ƠN

Lời đầu tiên, chúng em xin gửi lời cảm ơn sâu sắc đến Thầy Lâm Đức Khải, Giảng viên môn Thiết kế Luận Lý Số. Nhờ sự hướng dẫn và hỗ trợ trực tiếp từ Thầy trong suốt quá trình học tập, nhóm chúng em mới có thể tiến hành xây dựng, nghiên cứu và hoàn thành bài báo cáo này.

Trong thời gian học tập với Thầy, chúng em đã học được nhiều kiến thức quý giá, tiếp thu thêm nhiều kinh nghiệm, cũng như phát triển các kỹ năng sống và làm việc. Đó đều là những hành trang cần thiết cho chúng em hoàn thiện bản thân hiện tại và tiến xa hơn trong tương lai. Một lần nữa, chúng em xin chân thành cảm ơn Thầy. Chúc Thầy luôn dồi dào sức khỏe, vững bước trên con đường sự nghiệp giảng dạy cao quý. Mong rằng Thầy sẽ tiếp tục truyền cảm hứng, mang đến những bài học ý nghĩa và khai sáng tâm trí cho nhiều thế hệ học trò sau này.

TP. Hồ Chí Minh, thứ ba ngày 3 tháng 12 năm 2024

Nhóm sinh viên thực hiện

Đoàn Đăng Quang

Giản Thanh Sang

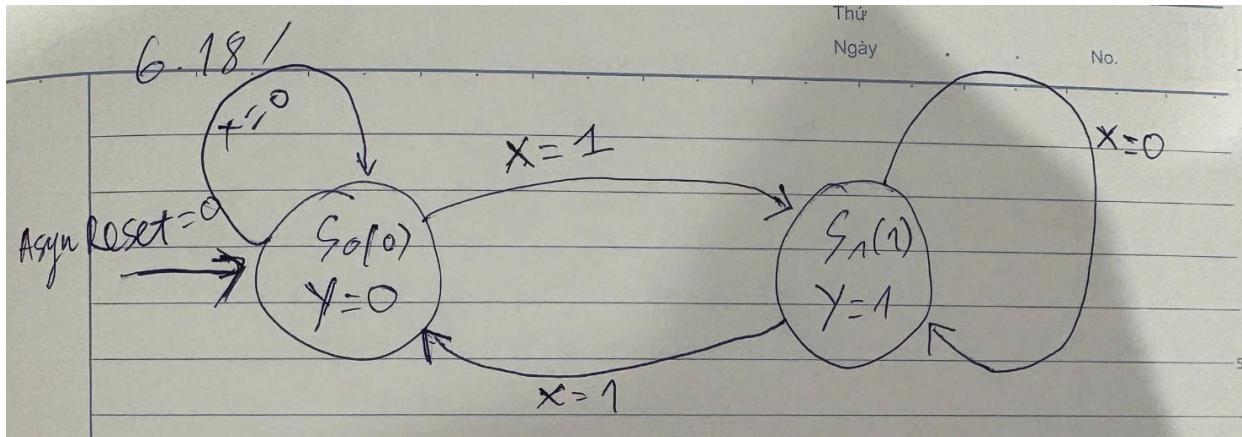
Nguyễn Ngọc Huy Hoàng

Bài 1: Thiết kế mạch kiểm tra parity trong chuỗi input

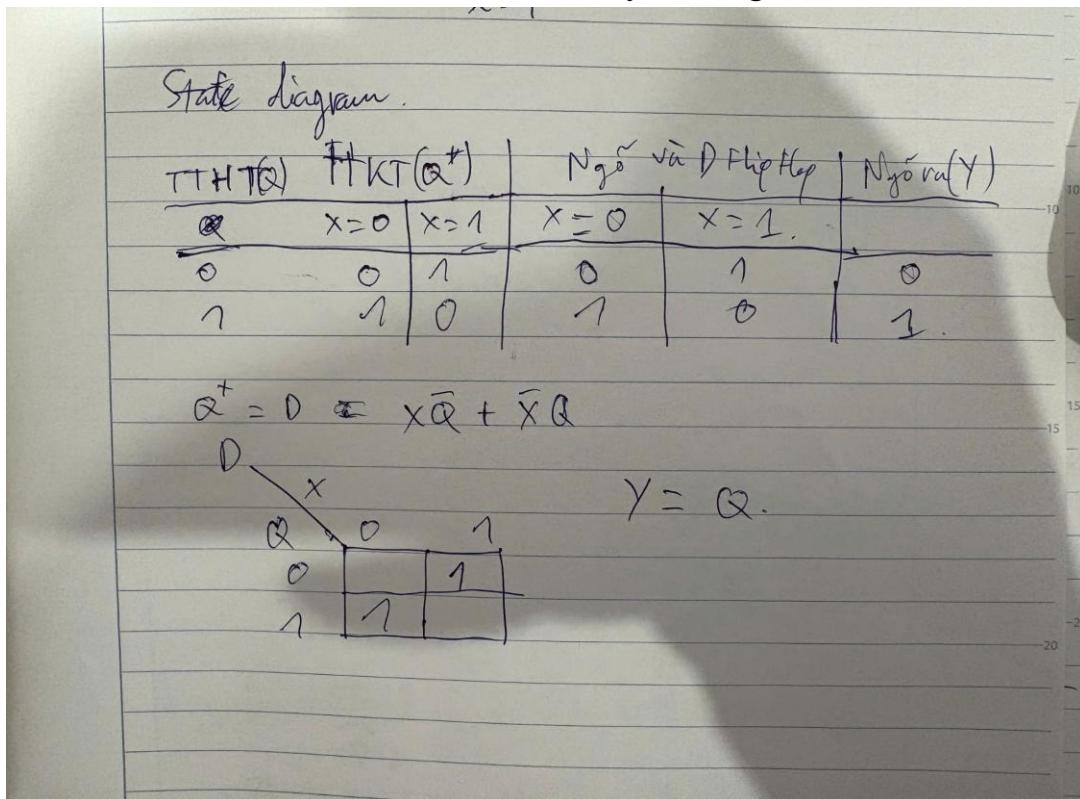
1. Mô tả chức năng:

- Mạch có input đầu vào 1 bit (X), tín hiệu reset bắt đồng bộ mức thấp và xung clock.
- Output đầu ra Y = 1 khi số lượng input X bằng '1' nhận vào là lẻ kể từ sau lúc reset.

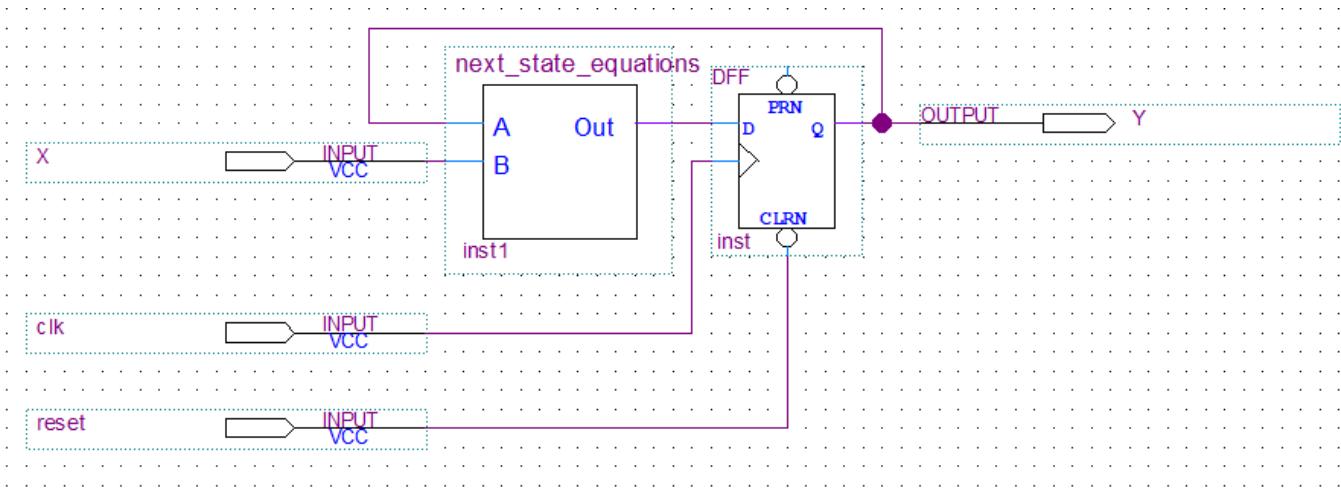
2. Sơ đồ chuyển trạng thái:



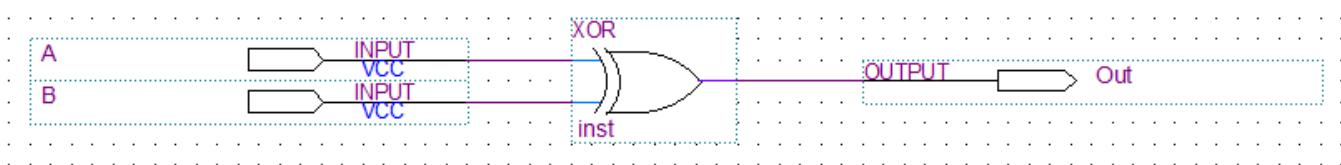
Hình 1.1: sơ đồ chuyển trạng thái



Hình 1.2: Phương trình ngõ vào và mã hóa ngõ ra

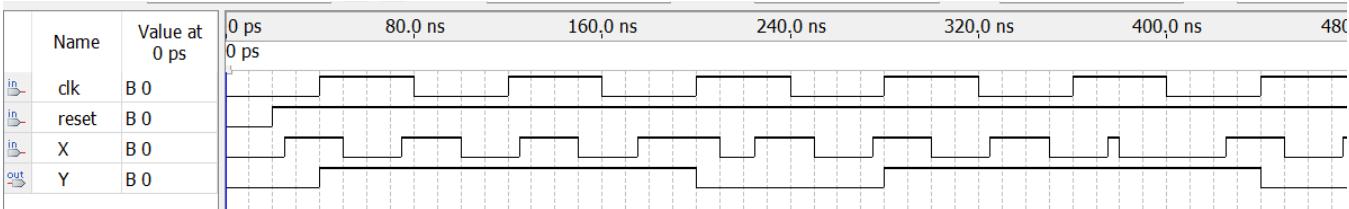


Hình 1.3:



Hình 1.4:

3. Kết quả mô phỏng:



Hình 1.5:

- Nhận input = 1 tại cạnh lên clk thứ 1,3,4,6.
- Từ cạnh lên 1 đến 3 output bằng 1. Từ cạnh lên 3 4 output bằng 0 (vì đã nhận 2 input ‘1’).
- Từ cạnh lên 4 đến 6 output bằng 1 vì đã nhận 3 input ‘1’; tại cạnh lên 1,3,4.

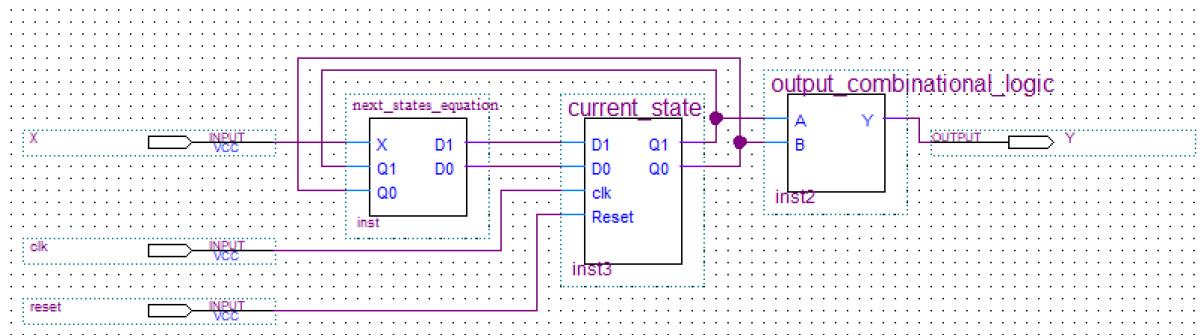
Bài 2: Thiết kế Mạch đếm đủ 3 bit '1' trong dãy input

1. Tổng quan:

- Thiết kế mạch với chức năng phát hiện đủ 3 bit '1' trong chuỗi input.
- Input đầu vào:
 - + X là giá trị của input tại 1 thời điểm.
 - + CLK

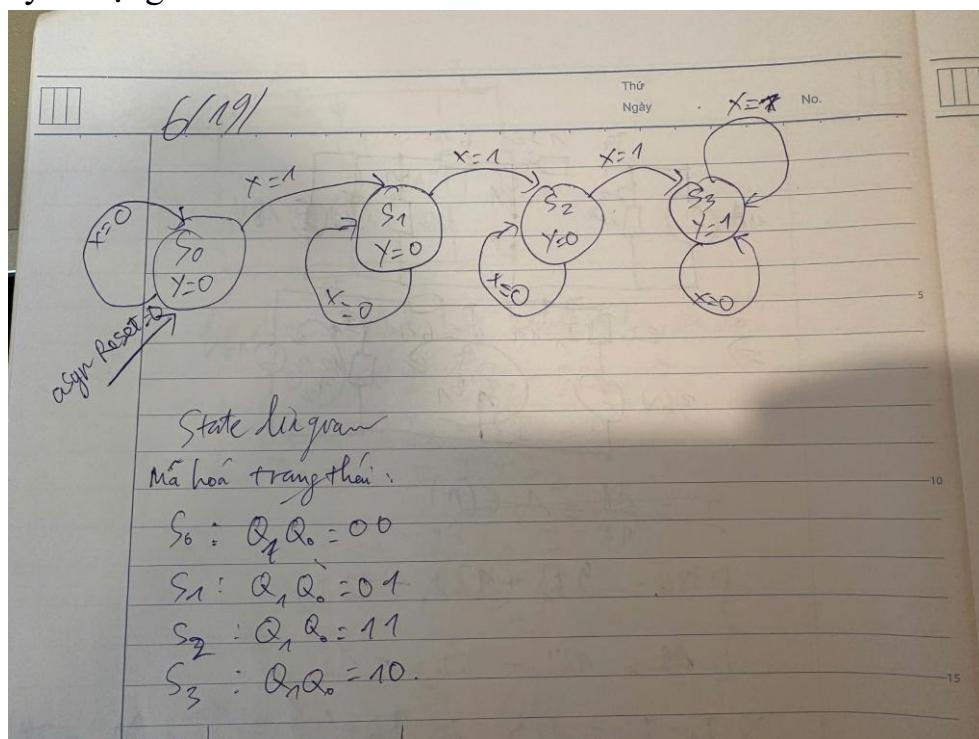
Mỗi CLK mạch nhận 1 giá trị của X tạo thành chuỗi Input.

- Tín hiệu Output bằng 1 khi có đủ 3 bit '1' được nhận từ khi reset.



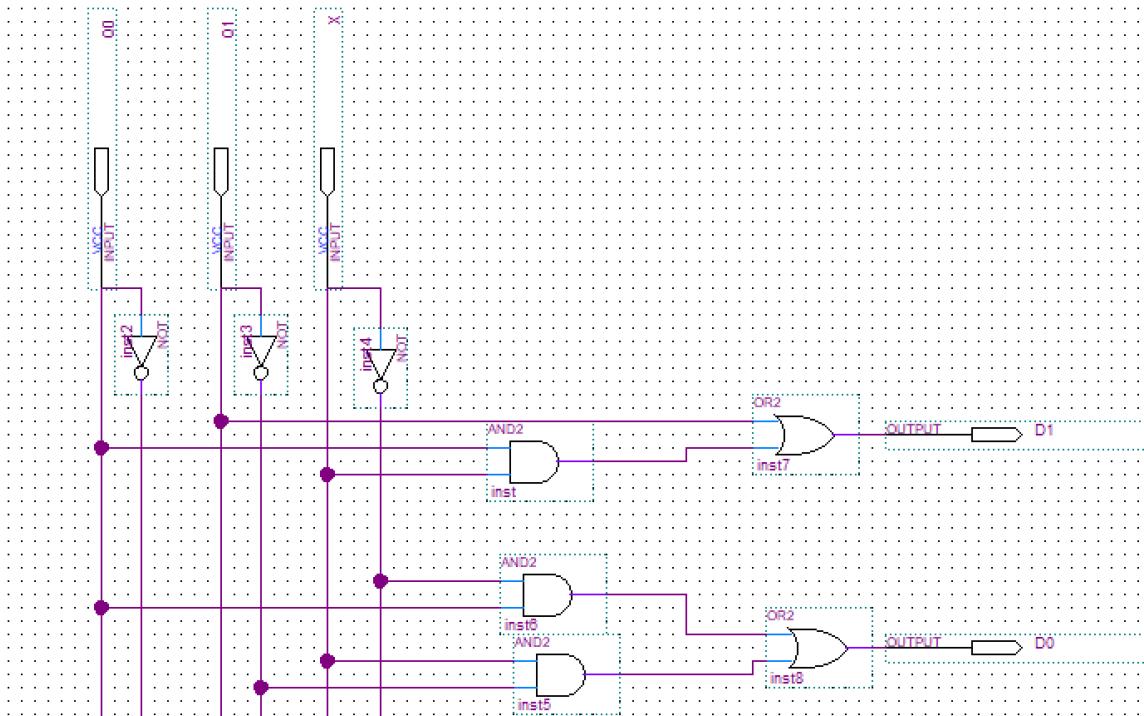
2. Thiết kế máy trạng thái:

Sơ đồ chuyển trạng thái:

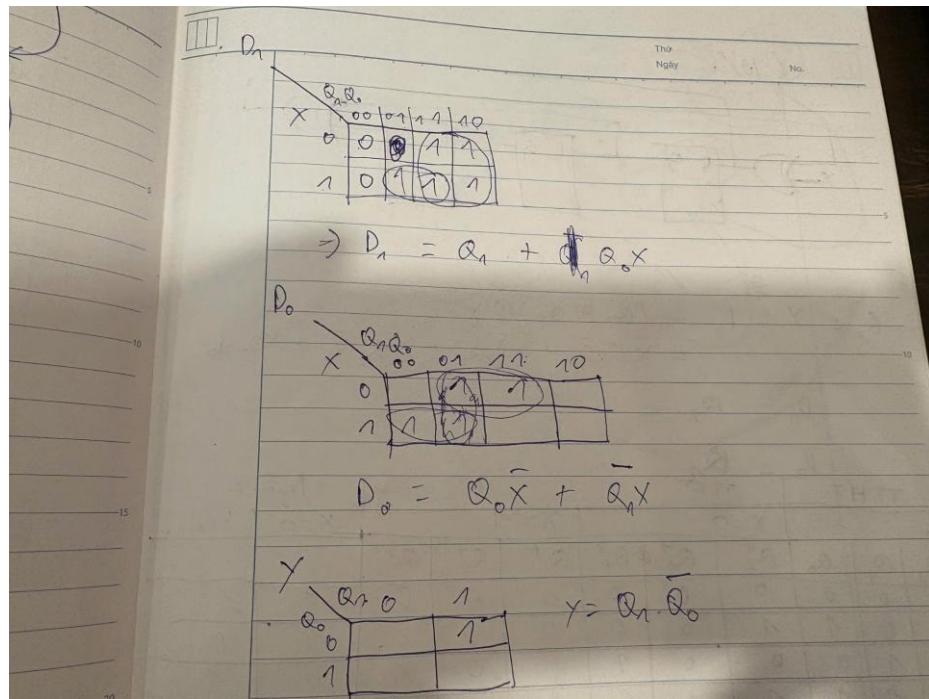


Sơ đồ chuyển trạng thái

2.1. Khối trạng thái kế tiếp:



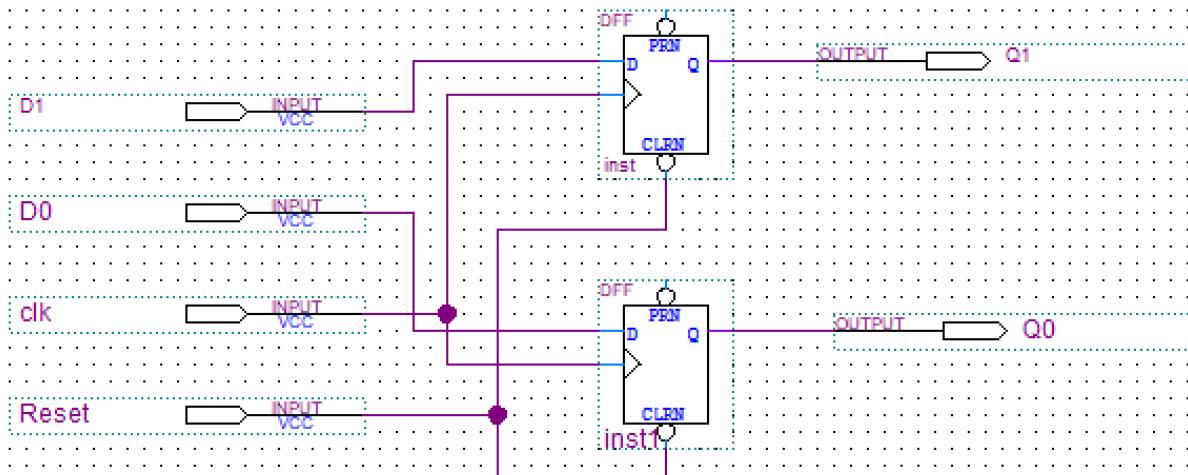
$x_3 \cdot x_2 x_0 = 110$			
$TTHT$	$TTKT$		
$Q_1 Q_0$	$Q_1^+ Q_0^+$		
$x=0$	$x=1$		
S_0	S_1		
S_1	S_2		
S_2	S_3		
S_3	S_3		
$TTHT$	$TTKT$	Ngo vào FF	
$Q_1 Q_0$	$Q_1^+ Q_0^+$	$x=0$	$x=1$
0 0	0 0	0 0	0 1
0 1	0 1	0 1	1 1
1 1	1 1	1 0	1 0
1 0	1 0	1 0	0 1



- Rút gọn tổ hợp trạng thái kế tiếp.

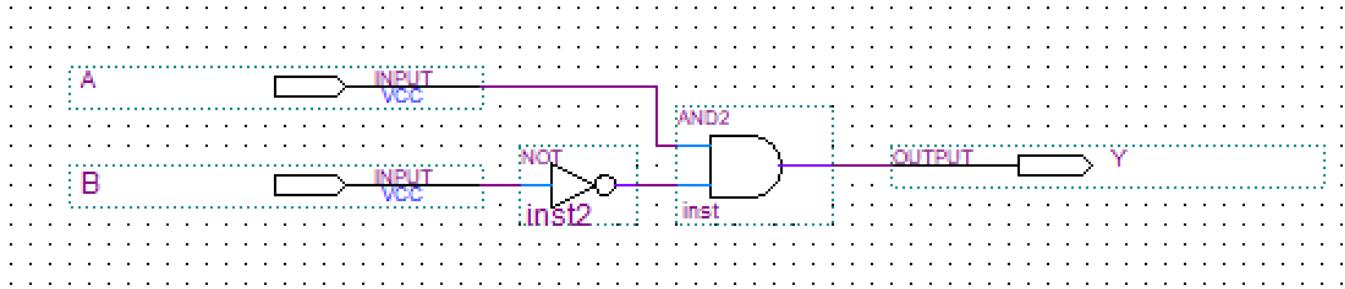
2.2. Khối trạng thái hiện tại:

- Tín hiệu reset mức thấp bắt đồng bộ.

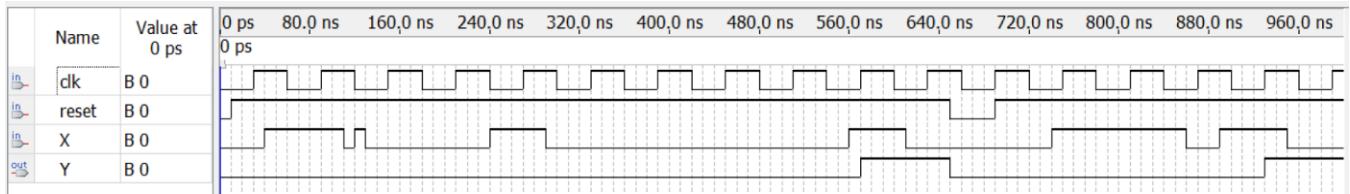


2.3. Khối mã hóa ngoã ra:

- Ngoã ra bằng 1 khi mạch ở trạng thái hiện tại ở State 3 (10).



3. Kết quả mô phỏng:



- Mạch hoạt động đúng chức năng, khi nhận đủ 3 bit ‘1’ thì tín hiệu output Y bằng 1.

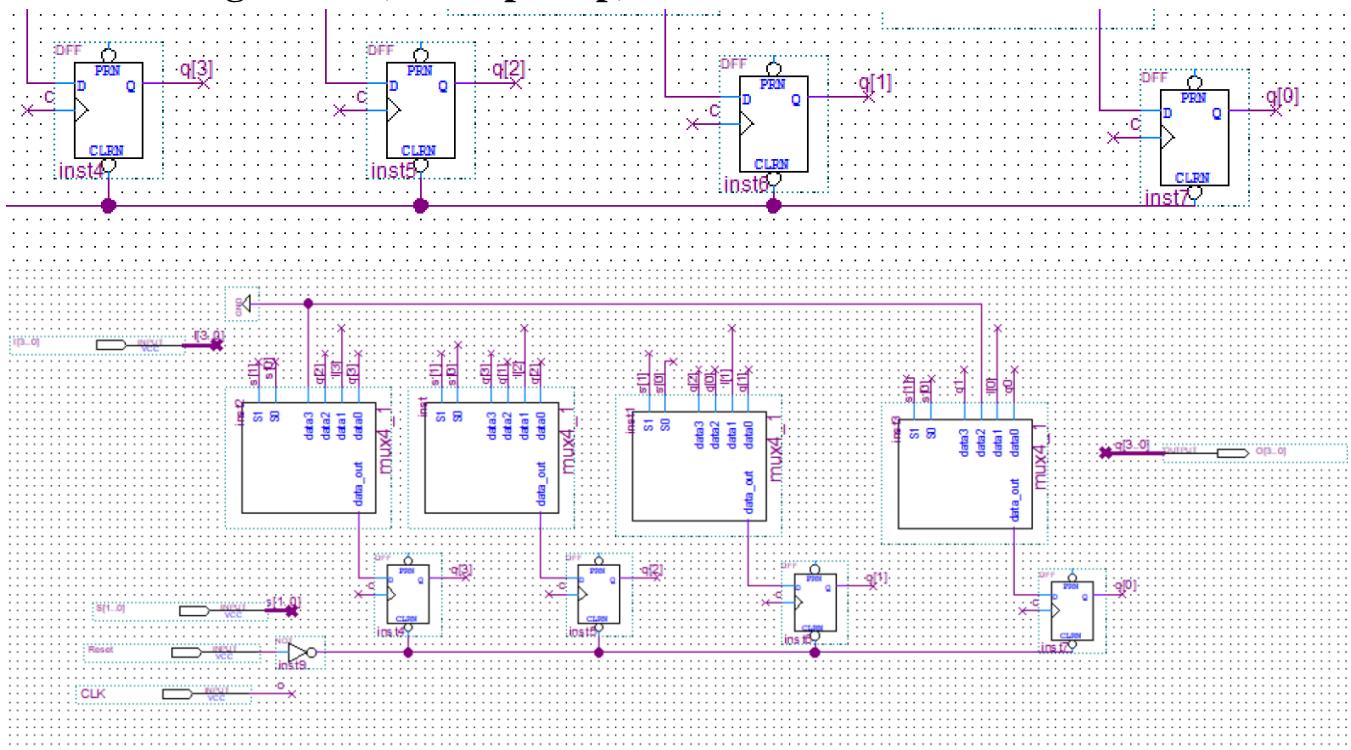
Bài 3: Thiết kế Register đa chức năng SRwPL

1. Tổng quan:

- Thiết kế thanh ghi 4 bit với khả năng dịch trái, phải, và nạp giá trị song song.
- Input đầu vào:
 - + Dữ liệu đầu vào 4bit.
 - + Tín hiệu điều khiển 2 bit để chọn chế độ.
 - + Tín hiệu Reset bắt đồng bộ mức cao.
 - + Tín hiệu CLK.

2. Thiết kế chi tiết:

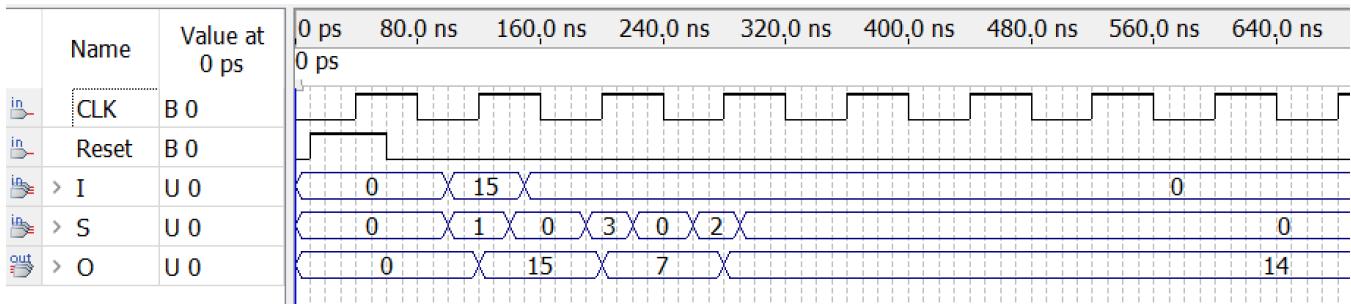
- Thanh ghi 4 bit (4 D Flip-Flop)



- Tích hợp các mux 4-1(1bit) để chọn ngõ vào cho flipflop.

Select S[1..0]		Operation	Next State Value Q[3..0]			
0	0	No Change	Q ₃	Q ₂	Q ₁	Q ₀
0	1	Parallel Load	I ₃	I ₂	I ₁	I ₀
1	0	Shift Left(<<)	Q ₂	Q ₁	Q ₀	0
1	1	Shift Right(>>)	0	Q ₃	Q ₂	Q ₁

3. Kết quả mô phỏng:

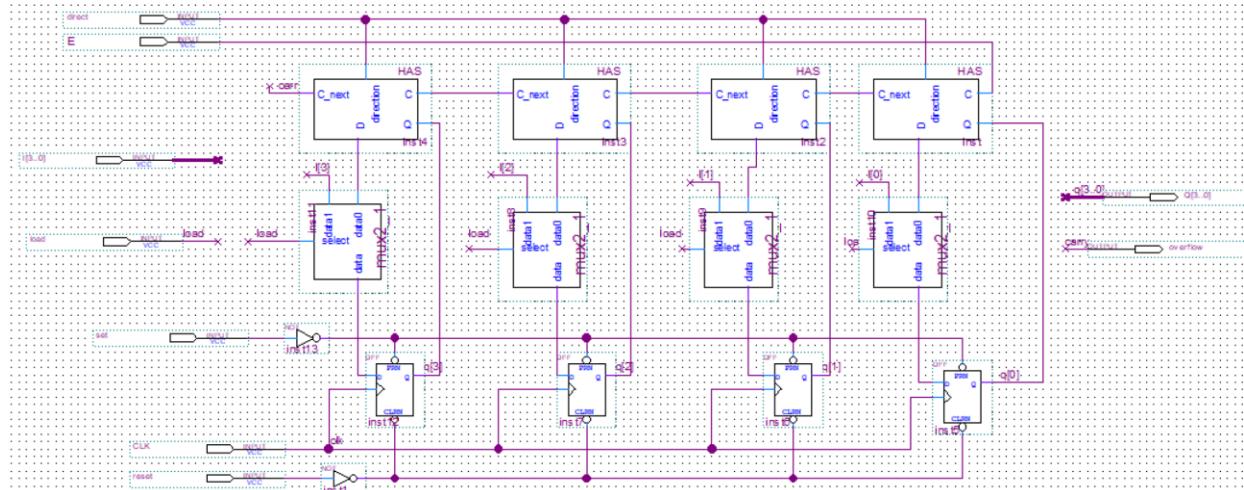


- Tín hiệu Select S[1..0] lựa chọn chế độ hoạt động cho thanh ghi SRPL
 - + 0 : thanh ghi giữ nguyên giá trị.
 - + 1: nạp giá trị song song.
 - + 2: Dịch trái 1 bit.
 - + 3: Dịch phải 1 bit.
- Giải thích mô phỏng:
 - + Đặt select=1 nạp giá trị 15 vào thanh ghi.
 - + Sau đó, đặt select = 3 để dịch phải ($1111 >>1 = 0111$),
 - + Tiếp tục, đặt select = 2 để dịch trái ($0111 <<1 = 1110$).

Bài 4: Thiết kế Up-Down Counter with Parallel Load

1. Tổng quan:

- Thiết kế bộ đếm 4bit có chức năng đếm lên/xuống và nạp giá trị song song.

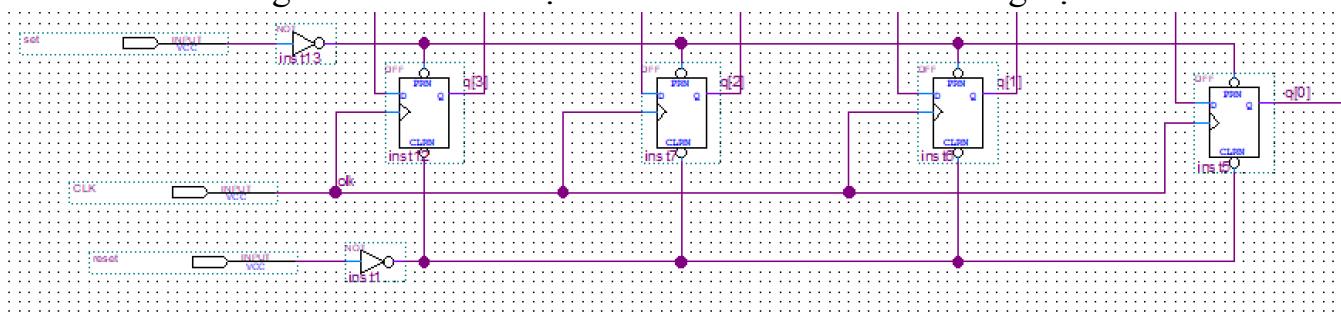


- Đầu vào:

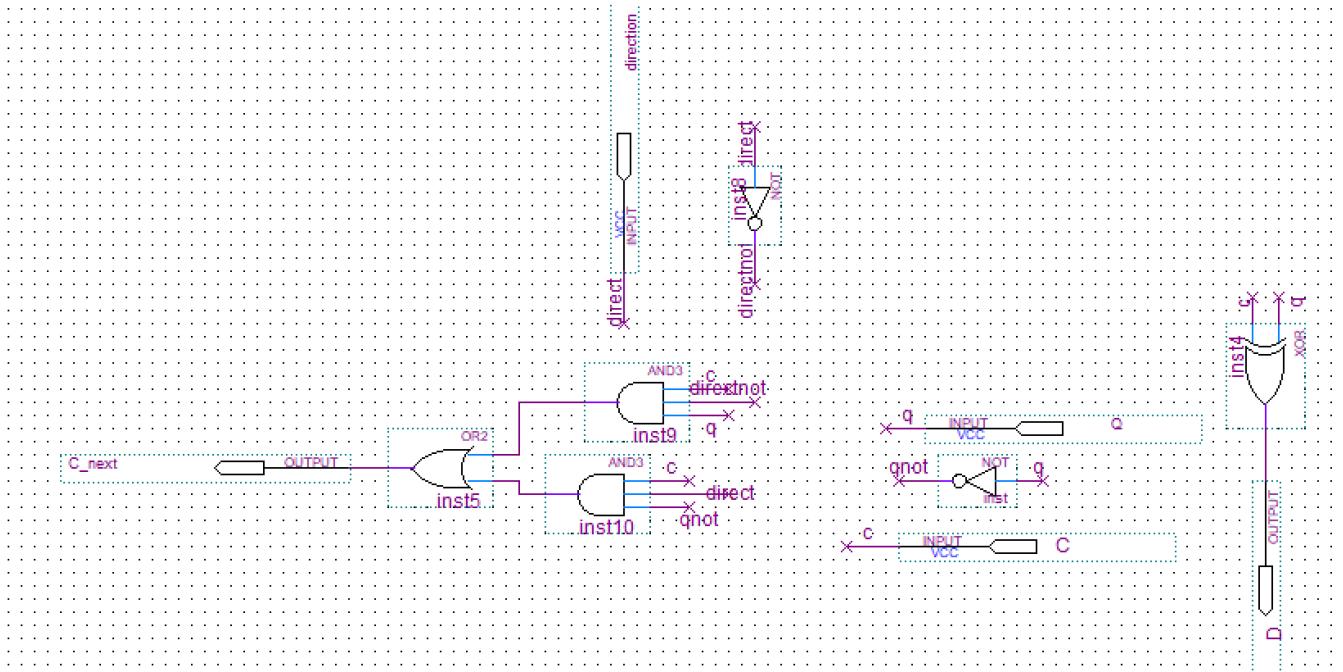
- + Dữ liệu đầu vào 4 bit I[3..0].
- + Tín hiệu D (direct) cấu hình chức năng đếm (0: đếm lên, 1 đếm xuống).
- + Tín hiệu En cho phép đếm.
- + Tín hiệu Load cho phép nạp giá trị.
- + Tín hiệu set và preset.
- Ngõ ra: giá trị bộ đếm.

2. Thiết kế chi tiết:

2.1. Thanh ghi 4 bit với tín hiệu set và reset mức cao bắt đồng bộ:



2.2. Khối HAS:



C	Q	D
0	0	0
0	1	1
1	0	1
1	1	0

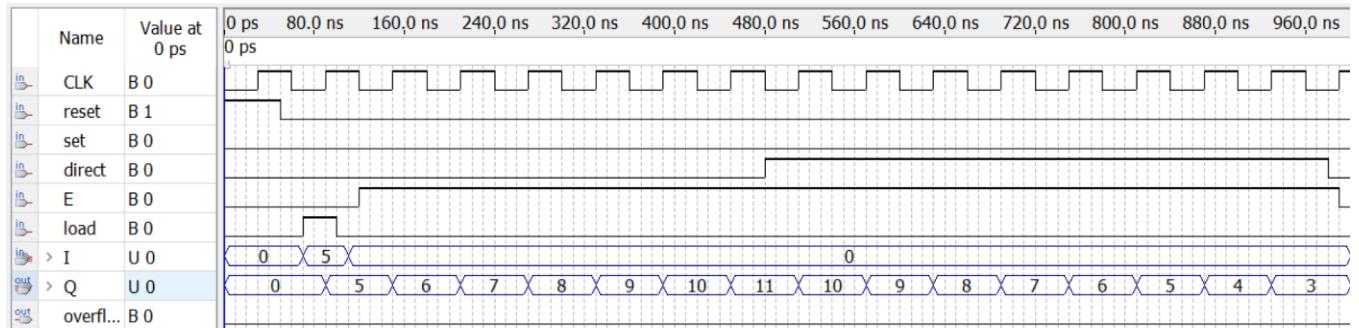
- Giá trị ngõ vào D Flip Flop sẽ đảo khi đầu vào C=1. (Ở bit LSB của thanh ghi ta nối tín hiệu En (cho phép đếm) vào đầu vào C của khối HAS tạo ngõ vào cho DFF của bit LSB).

C	Direct	Q	Carry_Next
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0

1	0	1	1
1	1	0	1
1	1	1	0

- Giá trị Carry_next=1 khi Q=1,Carry=1 và đang đếm lên D=0, hoặc Q=0,Carry=1, và đang đếm xuống D=1.

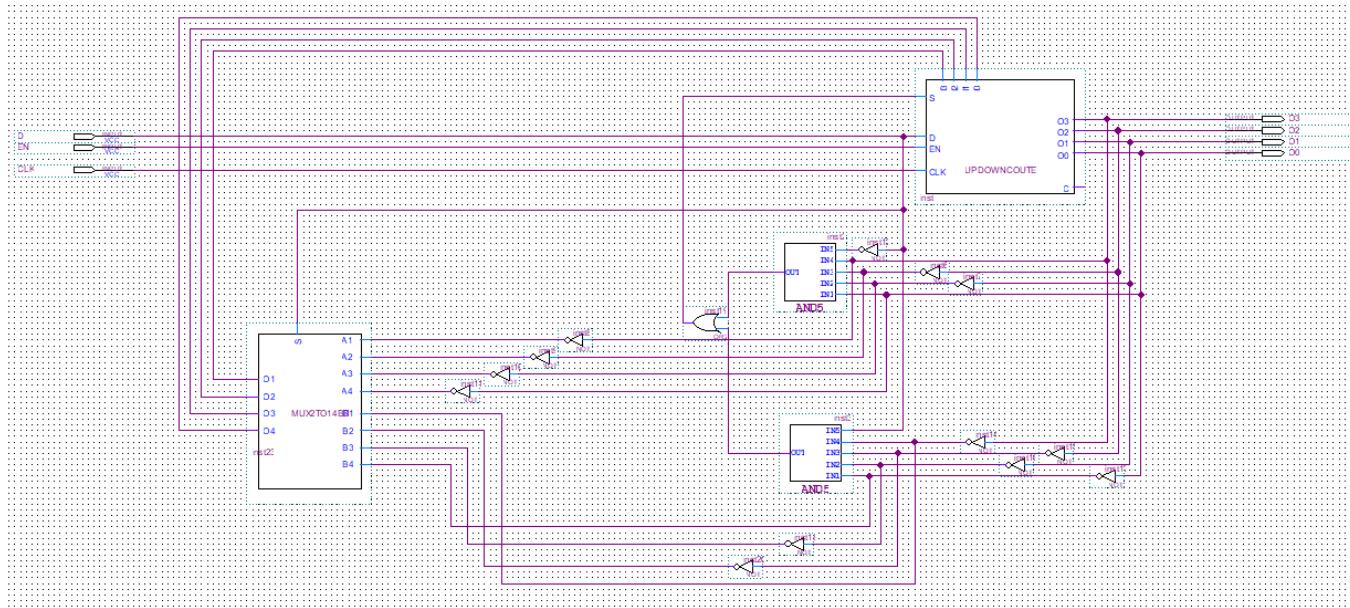
3. Kết quả mô phỏng:



- Giải thích mô phỏng:
 - + Reset mạch trước khi load giá trị song song vào bộ đếm.
 - + Đặt tín hiệu E=1, cho phép đếm,
 - + Direct =0 đếm lên, =1 đếm xuống. (khi direct từ 0 lên 1, tại cạnh lên clock tiếp theo bộ đếm đổi từ đếm lên thành đếm xuống 11->10).

Bài 5: Thiết kế BCD counter

1. Tổng quan:



Input:

- D: xác định việc đếm lên D = 0 hoặc đếm xuống D = 1
- EN: tín hiệu cho phép bộ đếm hoạt động
- CLK

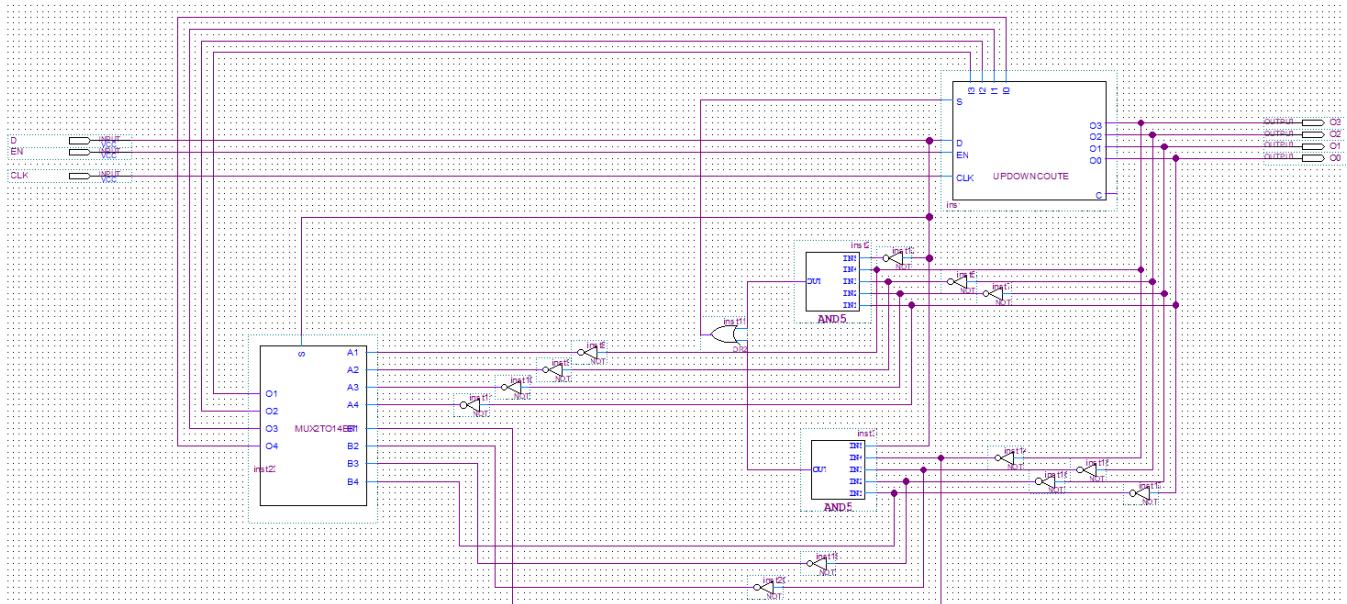
Output:

- Q3Q2Q1Q0: giá trị ngõ ra của bộ đếm có độ dài là 4 bit

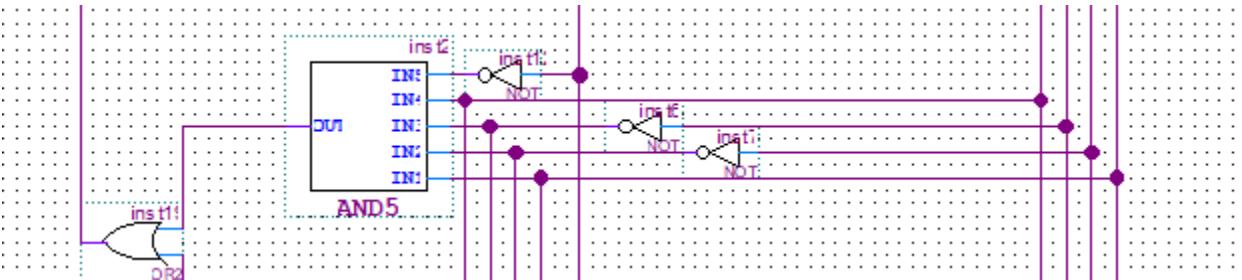
2. Thiết kế chi tiết:

Sử dụng UPDOWNCOUNTER 4 bit có chức năng nạp dữ liệu để thiết kế BCD UPDOWNCOUNTER

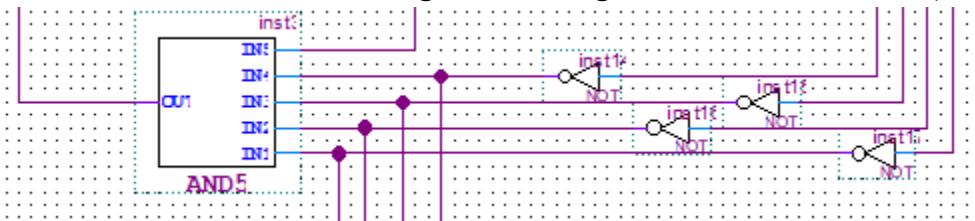
Thiết kế UPDOWNCOUNTER 4 bit được sử dụng lại từ bài trước



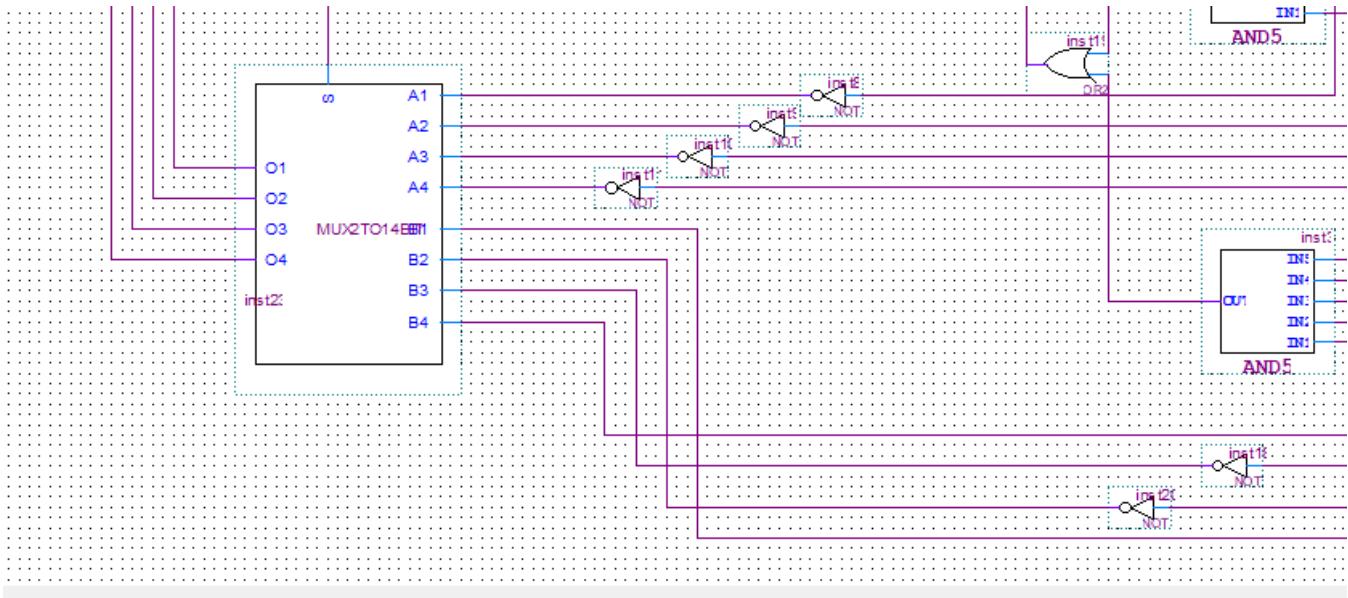
Dùng AND5 để nhận biết bộ đếm đang đếm lên $D = 0$ và đếm đến 9 ($Q_3Q_2Q_1Q_0 = 1001$)



Mạch nhận biết bộ đếm đang đếm xuống $D = 1$ và đếm đến 0 ($Q_3Q_2Q_1Q_0 = 0000$)



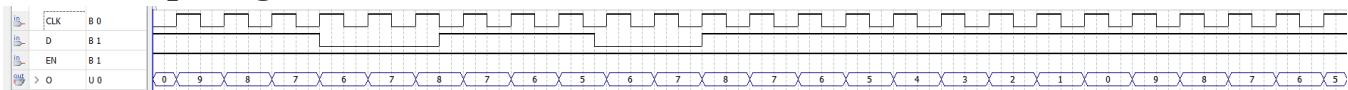
Dùng OR để tạo ra tín hiệu cho S, tín hiệu cho phép nạp dữ liệu 0 hoặc 9 vào bộ đếm



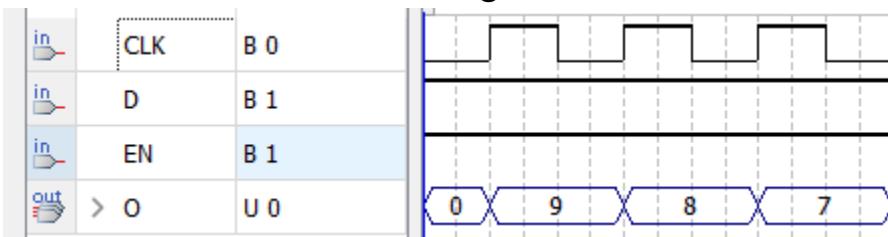
Mạch xác định:

- Nếu bộ đếm đếm lên thì nạp giá trị 0000
- Nếu bộ đếm đếm xuống thì nạp giá trị 1001

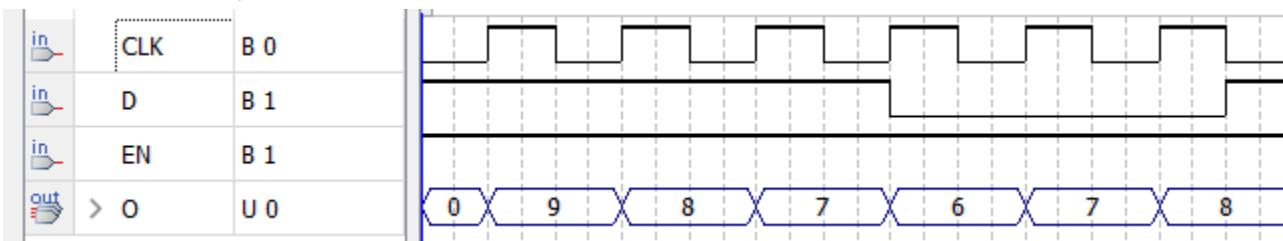
3. Mô phỏng



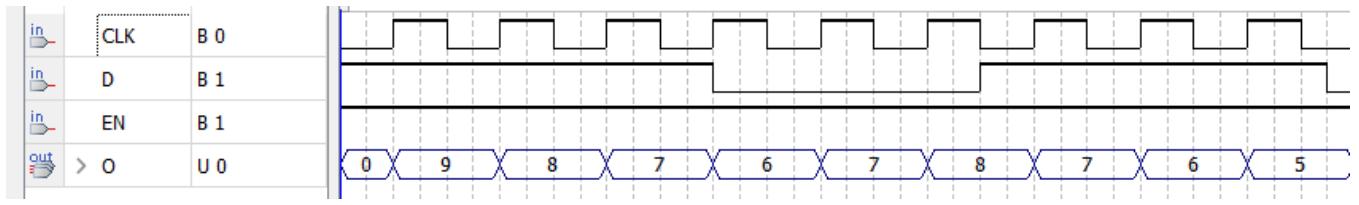
Khi D = 1 thì bộ đếm đếm xuống



Khi D = 0 thì bộ đếm đếm lên



Tiếp tục thay đổi D = 1 thì bộ đếm đếm xuống

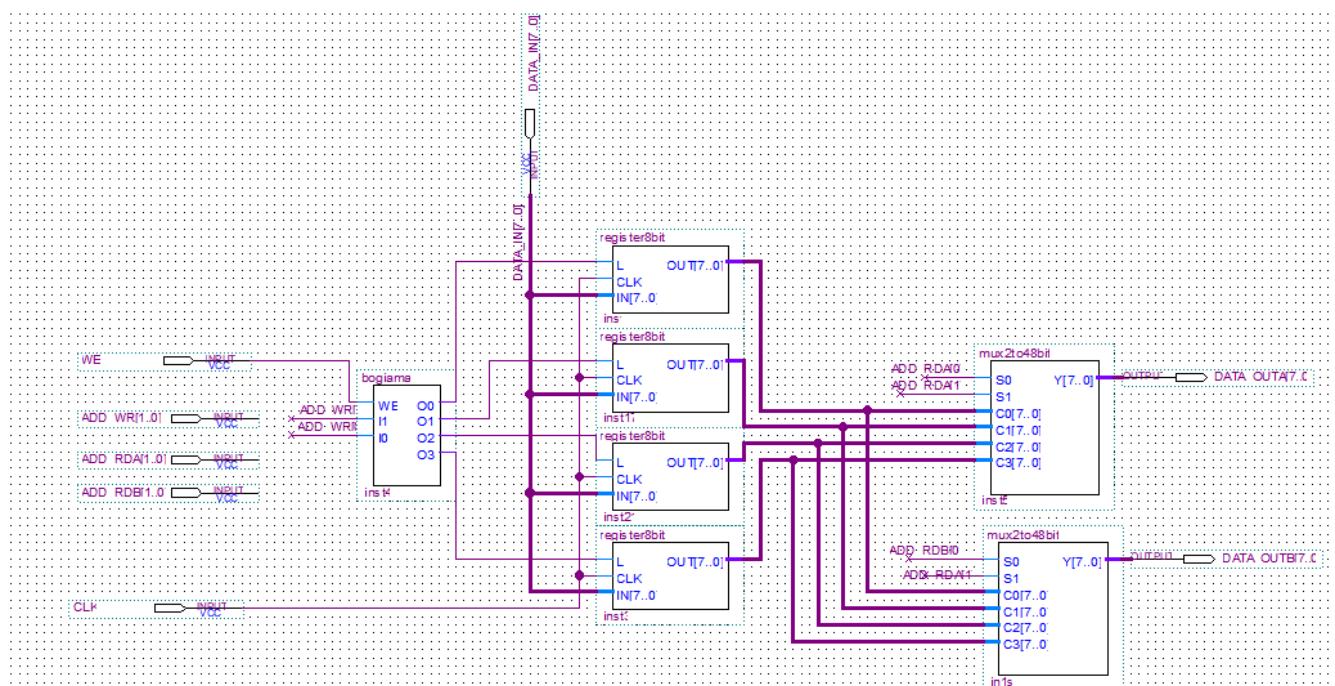


Bài 6: Thiết kế Register File 2 Read port 1 Write Port

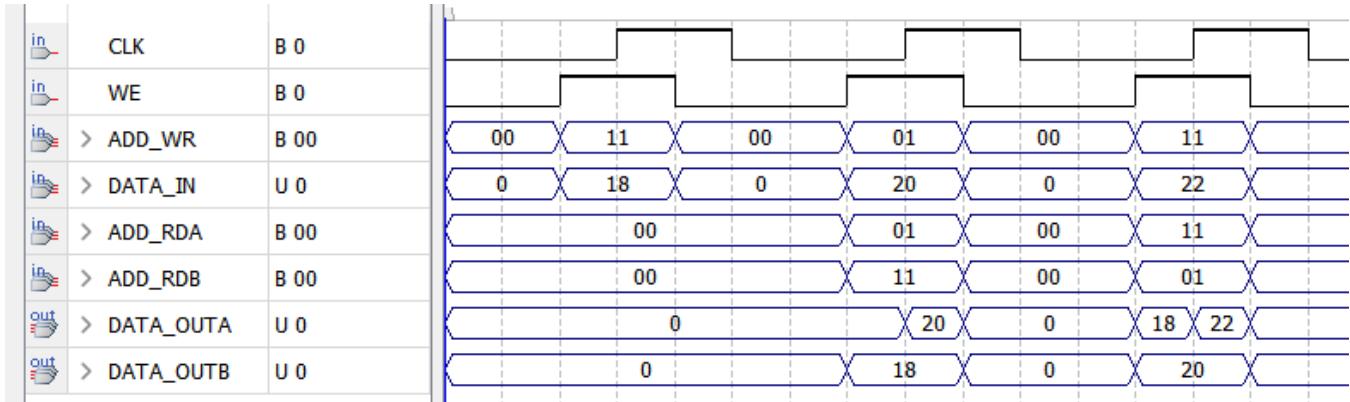
1. Tổng quan:

- Thiết kế Register file với 4 thanh ghi có 2 Read port, 1 Write port
- Đầu vào:
 - + Dữ liệu đầu vào DATA_IN[7..0] được lưu trong Register
 - + We: cho phép ghi dữ liệu
 - + ADD_WR[1..0]: địa chỉ để ghi dữ liệu, do Register file có 4 thanh ghi nên ADD_WR có độ dài 2 bit
 - + ADD_RDA[1..0]: địa chỉ đọc giá trị từ Register file ra cổng thứ 1
 - + ADD_RDB[1..0]: địa chỉ đọc giá trị từ Register file ra cổng thứ 2
 - + CLK
- Ngõ ra:

2. Thiết kế chi tiết



3. Kết quả mô phỏng



Ở CLK tích cực đầu tiên, ADD_WR = 11, DATA_IN = 18, tức là ghi giá trị 18 vào thanh ghi có địa chỉ 11

Ở CLK tiếp theo, ADD_WR = 01, DATA_IN = 20, ghi giá trị 20 vào địa chỉ 01, đồng thời, đọc giá trị ADD_RDA = 01, ADD_RDB = 11, đọc giá trị của thanh ghi có địa chỉ và 01 ra cổng A, và giá trị của thanh ghi có địa chỉ 11 ra cổng B

Ở CLK tiếp theo, ADD_WR = 11, DATA_IN = 22, ghi giá trị 22 vào thanh ghi có địa chỉ là 11, đồng thời ADD_RDA = 11, ADD_RDB = 01, đọc lần lượt giá trị của 2 thanh ghi có địa chỉ là 11 và 01 ra 2 cổng A và B

Bài 7: Thiết kế RAM

1. Tổng quan:

- Thiết kế RAM 16x8 (16 địa chỉ - 8 bit).
- Input:
 - + Tín hiệu Chip select (CS) cho phép RAM hoạt động.
 - + Tín hiệu Read/write select (RWS) cho phép ghi hoặc đọc.
 - + Tín hiệu Reset đặt tất cả giá trị gồm địa chỉ, dữ liệu thành 0.
 - + I/O là cổng cho phép cả nhập và xuất dữ liệu.
- Output: giá trị 8 bit từ địa chỉ trong RAM đã chọn.

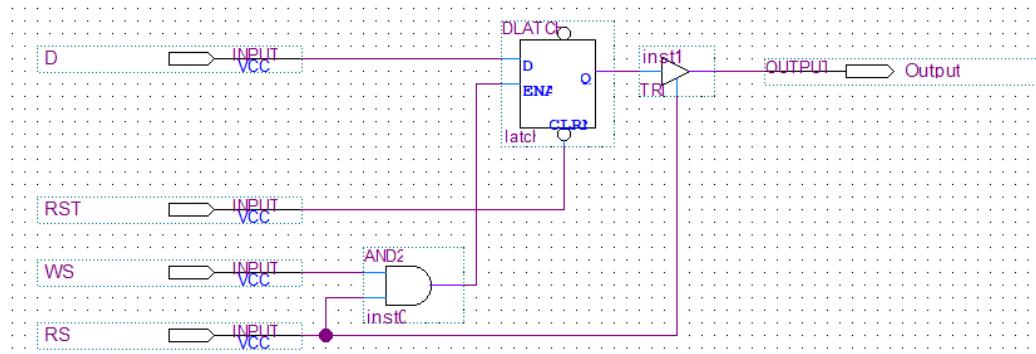
2. Thiết kế mạch:

2.1) Memory cell:

- Chức năng: lưu trữ giá trị 1 bit, có các tín hiệu điều khiển cho phép ghi, đọc và đặt giá trị về 0.
- Bảng sự thật:

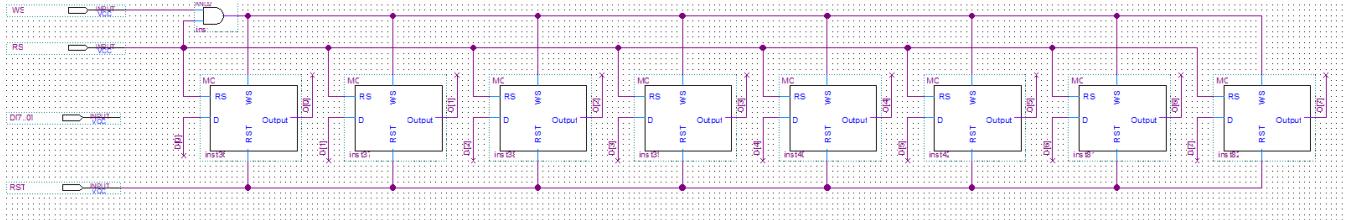
RST	RS	WS	D	Output
X	0	X	X	Z
0	1	X	X	0
1	1	0	X	Last D
1	1	1	0	0
1	1	1	1	1

Schematic:



2.2) RAM 8 bit:

- Chức năng: lưu trữ giá trị 8 bit, có các tín hiệu điều khiển cho phép ghi, đọc và đặt giá trị về 0.
- Schematic:

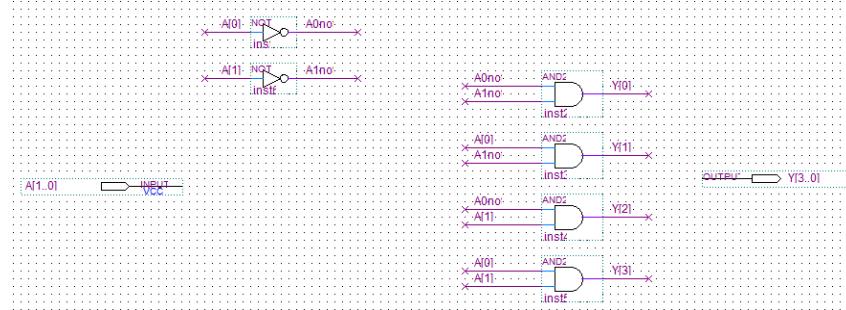


2.3) Decoder 2 to 4:

- Chức năng: nhập vào 2 bit có vai trò như địa chỉ và giải mã thành 4 bit để chọn hàng ram 8 bit muốn thao tác.
- Bảng sự thật:

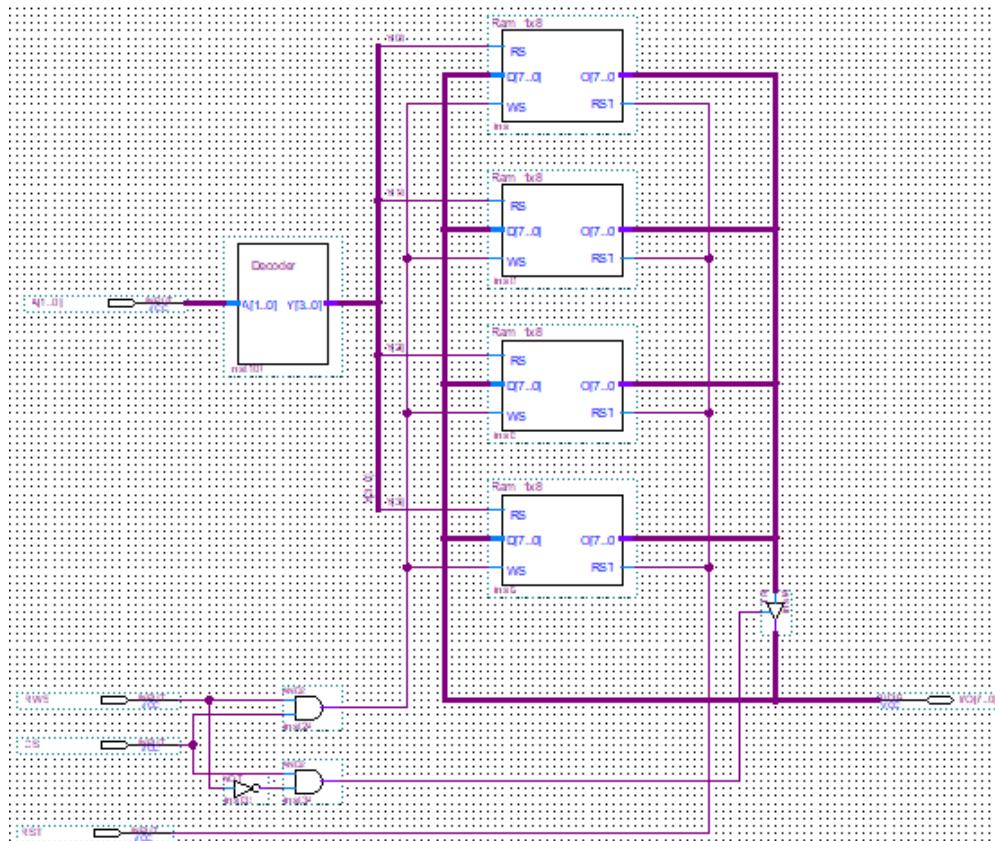
A[1..0]	Y[3..0]
00	0001
01	0010
10	0100
11	1000

- Schematic:



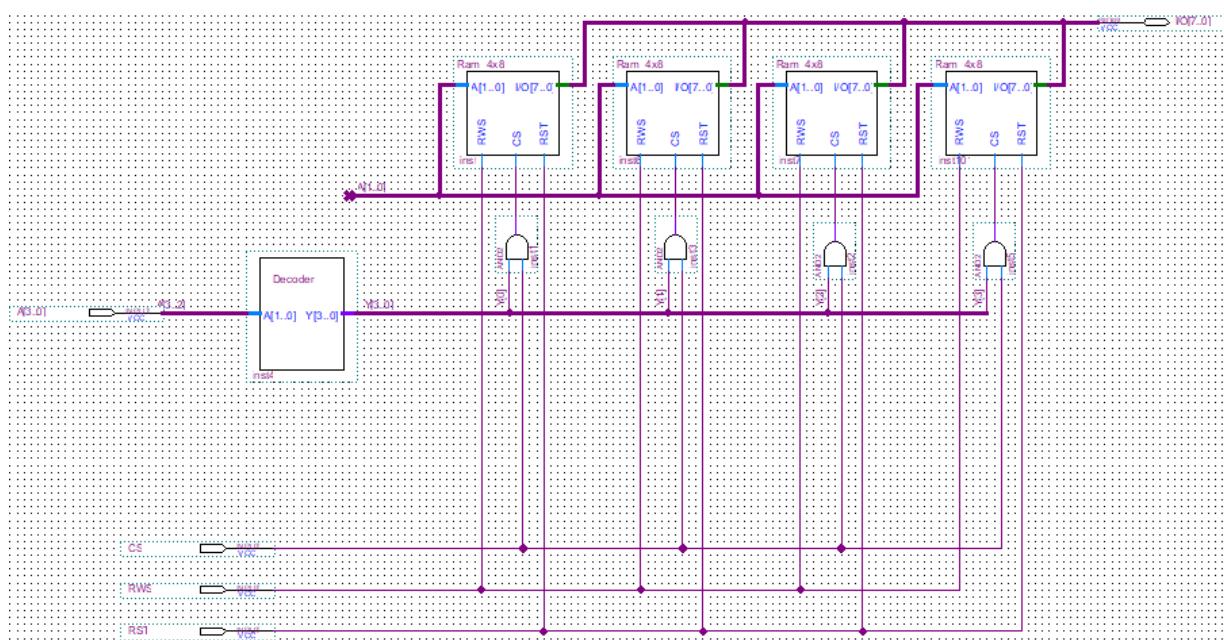
2.4) RAM 4x8:

- Chức năng: lưu trữ 4 giá trị 8 bit, giải mã 2 bit địa chỉ để chọn RAM 8 bit ở hàng tương ứng cần thao tác, có các tín hiệu để thao tác dữ liệu.
- Schematic:

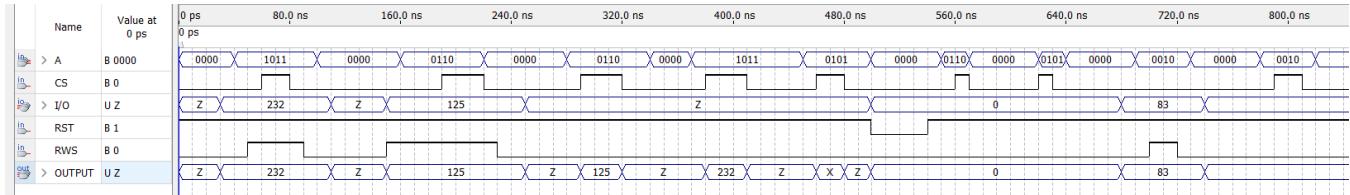


2.5) RAM 16x8:

- Chức năng: lưu trữ 16 giá trị 8 bit, nhận 4 bit địa chỉ và giải mã 2 bit có trọng số lớn nhất để chọn RAM 4x8 bit và gửi 2 bit có trọng số nhỏ nhất vào khối RAM 4x8, có các tín hiệu để thao tác dữ liệu.
- Schematic:



3. Kết quả mô phỏng:



- Nhập giá trị 232 tại địa chỉ 1011: CS = 1, RWS = 1, A = 1011, I/O = 232.
- Nhập giá trị 125 tại địa chỉ 0110: CS = 1, RWS = 1, A = 0110, I/O = 125.
- Xuất dữ liệu tại địa chỉ 0110: CS = 1, RWS = 0, A = 0110 => Output = 125.
- Xuất dữ liệu tại địa chỉ 1011: CS = 1, RWS = 0, A = 1011 => Output = 232.
- Xuất dữ liệu tại địa chỉ 0101: CS = 1, RWS = 0, A = 0101 => Output = X do chưa được nhập vào hay khởi tạo trước đó.
- Bật reset đưa giá trị toàn bộ ram trong địa chỉ thành 0.
- Xuất dữ liệu tại địa chỉ 0110: CS = 1, RWS = 0, A = 0110 => Output = 0 do đã được reset trước đó.
- Xuất dữ liệu tại địa chỉ 0101: CS = 1, RWS = 0, A = 0101 => Output = 0 do đã được reset trước đó.
- Nhập giá trị 83 tại địa chỉ 0010 nhưng không bật CS: CS = 0, RWS = 1, A = 0010, I/O = 83.
- Xuất dữ liệu tại địa chỉ 0010: CS = 1, RWS = 0, A = 0010 => Output = 0 do khi nhập thì CS = 0 nên không cho thao tác dữ liệu, chỉ giữ giá trị 0 do đã bật reset trước đó.

Bài 8: Thiết kế Stack

1. Tổng quan:

- Thiết kế mạch lưu trữ dữ liệu theo nguyên lý xếp chòng (first in last out).
- Đầu vào:
 - + Reset.
 - + Xung clock.
 - + Enable để cho phép thao tác lên ngăn xếp.
 - + Push/Pop để lựa chọn giữa thao tác thêm dữ liệu vào ngăn xếp và lấy dữ liệu từ ngăn xếp.
 - + Dữ liệu nhập vào là giá trị có độ dài 8 bit.
- Đầu ra: Giá trị được thêm gần nhất vào ngăn xếp.

2. Thiết kế mạch:

2.1) Control logic:

- Chức năng: Xử lý các tín hiệu điều khiển.
- Bảng sự thật:

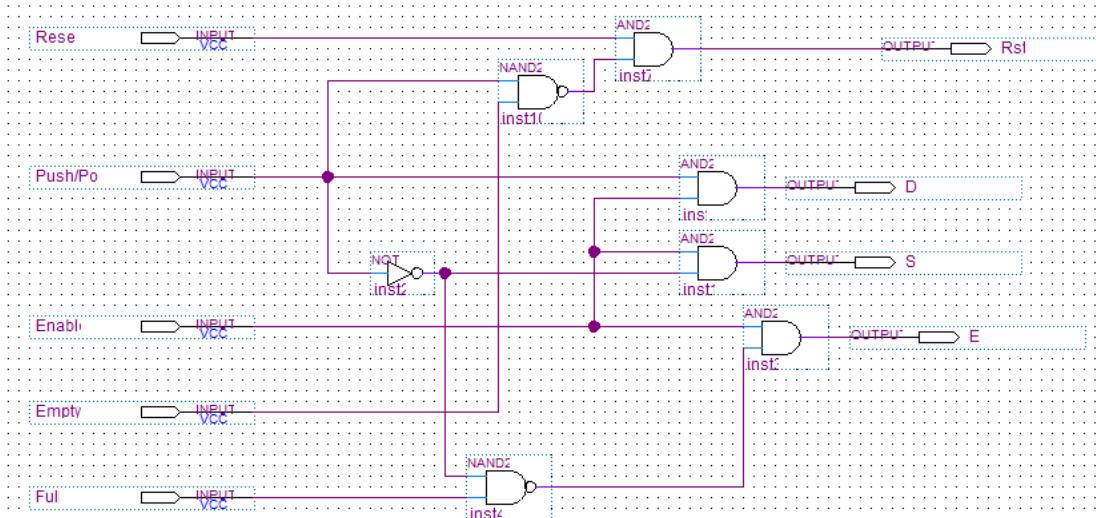
Reset	Push/pop	Empty	RST
0	X	X	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Enable	Push/pop	D
0	X	0
1	0	0
1	1	1

Enable	Push/pop	S
0	X	0
1	0	1
1	1	0

Enable	Push/pop	Full	E
0	X	X	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

- Schematic:



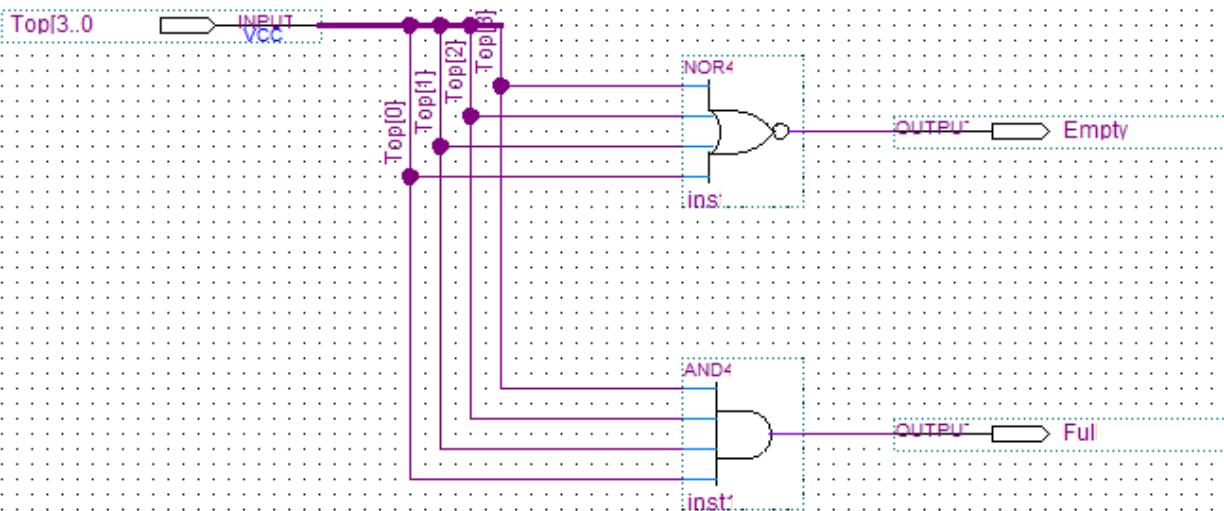
2.2) Stack check:

- Chức năng: Kiểm tra xem ngăn xếp đang trống hay đang đầy. Sử dụng ram 16 x 8 nên địa chỉ đầu tiên là 0000 và địa chỉ cuối cùng là 1111.
- Bảng sự thật:

Top	Empty
0000	1
XXX1	0
XX1X	0
X1XX	0
1XXX	0

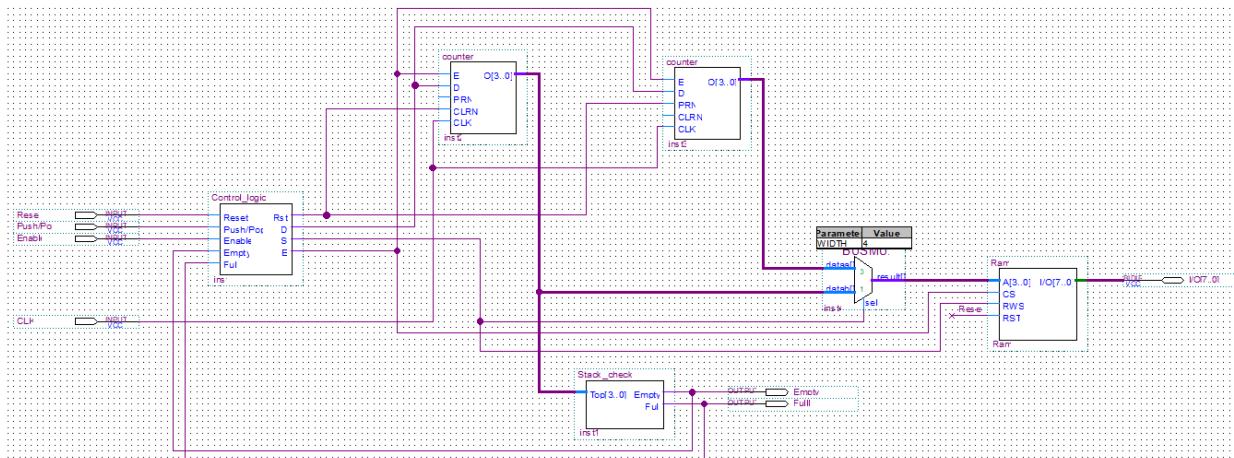
Top	Empty
1111	1
XXX0	0
XX0X	0
X0XX	0

- Schematic:

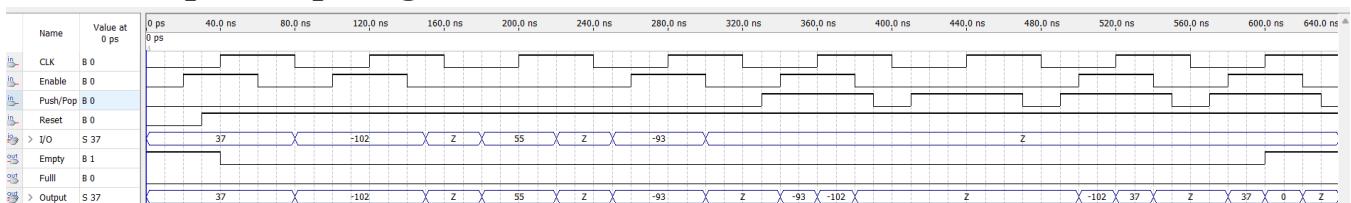


2.3) Stack:

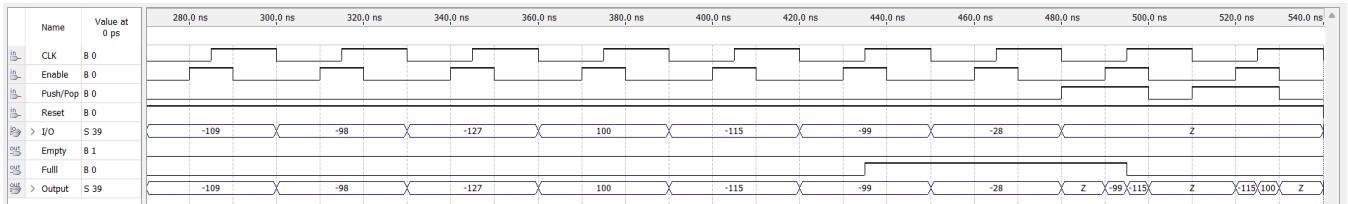
- Chức năng: Lưu trữ và truy xuất dữ liệu từ RAM theo nguyên tắc xếp chồng, sử dụng các tín hiệu để cho phép sử dụng, đặt lại, thêm hoặc lấy ra giá trị từ ngăn xếp.
- Schematic:



3. Kết quả mô phỏng:



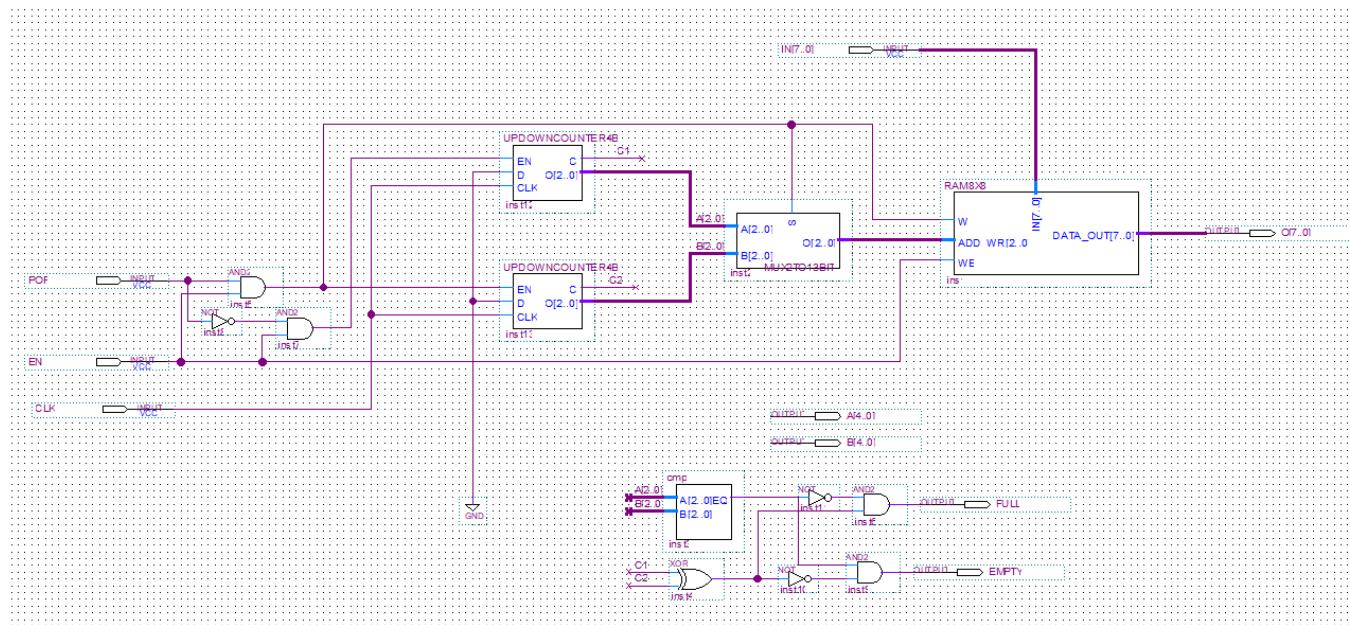
- Khi chưa nhập dữ liệu thì tín hiệu empty được bật lên 1.
 - Thực hiện nhập 37: Enable = 1, push/pop = 0, Input = 37, tín hiệu empty tắt đi vì Qtop != 0000.
 - Nhập -102: Enable = 1, push/pop = 0, Input = -102.
 - Nhập 55, không bật Enable: Enable = 0, push/pop = 0, Input = 55.
 - Nhập -93: Enable = 1, push/pop = 0, Input = -93.
 - Lấy dữ liệu từ ngăn xếp: Enable = 1, push/pop = 1 \Rightarrow Output = -93, do bộ đếm đồng thời giảm xuống 1 nên sẽ đổi qua giá trị ở vị trí trước đó.
 - Thực hiện lấy dữ liệu nhưng không bật Enable: Enable = 0, push/pop = 1 \Rightarrow Output = Z.
 - Lấy dữ liệu từ ngăn xếp: Output = -102.
 - Lấy dữ liệu từ ngăn xếp: Output = 37, lúc này vì đã lấy hết giá trị ra khỏi ngăn xếp nên empty được bật lên 1.
- => Giá trị Output được lấy ra đúng với thứ tự first in last out.



- Thực hiện nhập vào ngăn xếp đến khi đầy, tín hiệu Full được bật lên 1.
 - Khi tín hiệu Full lên 1, tiếp tục nhập thêm -28.
 - Thực hiện lấy dữ liệu thì được Output = -99 là giá trị được nhập trước khi tín hiệu Full = 1 \Rightarrow Giá trị -28 không được thêm vào ngăn xếp khi đã đầy.
- => Đúng với thiết kế của mạch.

Bài 9: Thiết kế Queue

1. Tổng quan:



Input:

- POP: tín hiệu ghi hoặc xóa dữ liệu trong QUEUE
 - EN: cho phép QUEUE hoạt động
 - CLK
 - IN[7..0]: dữ liệu để nạp vào QUEUE
 - RAM 8x8: dùng để lưu trữ dữ liệu

Output:

- O[7..0]: đọc dữ liệu từ hàng đợi

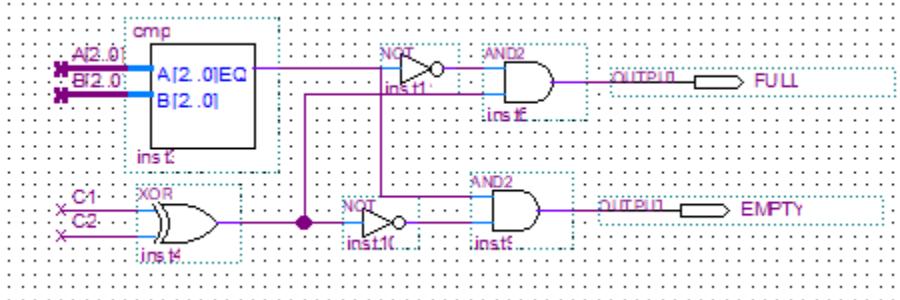
2. Thiết kế chi tiết

Gồm 2 khối counter đếm lên để lưu địa chỉ đầu và đuôi của hàng đợi

EN	POP	Bộ đếm đỉnh	Bộ đếm đuôi	RAM	Hàng đợi
0	x				

0	X	Không đổi			
1	0	Không đổi	Tăng	Đọc	Đọc dữ liệu
1	1	Tăng	Không đổi	Ghi	Ghi dữ liệu

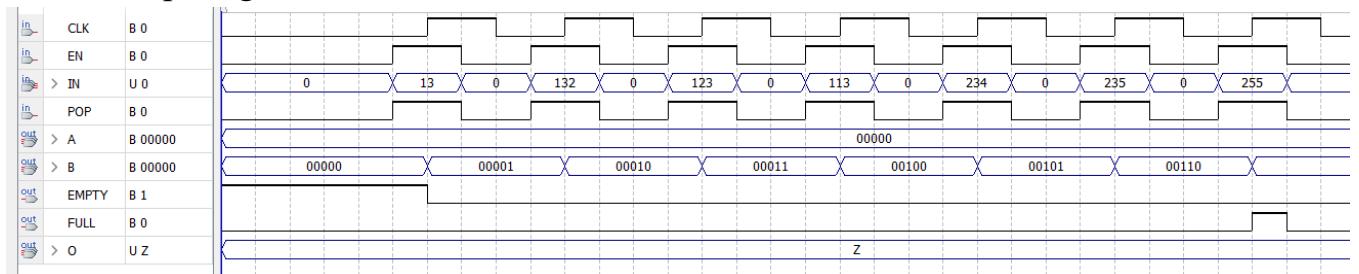
Mạch phát hiện Hàng đợi đang trống hay đã đầy



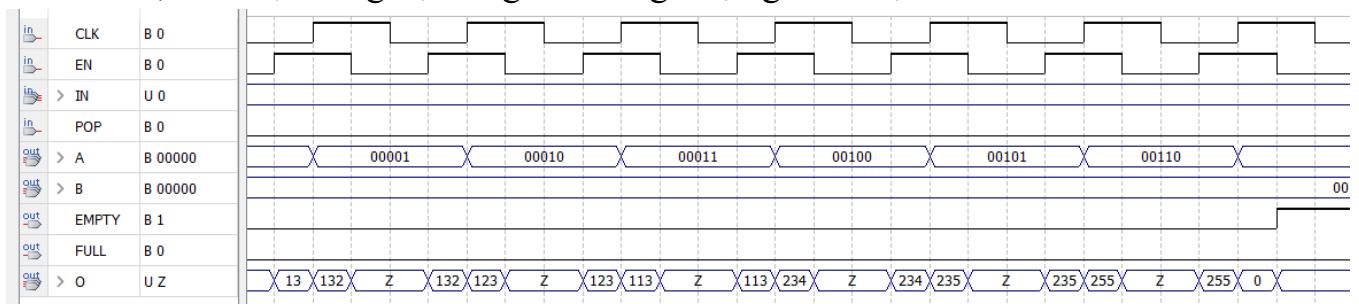
C1,C2 là 2 bit carry của counter

A[2..0] và B[2..0] là giá trị của 2 bộ đếm đinh và đích

3. Mô phỏng



Tiến hành ghi dữ liệu vào hàng đợi, sau khi bộ đếm đếm khi bộ đếm đinh đếm đến 7 thì FULL = 1, báo hiệu hàng đợi rỗng và không được ghi dữ liệu vào



Sau khi hàng đợi đầy thì ta đọc dữ liệu ra từ hàng đợi, sau khi đọc hết dữ liệu ra khỏi hàng đợi thì EMPTY = 1, báo hiệu hàng đợi rỗng

Bài 10: Tính tổng dãy số từ 1 đến n

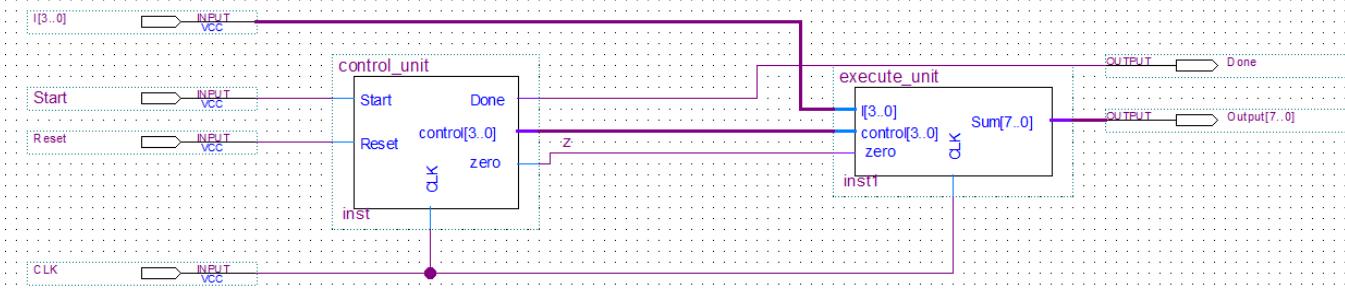
1. Tổng quan:

- Thiết kế một mạch có chức năng tính tổng các số từ 1 đến n.

- Input đầu vào:
 - + Số n: 8 bit
 - + Tín hiệu Start
 - + CLK

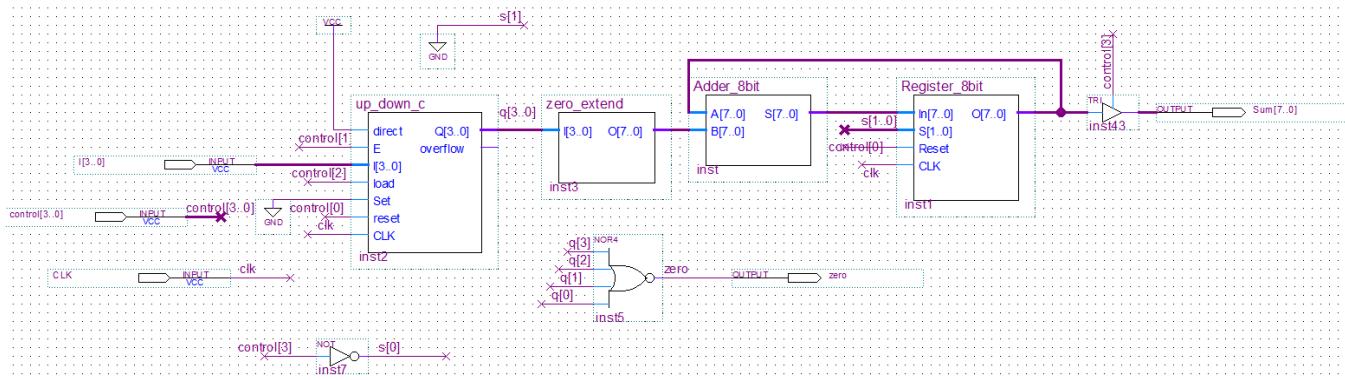
Đưa dữ liệu là số n và start=1 tại cạnh lên clk để bắt đầu thực thi.

- Output: 8bit là tổng dãy số từ 1 đến n.
 - Tín hiệu Done (dài 1 chu kỳ) tính cực khi đã có dữ liệu tổng ngõ ra.

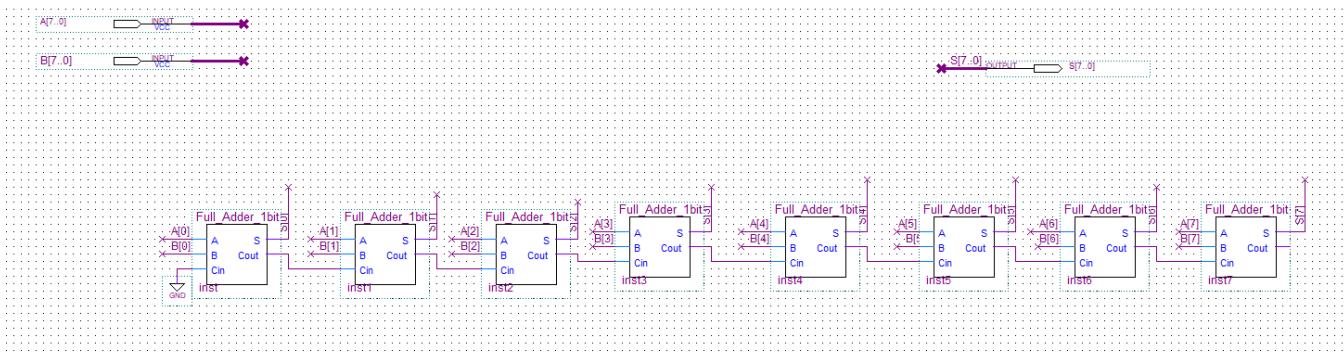


2. Thiết kế Datapath:

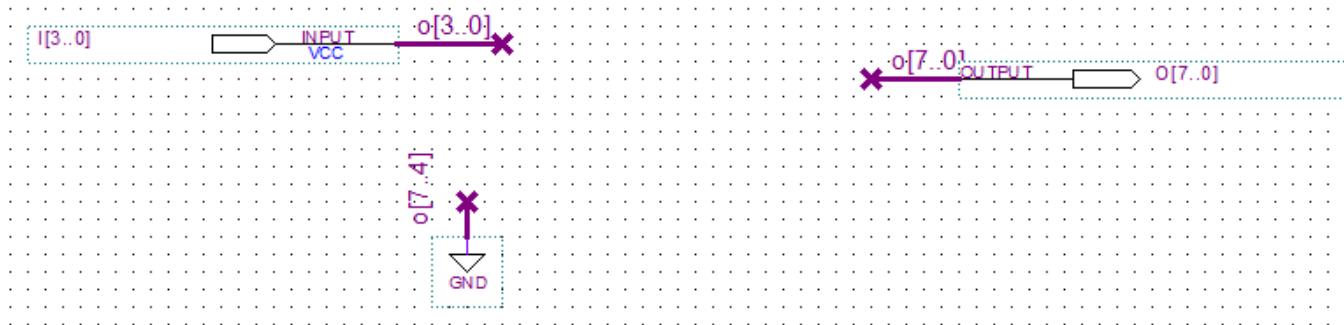
- Nạp giá trị n vào counter, tổng n và reg được cập nhật sau mỗi clk.
 - Ở cạnh lên clk đầu tiên nạp giá trị vào counter, FSM chuyển đến trạng thái S1(cho phép đếm xuống), tại cạnh lên clk tiếp theo REG cập nhật $n+0$, tại cạnh lên clk tiếp theo REG cập nhật $(n-1)+n, \dots$.
 - Tín hiệu Zero được gửi cho Controller để chuyển đến trạng thái S2(Done) tại cạnh lên clk tiếp theo.



- Khối Up-Down Counter và Register SRwPL đã được trình bày ở phần bài trước.
- Khối Adder

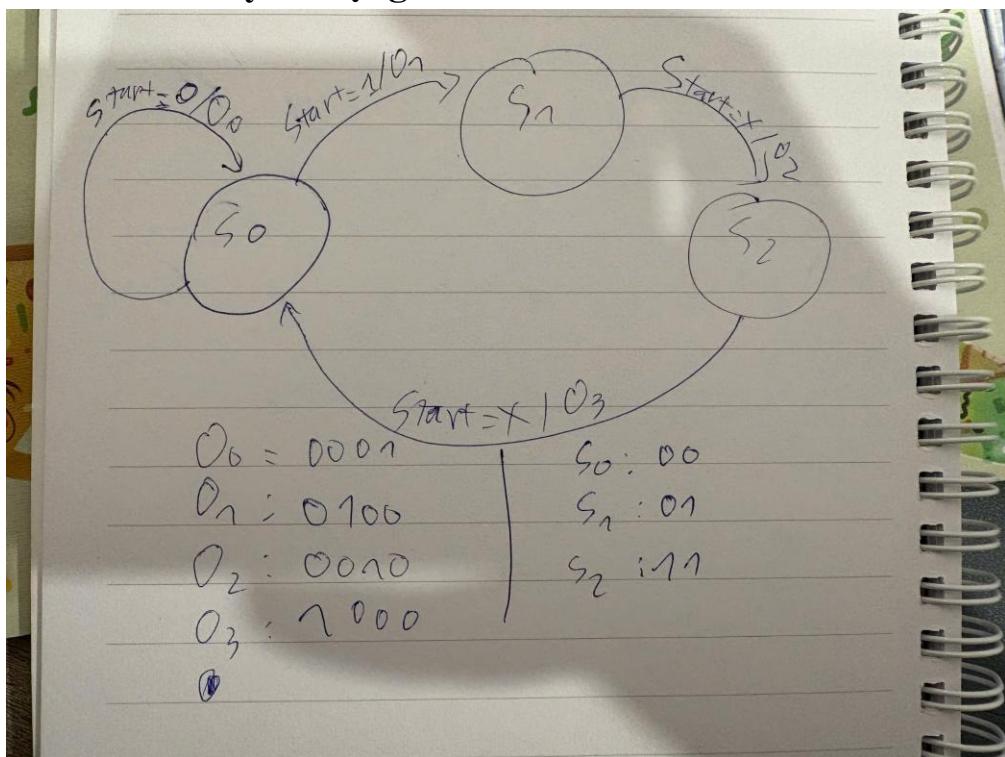


- Khối zero extend mở rộng dấu từ 4 bit thành 8 bit.



3. Thiết kế Controller:

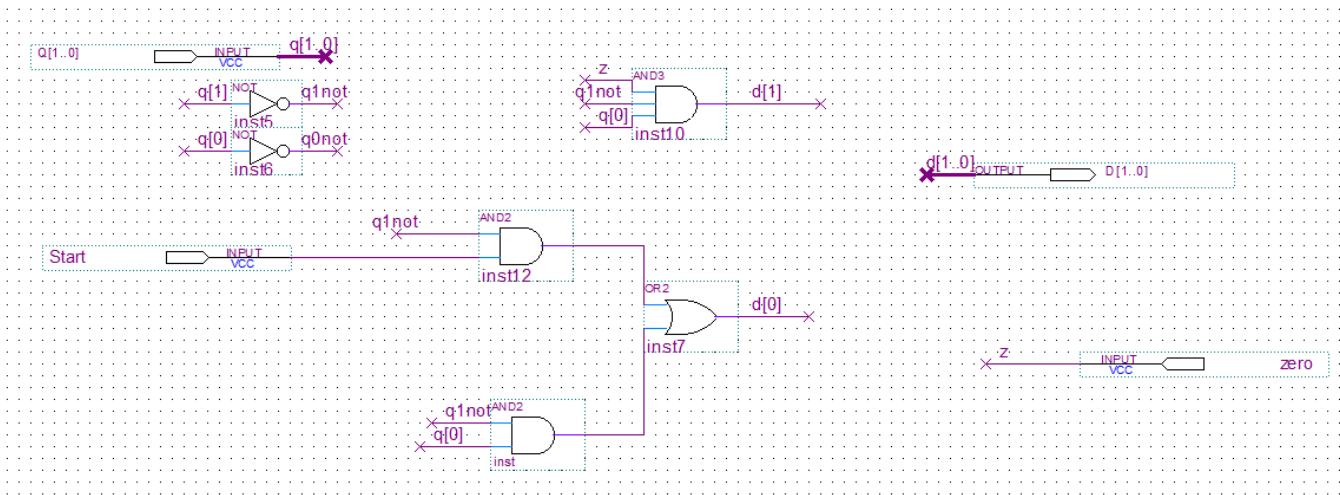
3.1. Sơ đồ chuyển trạng thái:



3.2. Khối Next State:

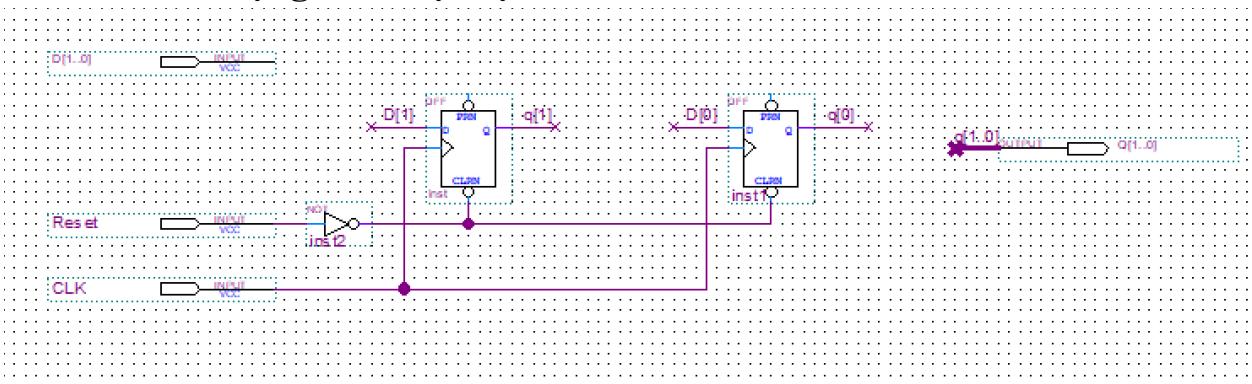
Start	Zero	Q1	Q0	D1	D0
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	X	X
0	0	1	1	0	0
0	1	0	0	0	0
0	1	0	1	1	1
0	1	1	0	X	X
0	1	1	1	0	0
1	0	0	0	0	1
1	0	0	1	0	1
1	0	1	0	X	X
1	0	1	1	0	0
1	1	0	0	0	1
1	1	0	1	1	1
1	1	1	0	X	X
1	1	1	1	0	0

- **D1:** zeroQ1'Q0
- **D0:** Q1'Q0 + StartQ1'



Hình 3.2. Khối trạng trạng kê tiếp

3.3. Khối trạng thái hiện tại:



Hình 3.3. Khối trạng thái hiện tại

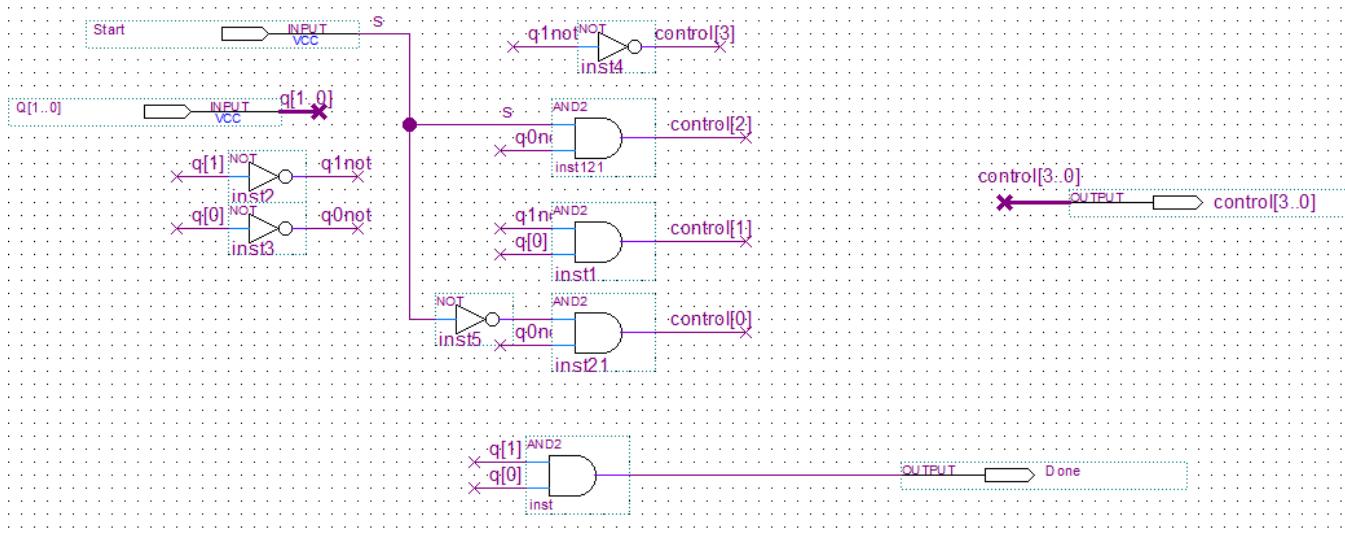
3.4. Khối mã hóa ngoã ra

Start	Q1	Q0	OE/Register Mode[3]	Load Counter[2]	En[1]	Reset[0]
0	0	0	0	0	0	1
0	0	1	0	0	1	0
0	1	0	X	X	X	X
0	1	1	1	0	0	0
1	0	0	0	1	0	0
1	0	1	0	0	1	0

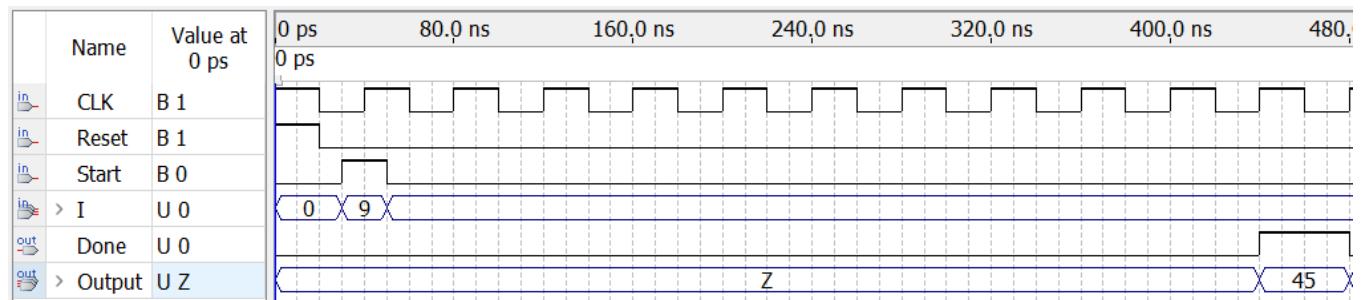
1	1	0	X	X	X	X
1	1	1	1	0	0	0

Hình 1.4. Khối mã hóa ngõ ra

- control[3]:Q1
- control[2]:StartQ0'
- control[1]:Q1'Q0
- control[0]:Start'Q0'



4. Kết quả mô phỏng:

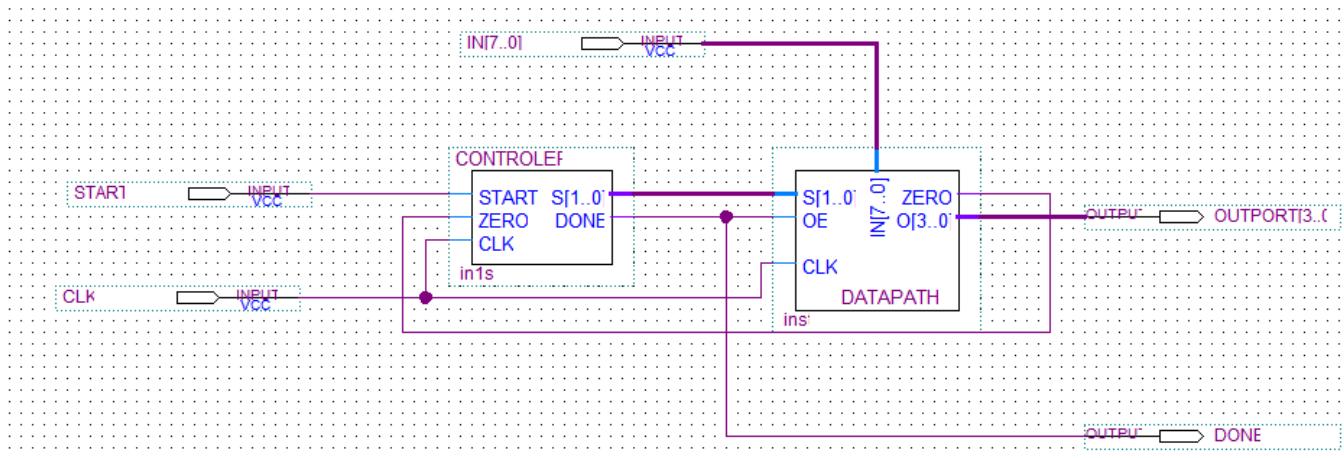


- Khi nhập input n thì sau $n+1$ chu kỳ sẽ có giá trị output. (có thể so sánh Down-counter với “1” hay “0” để tiết kiệm 1 chu kỳ).

Bài 11: Đếm số bit 1 trong input

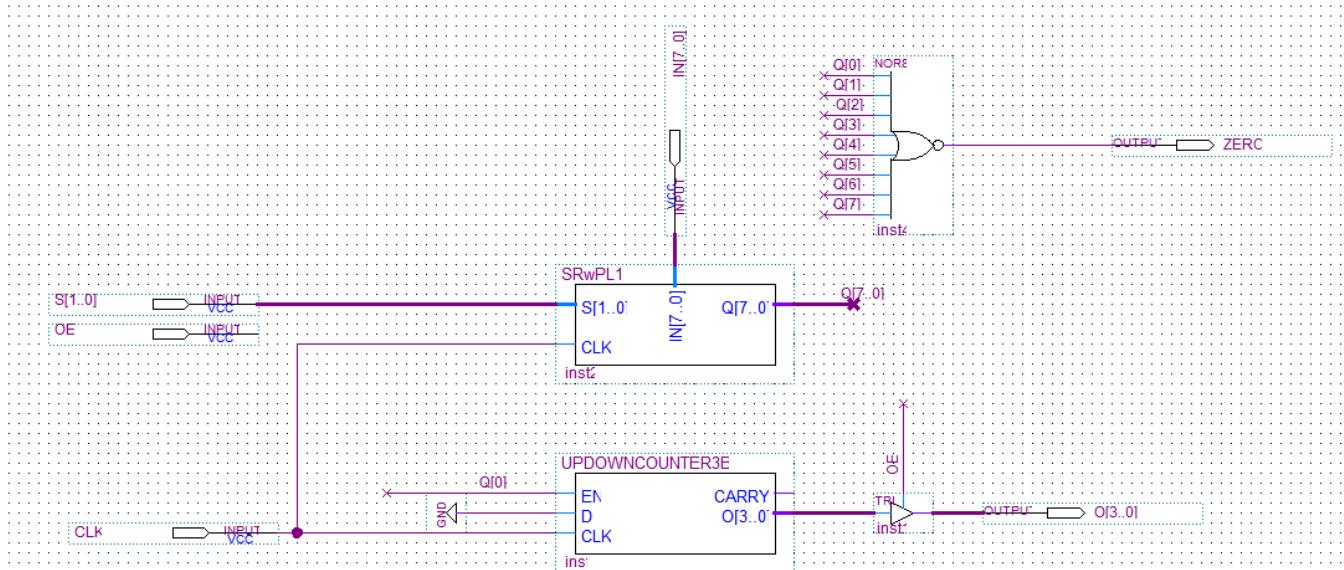
1. Tổng quan:

Kiến trúc tổng quát



- Thiết kế 1 mạch có khả năng đếm số bit 1 với input đầu vào có độ dài 8 bit
- Input:
 - + Số n: 8 bit
 - + Tín hiệu START
 - + CLK

2. Thiết kế Datapath:



Khối SRwPL là thanh ghi dịch đa chức năng

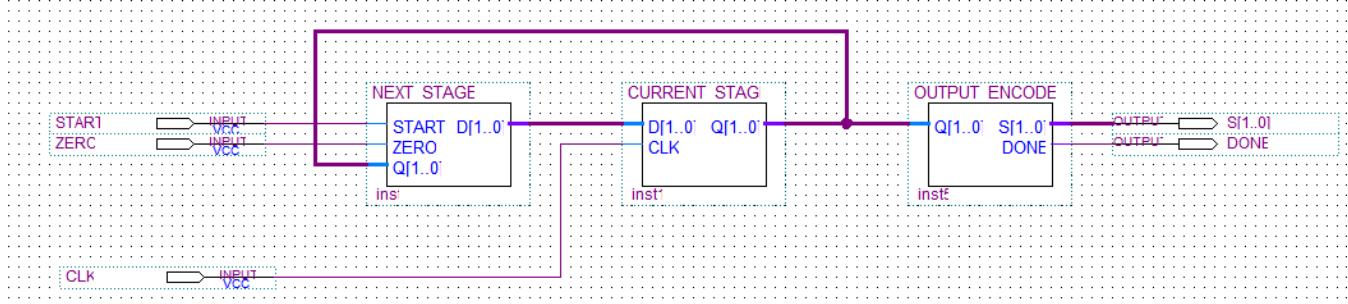
S[1..0]	Chức năng
00	Không đổi

01	Nạp dữ liệu
10	Dịch phải
11	Dịch trái

Khối UPDOWNCOUNTER3BIT: chứ năng đếm từ 0 - 15

Bit ZERO kiểm tra xem input # 0 có thể giảm số chu kì xuống thấp nhất

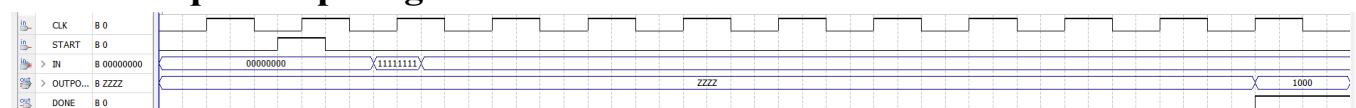
3. Thiết kế Controller:



Bảng trạng thái và các tín hiệu điều khiển

TTHT	TTKT	Output	
		S[1..0]	DONE
S0	[START = 0, S0] [START = 1, S1]	00	0
S1	s2	01	0
S2	[Data # 0, S1] [Data = 0, S3]	10	0
S3	S0	00	1

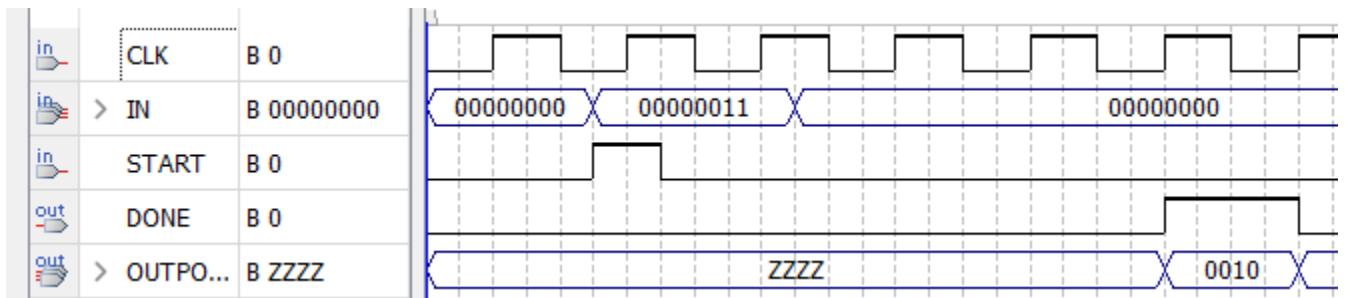
4. Kết quả mô phỏng:



Testcase1:

- Input: 11111111
- Output: 8

Nhận xét: sau 8 chu kì kể từ khi input được nạp vào thì ta có kết quả, DONE = 1 cho biết kết quả phép tính hợp lệ



Tesetcase2:

- Input: 00000011
- Output: 2

Nhận xét: sau 2 chu kì kể từ khi nhận output thì xuất ra kết quả, DONE = 1 cho biết kết quả hợp lệ

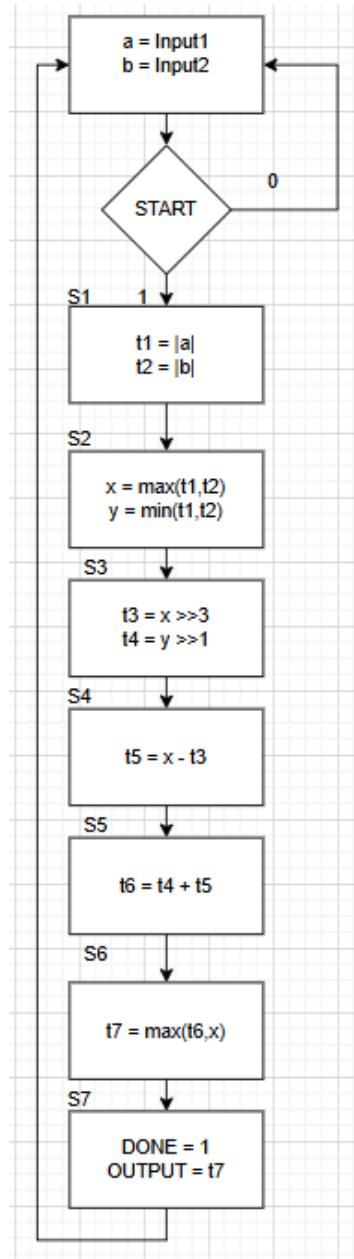
Bài 12: Thiết kế mạch tính $\sqrt{a^2 + b^2}$ xấp xỉ căn bậc 2 Register Sharing - (Graph - partitioning)

1. Tổng quan:

Mục tiêu thiết kế mạch có chức năng tính $\sqrt{a^2 + b^2} \approx \max((0.875x + 0.5y), x)$ với $x = \max(a,b)$, $y = \min(a,b)$

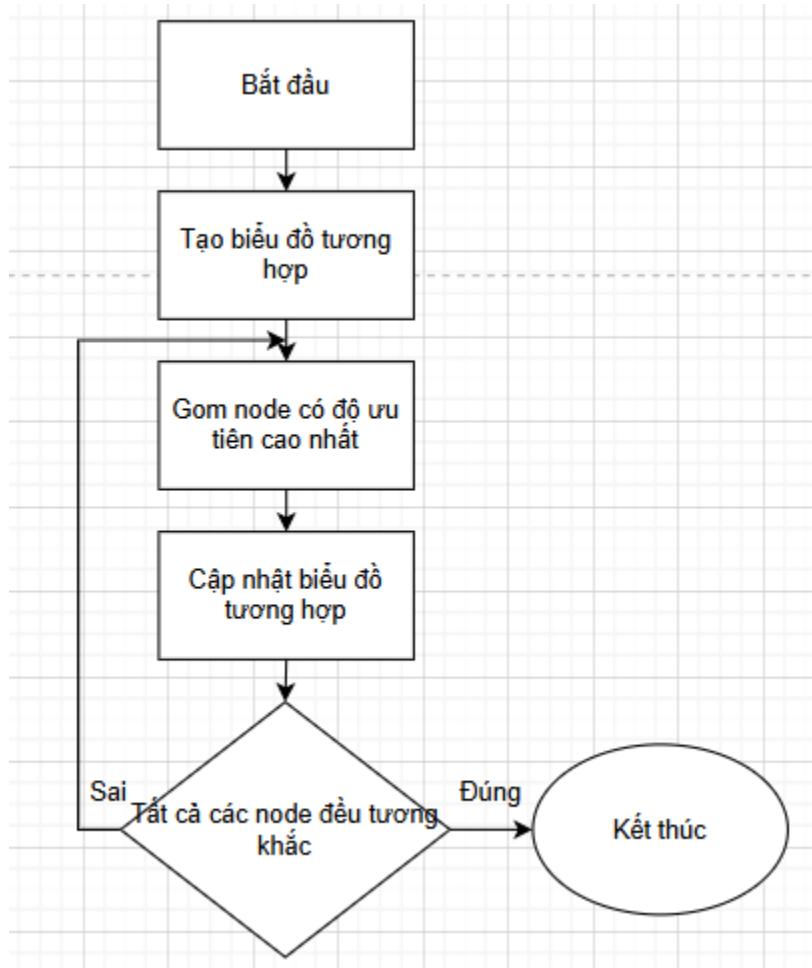
$$\sqrt{a^2 + b^2} \approx \max((0.875x + 0.5y), x)$$

Với $x = \max(a,b)$, $y = \min(a,b)$



Sơ đồ giải thuật

Lưu đồ giải thuật “graph-partitioning”



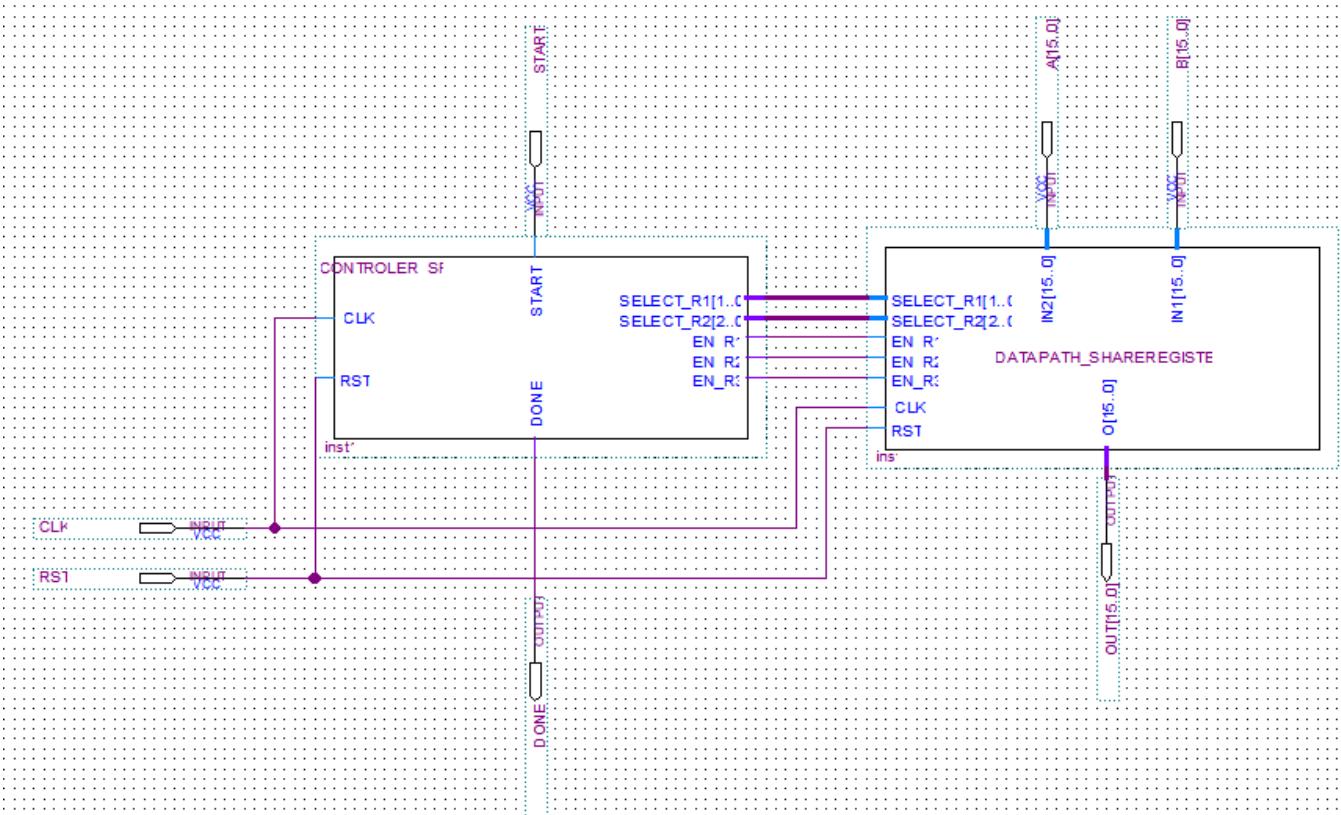
Dựa vào lưu đồ giải thuật “graph-partitioning”, ta có thể gom nhóm:

$$R1 = \{a, t1, x, t7\}$$

$$R2 = \{b, t2, y, t3, t5, t6\}$$

$$R3 = \{t4\}$$

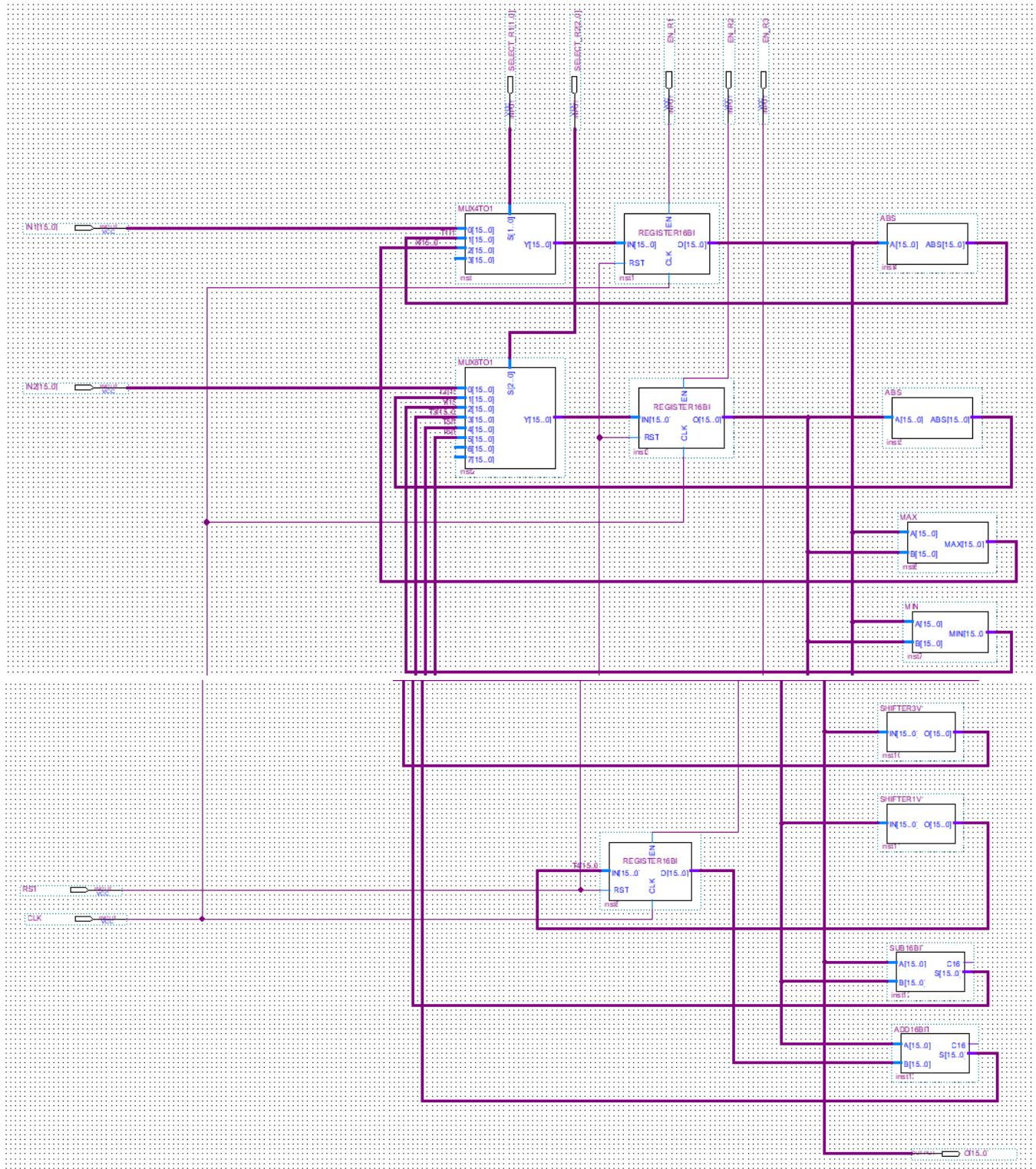
Kiến trúc tổng quan



Input:

- + START: tín hiệu bắt đầu
- + A[15..0], B[15..0]: dữ liệu đầu vào của phép tính căn bậc 2
- + DONE: tín hiệu cho biết kết quả phép tính hợp lệ
- + OUT[15..0]: kết quả của phép tính

2. Thiết kế Datapath:



Datapath bao gồm 3 thanh ghi 16 bit dùng để lưu giá trị của các biến sau khi đã phân hoạch

Bảng ngõ vào thanh ghi R1 với tín hiệu SELECT_R1[1..0]

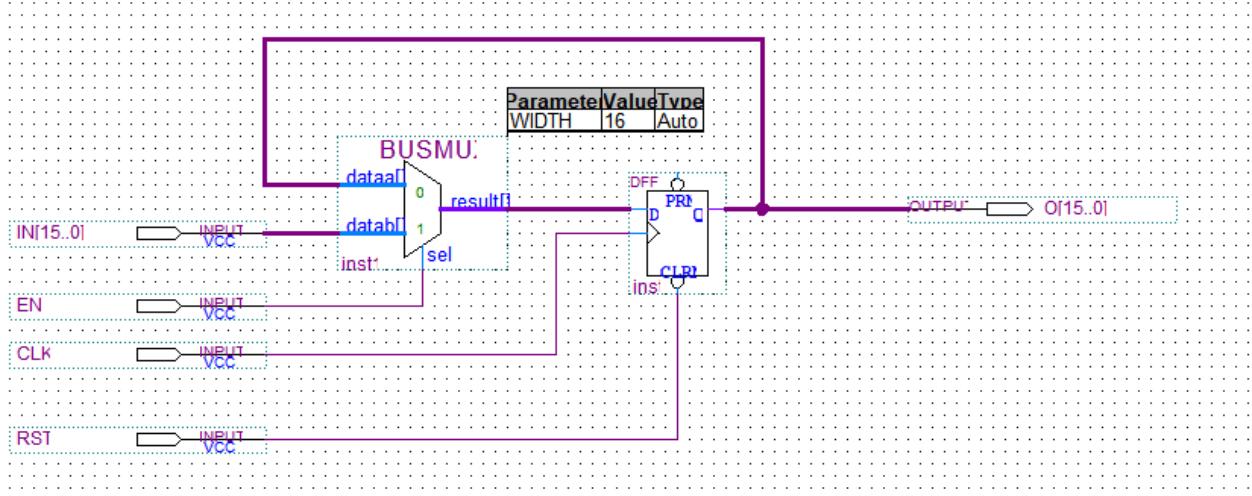
SELECT_R1[1..0]	Ngõ vào R1
00	a
01	t1,t7
10	x

Bảng ngõ vào thanh ghi R2 với tín hiệu SELECT_R2[1..0]

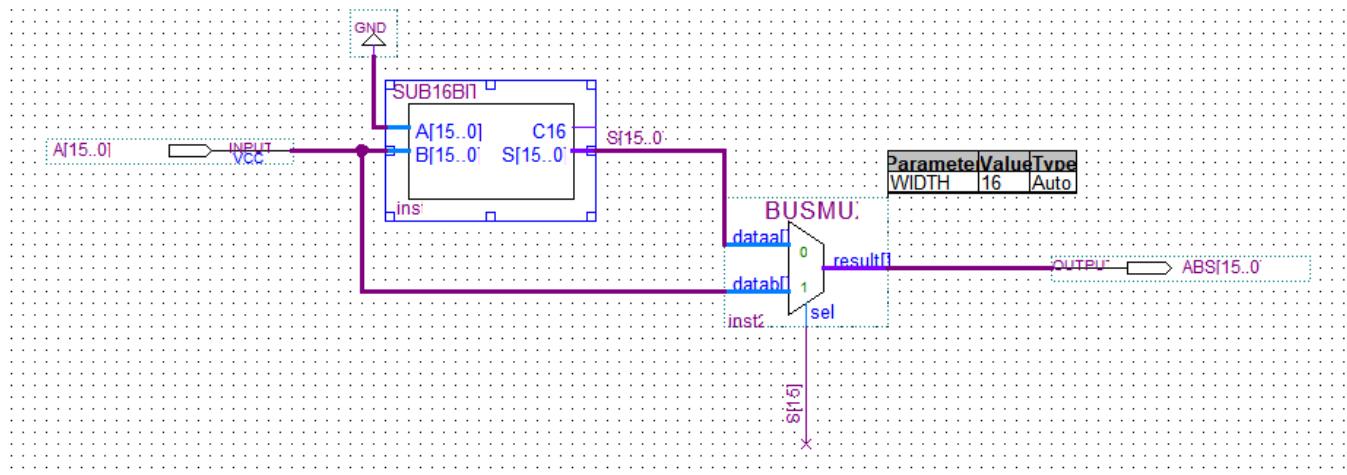
SELECT_R2[2..0]	Ngõ vào R2
000	b
001	t2
010	y
100	t2
101	t5
110	t6

Thiết kế các khối trong datapath

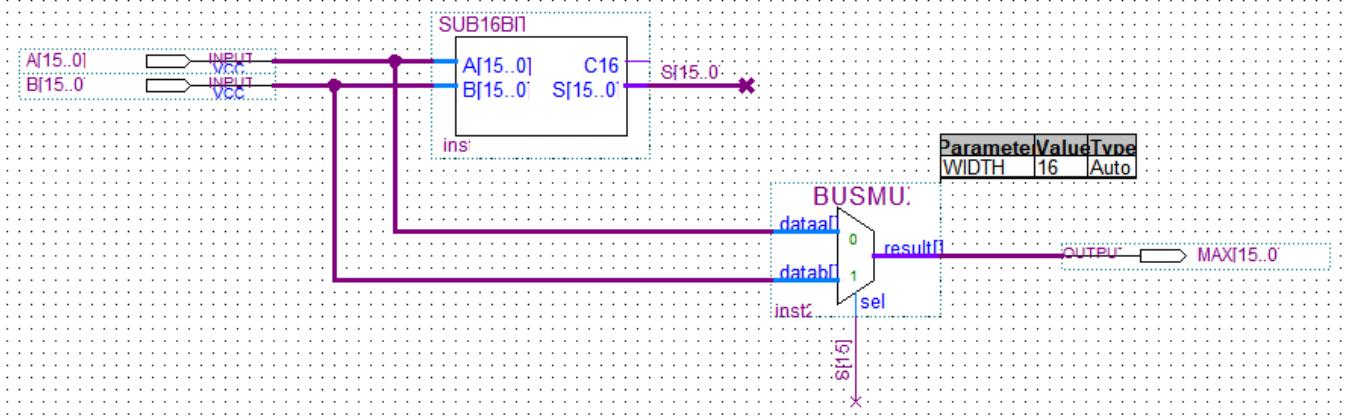
Thanh ghi 16 bit có chức năng nạp dữ liệu



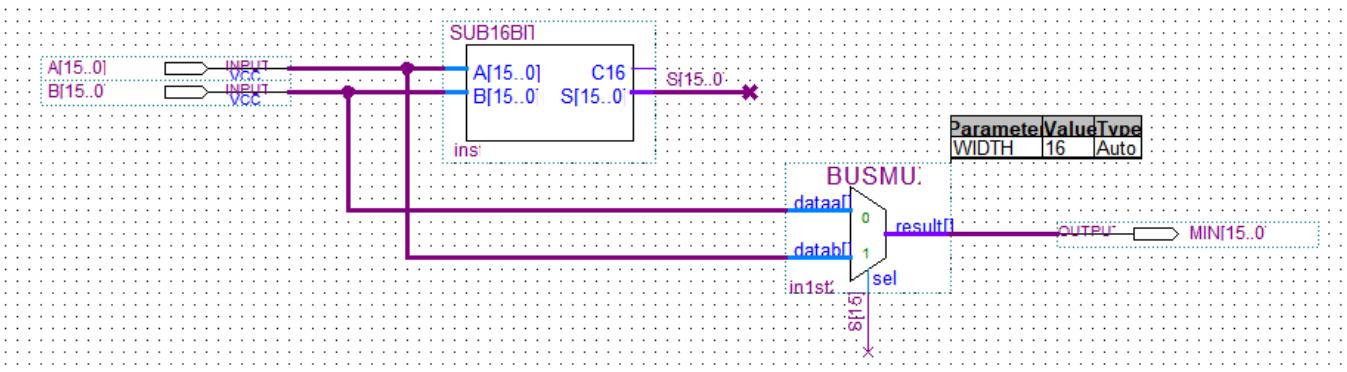
Khối tính giá trị tuyệt đối



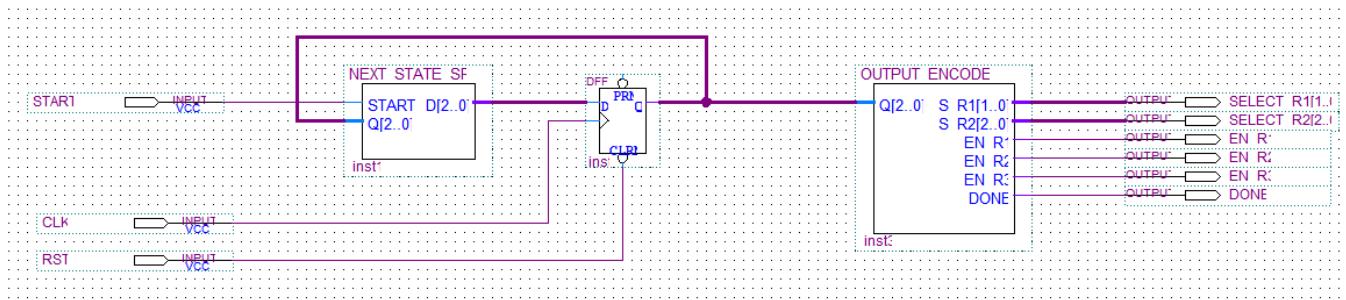
Khối tính giá trị lớn nhất



Khối tính giá trị nhỏ nhất



3. Thiết kế Controller:



Bảng trạng thái và các tín hiệu điều khiển

TTHT	TTKT		OUTPUT							
	STAR T	TT	SELECT_R1[1.. 0]	SELECT_R2[2.. 0]	EN_R 1	EN_R 2	EN_R 3	DONE	OE	
S0	0	S0	00	000	1	1	0	0	0	0
	1	S1								
S1		S2	01	001	1	1	0	0	0	0
S2		S3	10	010	1	1	0	0	0	0
S3		S4	xx	011	0	1	1	0	0	0
S4		S5	xx	100	0	1	0	0	0	0
S5		S6	xx	101	0	1	0	0	0	0
S6		S7	10	xxx	1	0	0	0	0	0
S7		S0	xx	xxx	0	0	0	1	1	

Tại S0, nếu START = 1, ta bật EN_R1 và EN_R2 để nạp giá trị input1 và input2 vào

Tại S1: EN_R1, EN_R2 cập nhật giá trị t1, t2; là kết quả của phép tính trị tuyệt đối

Tại S2: EN_R1, EN_R2 cập nhật giá trị x, y; là kết quả của phép tính min(t1,t2), max(t1,r2)

Tại S3: EN_R1 cập nhật giá trị của x >> 3, EN_R3 cập nhật giá trị của y >> 1

Tại S4: EN_R2 cập nhật kết quả của phép trừ x - t3

Tại S5: EN_R2 cập nhật kết quả của phép cộng t4 + t5

Tại S6: EN_R1 cập nhật giá trị max(t6,x)

Tại S7: OE mở lên để xuất giá trị R7 và DONE được bật để xác nhận kết quả hợp lệ

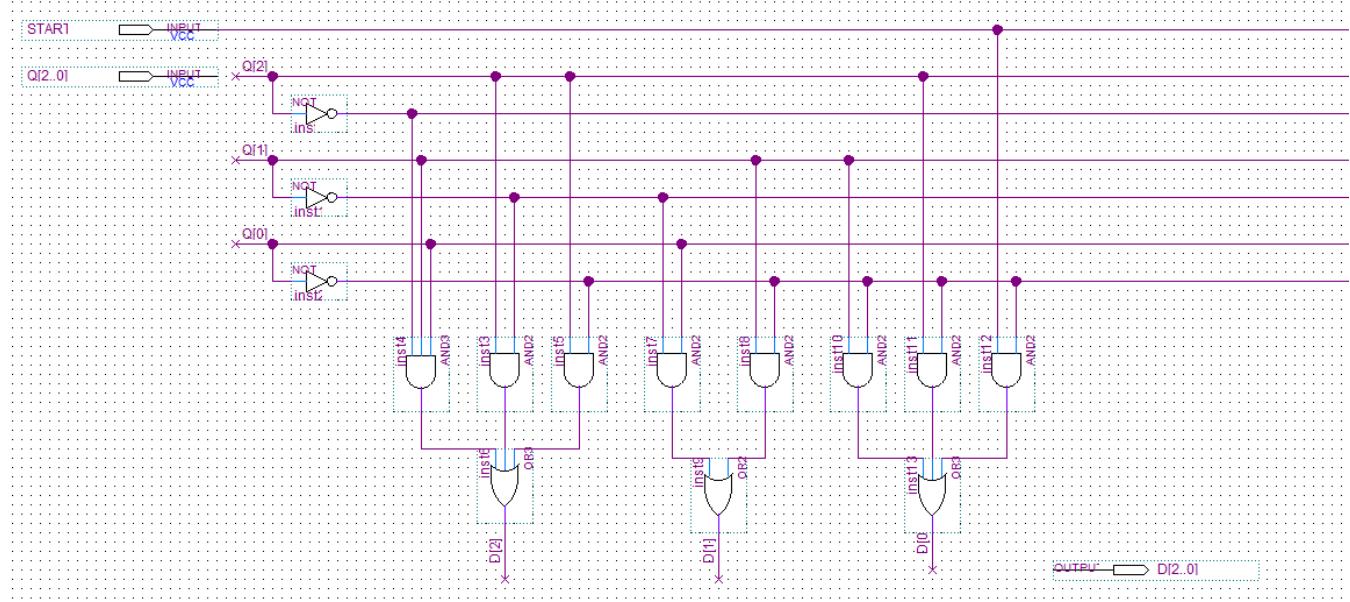
	Q2Q1Q0
S0	000
S1	001
S2	010
S3	011
S4	100
S5	101
S6	110
S7	111

NEXT_STATE

$$D2 = Q2'Q1Q0 + Q2Q1' + Q2Q0'$$

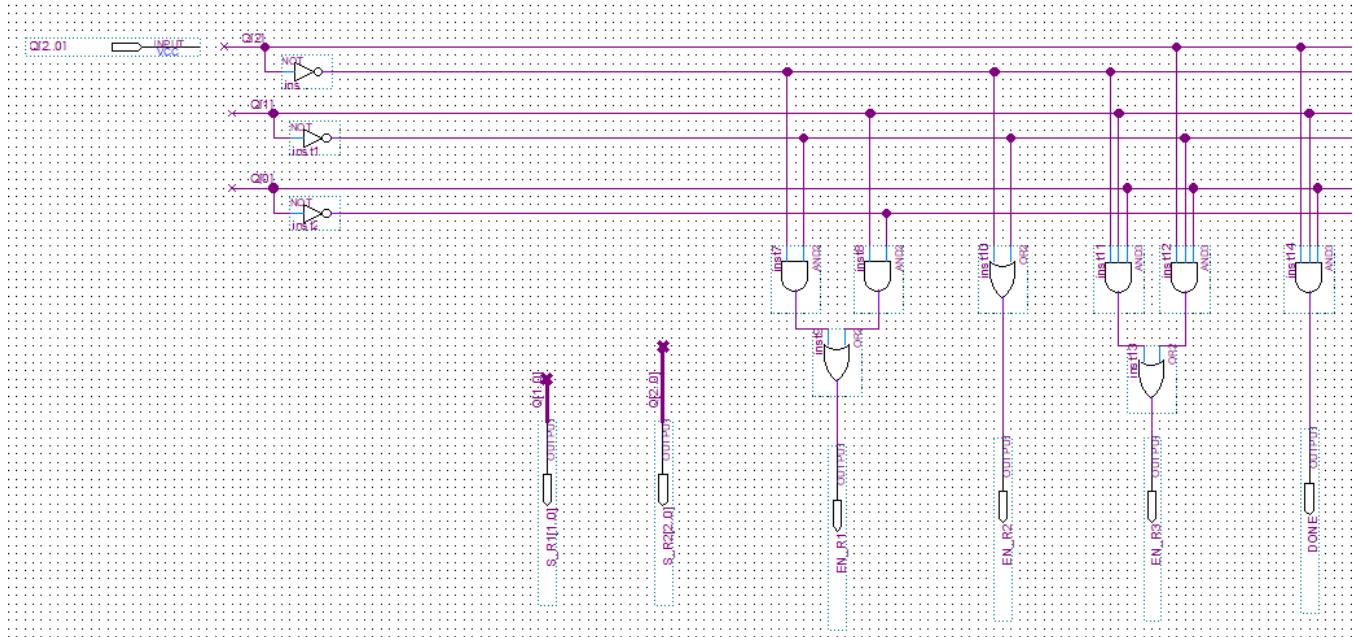
$$D1 = Q1'Q0 + Q1Q0'$$

$$D0 = Q1Q0' + Q2Q0' + \text{START}Q0$$

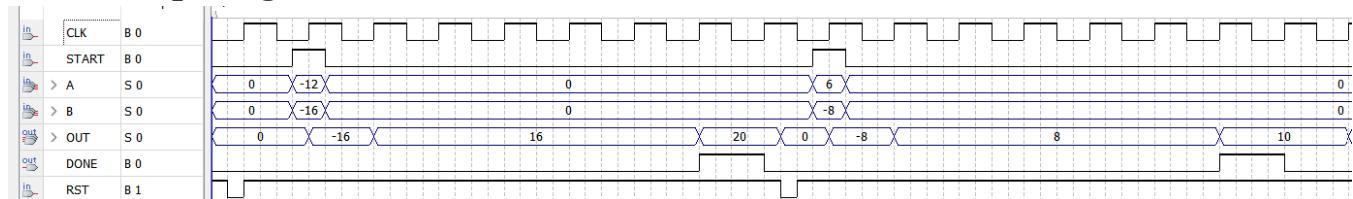


OUTPUT_ENCODER

```
SELECT_R1[1] = Q1  
SELECT_R1[0] = Q0  
SELECT_R2[2] = Q2  
SELECT_R2[1] = Q1  
SELECT_R2[0] = Q0  
EN_R1 = Q2'Q1' + Q1Q0'  
EN_R2 = Q2' + Q1' + Q0'  
EN_R3 = Q2'Q1Q0 + Q2Q1'Q0
```



4. Mô phỏng:



Testcase1:

- Input:
 - + A = -12
 - + B = -16
 - + START = 1
 - Output:
 - + OUT = 20 = $\sqrt{(-12)^2 + (-16)^2}$

Nhân xét:

- Sau 6 chu kỳ thì tín hiệu DONE bắt đầu, cho biết kết quả của ngõ OUT = 20 hợp lệ

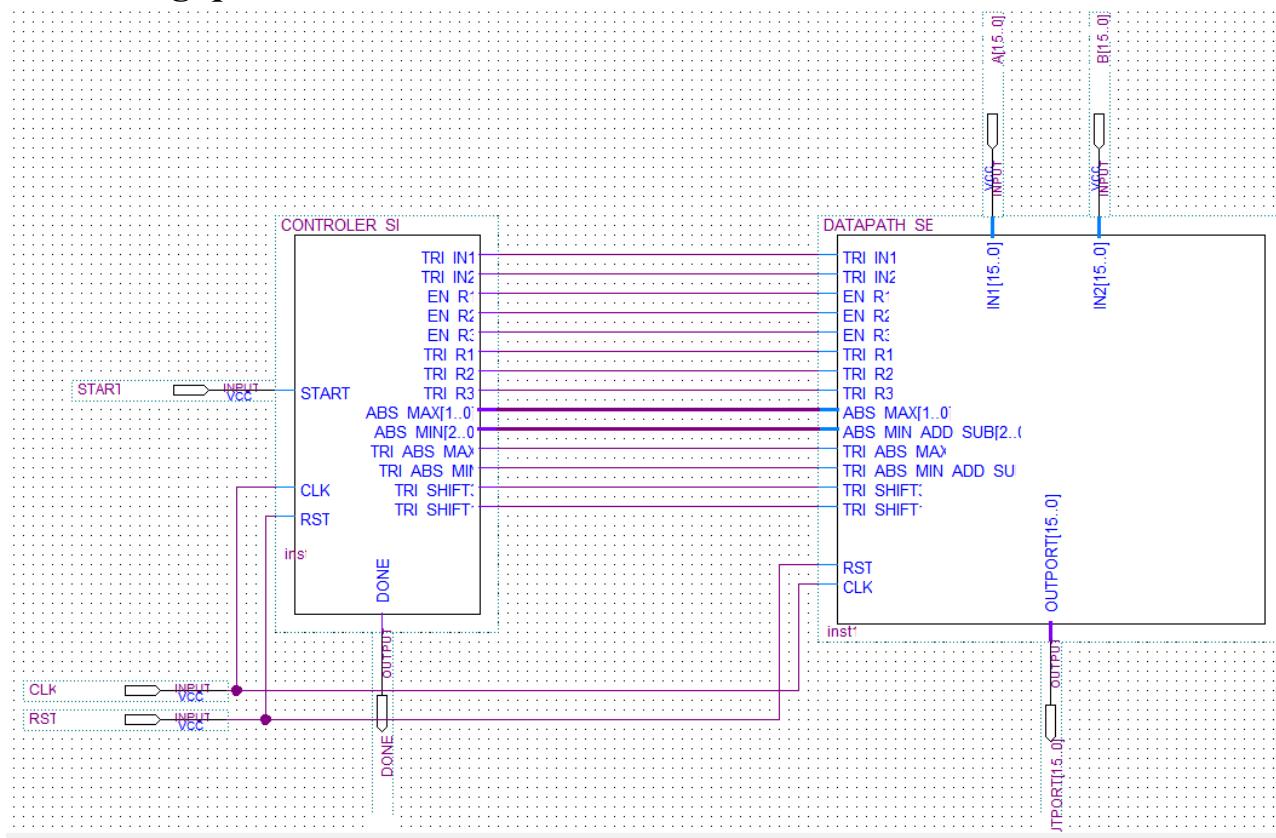
Testcase2:

- Input:
 - + A = 6
 - + B = -8
 - + START = 1
- Output:
 - + OUT = 10

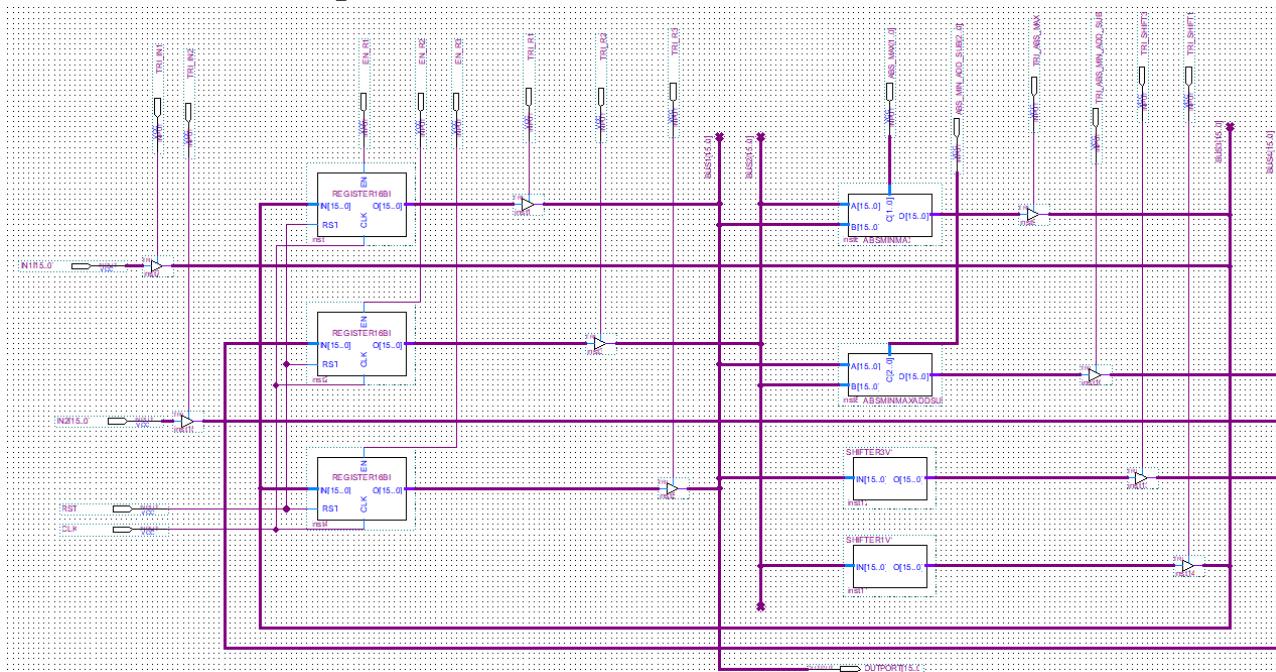
Nhận xét: sau khi tính ra kết quả của cặp input đầu tiên, ta reset hệ thống và nạp cặp input tiếp theo, tín hiệu START = 1 để tiến hành tính toán, và sau 6 chu kỳ thì kết quả được xuất ra

Bài 13: Thiết kế mạch tính xấp xỉ căn bậc 2 Bus Sharing

1. Tổng quan:



2. Thiết kế Datapath:



Gán BUS

$$BUS1 = \{A, C, D, E, H\}$$

BUS2 = {B, F, G}

BUS3 = {I, K, M}

BUS4 = {J, L, N}

Khối AU1 và AU2 được sử dụng từ bài SHARE_FUNCTION với chức năng tương tự

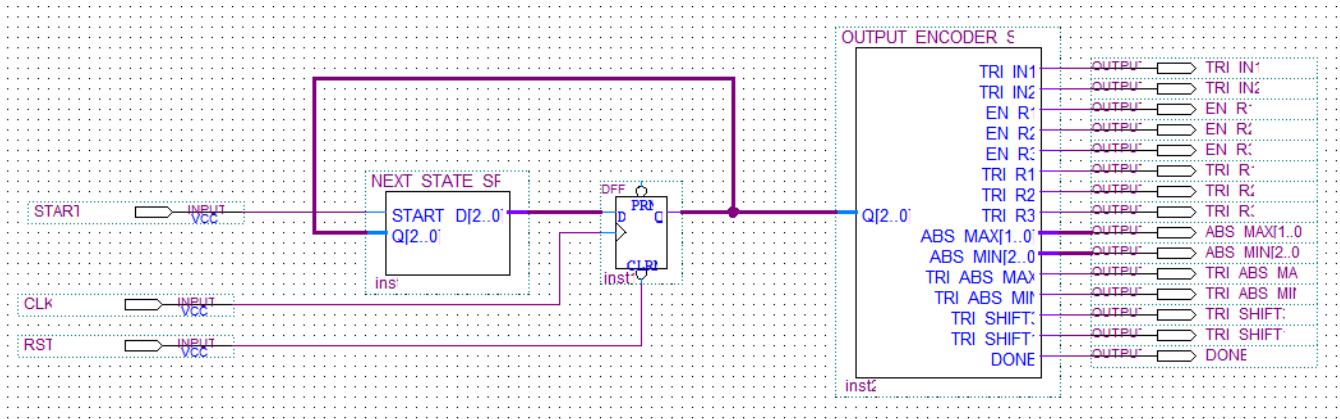
Input:

- Dữ liệu:
 - + IN1[15..0]:
 - + IN2[15..0]:
- Dữ liệu được gửi từ controller
 - + EN_R1: cho phép ghi vào R1
 - + EN_R2: cho phép ghi vào R2
 - + EN_R3: cho phép ghi vào R3
 - + TRI_R1: đọc dữ liệu từ R1
 - + TRI_R2: đọc dữ liệu từ R2
 - + TRI_R3: đọc dữ liệu từ R3
 - + ABS_MAX[1..0]: chọn chức năng của AU1
 - + ABS_MIN_ADD_SUB: chọn chức năng của AU2
 - + TRI_ABS_MAX: cho phép đọc dữ liệu từ khối AU1
 - + TRI_ABS_MIN_ADD_SUB: cho phép đọc dữ liệu từ khối AU2
 - + TRI_SHIFT3: cho phép đọc dữ liệu từ khối SHIFT3
 - + TRI_SHIFT1: cho phép đọc dữ liệu từ khối SHIFT1

Output:

- OUTPORT[15..0]: kết quả của phép tính tổng căn bậc 2

3. Thiết kế Controller:



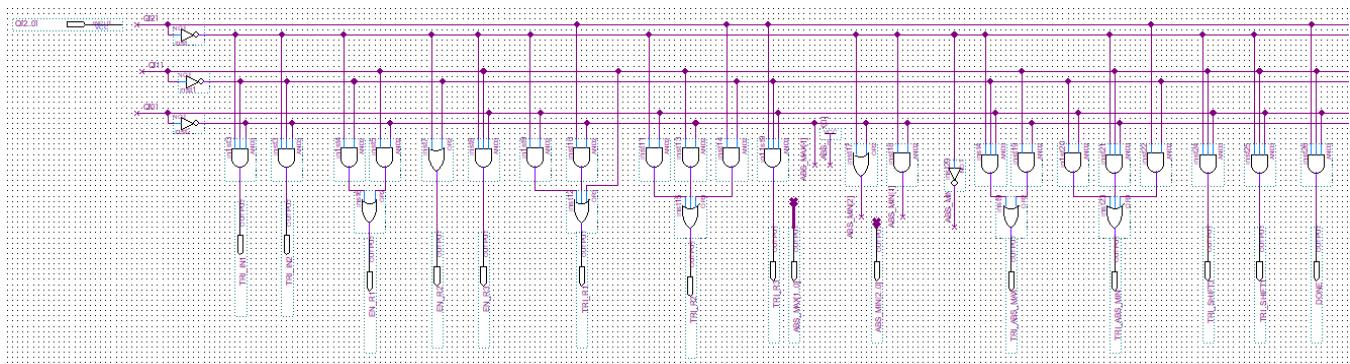
TTHT	TTKT		Output													
	STAR T		TRI_IN1	TRI_IN2	EN_R1	EN_R2	EN_R3	TRI_R1	TRI_R2	TRI_R3	ABS/MAX	ABS/MI N/ADD/ SUB	TR_AU1	TR_AU2	TRI_SH3	TRI_SH1
S0	0	S0	1	1	1	1	0	0	0	0	XX	XXX	0	0	0	0
	1	S1														
S1		S2	0	0	1	1	0	1	1	0	01	100	1	1	0	0
S2		S3	0	0	1	1	0	1	1	0	11	110	1	1	0	0
S3		S4	0	0	0	1	1	1	1	0	XX	XX	0	0	1	1
S4		S4	0	0	0	1	0	1	1	0	XX	101	0	1	0	0
S5		S6	0	0	0	1	0	0	1	1	XX	001	0	1	0	0
S6		S7	0	0	1	0	0	1	1	0	11	XXX	1	0	0	0
S7		S0	0	0	0	0	0	1	0	0	XX	XXX	0	0	0	0

Khối NEXT_STATE: sử dụng lại từ bài SHARE_REGISTER

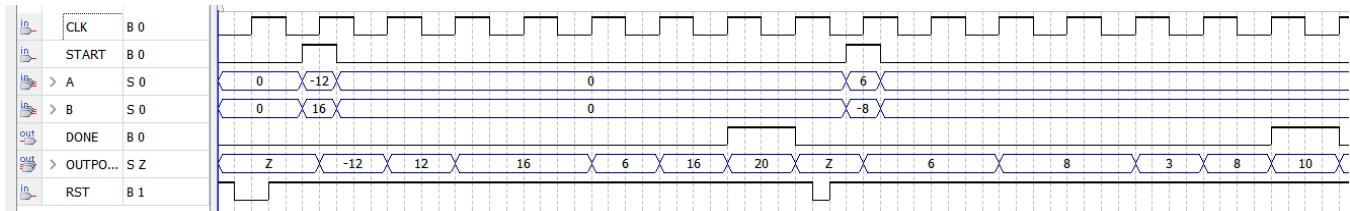
Khối OUTPUT_ENCODER

$$\text{TRI_IN1} = Q2'Q1'Q0'$$

$\text{TRI_IN1} = Q_2'Q_1'Q_0'$
 $\text{EN_R1} = Q_2'Q_1' + Q_1Q_0'$
 $\text{EN_R2} = Q_2' + Q_1'$
 $\text{EN_R3} = Q_2'Q_1Q_0$
 $\text{TRI_R1} = Q_2'Q_0 + Q_1 + Q_2Q_0'$
 $\text{TRI_R2} = Q_2'Q_0 + Q_1Q_0' + Q_2Q_1'$
 $\text{TRI_R3} = Q_2Q_1'Q_0$
 $\text{ABS_MAX[1]} = Q_0'$
 $\text{ABS_MAX[0]} = 1$
 $\text{ABS_MIN_ADD_SUB[2]} = Q_2' + Q_0'$
 $\text{ABS_MIN_ADD_SUB[1]} = Q_2'Q_0'$
 $\text{ABS_MIN_ADD_SUB[0]} = Q_2$
 $\text{TRI_ABS_MAX} = Q_2'Q_1'Q_0 + Q_1Q_0'$
 $\text{TRI_ABS_MIN_ADD_SUB} = Q_1'Q_0 + Q_2'Q_1Q_0' + Q_2Q_1'$
 $\text{TRI_SHIFT3} = Q_2'Q_1Q_0$
 $\text{TRI_SHIFT1} = Q_2'Q_1Q_0$



4. Mô phỏng:



Testcase1

- Input:
 - + A = -12
 - + B = 16
- Output:
 - + OUTPORT = 20

Nhận xét: sau 6 chu kì kể từ khi tín hiệu START = 1, OUTPORT = 20 và DONE = 1, cho biết 20 là kết quả hợp lệ của phép tính căn

Testcase2

- Input:
 - + A = 6
 - + B = -8
- Output:
 - + OUTPORT = 10

Nhận xét: Sau khi tính xong cặp dữ liệu thứ nhất, ta RST = 1, reset hệ thống và nạp tiếp bộ dữ liệu tiếp theo, kể từ khi tín hiệu START = 1 thì sau 6 chu kỳ tiếp theo OUTPORT = 10, DONE = 1 cho biết 10 là kết quả của phép tính căn

Bài 14: Thiết kế mạch tính xấp xỉ căn bậc 2 Functional Unit Sharing

1. Tổng quan:

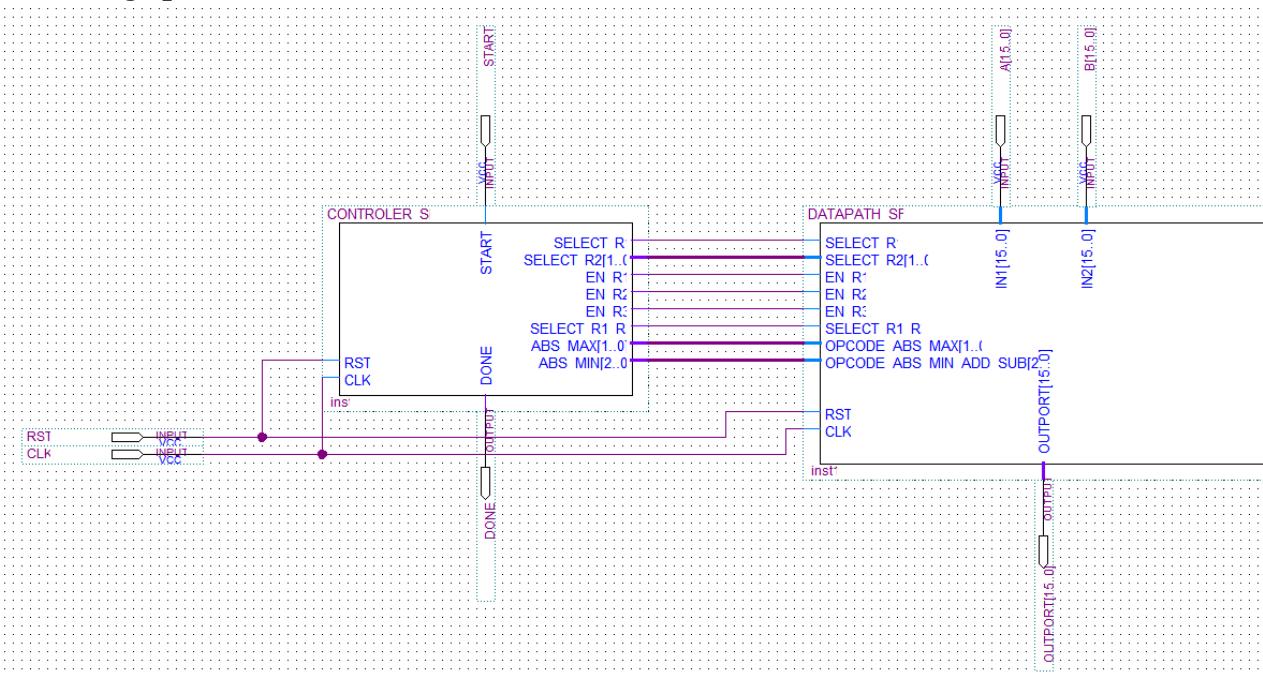
Với mục tiêu thiết kế tối ưu mạch thực hiện chức năng tính tổng căn bậc 2 của 2 số và sử dụng thuật toán đã trình bày ở mục trên

Sử dụng giải thuật “graph-partitioning” để thực hiện việc chia sê khói chức năng, ta có thể gom nhóm các khói tính toán với:

$$AU1 = [|b|, \min, +, -]$$

$$AU2 = [|a|, \max]$$

Tổng quan thiết kế



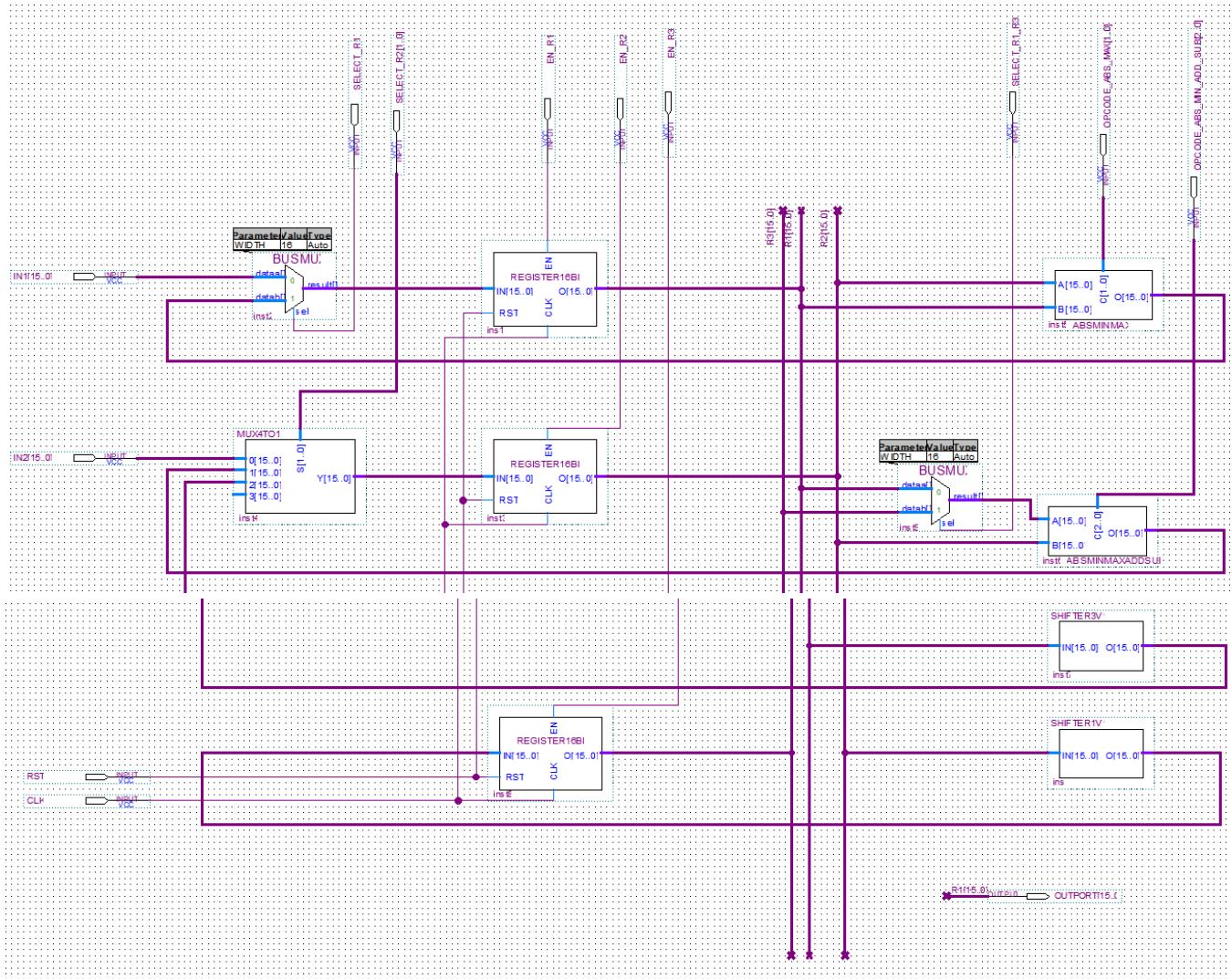
Input:

- + $A[15..0]$: toán hạng 1
- + $B[15..0]$: toán hạng 2
- + START: tín hiệu bắt đầu
- + RST: tín hiệu reset hệ thống
- + CLK: tín hiệu xung Clock

Output:

- + OUTPORT[15..0]: kết quả của phép tính căn bậc 2

2. Thiết kế Datapath:



Datapath gồm 3 thanh ghi:

$$R1 = \{a, t1, x, t7\}$$

$$R2 = \{b, t2, y, t3, t5, t6\}$$

$$R3 = \{t4\}$$

Ngõ vào của các thanh ghi được điều khiển bởi khối MUX trước nó
Như đã phân tích ở trên, ta sẽ xây dựng 2 khối AU1 và AU2 có các chức năng như sau:

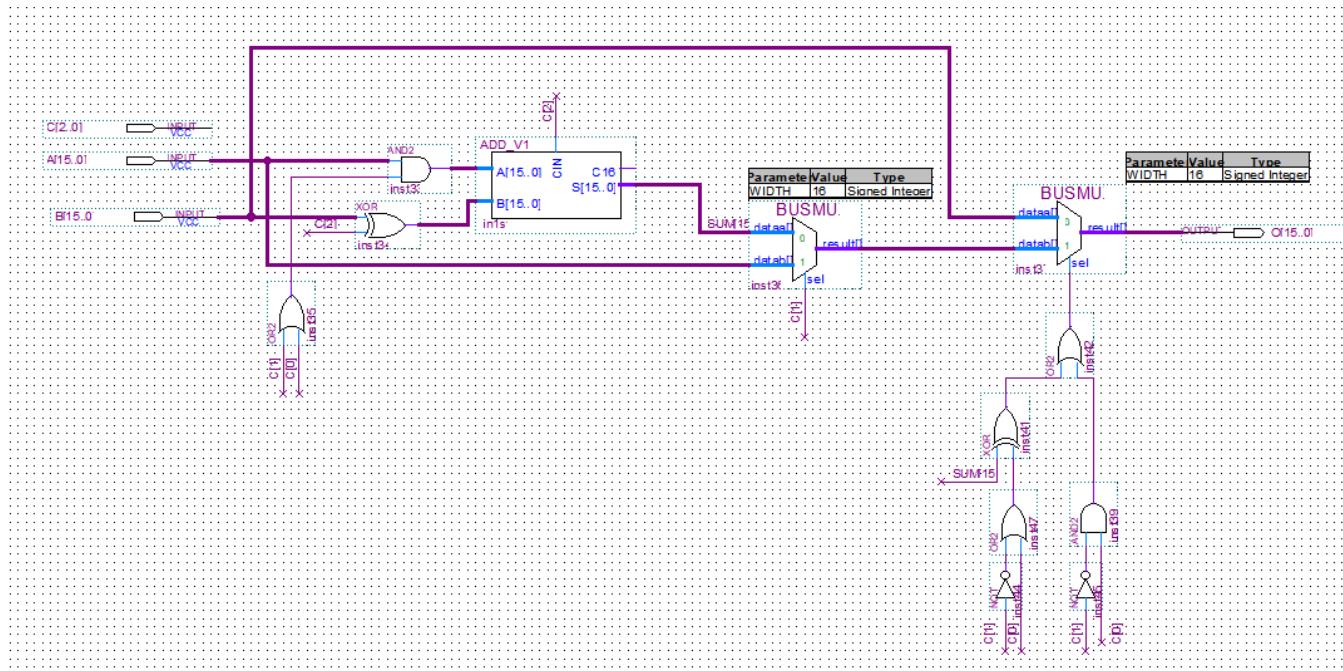
$$AU1 = [|b|, \min, +, -]$$

$$AU2 = [|a|, \max]$$

Bảng chức năng khối AU1

C[2..0]	Chức năng
001	add
100	abs
101	sub
111	max

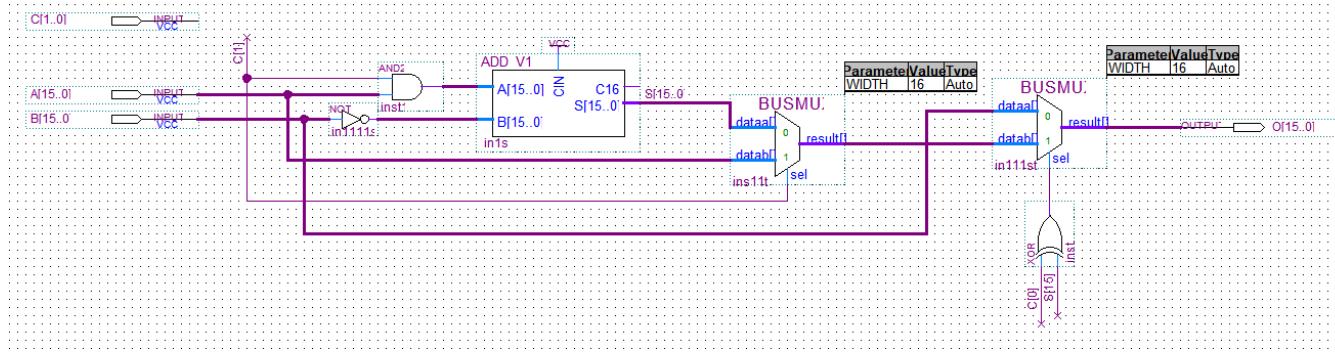
Thiết kế khối AU1:



Bảng chức năng khối AU2

C[1..0]	Chức năng
01	abs
11	max

Thiết kế khối AU2



Datapath:

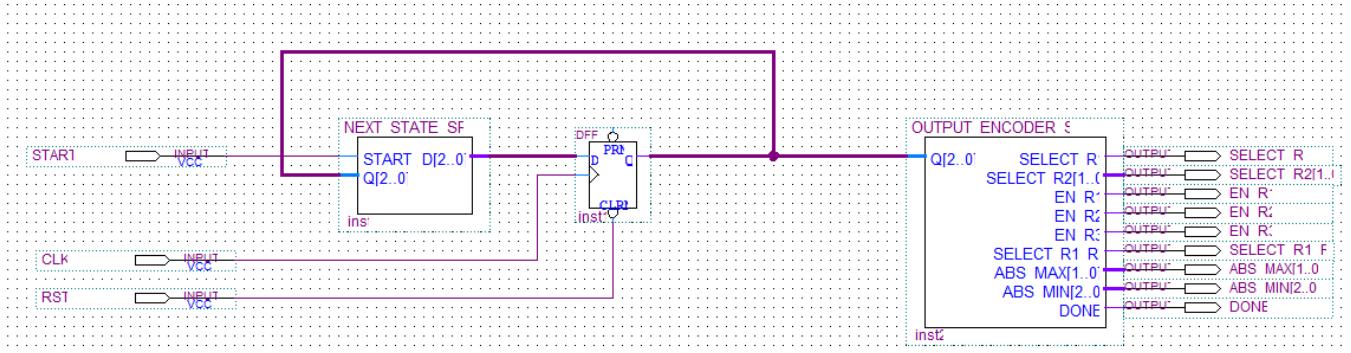
Input:

- Dữ liệu đầu vào: được nhập với người dùng
 - + A[15..0]
 - + B[15..0]
- Dữ liệu được gửi từ khối điều khiển:
 - + SELECT_R1: chọn địa chỉ để ghi vào thanh ghi 1
 - + SELECT_R2: chọn địa chỉ để ghi vào thanh ghi 2
 - + EN_R1: cho phép ghi vào R1
 - + EN_R2: cho phép ghi vào R2
 - + EN_R3: cho phép ghi vào R3
 - + SELECT_R1_R3: đọc địa chỉ để chọn R1 hoặc R3 là đầu vào của AU1
 - + OPCODE_ABS_MAX: chọn chức năng cho khối AU2
 - + OPCODE_ABS_MIN_ADD_SUB: chọn chức năng cho khối AU1

Output:

- OUTPORT[15..0]: là kết quả của phép tính

3. Thiết kế khối điều khiển:



Bảng trạng thái các khối điều khiển

TTHT	TTKT		Output								
	START		SELECT_R1	SELECT_R2	EN_R1	EN_R2	EN_R3	SELECT_R1_R3	OP_AU1	OP_AU2	DONE
S0	0	S0	0	00	1	1	0	0	000	00	0
	1	S1									
S1		S2	1	01	1	1	0	0	100	01	0
S2		S3	1	01	1	1	0	0	110	11	0
S3		S4	0	10	0	1	1	0	000	00	0
S4		S5	0	01	0	1	0	0	101	00	0
S5		S6	0	01	0	1	0	1	001	00	0
S6		S7	1	00	1	0	0	0	000	11	0
S7		S0	0	00	0	0	0	0	000	00	1

NEXT_STATE: sử dụng khối NEXT_STATE từ bài SHARE_REGISTER
OUTPUT_ENCODER:

$$\text{SELECT_R1} = Q_2'Q_1'Q_0 + Q_1Q_0'$$

$$\text{SELECT_R2[1]} = Q_2'Q_1Q_0$$

$$\text{SELECT_R2[0]} = Q_1'Q_0 + Q_2'Q_1'Q_0' + Q_2Q_1'$$

$$\text{EN_R1} = Q_2'Q_1' + Q_1Q_0'$$

$$\text{EN_R2} = Q_2' + Q_1'$$

$$\text{EN_R3} = Q_2'Q_1Q_0$$

$$\text{SELECT_R1_R3} = Q_2Q_1'Q_0$$

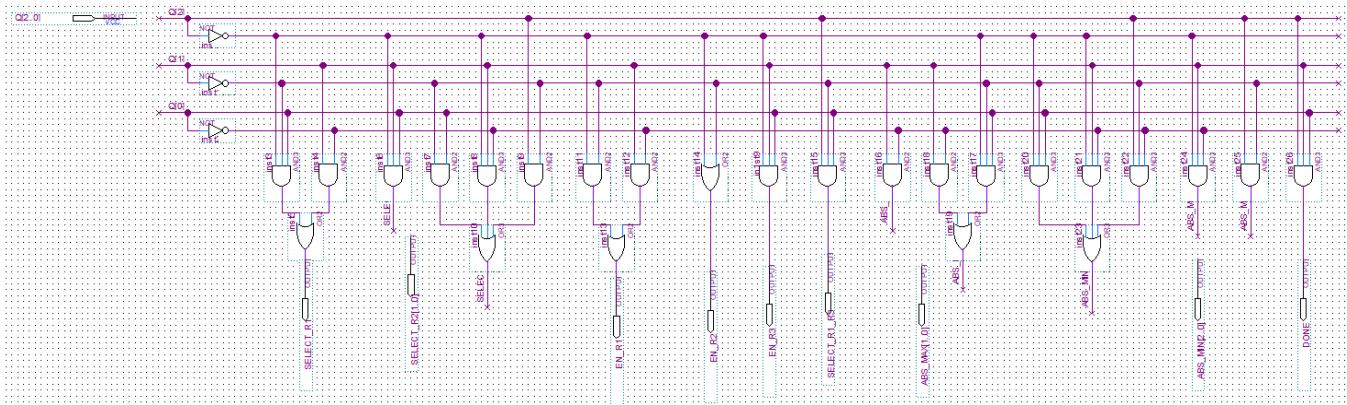
$$\text{OPCODE_AU1[2]} = Q_2'Q_1'Q_0' + Q_2'Q_1Q_0' + Q_2Q_1'Q_0'$$

$$\text{OPCODE_AU1[1]} = Q_2'Q_1Q_0'$$

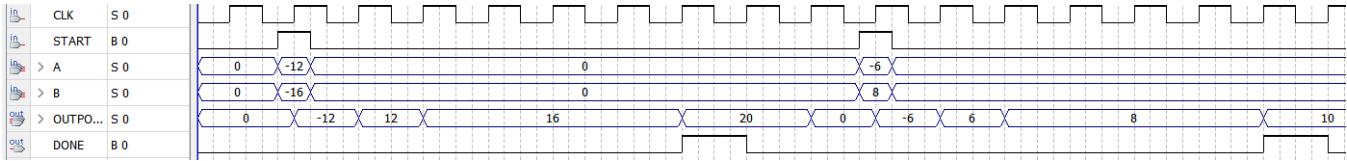
$$\text{OPCODE_AU1[0]} = Q_2Q_1'$$

$$\text{OPCODE_AU2[1]} = Q_1Q_0'$$

$$\text{OPCODE_AU2[0]} = Q_2'Q_1'Q_0 + Q_1Q_0'$$



4. Mô phỏng:



Testcase1

- Input:
 - + A = -12
 - + B = -16
 - + START = 1
- Output:
 - + OUTPORT = 20

Sau khi tín hiệu START bật lên thì sau 6 chu kì sẽ xuất ra kết quả, tin hiệu done bật cho biết kết quả của OUTPORT hợp lệ

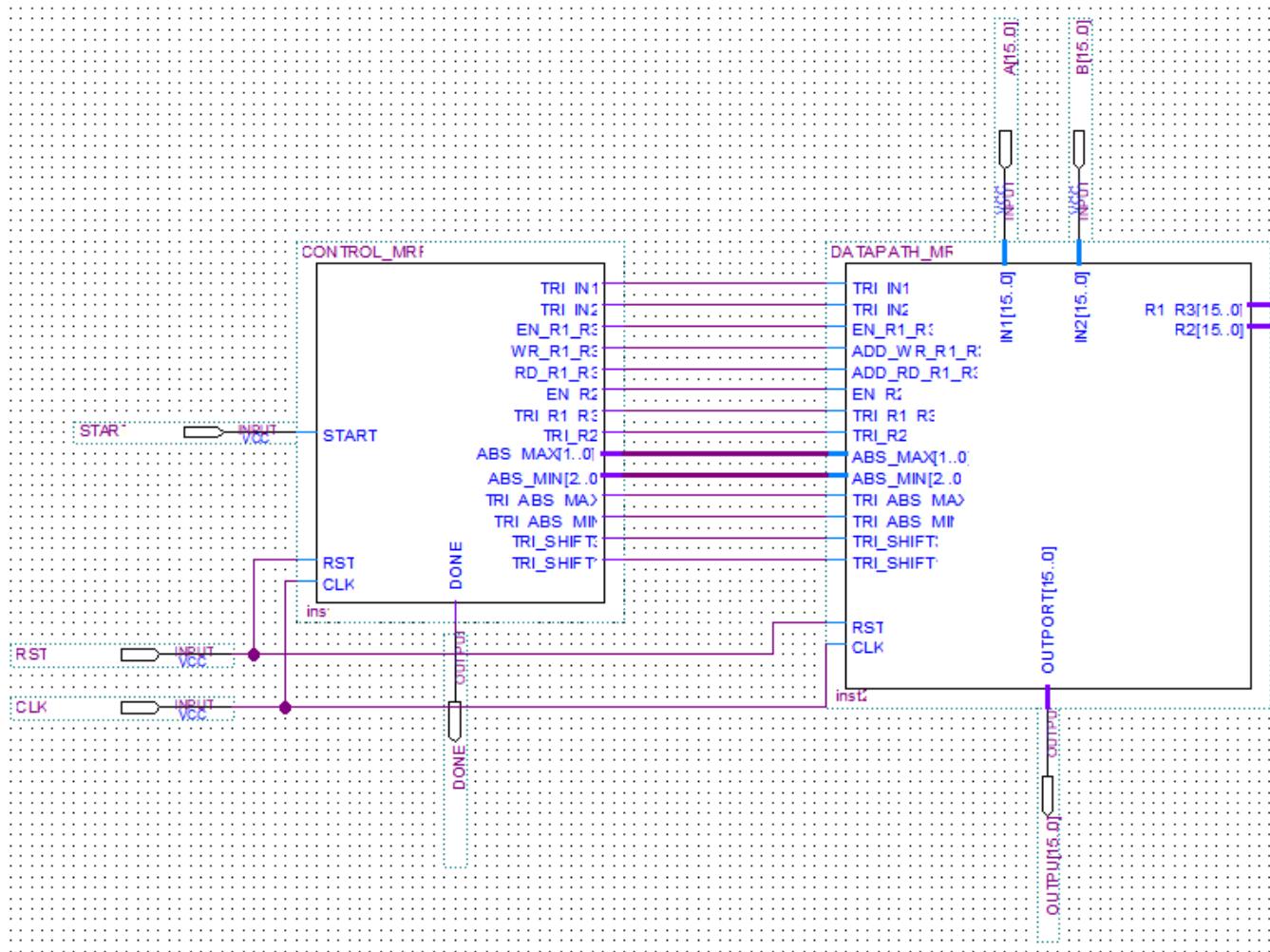
Testcase2

- Input:
 - + A = -6
 - + B = 8
- Output:
 - + OUTPORT = 10

Sau khi output của cặp dữ liệu A, B được tính xong thì ta reset hệ thống và nạp lại A, B để tiếp tục tính, và sau 6 chu kỳ kể từ khi START được bật thì DONE = 1, cho biết kết quả của OUTPORT = 10 là hợp lệ

Bài 15: Thiết kế mạch tính xấp xỉ căn bậc 2 Register Merging

1. Tổng quan:



Gán thanh ghi

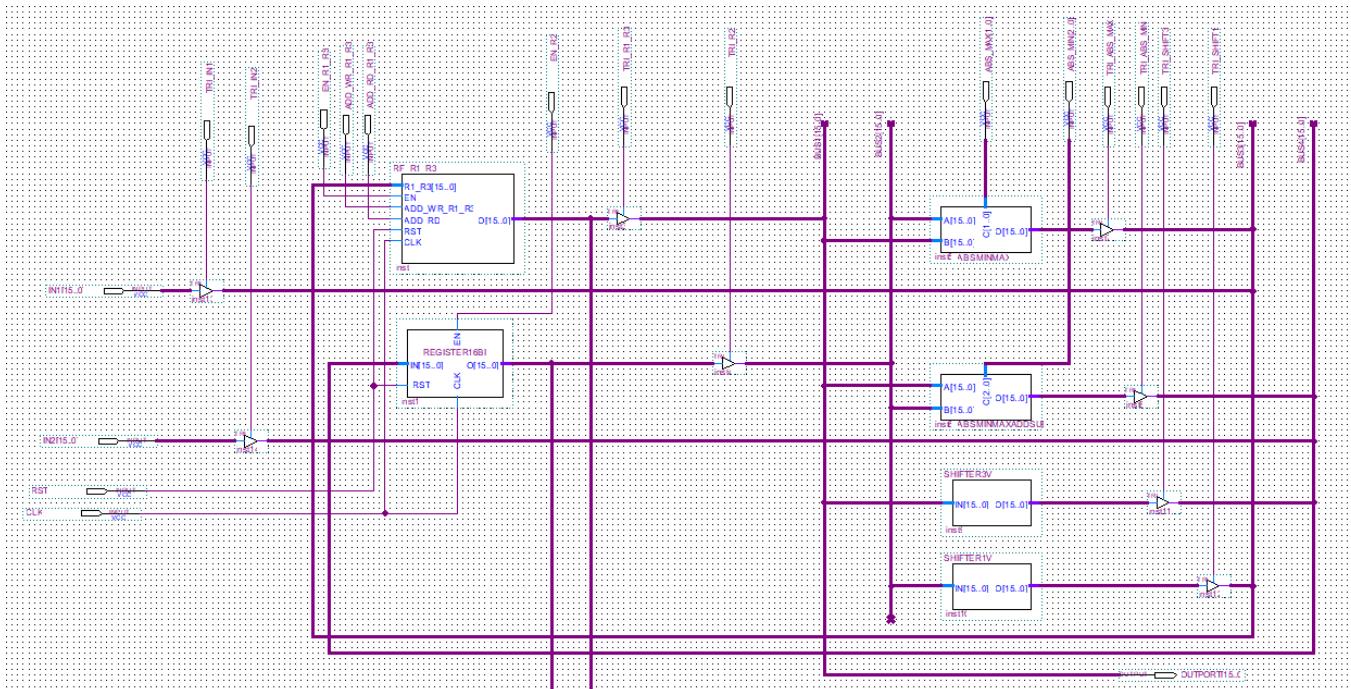
$$R1 = \{a, t1, x, t7\}$$

$$R2 = \{b, t2, y, t3, t5, t6\}$$

$$R3 = \{t4\}$$

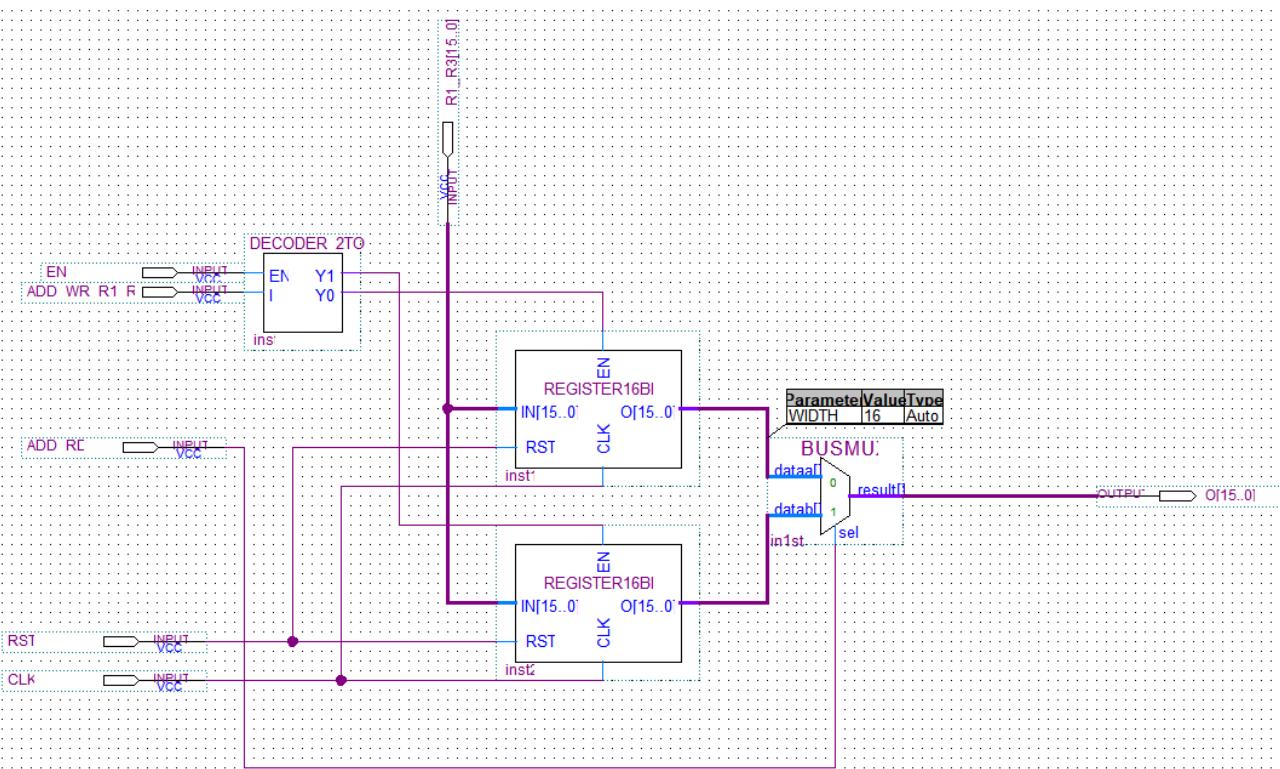
Bảng truy xuất thanh ghi

2. Thiết kế Datapath:



Sau khi hợp nhất thanh ghi thì datapath gồm 1 register file chứa R1 và R3, và thanh ghi R2

Register file R1 và R3



EN: tín hiệu cho phép register file hoạt động

ADD_WR_R1_R3: địa chỉ ghi (0: R1, 1:R3)

ADD_RD_R1_R3: địa chỉ đọc (0: R1, 1: R3)

RST: reset

CLK:

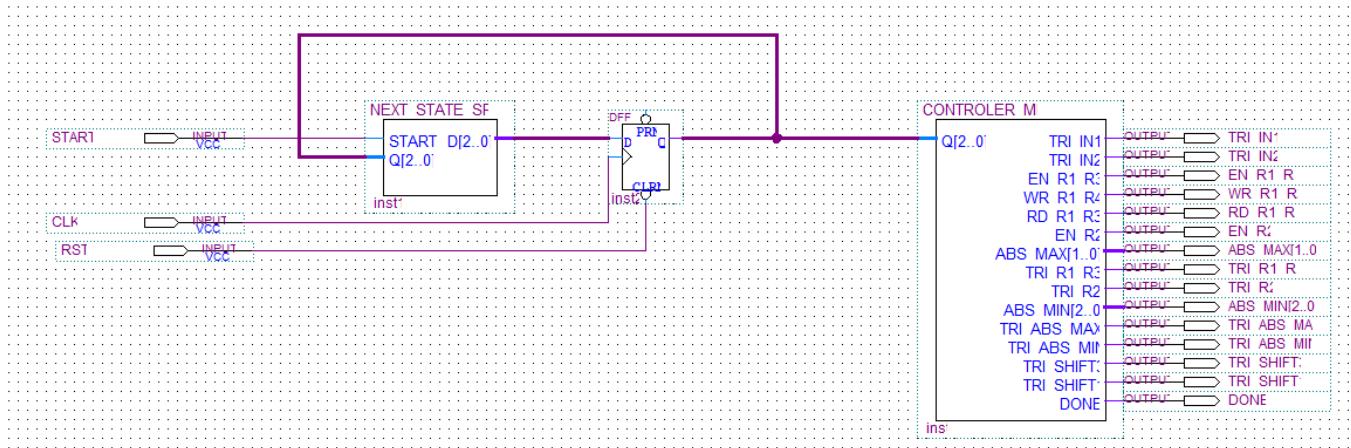
O[15..0]: dữ liệu ngõ ra của register file

Khối chức năng bao gồm 2 khối AU1 và AU2 được sử dụng lại từ bài trước

Input:

- Dữ liệu đầu vào:
 - + IN1[15..0]:
 - + IN1[15..0]:
- Dữ liệu được gửi từ khói điều khiển
 - + TRI_IN1: cho phép đọc dữ liệu từ IN1[15..0]
 - + TRI_IN2: cho phép đọc dữ liệu từ IN2[15..0]
 - + EN_R1_R3: tín hiệu cho phép ghi dữ liệu vào register file
 - + ADD_WR_R1_R3: xác định địa chỉ ghi (0: R1; 1: R3)
 - + ADD_RD_R1_R3: xác định địa chỉ đọc (0: R1; 1:R3)
 - + EN_R2: cho phép ghi dữ liệu vào R2
 - + TRI_R1_R3: đọc dữ liệu từ register file
 - + TRI_R2: đọc dữ liệu từ R2
 - + ABS_MAX[1..0]: chọn chức năng của AU1
 - + ABS_MIN_ADD_SUB[2..0]: chọn chức năng của AU2
 - + TRI_ABS_MAX: cho phép đọc dữ liệu từ AU1
 - + TRI_ABS_MIN_ADD_SUB: cho phép đọc dữ liệu từ AU2
 - + TRI_SHIFT3: cho phép đọc dữ liệu từ khói dịch phải 3 bit
 - + TRI_SHIFT1: cho phép đọc dữ liệu từ khói dịch phải 1 bit

3. Thiết kế Controller:



Bảng trạng thái và tín hiệu ngõ ra

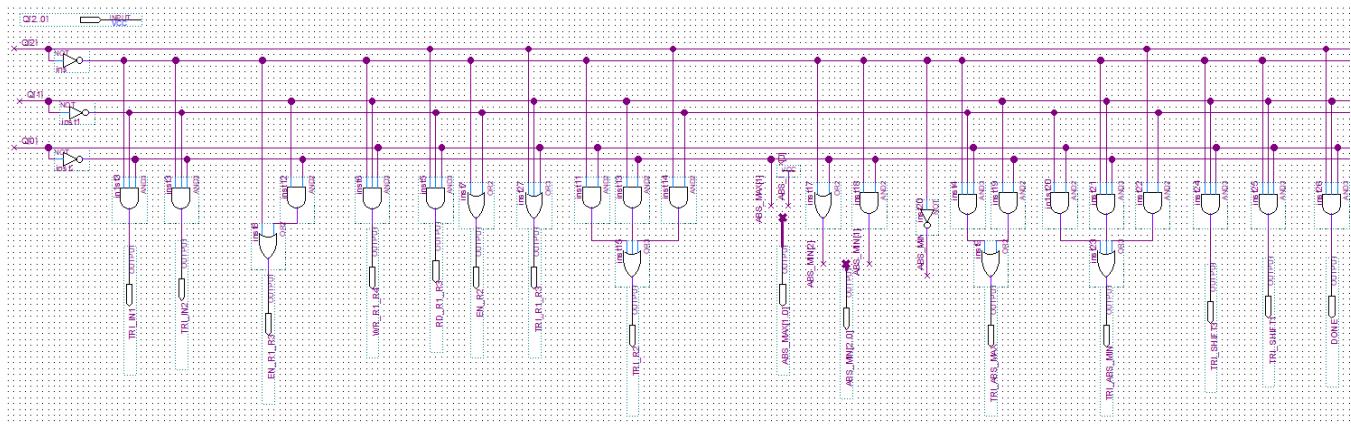
	TRI_I_N1	TRI_IN2	EN_R1_R3	ADD_WR_R1_R3	ADD_RD_R1_R3	EN_R2	TRI_R1_R3	TRI_R2	ABS_MAX	ABS_MIN_A_DD_SUB	TRI_ABS_MAX	TRI_ABS_MIN_ADD_SUB	TRI_SHIFT_3	TRI_SHIFT_1
S0	1	1	1	0	X	1	0	0	XX	XXX	0	0	0	0
S1	0	0	1	0	0	1	1	1	01	100	1	1	0	0
S2	0	0	1	0	0	1	1	1	11	110	1	1	0	0
S3	0	0	1	1	0	1	1	1	XX	XXX	0	0	1	1
S4	0	0	0	0	0	1	1	1	XX	101	0	1	0	0
S5	0	0	0	0	1	1	1	1	XX	001	0	1	0	0
S6	0	0	1	0	0	0	1	1	11	XXX	0	0	0	0
S7	0	0	0	0	0	0	0	1	0	XX	XXX	0	0	0

Khối NEXT_STATE được sử dụng lại từ bài trước

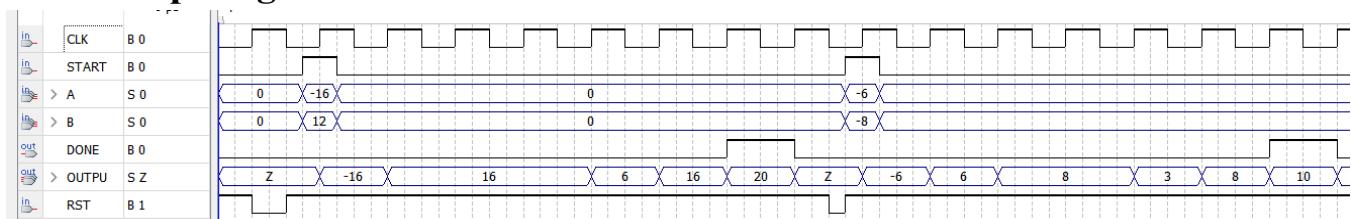
Khối OUTPUT_ENCODER

$$TRI_IN1 = Q2'Q1'Q0'$$

$\text{TRI_IN1} = Q_2'Q_1'Q_0'$
 $\text{EN_R1_R3} = Q_2' + Q_1Q_0'$
 $\text{ADD_WR_R1_R3} = Q_2'Q_1Q_0$
 $\text{ADD_RD_R1_R3} = Q_2Q_1'Q_0$
 $\text{EN_R2} = Q_2' + Q_1'$
 $\text{TRI_R1_R3} = Q_0 + Q_1 + Q_2$
 $\text{TRI_R2} = Q_2'Q_0 + Q_1Q_0' + Q_2Q_1'$
 $\text{ABS_MAX[1]} = Q_0'$
 $\text{ABS_MAX[0]} = 1$
 $\text{ABS_MIN_ADD_SUB[2]} = Q_2' + Q_0'$
 $\text{ABS_MIN_ADD_SUB[1]} = Q_2'Q_0'$
 $\text{ABS_MIN_ADD_SUB[0]} = Q_2$
 $\text{TRI_ABS_MAX} = Q_2'Q_1'Q_0 + Q_1Q_0'$
 $\text{TRI_ABS_MIN_ADD_SUB} = Q_1'Q_0 + Q_2'Q_1Q_0' + Q_2Q_1'$
 $\text{TRI_SHIFT3} = Q_2'Q_1Q_0$
 $\text{TRI_SHIFT1} = Q_2'Q_1Q_0$



4. Mô phỏng:



- Sau khi tín hiệu START = 1 thì sau 6 chu kỳ DONE = 1 cho biết kết quả của phép tính căn, sau đó ta RST = 1 để reset hệ thống sau đó nạp giá trị tiếp theo để tính,

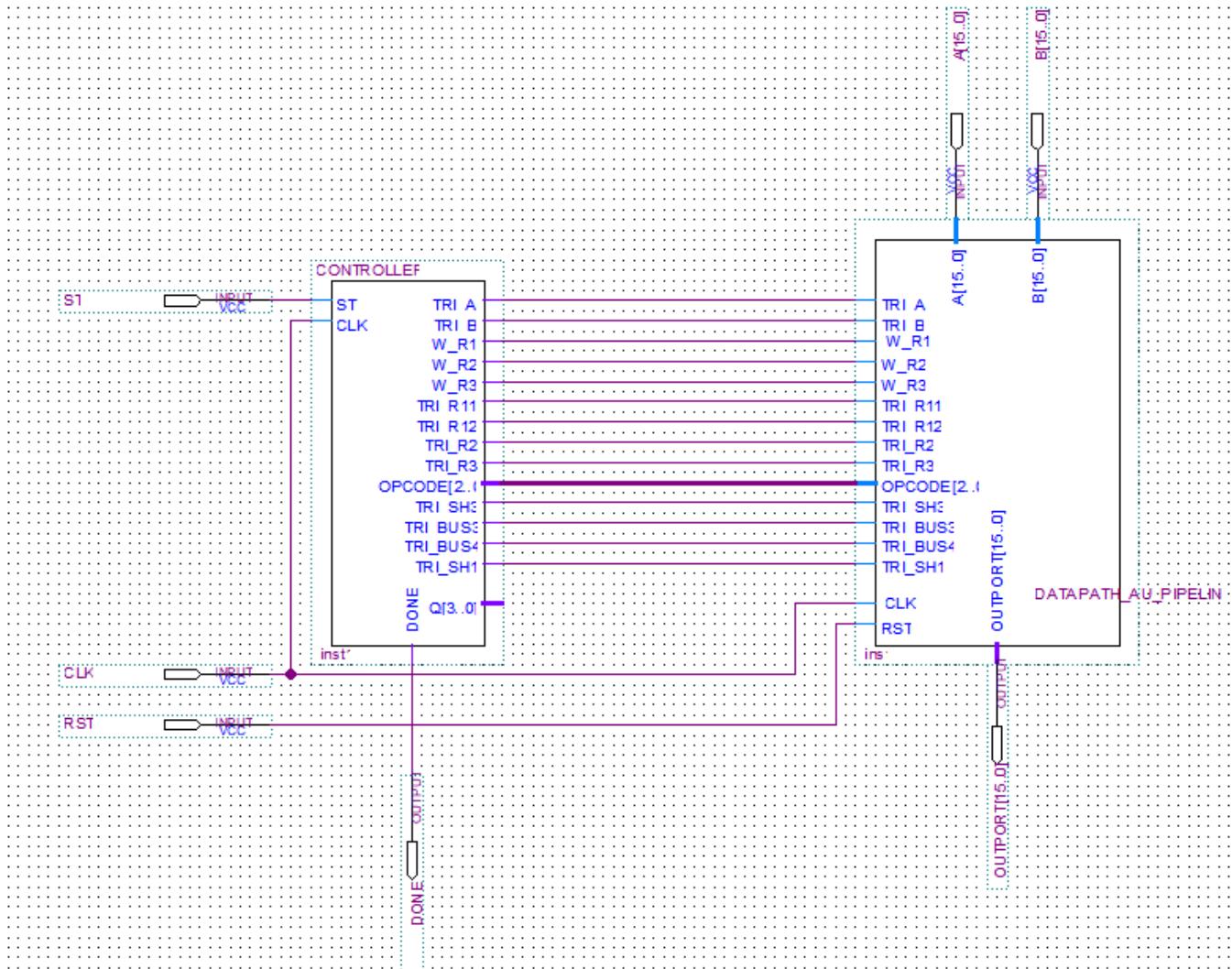
sau 6 chu kỳ từ khi START = 1 thì DONE = 1 cho biết kết quả của phép tính căn là hợp lệ

Bài 16: Thiết kế mạch tính xấp xỉ căn bậc 2 Functional Unit Pipelining

1. Tổng quan:

Giản đồ thời gian của khối datapath thực thi Pipeline khối chức năng

	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12
Đọc R1		a			t1	t1	x			x			
Đọc R2			b		t2	t2	t3		t5		t6		
Đọc R3									t4				
AU tầng 1		a	b		max	min	-		+		max		
AU tầng 2			a	b		max	min	-	+		max		
Dịch bit						>>3	>>1						
Ghi R1	a		t1			x			t6		t7		
Ghi R2	b			t2		t3		t5					
Ghi R3							t4						
Xuất ngoại ra													t7



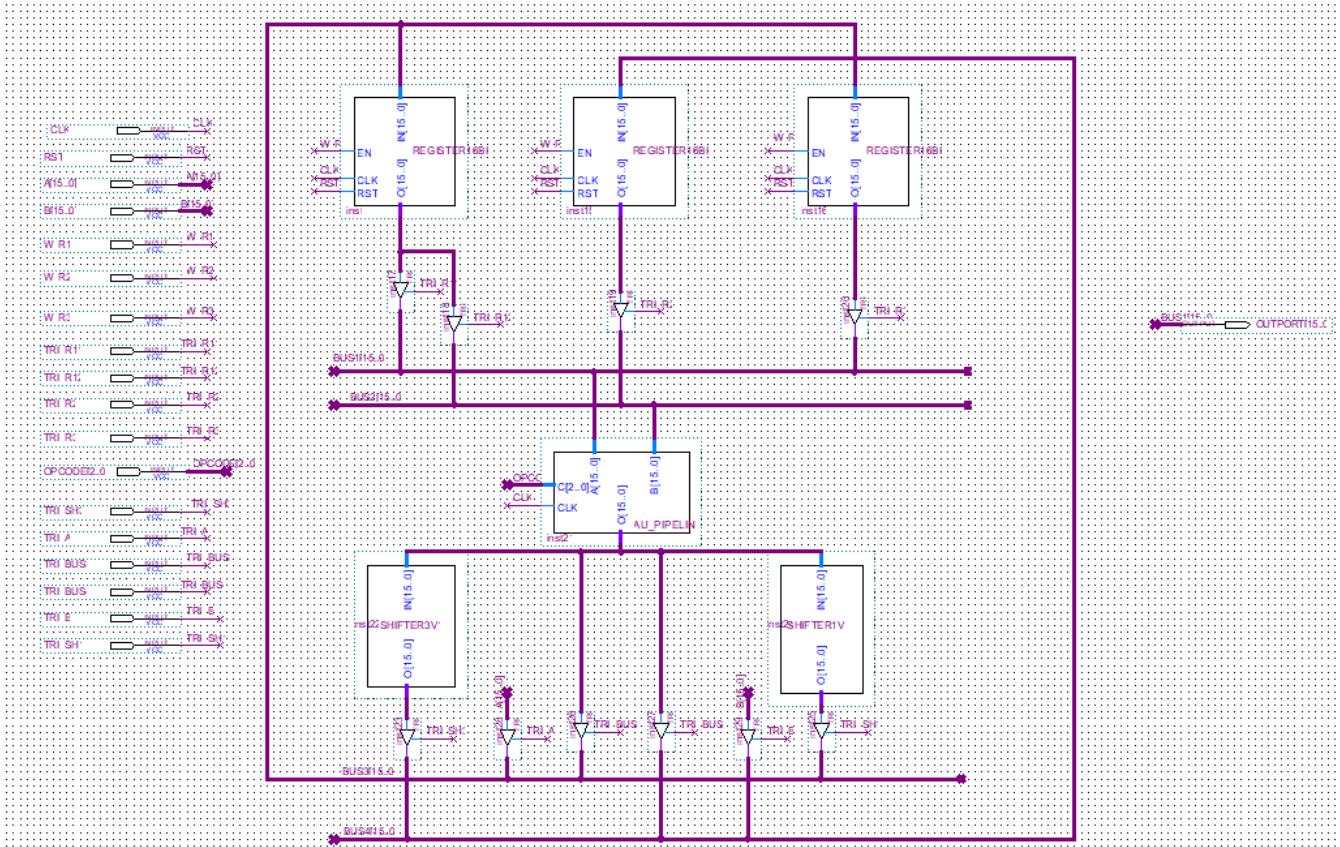
Input:

- + A[15..0]
- + B[15..0]
- + START: tín hiệu bắt đầu
- + CLK: tín hiệu xung clock
- + RST: tín hiệu reset của hệ thống

Output:

- + OUTPORT[15..0]: kết quả của phép tính tổng căn bậc 2
- + DONE: cho biết kết quả của OUTPORT là hợp lệ

2. Thiết kế datapath

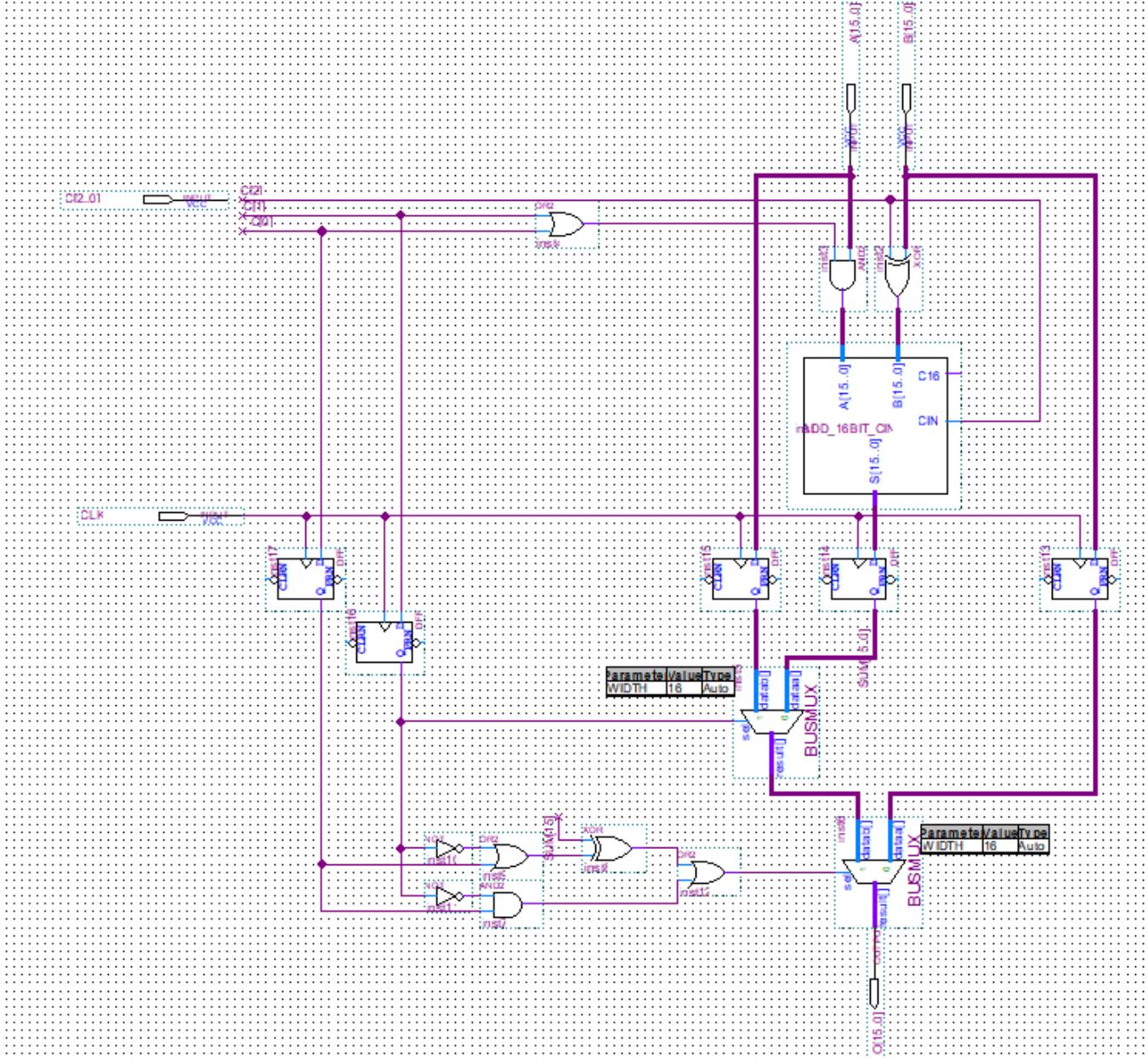


Input:

- Dữ liệu đầu vào
 - + A[15..0]
 - + B[15..0]
- Dữ liệu được gửi từ khối điều khiển
 - + W_R1: cho phép ghi dữ liệu vào R1
 - + W_R2: cho phép ghi dữ liệu vào R2
 - + W_R3: cho phép ghi dữ liệu vào R3
 - + TRI_R11: cho phép đọc dữ liệu từ R1 đến BUS1
 - + TRI_R12: cho phép đọc dữ liệu từ R1 đến BUS2
 - + TRI_R2: cho phép đọc dữ liệu từ R2
 - + TRI_R3: cho phép đọc dữ liệu từ R3
 - + OPCODE[2..0]: chọn chức năng cho các toán hạng
 - + TRI_SH3: cho phép đọc dữ liệu từ khối dịch phải 3 bit
 - + TRI_A: cho phép đọc dữ liệu từ đầu vào A
 - + TRI_BUS3: cho phép đọc dữ liệu từ AU xuống BUS3
 - + TRI_BUS4: cho phép đọc dữ liệu từ AU xuống BUS4
 - + TRI_B: cho phép đọc dữ liệu từ đầu vào B

- + TRI_SH1: cho phép đọc dữ liệu từ khối dịch phải 1 bit

Thiết kế AU_PIPELINE



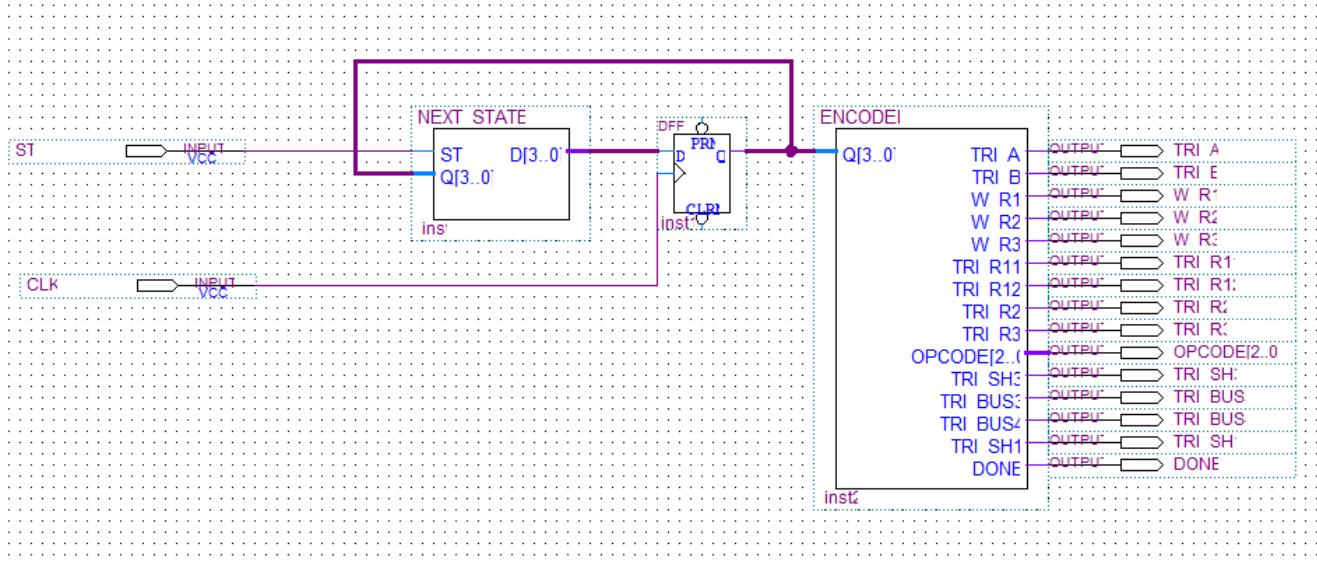
Dùng cái DFF chặn giá trị, chia AU thành 2 tầng để tính toán
Bảng chức năng khối AU

C[2..0]	Chức năng
001	add
100	abs

101	sub
110	min
111	max

3. Thiết kế Controller:

Kiến trúc tổng quát của khối điều khiển



Bảng trạng thái và tín hiệu ngõ ra

TTHT	TTKT		OUTPUT														
	ST A RT		TRI _A	TRI _B	W_ R1	W_ R2	W_ R3	TR I_ R1 1	TR I_ R1 2	TRI _R2	TRI _R3	OPC ODE	TRI _S H3	TR I_ BU S3	TR I _B US 4	TR I _S H1	D O NE
S0	0	S0	1	1	1	1	0	0	0	0	0	XXX	0	0	0	0	0
	1	S1															
S1		S2	0	0	0	0	0	0	1	0	0	100	0	0	0	0	0
S2		S3	0	0	1	0	0	0	0	1	0	100	0	1	0	0	0
S3		S4	0	0	0	1	0	0	0	0	0	XXX	0	0	1	0	0
S4		S5	0	0	0	0	0	1	0	1	0	111	0	0	0	0	0

S5		S6	0	0	1	1	0	1	0	1	0	110	1	1	0	0	0
S6		S7	0	0	0	0	1	1	0	1	0	101	0	0	0	1	0
S7		S8	0	0	0	1	0	0	0	0	0	XXX	0	0	1	0	0
S8		S9	0	0	0	0	0	0	0	1	1	001	0	0	0	0	0
S9		S10	0	0	0	1	0	0	0	0	0	XXX	0	0	1	0	0
S10		S11	0	0	0	0	0	1	0	1	0	111	0	0	0	0	0
S11		S12	0	0	1	0	0	0	0	0	0	XXX	0	1	0	0	0
S12		S0	0	0	0	0	0	1	0	0	0	XXX	0	0	0	0	1

TRI_A = Q3'Q2'Q1'Q0'

TRI_B = Q3'Q2'Q1'Q0'

W_R1 = Q3'Q2'Q0' + Q2Q1'Q0 + Q3Q1Q0

W_R2 = Q3'Q2'Q1'Q0' + Q3'Q1Q0 + Q2Q0 + Q3Q1'Q0

W_R3 = Q2Q1Q0'

TRI_R11 = Q2Q1' + Q2Q0' + Q3Q1Q0'

TRI_R12 = Q3'Q2'Q1'Q0

TRI_R2 = Q1Q0' + Q3Q2'Q0' + Q3'Q2Q1'

TRI_R3 = Q3Q2'Q1'Q0'

OPCODE[2] = Q1 + Q3'

OPCODE[1] = Q3Q1 + Q2Q1'

OPCODE[0] = Q3 + Q2Q0'

TRI_SH3 = Q2Q1'Q0

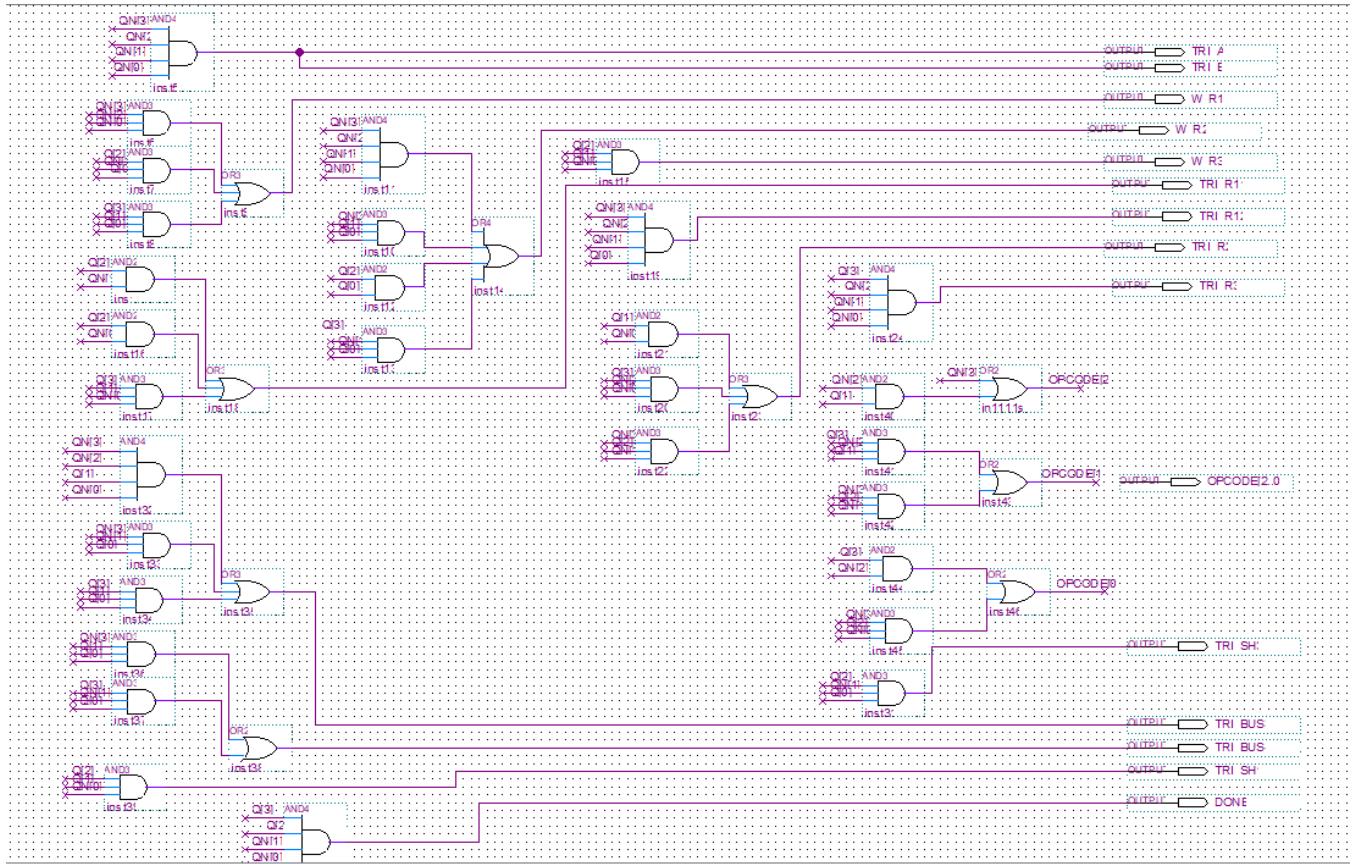
TRI_BUS3 = Q3'Q2'Q1Q0' + Q2Q1'Q0 + Q3Q1Q0

TRI_BUS4 = Q3'Q1Q0 + Q3Q1'Q0

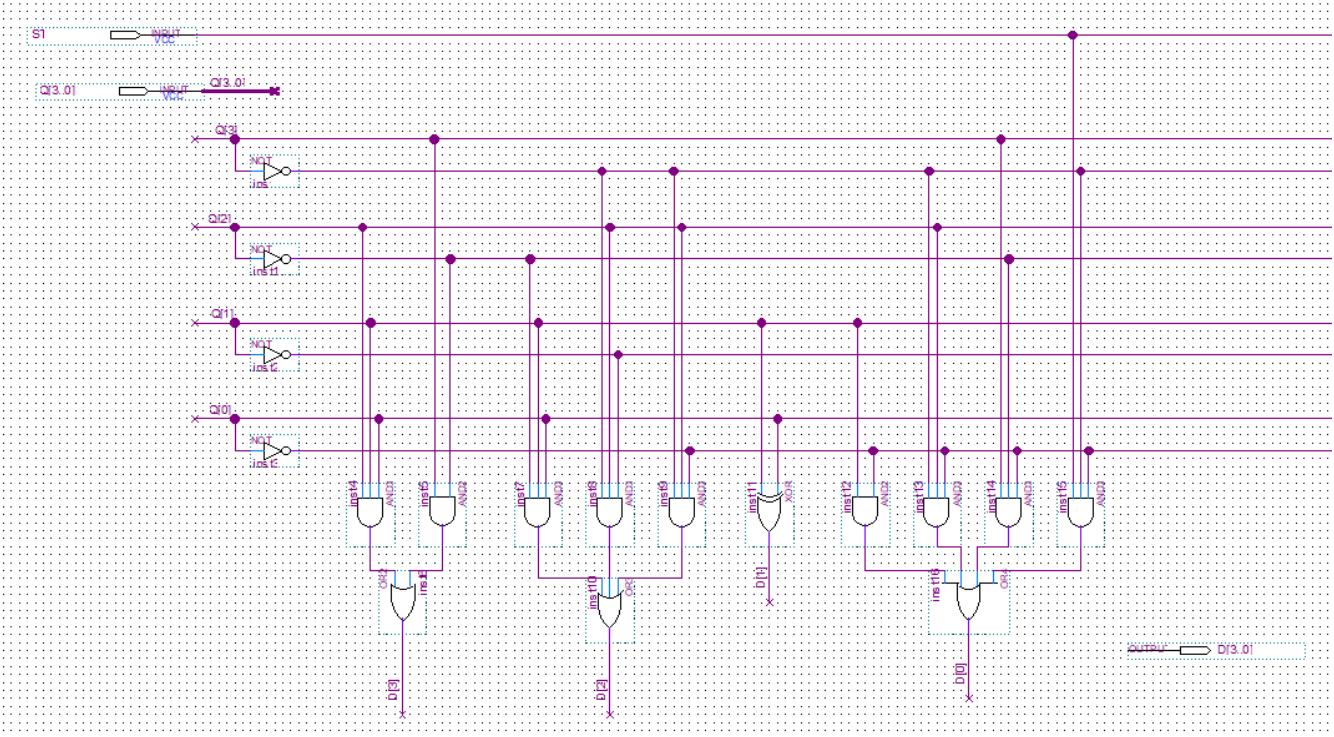
TRI_SH1 = Q3'Q2Q1Q0

DONE = Q3Q2Q1'Q0'

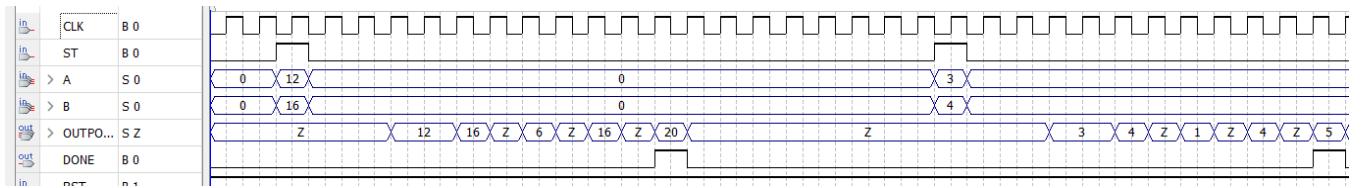
Mạch khởi ENCODER



Thiết kế khối trạng thái kế tiếp



4. Mô phỏng:



Testcase1:

- Input:

- + $A = 12$
- + $B = 16$

Output:

- $OUTPUT = 20$, $DONE = 1$, cho biết kết quả hợp lệ

Sau khi tín hiệu $START = 1$, thì sau 11 chu kì sẽ có ra kết quả là 20, $DONE$ cho biết 20 là kết quả đúng

Testcase2:

- Input:

- + $A = 3$
- + $B = 5$

- Output:

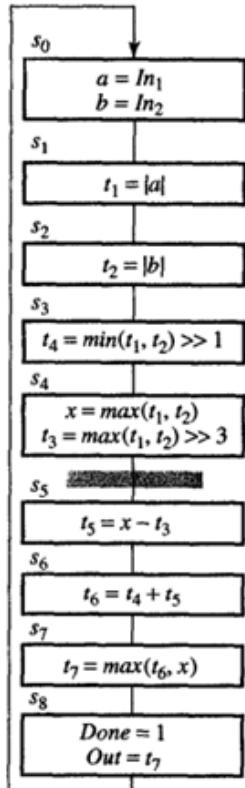
- + $OUTPUT = 5$, $DONE = 1$, cho biết kết quả đúng

Sau khi xuất ra được output đầu tiên thì máy trạng thái quay lại $S0$ nên ta chỉ cần nạp dữ liệu vào và $START = 1$ để tính kết quả tiếp theo

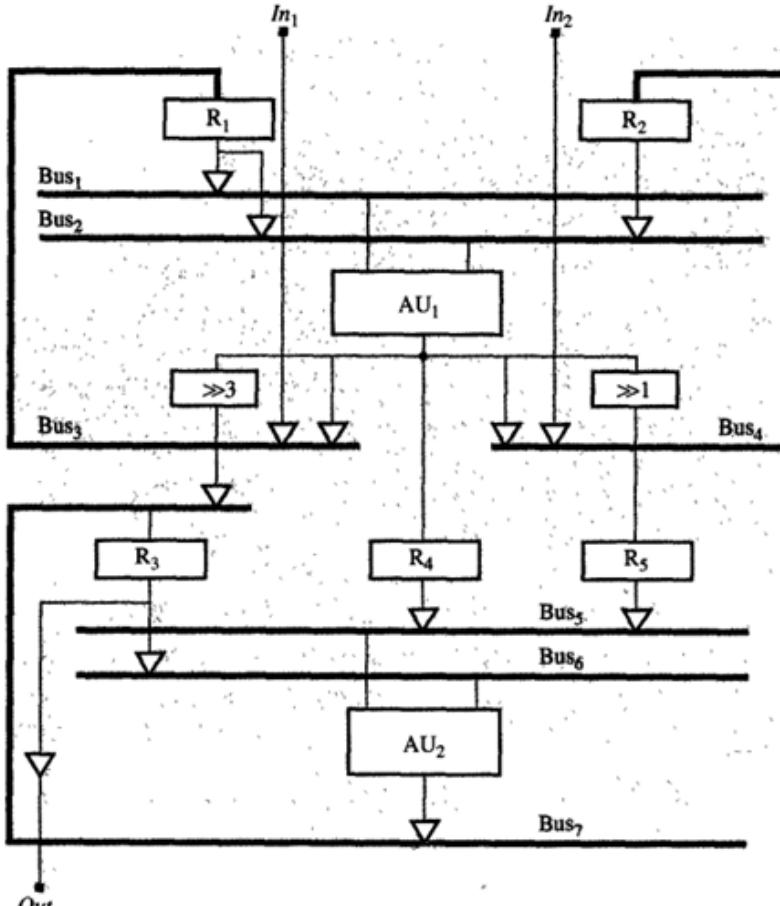
Bài 17: Thiết kế mạch tính xấp xỉ căn bậc 2 Datapath Pipelining.

1. Tổng quan:

- Thiết kế mạch tính xấp xỉ căn bậc hai dựa trên kỹ thuật datapath pipelining nhằm cải thiện hiệu suất xử lý bằng cách phân chia quá trình tính toán thành các giai đoạn độc lập, giúp cải thiện hiệu suất hệ thống bằng việc giảm độ trễ khi cần tính toán nhiều input.



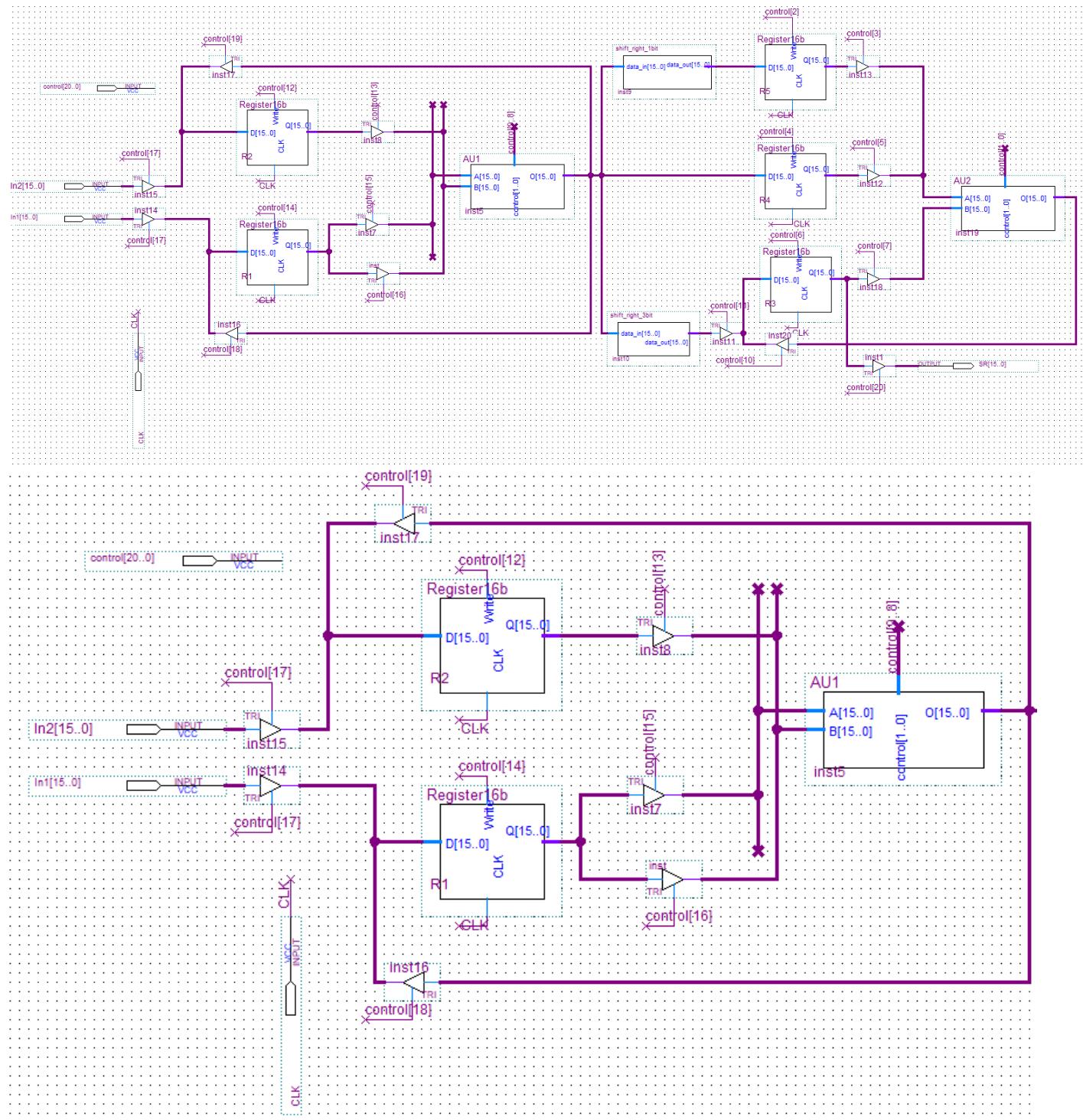
(a) ASM chart

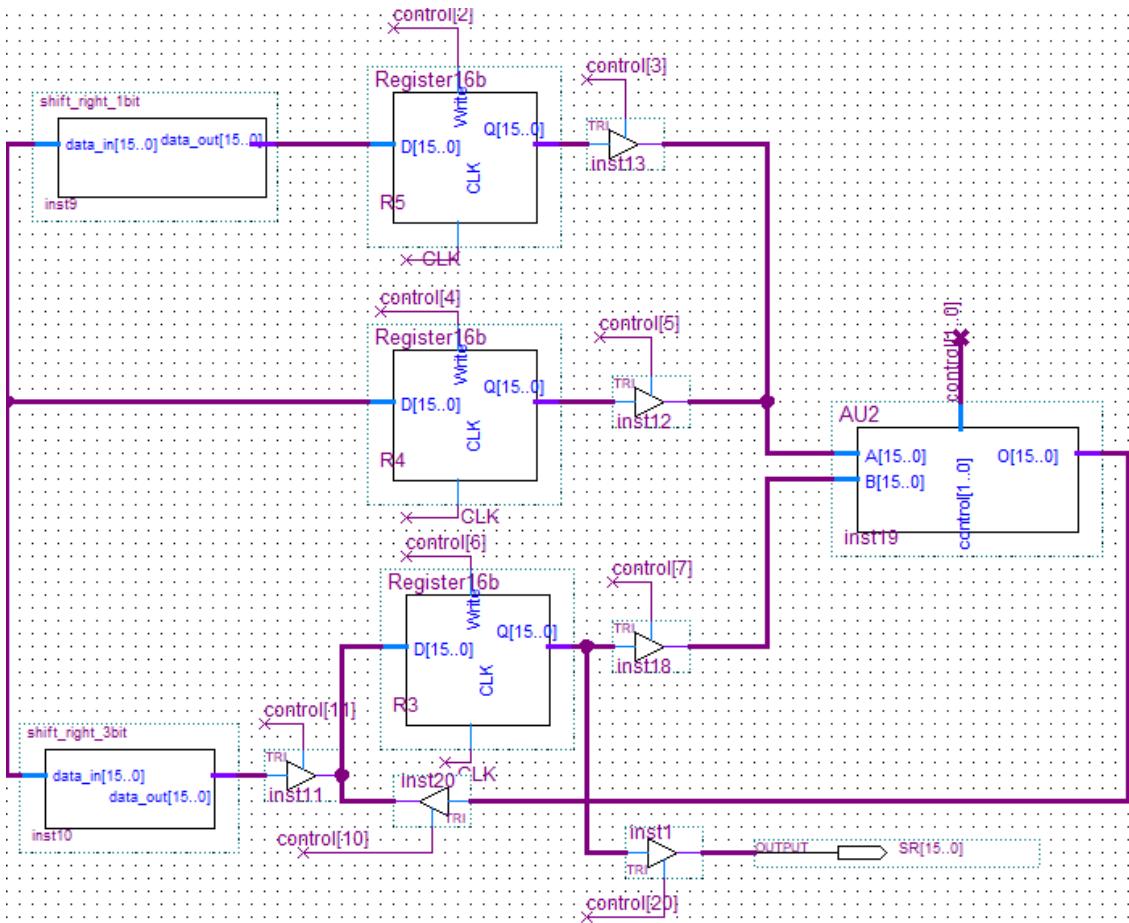


(b) Pipelined datapath

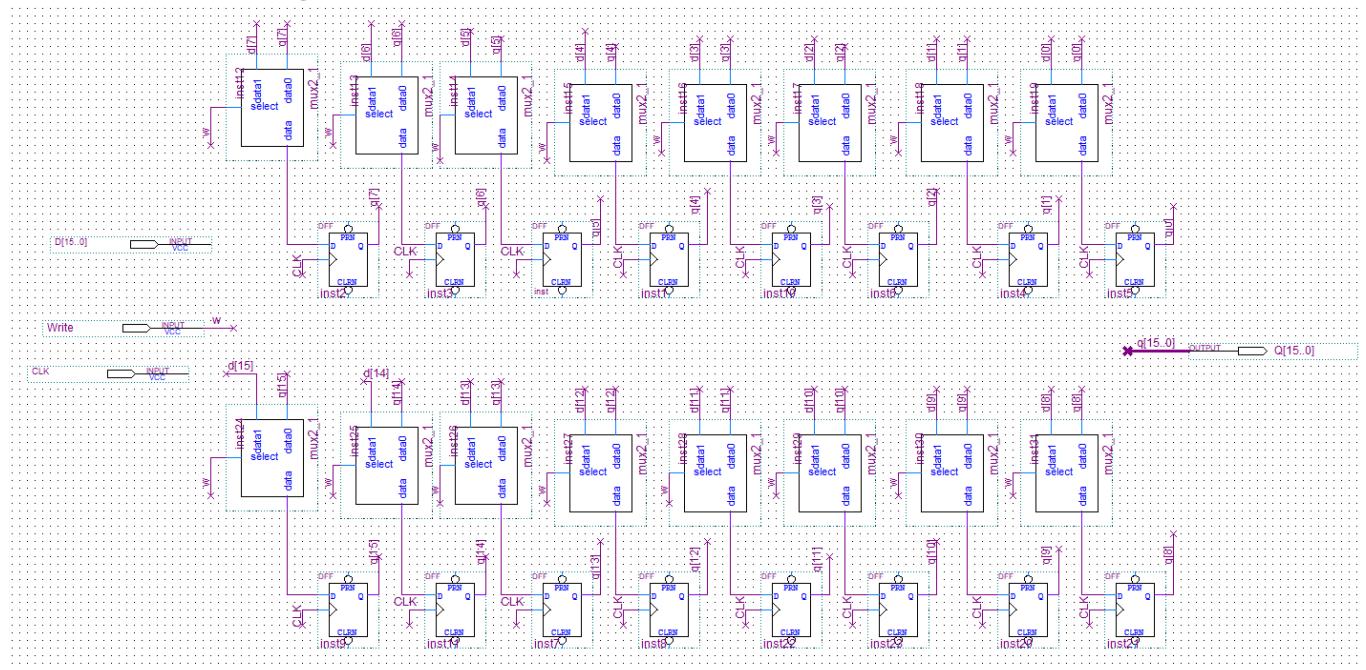
$$\begin{array}{ll}
 R_1 = [a, t_1] & R_3 = [t_3, t_5, t_6, t_7] \\
 R_2 = [b, t_2] & R_4 = [x] \\
 AU_1 = [abs/min/max] & R_5 = [t_4] \\
 & AU_2 = [+/-/max]
 \end{array}$$

2. Thiết kế Datapath:



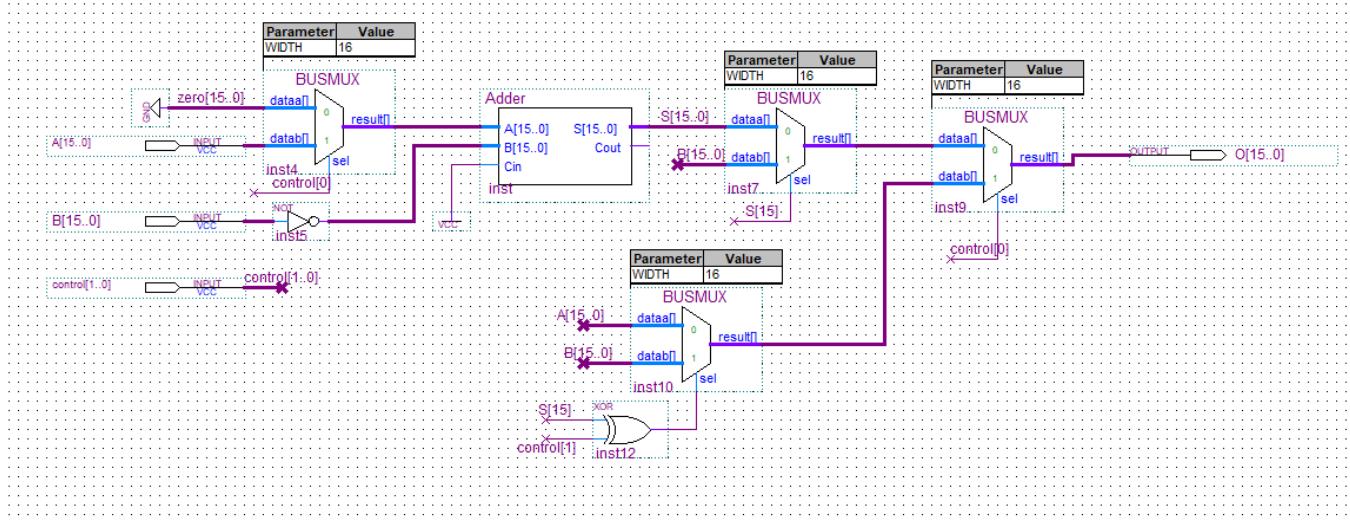


2.1. Thanh ghi 16 bit:



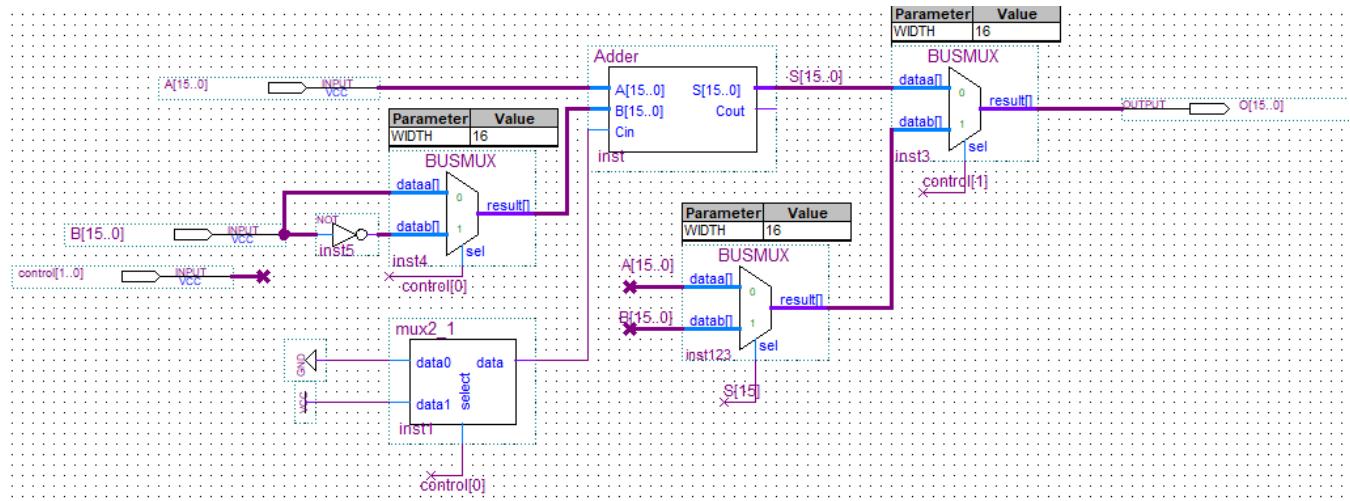
- Tín hiệu write cho phép ghi dữ liệu.
- Các Tristate trong datapath điều khiển việc đọc giá trị của thanh ghi.

2.2. Khối AU1:



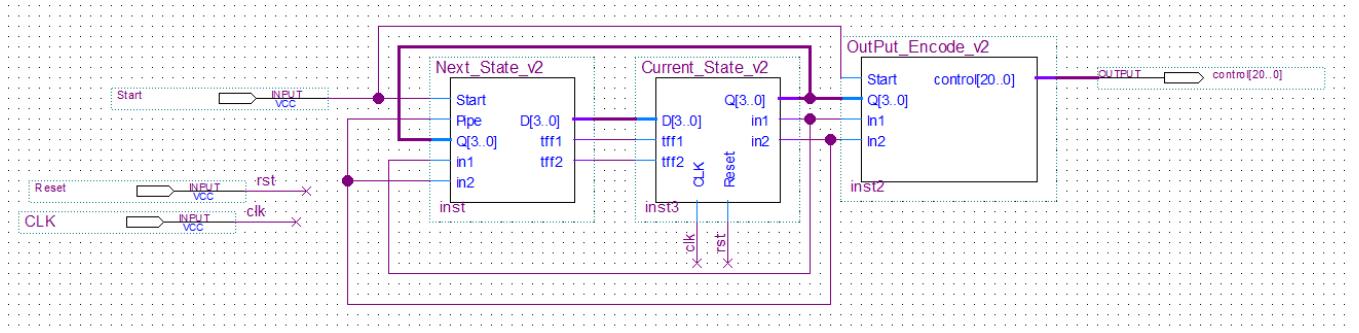
control[1..0]	Operation
00	Abs
01	Max
10	X
11	Min

2.3. Khối AU2:



control[1..0]	Operation
00	+
01	-
10	X
11	Max

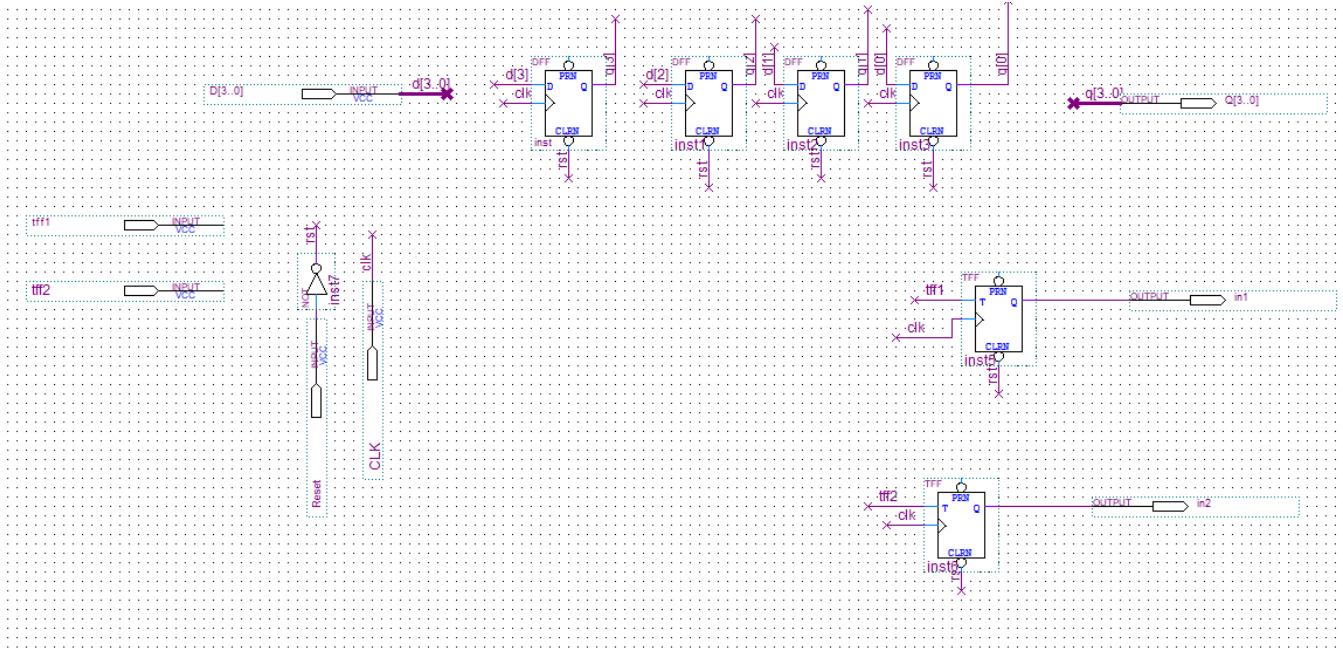
3. Thiết kế Controller:



3.1. Bảng sử dụng tài nguyên:

	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9
Read_R1		a		t1	t1		a		t1	t1
Read_R2			b	t2	t2			b	t2	t2
AU1		a	b	min	max		a	b	min	max
Shifter				>>1	>>3				>>1	>>3
Write_R1	a	t1				a	t1			
Write_R2	b		t2			b		t2		
Read_R3	t3	t5	t6	t7		t3	t5	t6	t7	
Read_R4	x		x			x		x		
Read_R5		t4					t4			
AU2	-	"+"	max			-	"+"	max		
Write_R3	t5	t6	t7		t3	t5	t6	t7		t3
Write_R4				x					x	
Write_R5				t4					t4	

3.2. Khối trạng thái hiện tại:



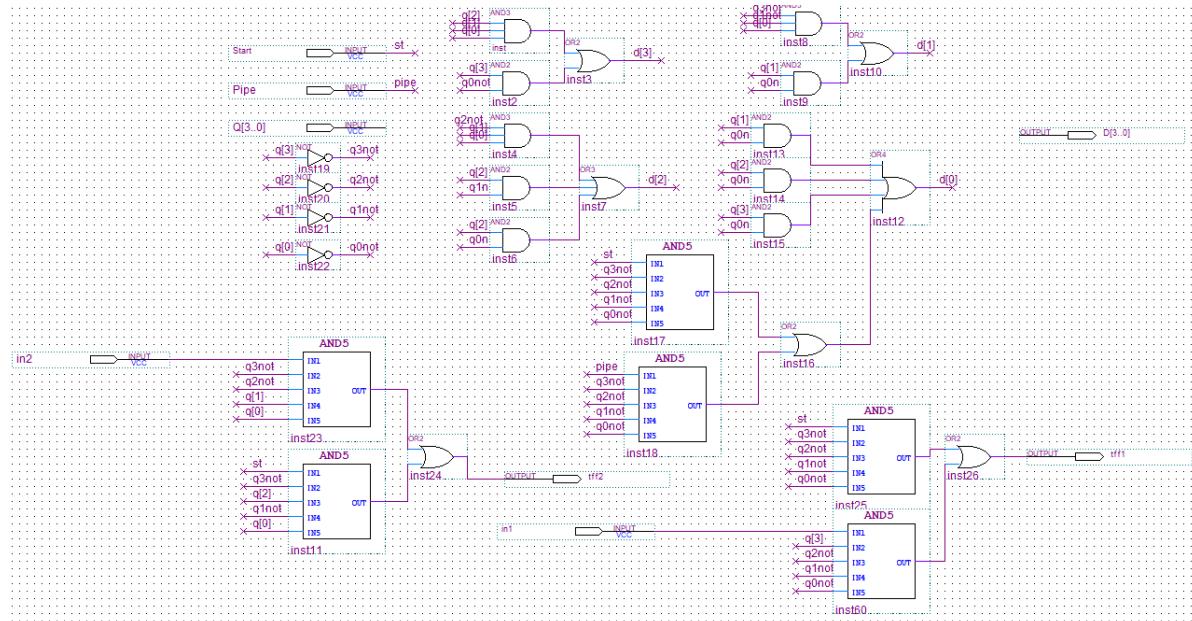
- 2 TFF kiểm tra input đã nhận tại S0 và S5. Tại S0 nếu nhận input thì TFF1=1, tại S5 nếu nhận input thì TFF2=1.
- Tại S8 nếu OE=1 thì TFF1 bị đảo về 0, tại S3 nếu OE=1 thì TFF2 bị đảo về 0.

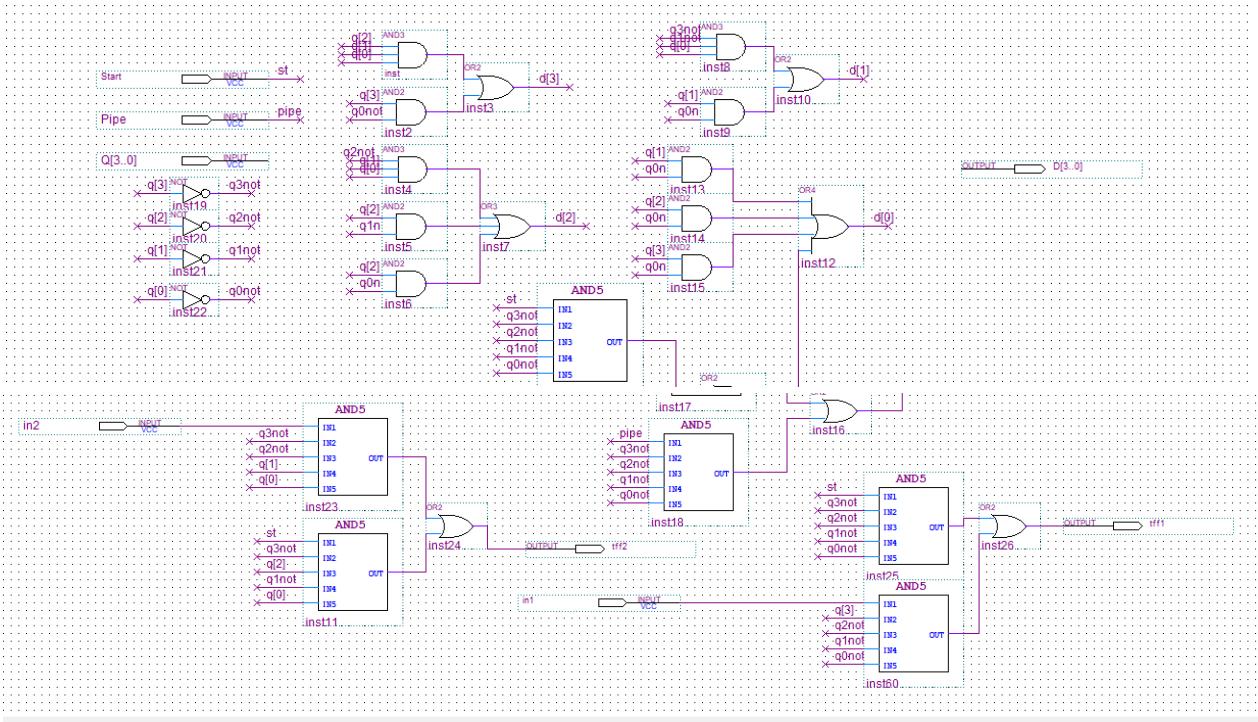
- OE=1 tại S8 khi TFF1=1, hoặc tại S3 khi TFF2 =1.

3.3. Khối trạng thái kế tiếp:

						Q2Q1Q0 + Q3Q0'	Q2'Q1Q0 + Q2Q1' + Q2Q0'	Q3'Q1'Q0 + Q1Q0'	Q1Q0' + Q2Q0' + Q3Q0'
	Q3	Q2	Q1	Q0		D3	D2	D1	D0
S0	0	0	0	0	Start/orPipe	0	0	0	1
S1	0	0	0	1		0	0	1	0
S2	0	0	1	0		0	0	1	1
S3	0	0	1	1		0	1	0	0
S4	0	1	0	0		0	1	0	1
S5	0	1	0	1		0	1	1	0
S6	0	1	1	0		0	1	1	1
S7	0	1	1	1		1	0	0	0
S8	1	0	0	0		1	0	0	1
S9	1	0	0	1		0	0	0	0
S10-15	X	X	X	X		X	X	X	X

- Mạch chuyển trạng thái từ S0 đến S9 lần lượt và quay về S0.
- Tại S0, mạch chuyển trạng thái lên S1 khi có tín hiệu Start=1 hoặc Pipe =1.
- Pipe =1 nếu tại S5 có nhận input.

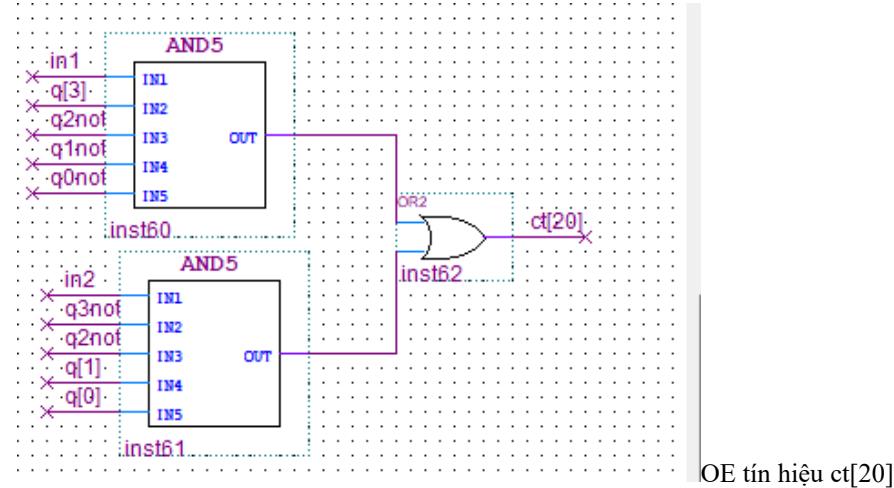




3.4. Khôi mã hóa ngo ra:

- Ngõ ra điều khiển 21 bit control[20..0]

	OE	AU1Bus4	AU1Bus3	IN_Tristate	Read_R1_abso_lute
S0		0		Start=1,1 0 Start=0,0	0
S1		0	1	0	1
S2		1	0	0	0
S3		0	0	0	0
S4		0	0	0	0
S5		0		Start=1,1 0 Start=0,0	0
S6		0	1	0	1
S7		1	0	0	0
S8		0	0	0	0
S9		0	0	0	0
S10-S15	X	X	X	X	X
		$Q2'Q1Q0' + Q2Q1Q0$	$Q3'Q2'Q1'Q0 + Q2Q1Q0'$	$StartQ3'Q2'Q1'Q0' + StartQ3'Q2Q1'Q0$	$Q3'Q2'Q1'Q0 + Q2Q1Q0'$



- OE =1 tại S8 khi in1=1(có nhận input tại S0).

- OE =1 tại S3 khi in2=1(có nhận input tại S5).
- In1, in2 là giá trị của TFF1 và TFF2, được sử dụng để kiểm tra việc nhận input.
- ct[19] (AU1Bus4) điều khiển Tristate của AU1 tại bus4.
- ct[18] (AU1Bus3) điều khiển Tristate của AU1 tại bus3.
- ct[17] điều khiển tristate của 2 input.
- Ct[16] điều khiển tristate (cổng đọc) của R1 tại bus2.

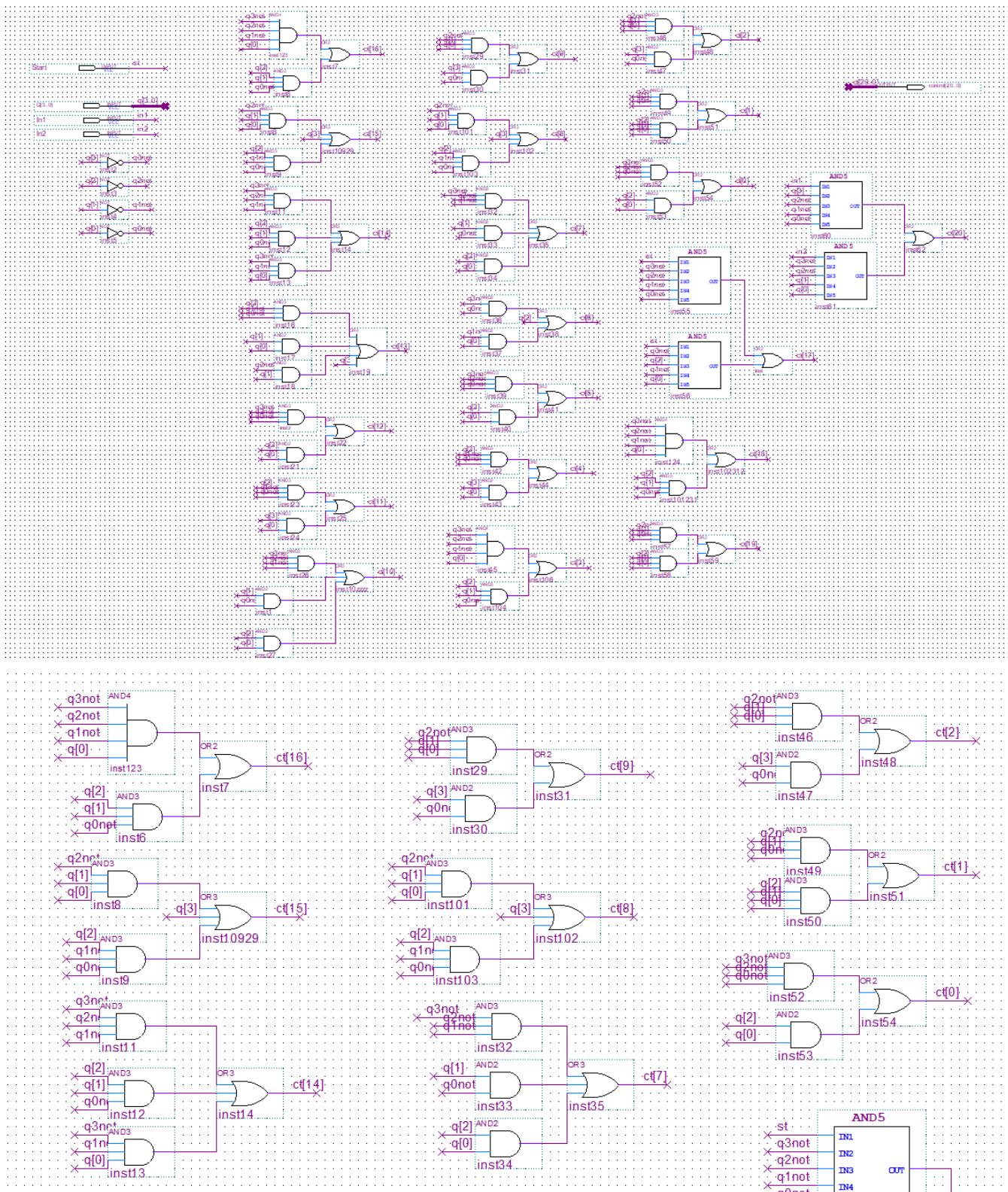
Read_R1	Write_R1	Read_R2	Write_R2	Shifter>>3	AU2_Tristate	AU1[1]	AU1[0]
0	1	0	1	0	1	0	0
0	1	0	0	0	1	0	0
0	0	1	1	0	1	0	0
1	0	1	0	0	0	1	1
1	0	1	0	1	0	0	1
0	1	0	1	0	1	0	0
0	1	0	0	0	1	0	0
0	0	1	1	0	1	0	0
1	0	1	0	0	0	1	1
1	0	1	0	1	0	0	1
X	X	X	X	X	X	X	X
Q2'Q1Q0 + Q2Q1'Q0' + Q3	Q3'Q2'Q1' + Q2Q1Q0' + Q3'Q1'Q0	Q2'Q1 + Q2Q1'Q0' + Q1Q0 + Q3	Q3'Q2'Q0' + Q2Q0	Q2Q1'Q0' + Q3Q0	Q3'Q2'Q1' + Q1Q0' + Q2Q0	Q2'Q1Q0 + Q3Q0'	Q2'Q1Q0 + Q2Q1'Q0' + Q3

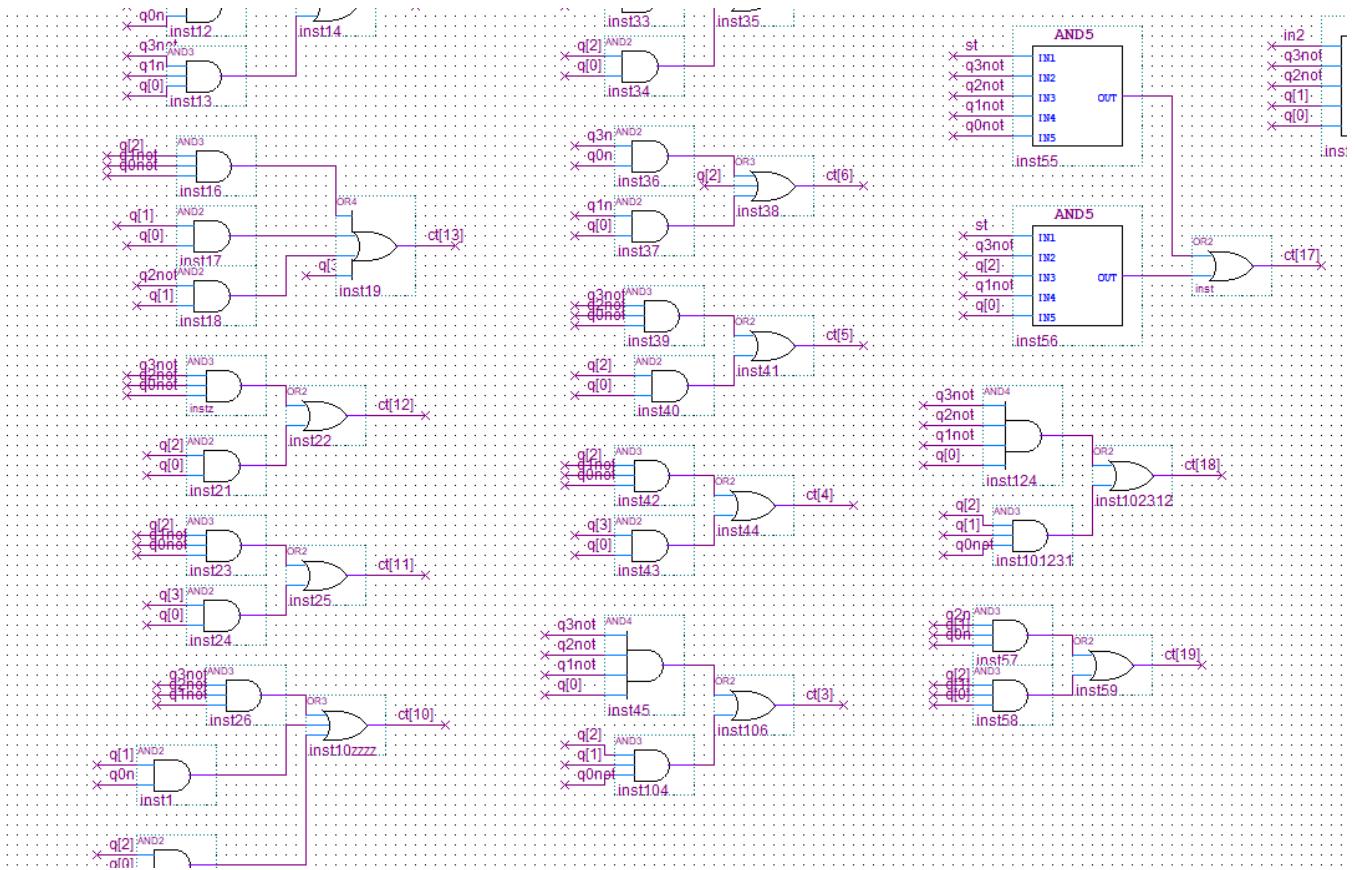
- Ct[15] điều khiển tristate của R1 tại bus1.
- Ct[14] tín hiệu cho phép ghi vào R1.
- Ct[13] điều khiển tristate(Cổng đọc) của R2.
- Ct[12] tín hiệu cho phép ghi vào R2.
- Ct[11] điều khiển tristate của khối Shift>>3.

- Ct[10] điều khiển tristate của AU2.
- Ct[9..8] chọn chế độ hoạt động cho AU1 (00: ABS, 01:MAX, 11:MIN) .

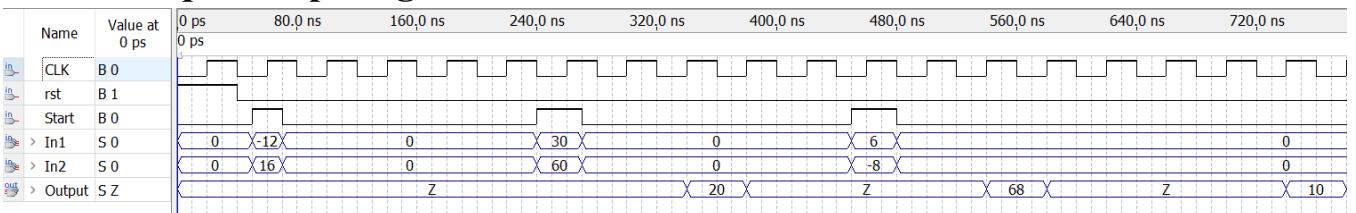
Read_R3	Write_R3	Read_R4	Write_R4	Read_R5	Write_R5	AU2	
1	1	1	0	0	0	0	1
1	1	0	0	1	0	0	0
1	1	1	0	0	0	1	1
0	0	0	0	0	1	0	0
0	1	0	1	0	0	0	0
1	1	1	0	0	0	0	1
1	1	0	0	1	0	0	0
1	1	1	0	0	0	1	1
0	0	0	0	0	1	0	0
0	1	0	1	0	0	0	0
X	X	X	X	X	X	X	X
Q3'Q2'Q1' + Q1Q0' + Q2Q0	Q2 + Q3'Q0' + Q1'Q0	Q3'Q2'Q0' + Q2Q0	Q2Q1'Q0' + Q3Q0	Q3'Q2'Q1'Q0 + Q2Q1Q0'	Q2'Q1Q0 + Q3Q0'	Q2'Q1Q0' + Q2Q1Q0	Q3'Q2'Q0' + Q2Q0

- Ct[7] điều khiển tristate của R3.
- Ct[6] tín hiệu cho phép ghi R3.
- Ct[5] điều khiển tristate của R4.
- Ct[4] tín hiệu cho phép ghi R4.
- Ct[3] điều khiển tristate của R5.
- Ct[2] tín hiệu cho phép ghi R5.
- Ct[1..0] chọn chế độ hoạt động AU2 (00:'+',01:'-',11:'max').





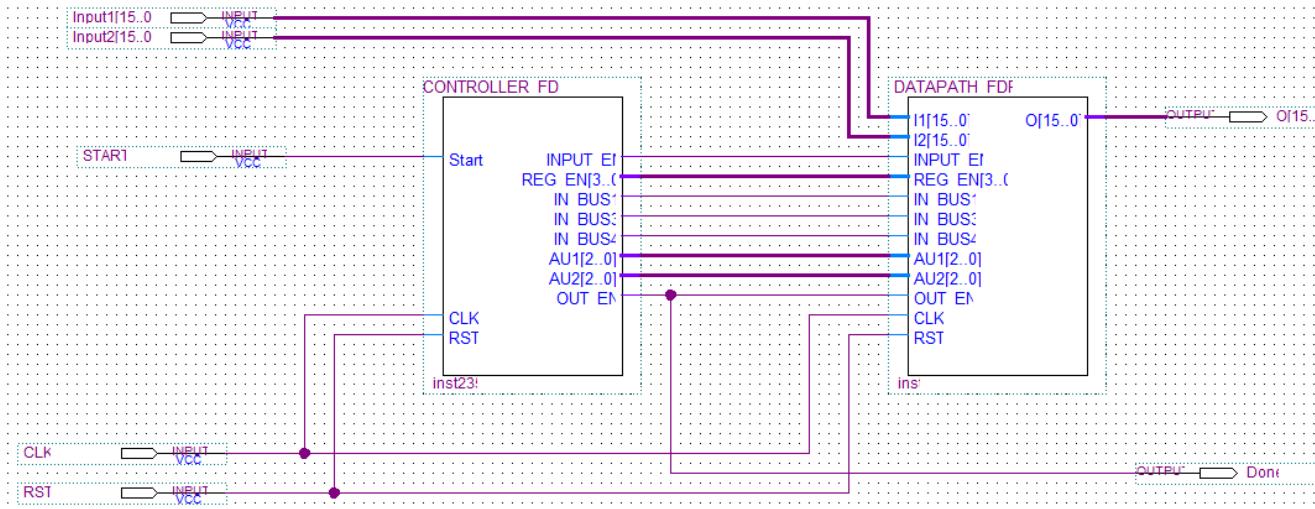
4. Kết quả mô phỏng:



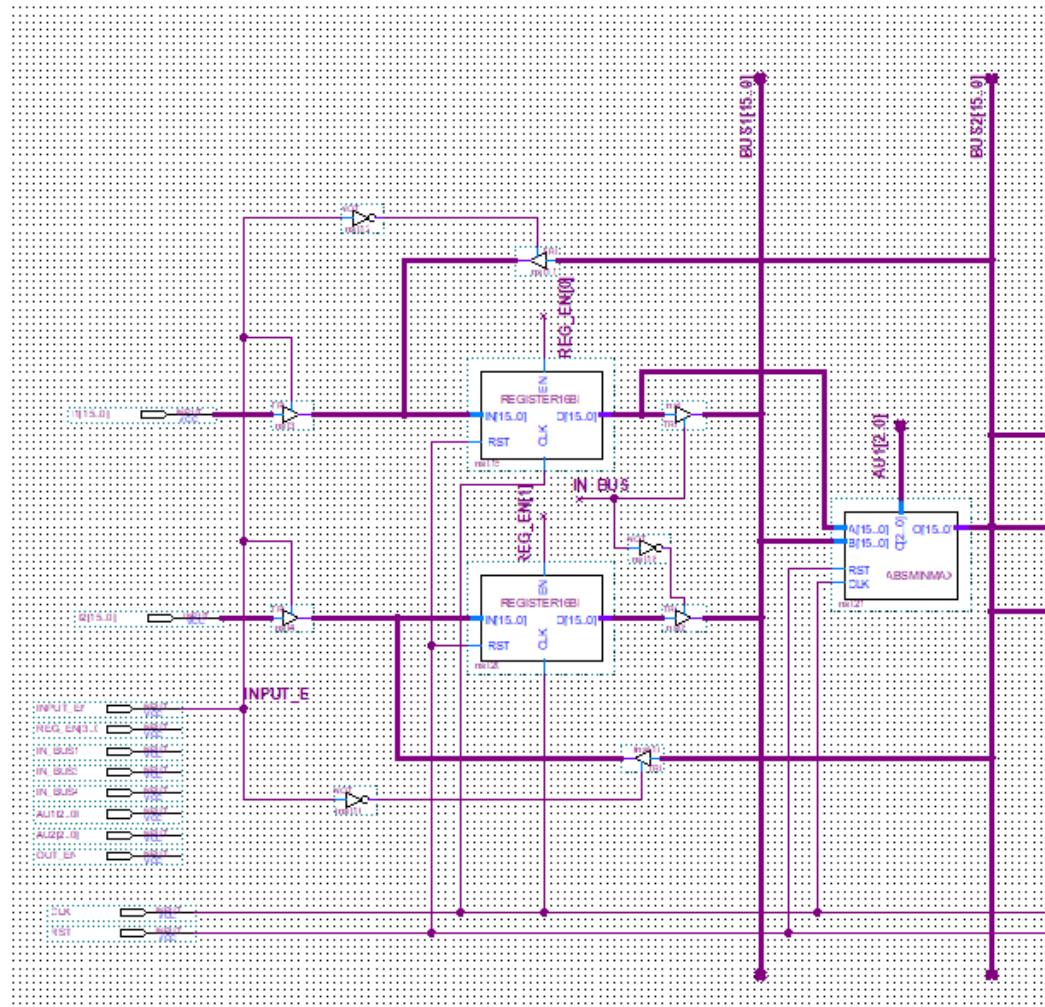
- Mạch hoạt động đúng chức năng Pipeline như mong muốn. Kể từ cạnh lên clk nhận input đến lúc cho ra output mất 7 chu kỳ.
- Sau 5 chu kỳ kể từ khi nhận input trước có thể nạp input tiếp theo.

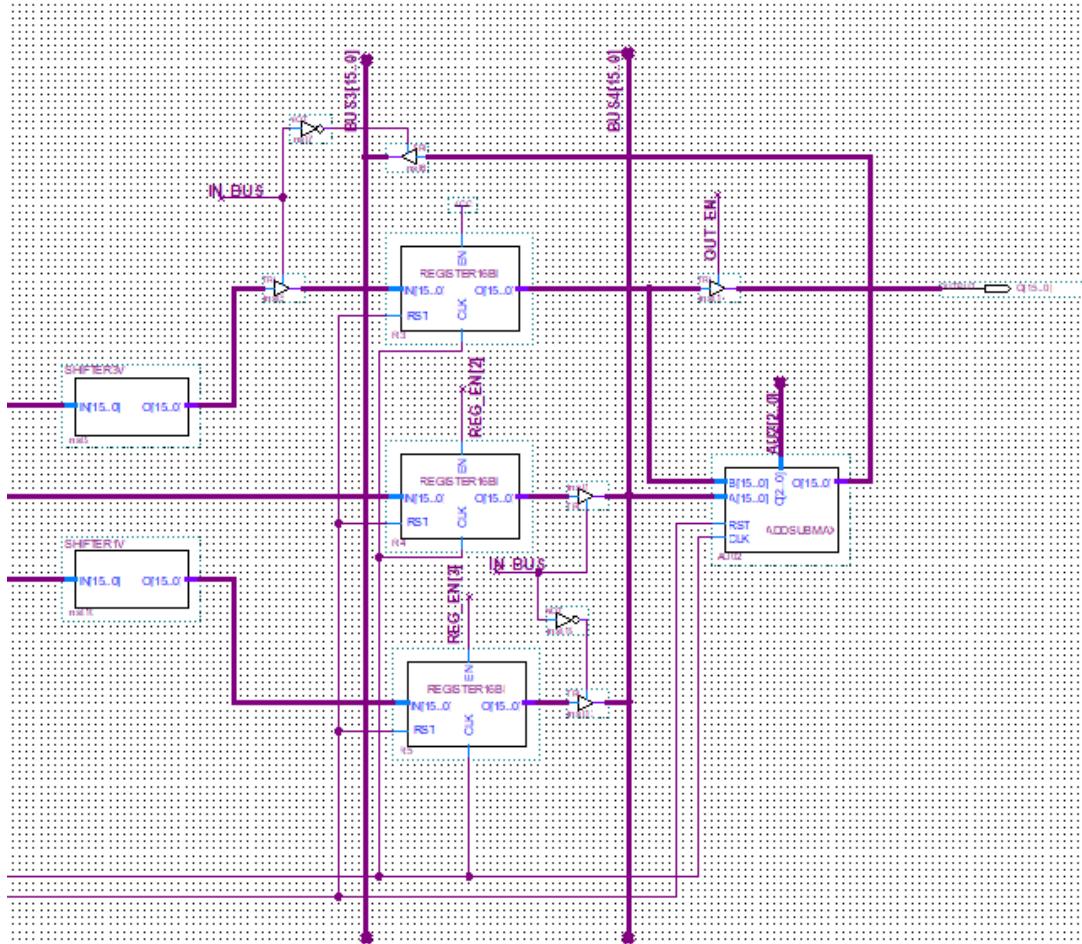
Bài 18: Thiết kế mạch tính xấp xỉ căn bậc 2 Datapath & Functional Unit Pipelining

1. Tổng quan:



2. Thiết kế Datapath:





$$R1 = [a, t1] \quad R3 = [t3, t5, t6, t7]$$

$$R2 = [b, t2] \quad R4 = [x]$$

$$R5 = [t4]$$

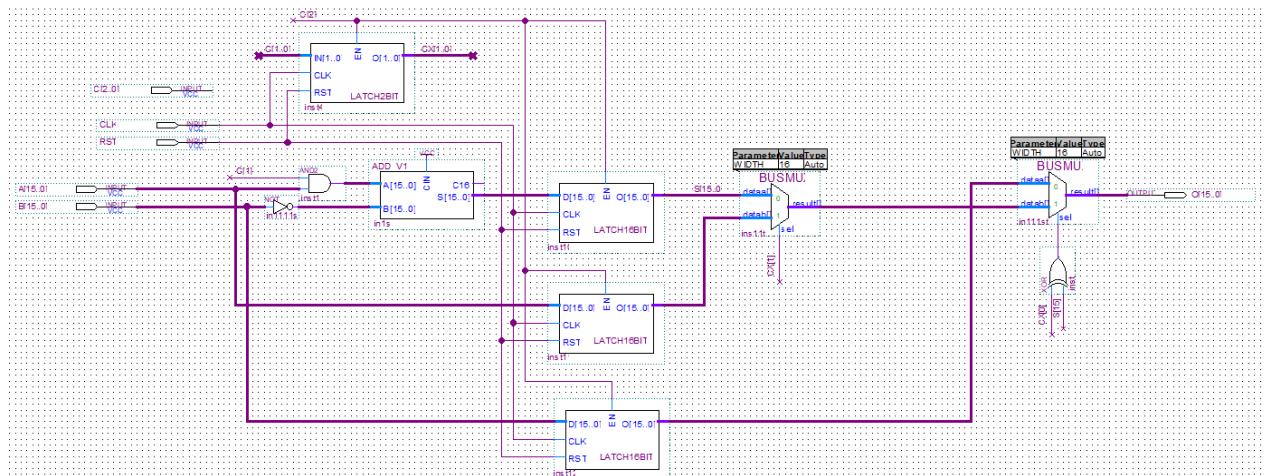
AU1 = [abs/min/max]

AU2[+/-max]

Giải thích các tín hiệu:

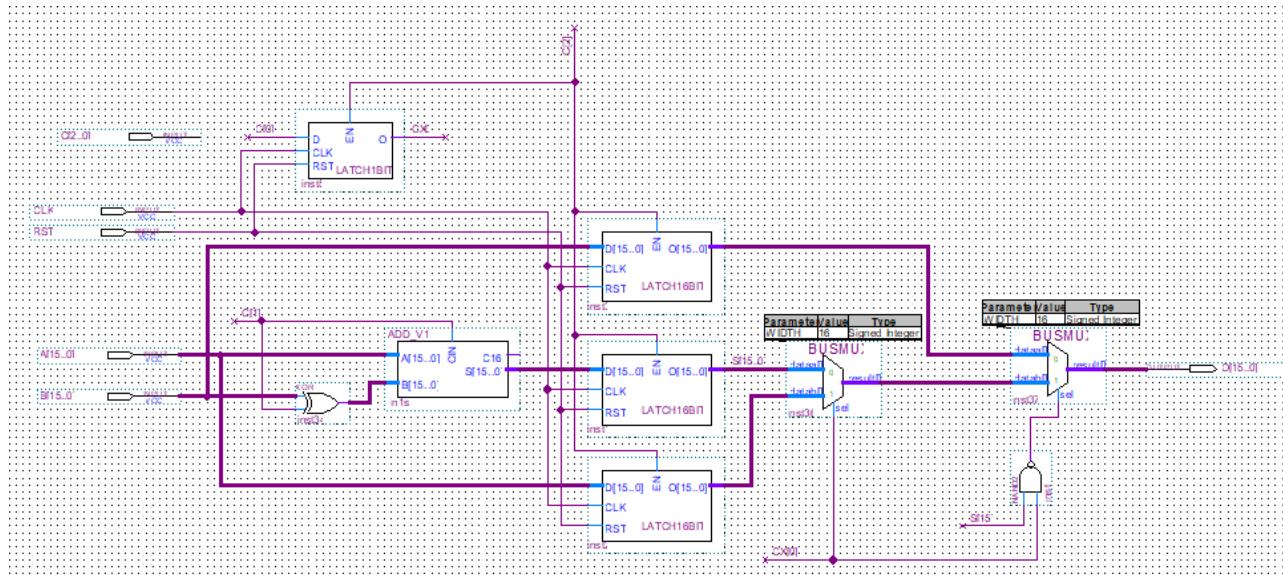
- INPUT_EN: Chọn đầu vào cho R1, R2.
 - + INPUT_EN = 1: Nhận giá trị được nhập vào.
 - + INPUT_EN = 0: Nhận giá trị đầu ra của khối AU1.
- REG_EN[3..0]: Cho phép ghi đè dữ liệu vào thanh ghi khi REG_EN[x] = 1 và không cho phép ghi đè khi REG_EN[x] = 0. Thanh ghi R3 trong mạch có thể luôn ở trạng thái được ghi đè mà không ảnh hưởng đến kết quả.
 - + REG_EN[0]: điều khiển thanh ghi R1.
 - + REG_EN[1]: điều khiển thanh ghi R2.

- + REG_EN[2]: điều khiển thanh ghi R4.
- + REG_EN[3]: điều khiển thanh ghi R5.
- IN_BUS1: Chọn đường đi dữ liệu được đi vào dây BUS 1.
 - + IN_BUS1 = 1: R1 được ghi vào BUS 1.
 - + IN_BUS1 = 0: R2 được ghi vào BUS 1.
- IN_BUS3: Chọn đường đi dữ liệu được đi vào dây BUS 3.
 - + IN_BUS1 = 1: đầu ra của AU1 được ghi vào BUS 3.
 - + IN_BUS1 = 0: đầu ra của AU2 được ghi vào BUS 3.
- IN_BUS4: Chọn đường đi dữ liệu được đi vào dây BUS 4.
 - + IN_BUS1 = 1: R4 được ghi vào BUS 4.
 - + IN_BUS1 = 0: R5 được ghi vào BUS 4.
- AU1[2..0]: Chọn chức năng thực hiện cho AU1.
 - + AU1 = 101: tính $|B|$.
 - + AU1 = 110: tìm $\min(A, B)$.
 - + AU1 = 111: tìm $\max(A, B)$.



AU2[2..0]: Chọn chức năng thực hiện cho AU2.

- + AU2 = 100: tính $A + B$.
- + AU2 = 110: tính $A - B$.
- + AU2 = 111: tìm $\max(A, B)$.



- OUT_EN: Cho phép xuất kết quả thuật toán.
 - + OUT_EN = 0: Không cho phép xuất kết quả.
 - + OUT_EN = 1: Cho phép xuất kết quả.
- RST: Đặt giá trị của tất cả các flip flop, latch trong mạch thành 0.
 - + RST = 0: Đặt tất cả các flip flop, latch trong mạch thành 0.
 - + RST = 1: Không ảnh hưởng lên mạch.

	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13
Đọc R1					t1	t1								
Đọc R2		a	b		t2	t2								
AU ₁ tầng 1		a	b		min	max								
AU ₁ tầng 2			a	b		min	max							
Dịch bit						>>1	>>3							
Ghi R1	a		t1											
Ghi R2	b			t2										

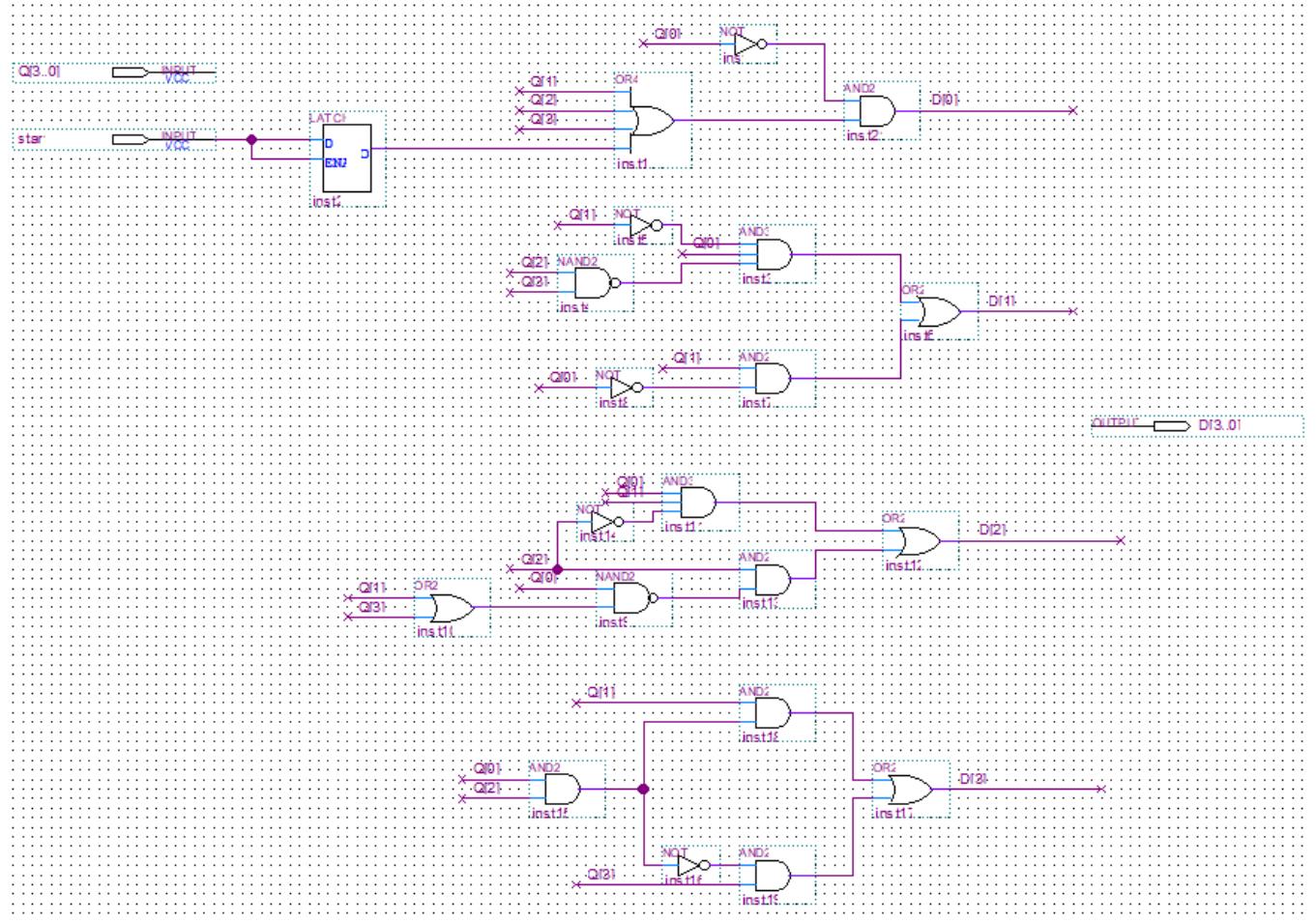
Đọc R3							t3	x	-		t5		t6	x		t7
Đọc R4										t4						
Đọc R5																
AU ₂ tầng 1																
AU ₂ tầng 2																
Ghi R3							t3	x	-	t5		t6		t7	t3	
Ghi R4																x
Ghi R5																
In kết quả							t4								t4	t7

3. Thiết kế Controller:

- Có 14 trạng thái \Rightarrow sử dụng 4 flip flop.
- Khi có tín hiệu start = 1 thì mạch bắt đầu chạy.

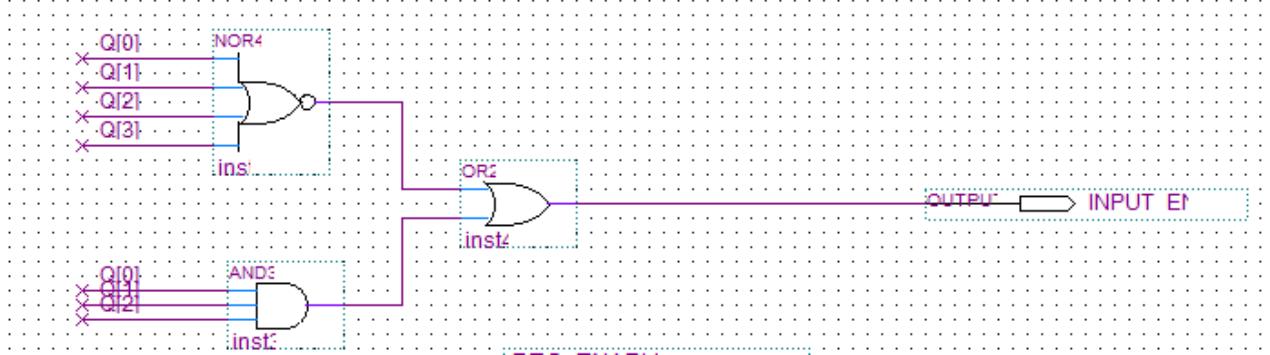
RST	Start	Q[3..0]	D[3..0]
0	X	X	0
1	0	0000	0000
1	1	0000	0001
1	X	0001	0010
1	X	0010	0011
1	X	0011	0100
1	X	0100	0101
1	X	0101	0110

1	X	0110	0111
1	X	0111	1000
1	X	1000	1001
1	X	1001	1010
1	X	1010	1011
1	X	1011	1100
1	X	1100	1101
1	X	1101	0000

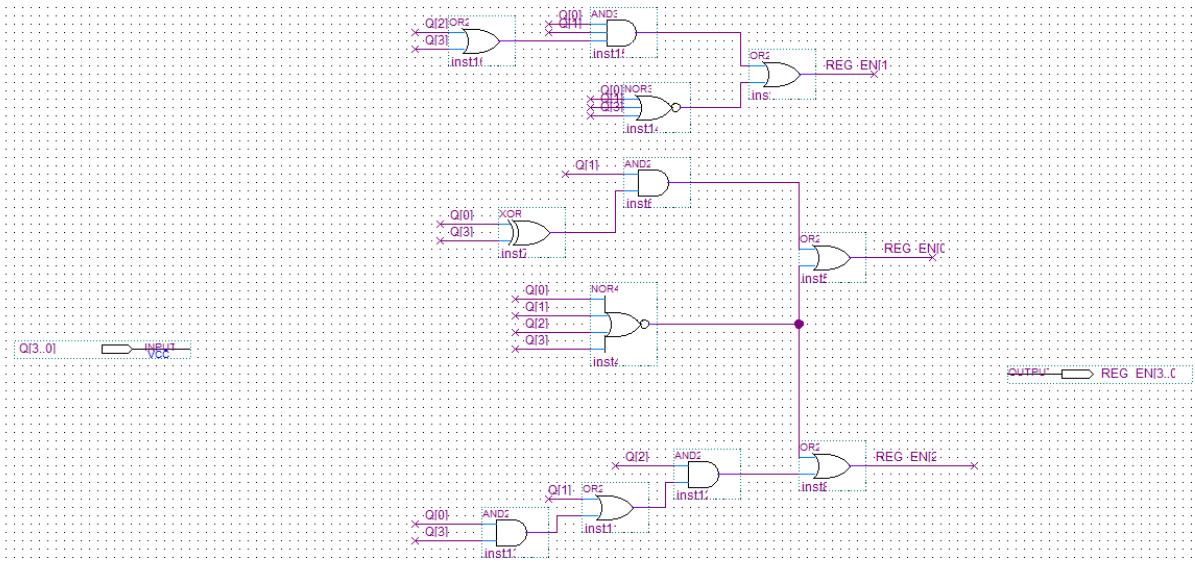


	INPUT_EN	REG_EN	IN_BUS1	IN_BUS3	IN_BUS4	AU1	AU2	OUT_EN
s0	1	0111	0	1	0	000	000	1
s1	0	0000	1	0	1	000	110	0
s2	0	0000	1	0	0	101	000	0
s3	0	0001	0	0	0	101	100	0
s4	0	0010	0	0	0	000	000	0
s5	0	1000	0	0	1	110	111	0
s6	0	1100	0	0	0	111	000	0
s7	1	0111	0	1	0	000	000	1
s8	0	0000	1	0	1	000	110	0
s9	0	0000	1	0	0	101	000	0
s10	0	0001	0	0	0	101	100	0
s11	0	0010	0	0	0	000	000	0
s12	0	1000	0	0	1	110	111	0
s13	0	1100	0	0	0	111	000	0

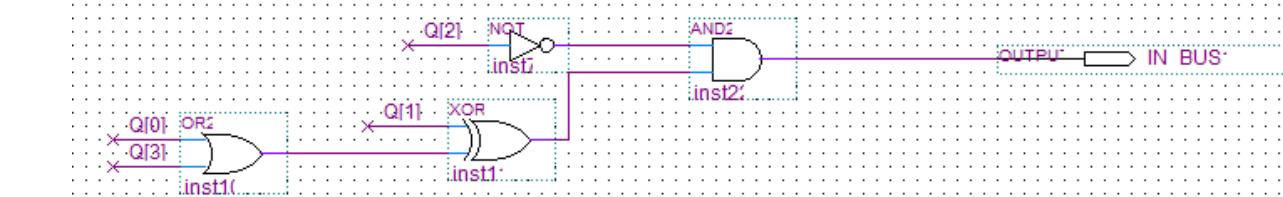
- INPUT_EN:



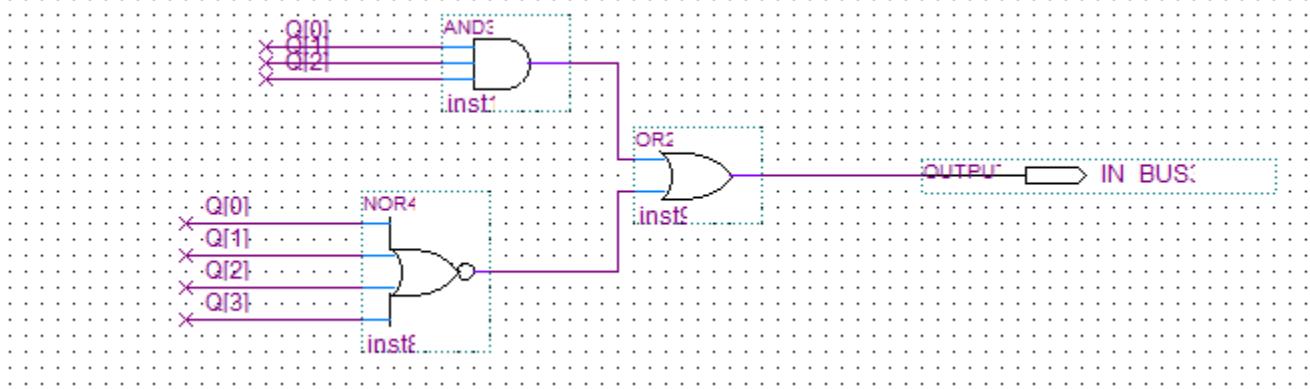
- REG_EN:



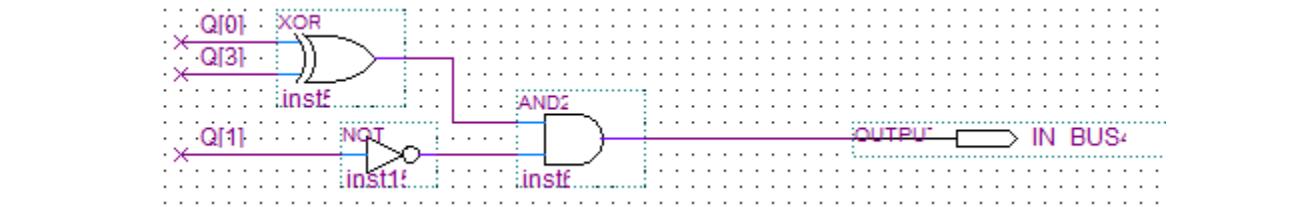
- IN_BUS1:



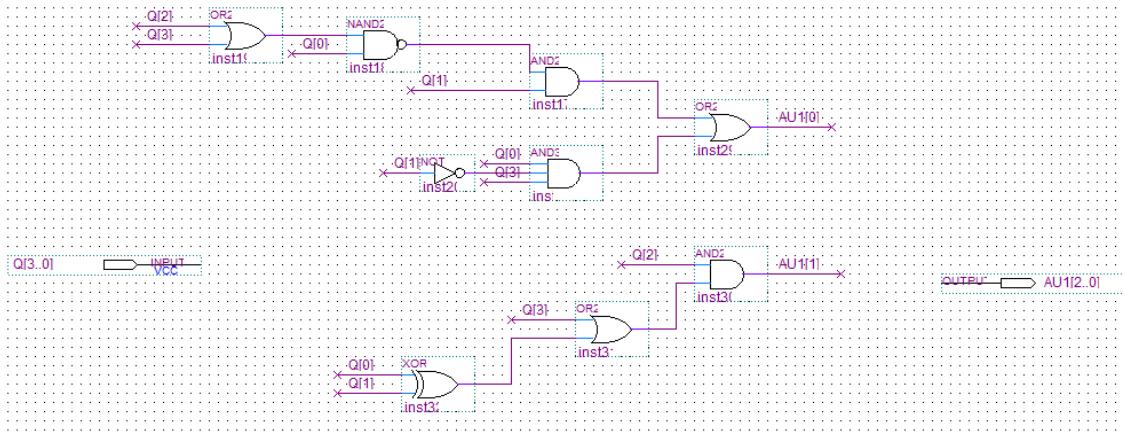
- IN_BUS3:



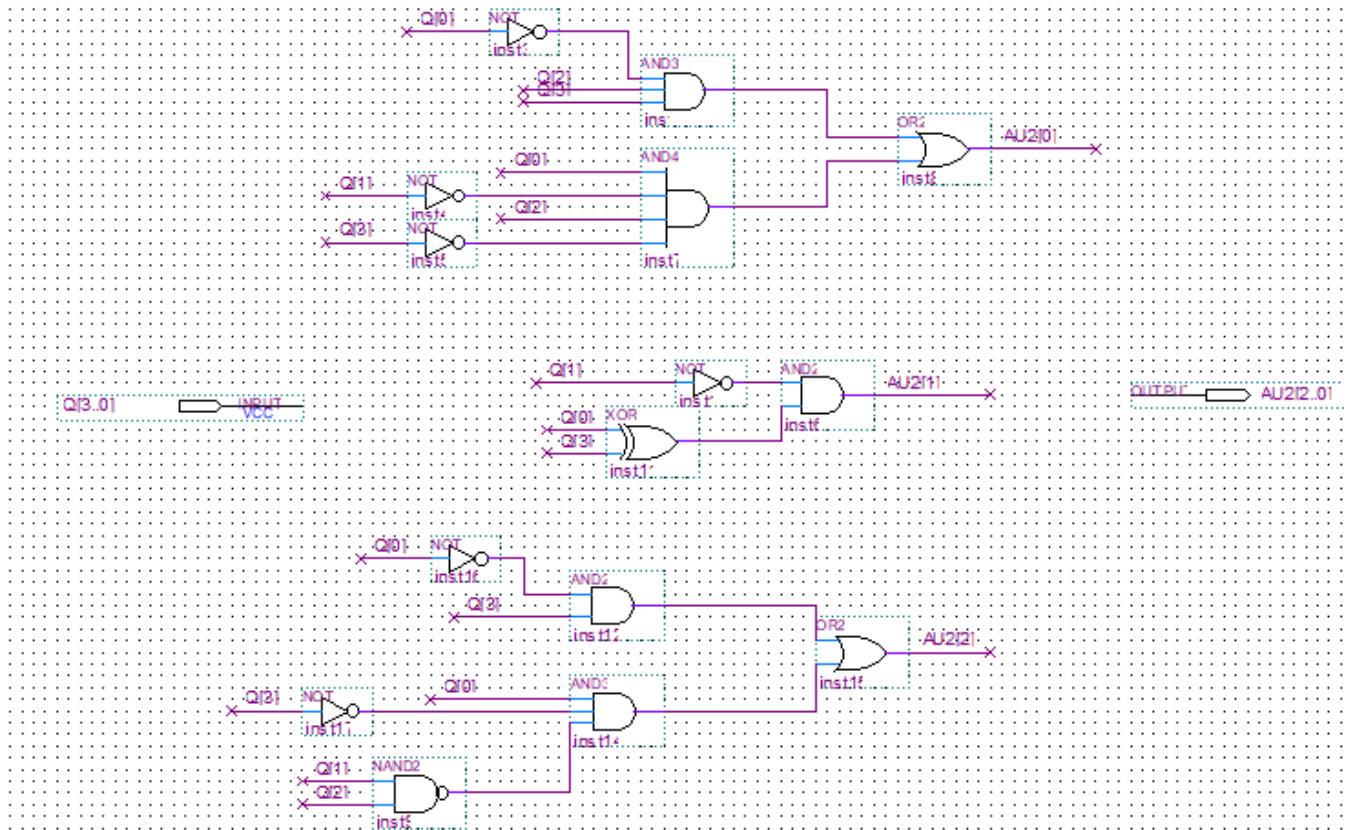
- IN_BUS4:



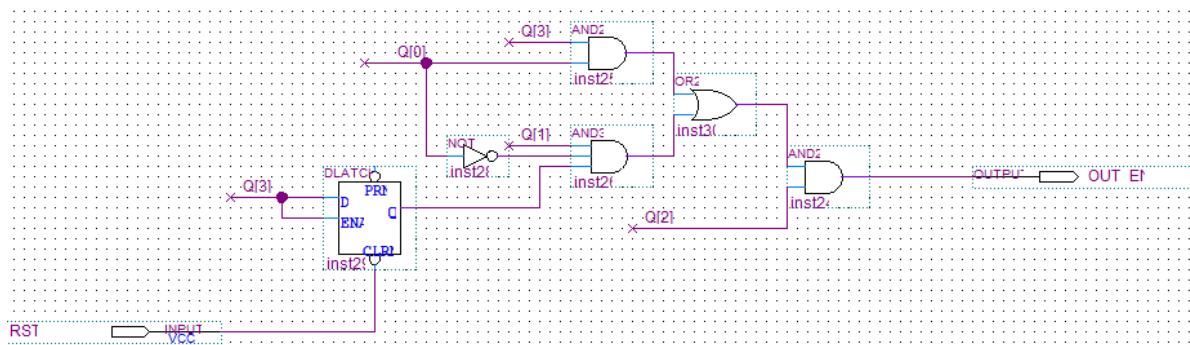
- AU1:

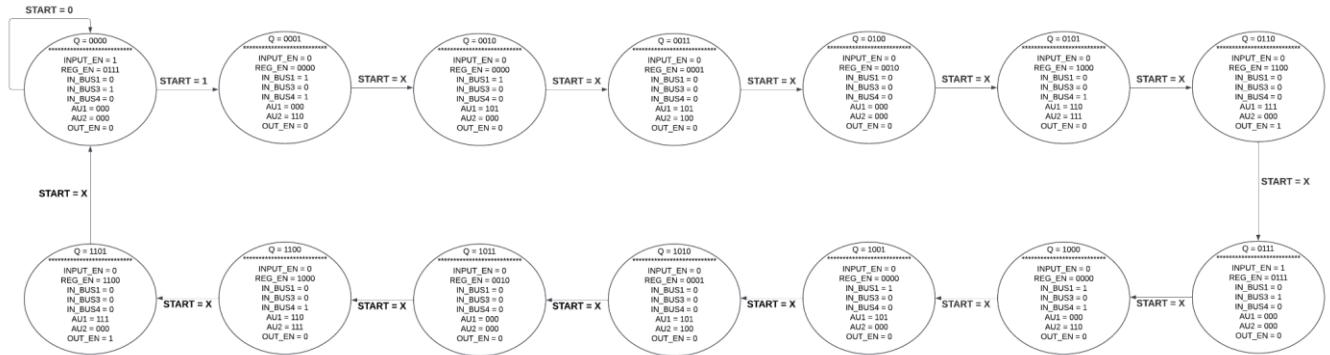


- AU2:

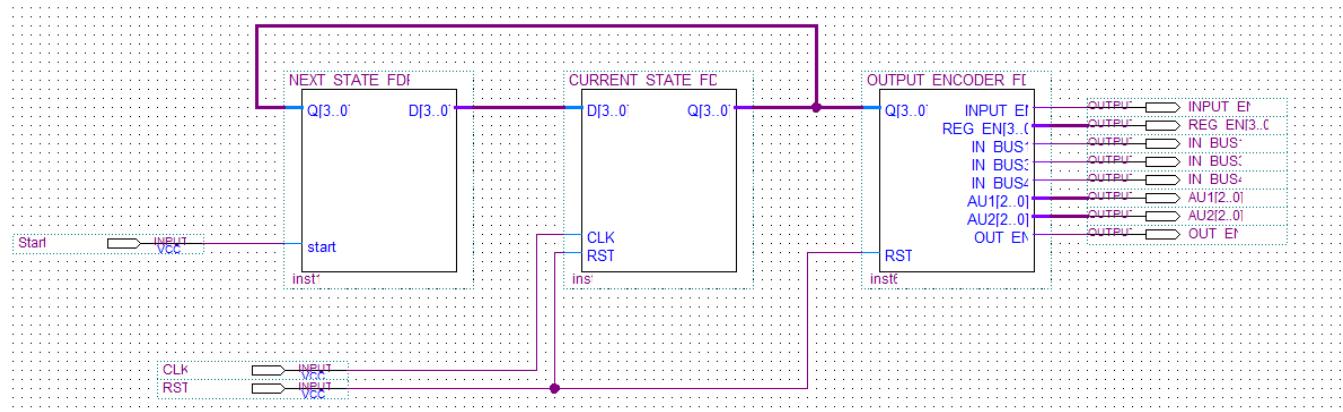


- OUT_EN:

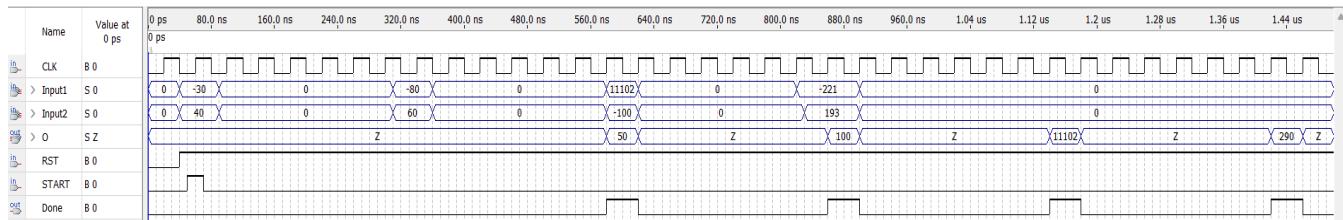




* Khối controller:



4. Kết quả mô phỏng:



- Ở chu kỳ đầu, đặt reset và không bật start \rightarrow mạch không hoạt động.
- Bật start và nhập Input1 = -30, Input2 = 40.
- Đến chu kỳ thứ 7, thực hiện nhập tiếp Input1 = -80, Input2 = 60.
- Đến chu kỳ thứ 14, đầu ra hiển thị 50 $\Rightarrow (-30)^2 + 40^2 = 50^2 \Rightarrow$ Mạch cho ra kết quả của lần nhập đầu tiên chính xác sau 14 chu kỳ. Đồng thời nhập thêm Input1 = 11102, Input2 = -100 ở chu kỳ này.
- Đến chu kỳ thứ 21, đầu ra hiển thị 100 $\Rightarrow (-80)^2 + 60^2 = 100^2 \Rightarrow$ Mạch cho ra kết quả của lần nhập thứ 2 chính xác sau 7 chu kỳ kể từ lần hiển thị kết quả của lần nhập thứ nhất. Đồng thời nhập thêm Input1 = -221, Input2 = 193 ở chu kỳ này.

- Đến chu kỳ thứ 28, đầu ra hiển thị $11102 \Rightarrow 11102^2 + (-100)^2 = (11102.45)^2$
 \Rightarrow Mạch cho ra kết quả của lần nhập thứ 3 xấp xỉ đúng sau 7 chu kỳ kể từ lần hiển thị kết quả của lần nhập thứ 2.
- Đến chu kỳ thứ 35, đầu ra hiển thị $290 \Rightarrow (-221)^2 + 193^2 = (293.41)^2 \Rightarrow$
Mạch cho ra kết quả của lần nhập thứ 4 xấp xỉ đúng sau 7 chu kỳ kể từ lần hiển thị kết quả của lần nhập thứ 3.
 \Rightarrow Ở lần tính đầu tiên cần 14 trạng thái, nhưng các lần sau chỉ mất 7 trạng thái, thời gian thực thi n lệnh sẽ là $7n + 7$ chu kỳ.