

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

-----☆-----



BÁO CÁO CUỐI KỲ
BÀI TẬP MÔN THIẾT KẾ LUẬN LÝ SỐ

Lớp: CE118.N21

Giảng viên hướng dẫn: TS. Lâm Đức Khải

Nhóm sinh viên thực hiện:

- | | |
|--------------------------------|----------|
| 1. Đào Phước Tài (50%) | 21521391 |
| 2. Nguyễn Quốc Trường An (50%) | 21521810 |

TP.Hồ Chí Minh, ngày 31 tháng 05 năm 2023

MỤC LỤC

DANH MỤC HÌNH ẢNH	8
DANH MỤC BẢNG	12
BÀI TẬP 1. REGISTER SHARING	15
I. TỔNG QUÁT THIẾT KẾ	15
II. THIẾT KẾ KHỐI DATAPATH	19
1. Mô hình tổng quát khối datapath	19
2. Phân tích các thành phần trong khối datapath	19
3. Thiết kế Register 8 bit	20
4. Thiết kế bộ chọn MUX 2to1 (8 bit)	20
5. Thiết kế các bộ chọn MUX 3to1, MUX 5to1	21
6. Thiết kế khối cộng/trừ	22
7. Thiết kế khối tính ABS	24
8. Thiết kế khối tính MAX	25
9. Thiết kế khối tính MIN	26
10. Thiết kế khối dịch phải (1 bit, 3 bit)	27
11. Tổng hợp thiết kế datapath	28
12. Mô phỏng datapath và phân tích	28
III. THIẾT KẾ KHỐI CONTROLLER	31
1. Sơ đồ trạng thái	31
2. Mã hóa trạng thái	32
3. Chọn loại flipflop và lập bảng chuyển trạng thái.	33
4. Thiết kế khối trạng thái kế tiếp	33
5. Thiết kế khối nhớ	34
6. Thiết kế khối ngõ ra tạo tín hiệu điều khiển	34
7. Tổng hợp thiết kế controller	35
IV. TỔNG HỢP THIẾT KẾ VÀ MÔ PHỎNG	36
1. Tổng hợp thiết kế	36
2. Mô phỏng và phân tích	37

BÀI TẬP 2. FUNCTIONAL - UNIT SHARING.....	38
I. TỔNG QUÁT THIẾT KẾ	38
II. THIẾT KẾ KHỐI DATAPATH.....	40
1. Mô hình tổng quát khối datapath	40
2. Phân tích các thành phần trong khối datapath.....	40
3. Thiết kế khối tính ABS/MAX.....	41
4. Thiết kế khối tính ADD/MIN/SUB/ABS.....	43
5. Tổng hợp thiết kế datapath	45
6. Mô phỏng datapath và phân tích.....	46
III. THIẾT KẾ KHỐI CONTROLER.....	49
1. Sơ đồ trạng thái	49
2. Mã hóa trạng thái.....	50
3. Chọn loại flipflop và lập bảng chuyển trạng thái.....	50
4. Thiết kế khối trạng thái kế tiếp.....	51
5. Thiết kế khối nhớ.....	51
6. Thiết kế khối ngõ ra tạo tín hiệu điều khiển.....	52
7. Tổng hợp thiết kế controller.....	54
IV. TỔNG HỢP THIẾT KẾ VÀ CHẠY MÔ PHỎNG	54
1. Tổng hợp thiết kế.....	54
2. Mô phỏng và phân tích	55
BÀI TẬP 3. REGISTER MERGING.....	56
I. TỔNG QUÁT THIẾT KẾ	56
II. THIẾT KẾ KHỐI DATAPATH	58
1. Mô hình tổng quát khối datapath	58
2. Phân tích các thành phần trong khối datapath.....	58
3. Thiết kế Register Files R1.R3.....	59
4. Tái sử dụng thiết kế Register 8 bit.....	60
5. Tái sử dụng thiết kế khối tính ABS/MAX.....	60
6. Tái sử dụng thiết kế khối tính ADD/MIN/SUB/ABS	60

7.	Tổng hợp thiết kế datapath	61
III.	THIẾT KẾ KHỐI CONTROLLER	62
1.	Sơ đồ trạng thái	62
2.	Mã hóa trạng thái	63
3.	Chọn loại flipflop và lập bảng chuyển trạng thái.....	63
4.	Thiết kế khối trạng thái kế tiếp.....	64
5.	Thiết kế khối nhớ.....	64
6.	Thiết kế khối ngõ ra	65
7.	Tổng hợp thiết kế controller.....	67
IV.	TỔNG HỢP THIẾT KẾ VÀ CHẠY MÔ PHỎNG	67
1.	Tổng hợp thiết kế.....	67
2.	Mô phỏng và phân tích	68
BÀI TẬP 4. PIPELINED FUNCTIONAL UNIT	69	
I.	TỔNG QUÁT THIẾT KẾ	69
II.	THIẾT KẾ KHỐI DATAPATH	72
1.	Mô hình tổng quát khối datapath	72
2.	Phân tích các thành phần trong khối datapath.....	72
3.	Thiết kế khối Pipeline AU [ADD/ABS/SUB/MIN/MAX]	73
4.	Tổng hợp thiết kế datapath	75
III.	THIẾT KẾ KHỐI CONTROLLER	76
1.	Sơ đồ trạng thái	76
2.	Mã hóa trạng thái	77
3.	Chọn loại flipflop và lập bảng chuyển trạng thái.....	78
4.	Thiết kế khối trạng thái kế tiếp.....	79
5.	Thiết kế khối nhớ.....	79
6.	Thiết kế khối ngõ ra	80
7.	Tổng hợp thiết kế controller.....	82
IV.	TỔNG HỢP THIẾT KẾ VÀ MÔ PHỎNG	82
1.	Tổng hợp thiết kế.....	82

2. Mô phỏng và phân tích	83
BÀI TẬP 5. DATAPATH PIPELINING.....	84
I. TỔNG QUÁT THIẾT KẾ	84
II. THIẾT KẾ KHỐI DATAPATH.....	87
1. Mô hình tổng quát khối datapath.....	87
2. Phân tích các thành phần trong khối datapath.....	87
3. Thiết kế khối AU 1 (ABS/MIN/MAX)	88
4. Thiết kế khối AU 2 (ADD/SUB/MAX).....	90
5. Tái sử dụng các thanh ghi	92
6. Tái sử dụng khối dịch phải 1 bit	92
7. Tái sử dụng khối dịch phải 3 bit	92
8. Tổng hợp thiết kế datapath	93
III. THIẾT KẾ KHỐI CONTROLLER	94
1. Sơ đồ chuyển trạng thái.....	94
2. Mã hóa trạng thái.....	96
3. Chọn loại flipflop và lập bảng chuyển trạng thái.....	97
4. Thiết kế khối trạng thái kế tiếp.....	98
5. Thiết kế khối nhớ.....	98
6. Thiết kế khối ngõ ra	99
7. Tổng hợp thiết kế controller.....	102
IV. TỔNG HỢP THIẾT KẾ VÀ MÔ PHỎNG.....	102
1. Tổng hợp thiết kế.....	102
2. Mô phỏng và phân tích	103
BÀI TẬP 6. DATAPATH PIPELINE - AU PIPELINE.....	104
I. TỔNG QUÁT THIẾT KẾ	104
II. THIẾT KẾ KHỐI DATAPATH	106
1. Mô hình tổng quát khối datapath.....	106
2. Phân tích các thành phần trong khối datapath.....	106
3. Thiết kế khối AU 1.....	107

4.	Thiết kế khối AU 2.....	109
5.	Tái sử dụng Register 2 bit.....	110
6.	Tái sử dụng Register 8 bit.....	111
7.	Tái sử dụng khối dịch phải 1 bit	111
8.	Tái sử dụng khối dịch phải 3 bit	111
9.	Tổng hợp thiết kế datapath	112
III.	THIẾT KẾ KHỐI CONTROLLER	113
1.	Sơ đồ trạng thái	113
2.	Mã hóa trạng thái.....	114
3.	Chọn loại flipflop và lập bảng chuyển trạng thái.....	115
4.	Thiết kế khối trạng thái kế tiếp.....	116
5.	Thiết kế khối nhớ.....	117
6.	Thiết kế khối ngõ ra tạo tín hiệu điều khiển.....	117
7.	Tổng hợp thiết kế controller.....	121
IV.	TỔNG HỢP THIẾT KẾ VÀ MÔ PHỎNG	122
1.	Tổng hợp thiết kế.....	122
2.	Mô phỏng và phân tích	123

DANH MỤC HÌNH ẢNH

Hình 1.1: Tổng quát thiết kế Register Sharing	15
Hình 1.2: Giải thuật left-edge	16
Hình 1.3: Biểu đồ ASM cho bài toán tính xấp xỉ căn bậc hai theo thiết kế Register Sharing	17
Hình 1.4: Sơ đồ thiết kế khối đường dữ liệu của bài toán Register Sharing....	19
Hình 1.5: Register 8 bit.....	20
Hình 1.6: Mạch MUX 2to1 (1 bit)	20
Hình 1.7: Mạch MUX 2to1 (8 bit)	21
Hình 1.8: Mạch MUX 3to1	21
Hình 1.9: Mạch MUX 5to1	21
Hình 1.10: Sơ đồ thiết kế khối cộng/trừ.....	22
Hình 1.11: Mạch khối Full_Adder 1 bit	23
Hình 1.12: Mạch khối cộng/trừ 8 bit.....	23
Hình 1.13: Sơ đồ thiết kế khối tính ABS	24
Hình 1.14: Mạch khối tính ABS.....	24
Hình 1.15: Sơ đồ thiết kế khối tính MAX	25
Hình 1.16: Mạch khối tính MAX.....	25
Hình 1.17: Sơ đồ thiết kế khối tính MIN	26
Hình 1.18: Mạch khối tính MIN	26
Hình 1.19: Mạch khối dịch phải 1 bit (>>1)	27
Hình 1.20: Mạch khối dịch phải 3 bit (>>3)	27
Hình 1.21: Tổng hợp datapath theo thiết kế Register Sharing.....	28
Hình 1.22: Kết quả chạy mô phỏng datapath	28
Hình 1.23: Sơ đồ trạng thái theo thiết kế Register Sharing.....	31
Hình 1.24: Mạch khối trạng thái kế tiếp.....	33
Hình 1.25: Mạch khối nhớ.....	34
Hình 1.26: Mạch khối ngõ ra tạo tín hiệu điều khiển.....	35
Hình 1.27: Tổng hợp thiết kế controller	35
Hình 1.28: Tổng hợp thiết kế Register Sharing	36
Hình 1.29: Kết quả mô phỏng theo thiết kế Register Sharing.....	37
Hình 2.1: Tổng quát thiết kế Functional – Unit Sharing.....	38
Hình 2.2: Biểu đồ ASM cho bài toán tính xấp xỉ căn bậc hai theo thiết kế Functional - Unit Sharing.....	39
Hình 2.3: Sơ đồ thiết kế khối đường dữ liệu của bài toán Functional – Unit Sharing	40
Hình 2.4: Sơ đồ thiết kế khối tính ABS/MAX	41

Hình 2.5: Mạch khối tính ABS/MAX	42
Hình 2.6: Kết quả mô phỏng khối tính ABS/MAX	42
Hình 2.7: Sơ đồ thiết kế khối tính ADD/MIN/SUB/ABS	43
Hình 2.8: Mạch khối tính ADD/MIN/SUB/ABS	44
Hình 2.9: Mô phỏng khối tính ADD/MIN/SUB/ABS	45
Hình 2.10: Tổng hợp datapath theo thiết kế Functional - Unit Sharing	45
Hình 2.11: Mô phỏng datapath	46
Hình 2.12: Sơ đồ trạng thái theo thiết kế Functional – Unit Sharing	49
Hình 2.13: Mạch khối trạng thái kế tiếp	51
Hình 2.14: Mạch khối nhớ	51
Hình 2.15: Mạch khối ngõ ra	53
Hình 2.16: Tổng hợp thiết kế controller	54
Hình 2.17: Mô phỏng khối controller	54
Hình 2.18: Tổng hợp thiết kế Functional - Unit Sharing	54
Hình 2.19: Kết quả mô phỏng theo thiết kế Functional - Unit Sharing	55
Hình 3.1: Tổng quát thiết kế Register Merging	56
Hình 3.2: Biểu đồ ASM cho bài toán tính xấp xỉ căn bậc hai theo thiết kế Register Merging	57
Hình 3.3: Sơ đồ thiết kế khối đường dữ liệu thực thi Register Merging	58
Hình 3.4: Mạch Register Files Cell	59
Hình 3.5: Mạch Register Files 8x1	59
Hình 3.6: Mạch Register Files 8x2	59
Hình 3.7: Khối REGISTER 8 bit	60
Hình 3.8: Khối tính ABS/MAX	60
Hình 3.9: Khối tính ADD/MIN/SUB/ABS	60
Hình 3.10: Tổng hợp datapath theo thiết kế Register Merging	61
Hình 3.11: Sơ đồ trạng thái theo thiết kế Register Merging	62
Hình 3.12: Mạch khối trạng thái kế tiếp	64
Hình 3.13: Mạch khối nhớ	64
Hình 3.14: Mạch khối ngõ ra	66
Hình 3.15: Mạch tổng quát khối controller	67
Hình 3.16: Tổng hợp thiết kế Register Merging	67
Hình 3.17: Kết quả mô phỏng theo thiết kế Register Merging	68
Hình 4.1: Tổng quát thiết kế Pipelined Functional Unit	69
Hình 4.2: Biểu đồ ASM cho bài toán tính xấp xỉ căn bậc hai theo thiết kế AU không pipeline	70
Hình 4.3: Sơ đồ thiết kế khối đường dữ liệu thực thi Pipeline AU	72

Hình 4.4: Sơ đồ khối AU Pipeline [ADD/ABS/SUB/MIN/MAX].....	73
Hình 4.5: Mạch khối Pipeline AU	74
Hình 4.6: Tổng hợp thiết kế khối datapath	75
Hình 4.7: Sơ đồ trạng thái theo thiết kế Pipelined Functional Unit	76
Hình 4.8: Mạch khối trạng thái kế tiếp.....	79
Hình 4.9: Mạch khối nhớ.....	79
Hình 4.10: Mạch khối ngõ ra	81
Hình 4.11: Tổng hợp thiết kế controller.....	82
Hình 4.12: Tổng hợp thiết kế Pipelined Functional Unit	82
Hình 4.13: Kết quả mô phỏng theo thiết kế Pipelined Functional Unit	83
Hình 5.1: Tổng quát thiết kế Datapath Pipelining.....	84
Hình 5.2: Biểu đồ ASM cho bài toán tính xấp xỉ căn bậc hai theo thiết kế Datapath Pipelining	85
Hình 5.3: Sơ đồ thiết kế khối đường dữ liệu thực thi Datapath Pipelining	87
Hình 5.4: Sơ đồ thiết kế khối AU1 [ABS/MIN/MAX]	88
Hình 5.5: Mạch khối AU 1.....	89
Hình 5.6: Sơ đồ thiết kế khối tính AU2 [ADD/SUB/MAX]	90
Hình 5.7: Mạch khối AU 2.....	91
Hình 5.8: Thanh ghi 8 bit	92
Hình 5.9: Khối dịch phải 1 bit.....	92
Hình 5.10: Khối dịch phải 3 bit.....	92
Hình 5.11: Tổng hợp datapath theo thiết kế Datapath Pipelining	93
Hình 5.12: Sơ đồ chuyển trạng thái theo thiết kế Datapath Pipelining	94
Hình 5.13: Mạch khối trạng thái kế tiếp.....	98
Hình 5.14: Mạch khối nhớ.....	98
Hình 5.15: Mạch khối ngõ ra	101
Hình 5.16: Tổng hợp thiết kế controller	102
Hình 5.17: Tổng hợp thiết kế Datapath Pipelining.....	102
Hình 5.18: Kết quả mô phỏng theo thiết kế Datapath Pipelining	103
Hình 6.1: Tổng quát thiết kế Datapath pipeline – AU pipeline.....	104
Hình 6.2: Sơ đồ thiết kế khối đường dữ liệu thực thi Datapath pipeline - AU pipeline	106
Hình 6.3: Sơ đồ thiết kế khối tính AU1 [ABS/MIN/MAX] 2 tầng.....	107
Hình 6.4: Mạch khối tính AU1 [ABS/MIN/MAX] 2 tầng.....	108
Hình 6.5: Sơ đồ khối tính AU 2 [ADD/SUB/MAX] 2 tầng	109
Hình 6.6: Mạch khối AU 2 [ADD/SUB/MAX] 2 tầng.....	110
Hình 6.7: Register 2 bit.....	110

Hình 6.8: Register 8 bit.....	111
Hình 6.9: Khối dịch phải 1 bit.....	111
Hình 6.10: Khối dịch phải 3 bit.....	111
Hình 6.11 Tổng hợp datapath theo thiết kế Datapath pipeline - AU pipeline	112
Hình 6.12: Sơ đồ trạng thái theo thiết kế Datapath pipeline - AU pipeline ...	113
Hình 6.13: Mạch khối trạng thái kế tiếp.....	116
Hình 6.14: Mạch khối nhớ.....	117
Hình 6.15: Mạch khối ngõ ra	121
Hình 6.16: Mạch khối controller	121
Hình 6.17: Tổng hợp thiết kế Datapath pipeline - AU pipeline	122
Hình 6.18: Kết quả mô phỏng thiết kế Datapath pipeline - AU pipeline.....	123

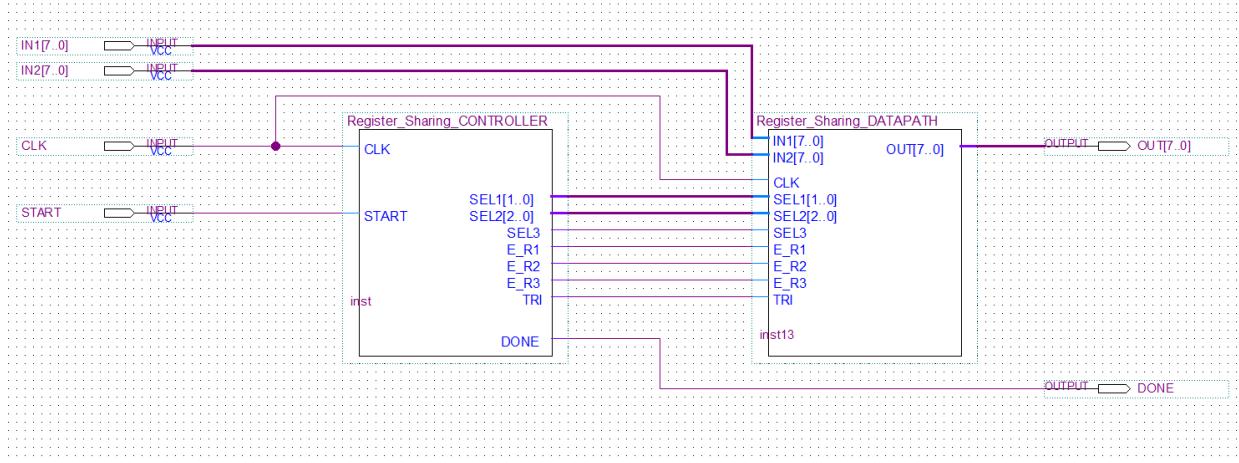
DANH MỤC BẢNG

Bảng 1.1: Bảng phân tích thời gian sống của các biến	18
Bảng 1.2: Bảng sự thật MUX 2to1 (1 bit)	20
Bảng 1.3: Bảng sự thật khối Full_Adder	23
Bảng 1.4: Bảng gán trạng thái theo bìa K-Map	32
Bảng 1.5: Bảng mã hóa trạng thái	32
Bảng 1.6: Bảng chuyển trạng thái	33
Bảng 1.7: Bảng sự thật ngõ ra tạo tín hiệu điều khiển	34
Bảng 2.1: Bảng mã hóa chức năng	41
Bảng 2.2: Bảng sự thật các chân kết nối	41
Bảng 2.3: Bảng mã hóa chức năng khối tính ADD/MIN/SUB/ABS	43
Bảng 2.4: Bảng sự thật các chân kết nối	44
Bảng 2.5: Bảng gán trạng thái theo bìa K-Map	50
Bảng 2.6: Bảng mã hóa trạng thái	50
Bảng 2.7: Bảng chuyển trạng thái	50
Bảng 2.8: Bảng sự thật tín hiệu ngõ ra	52
Bảng 3.1: Bảng gán trạng thái theo bìa K-Map	63
Bảng 3.2: Bảng mã hóa trạng thái	63
Bảng 3.3: Bảng chuyển trạng thái	63
Bảng 3.4: Bảng sự thật các ngõ ra	65
Bảng 4.1: Bảng giản đồ thời gian với các khối chức năng được pipeline	71
Bảng 4.2: Bảng sự thật các chức năng khối AU	74
Bảng 4.3: Bảng gán trạng thái theo bìa K-Map	77
Bảng 4.4: Bảng mã hóa trạng thái	77
Bảng 4.5: Bảng chuyển trạng thái	78
Bảng 4.6: Bảng sự thật các ngõ ra	80
Bảng 5.1: Giản đồ thời gian cho việc thực thi pipeline đường dữ liệu	86
Bảng 5.2: Bảng sự thật các tín hiệu điều khiển khối AU 1	88
Bảng 5.3: Bảng sự thật tín hiệu điều khiển khối AU 1	90
Bảng 5.4: Bảng gán trạng thái dựa trên bìa K-Map	96
Bảng 5.5: Bảng mã hóa trạng thái	96
Bảng 5.6: Bảng chuyển trạng thái	97
Bảng 5.7: Bảng sự thật các tín hiệu ngõ ra	99
Bảng 6.1: Bảng giản đồ thời gian thiết kế Datapath pipeline – AU pipeline	105
Bảng 6.2: Bảng sự thật các tín hiệu điều khiển chức năng AU1 2 tầng	108
Bảng 6.3: Bảng sự thật các tín hiệu điều khiển chức năng AU 2 2 tầng	109
Bảng 6.4: Bảng gán trạng thái theo bìa K-Map	114

Bảng 6.5: Bảng mã hóa trạng thái.....	114
Bảng 6.6: Bảng chuyển trạng thái	115
Bảng 6.7: Bảng sự thật các ngõ ra tạo tín hiệu điều khiển khối datapath.....	117

BÀI TẬP 1. REGISTER SHARING

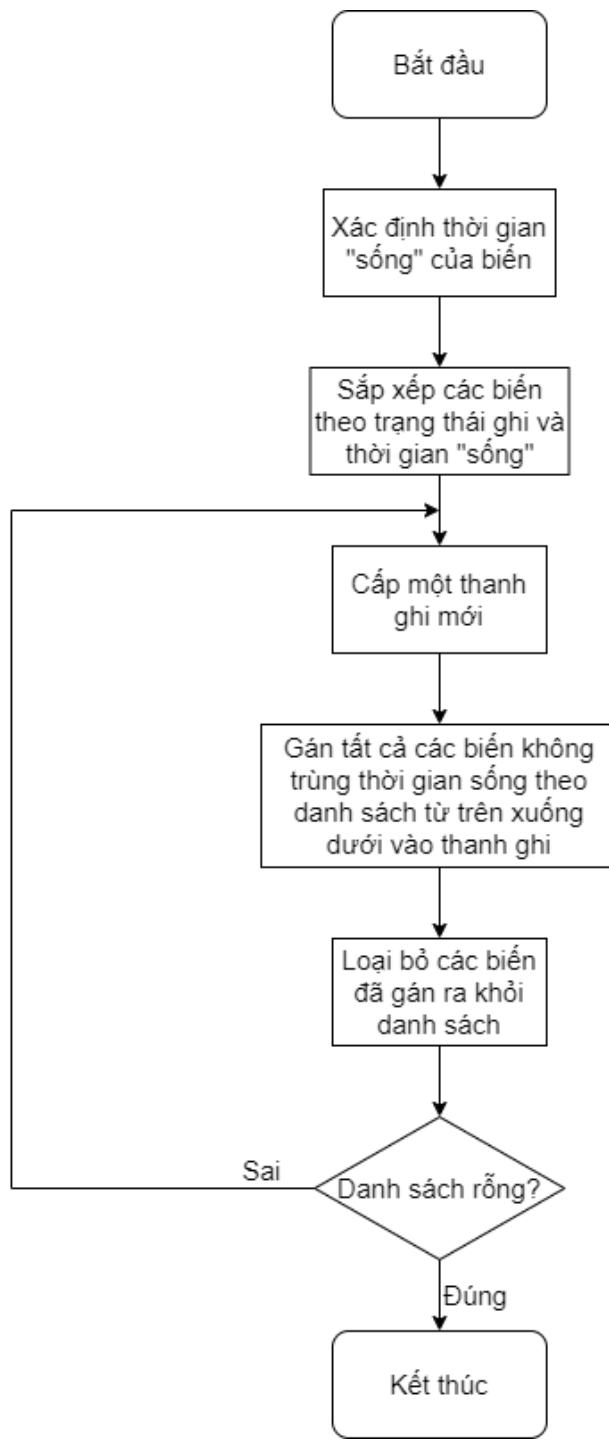
I. TỔNG QUÁT THIẾT KẾ



Hình 1.1: Tổng quát thiết kế Register Sharing

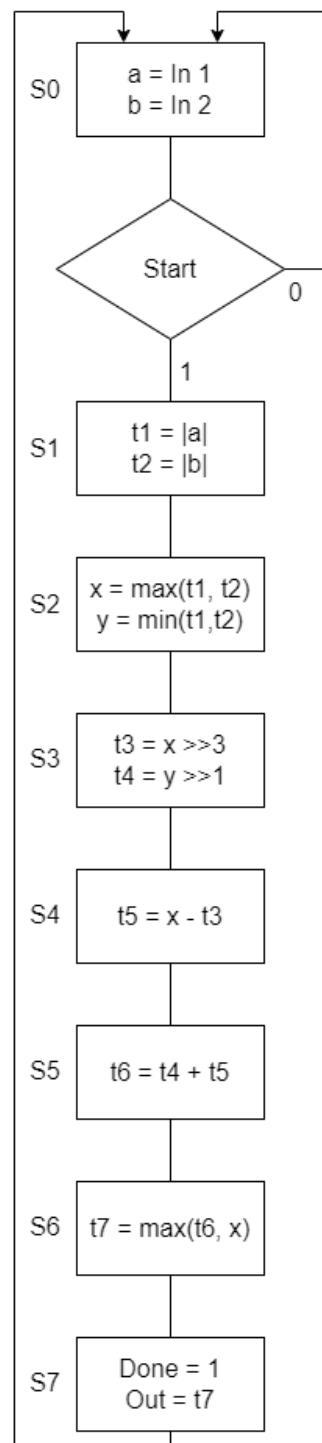
Mục tiêu chung của bài toán Register Sharing là cố gắng sử dụng số lượng thanh ghi ít nhất có thể thông qua việc gom nhóm các biến không cùng thời gian sống sao cho số lượng nhóm là ít nhất mà vẫn đảm bảo tất cả các biến đều thuộc một trong các nhóm đó. Bài toán này sử dụng giải thuật “left-edge” để gom nhóm, đây là giải thuật cố gắng gom nhiều biến nhất có thể vào một thanh ghi.

Lưu đồ giải thuật thuật toán “left-edge” được miêu tả thông qua Hình 1.2.



Hình 1.2: Giải thuật left-edge

Sơ đồ khối minh họa từng bước tính xấp xỉ căn bậc hai được mô tả như sau:



Hình 1.3: Biểu đồ ASM cho bài toán tính xấp xỉ căn bậc hai theo thiết kế Register Sharing

Thông qua sơ đồ khối của bài toán và giải thuật left-edge, ta lập được bảng phân tích thời gian sống của các biến qua từng trạng thái, bảng được miêu tả như sau:

Bảng 1.1: Bảng phân tích thời gian sống của các biến

	S0	S1	S2	S3	S4	S5	S6	S7
a		x						
b		x						
t1			x					
t2			x					
x				x	x	x	x	
y				x				
t4					x	x		
t3					x			
t5						x		
t6							x	
t7								x

Sau khi gom nhóm các biến vào thanh ghi, ta được danh sách các biến như sau:

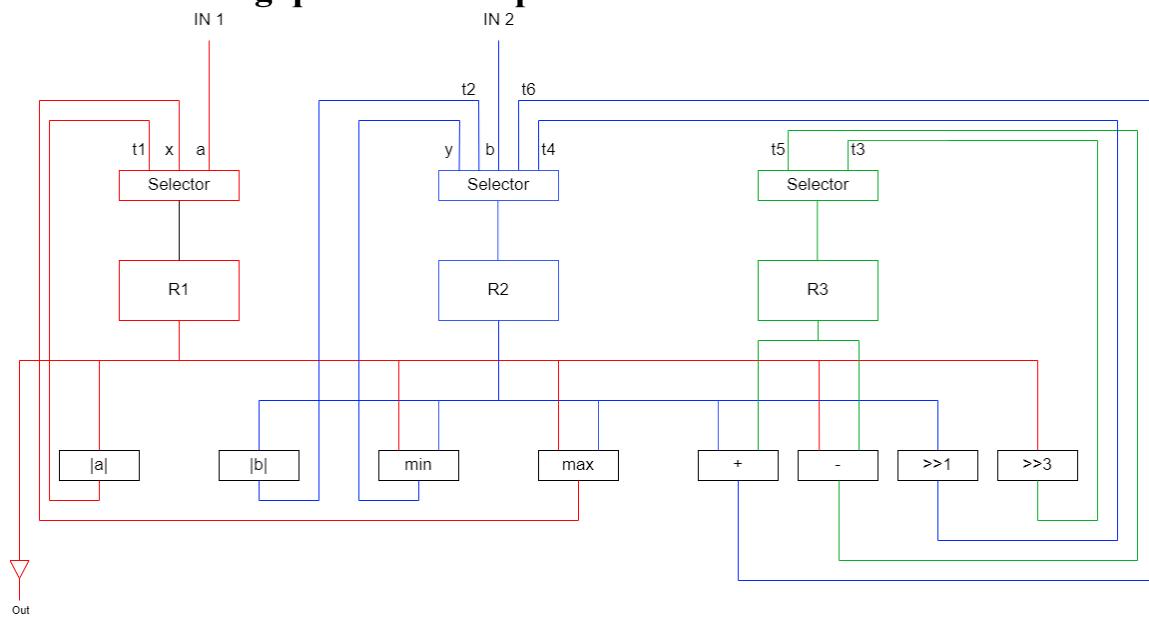
- R1 = [a, t1, x, t7]
- R2 = [b, t2, y, t4, t6]
- R3 = [t3, t5]

Từ đó, chúng ta sẽ sử dụng các thanh ghi này cho mục đích lưu trữ các giá trị đầu vào và các giá trị tính toán, cụ thể như sau:

- R1:
 - a = input 1
 - t1 = |a|
 - x = max(t1, t2)
- R2:
 - b = input 2
 - t2 = |b|
 - y = min(t1, t2)
 - t4 = y >> 1
 - t6 = t4 + t5
- R3:
 - t3 = x >> 3
 - t5 = x - t3

II. THIẾT KẾ KHỐI DATAPATH

1. Mô hình tổng quát khối datapath



Hình 1.4: Sơ đồ thiết kế khối đường dữ liệu của bài toán Register Sharing

2. Phân tích các thành phần trong khối datapath

Datapath trong bài toán này gồm các thành phần sau:

- Register: R1, R2, R3
- Bộ chọn: MUX 3to1, MUX 5to1, MUX 2to1
- Khối tính ABS
- Khối tính MAX
- Khối tính MIN
- Khối cộng/trừ
- Khối dịch phải 3 bit
- Khối dịch phải 1 bit

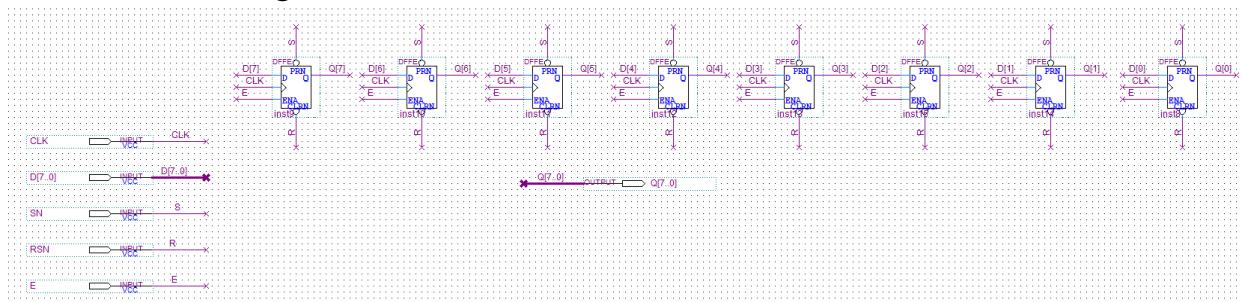
3. Thiết kế Register 8 bit

a. Phân tích thiết kế register 8 bit

Thiết kế này sử dụng flipflop D làm phần tử nhớ, gồm các tín hiệu:

- Ngõ vào D.
- Ngõ ra Q.
- Tín hiệu xung clock.
- Tín hiệu E (cho phép flipflop hoạt động).
- Các ngõ vào bắt đồng bộ PRN và CRLN (không sử dụng).

b. Vẽ mạch register 8 bit



Hình 1.5: Register 8 bit

4. Thiết kế bộ chọn MUX 2to1 (8 bit)

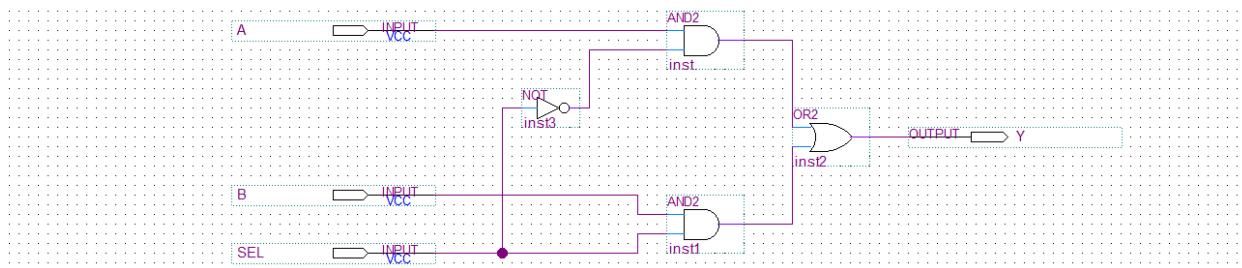
a. Bảng sự thật MUX 2to1 (1 bit)

Bảng 1.2: Bảng sự thật MUX 2to1 (1 bit)

A	B	SEL	Y
0	0	0	0
0	1	0	0
1	0	0	1
1	1	0	1
0	0	1	0
0	1	1	1
1	0	1	0
1	1	1	1

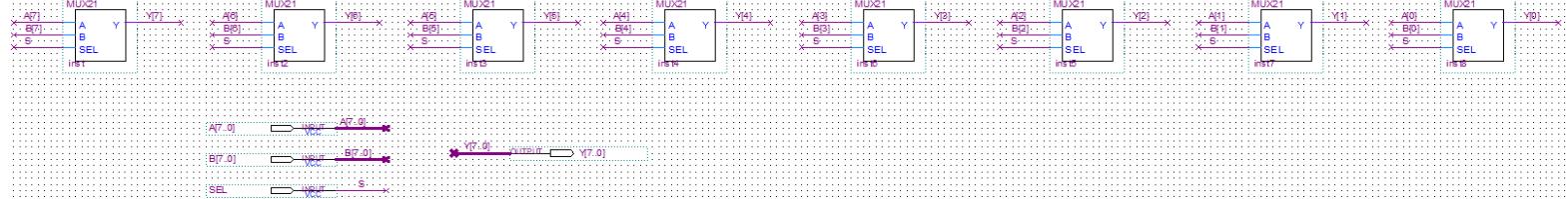
$$\Rightarrow Y = A \cdot SEL' + B \cdot SEL$$

b. Vẽ mạch MUX 2to1 (1 bit)



Hình 1.6: Mạch MUX 2to1 (1 bit)

c. Thiết kế MUX 2to1 8 bit từ MUX 2to1 1 bit



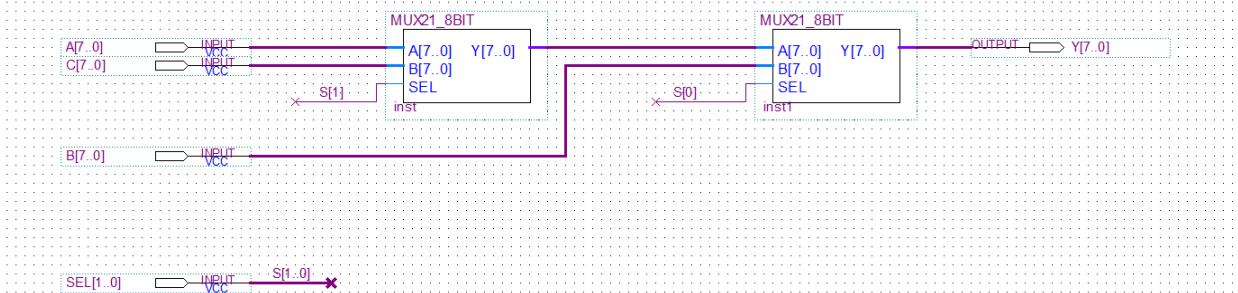
Hình 1.7: Mạch MUX 2to1 (8 bit)

5. Thiết kế các bộ chọn MUX 3to1, MUX 5to1

a. Phân tích MUX 3to1, MUX 5to1

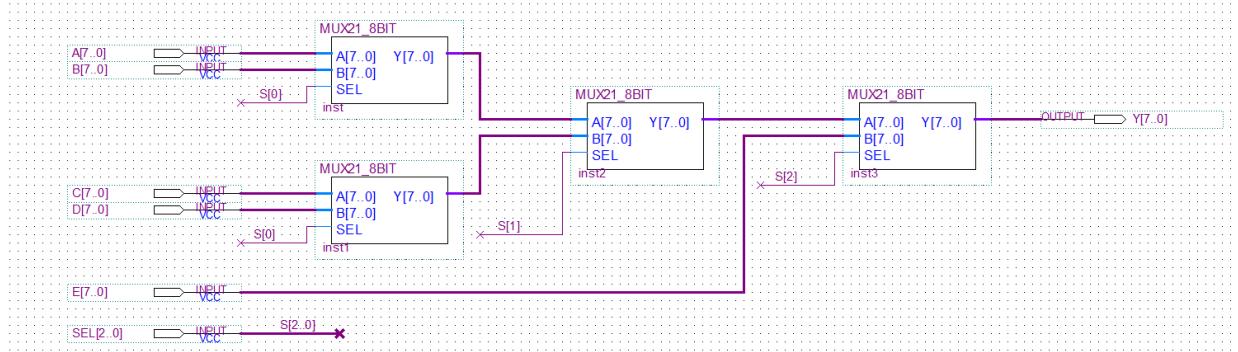
- MUX 3to1: để chọn 1 trong 3 ngõ vào, chúng ta cần 2 bit dùng để lựa chọn, thiết kế này tận dụng 2 khối MUX 2to1 đã thiết kế trước đó.
- MUX 5to1: để chọn 1 trong 5 ngõ vào, chúng ta cần 3 bit dùng để lựa chọn, thiết kế này tận dụng 3 khối MUX 2to1 đã thiết kế trước đó.

b. Vẽ mạch MUX 3to1



Hình 1.8: Mạch MUX 3to1

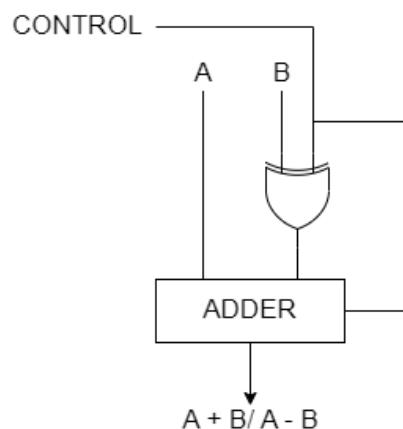
c. Vẽ mạch MUX 5to1



Hình 1.9: Mạch MUX 5to1

6. Thiết kế khối cộng/trừ

a. Phân tích thiết kế khối cộng/trừ



Hình 1.10: Sơ đồ thiết kế khối cộng/trừ

Khối cộng/trừ 8 bit này được thiết kế thông qua 8 khối Full_Adder (khối cộng hai bit đơn giản có giá trị nhớ đầu vào) kết hợp với công XOR và một tín hiệu CTL (CONTROL) cho việc điều khiển khối thực hiện cộng hay trừ.

Để thiết kế khối cộng hai số A và B, ta sẽ cho đầu vào CTL bằng 0, thông qua công XOR, giá trị B sẽ là chính nó và được đưa vào khối cộng để thực hiện. Ngược lại, để thiết kế khối trừ hai số A và B, ta sẽ cho đầu vào CTL (CONTROL) bằng 1, thông qua công XOR, giá trị B sẽ là giá trị đảo của nó, đồng thời ta gán tín hiệu CTL cho Cin đầu tiên cho khối Full_Adder có trọng số nhỏ nhất để thực hiện phép trừ theo nguyên lý: $A - B = A + (-B)$. Với $-B$ sẽ là số âm được biểu diễn bù hai bằng cách lấy phần bù của chính nó và cộng với 1.

b. Thiết kế khối Full_Adder

- Bảng sự thật khối Full_Adder

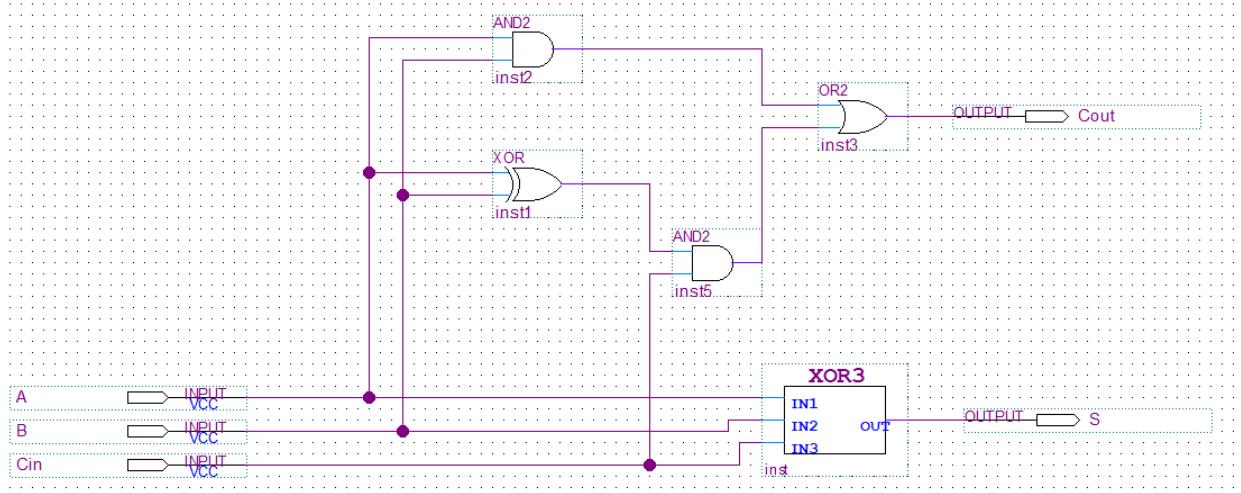
Bảng 1.3: Bảng sự thật khối Full Adder

Cin	A	B	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$\Rightarrow S = \text{Cin} \oplus A \oplus B$$

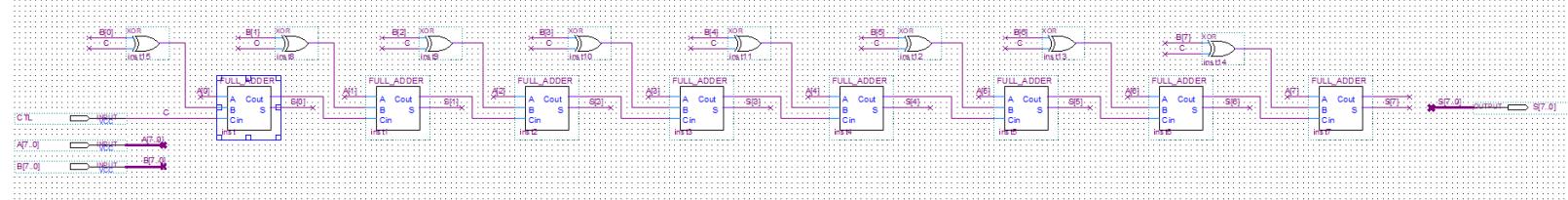
$$\Rightarrow \text{Cout} = (A \oplus B) \cdot \text{Cin} + A \cdot B$$

- Vẽ mạch khối Full_Adder



Hình 1.11: Mạch khối Full_Adder 1 bit

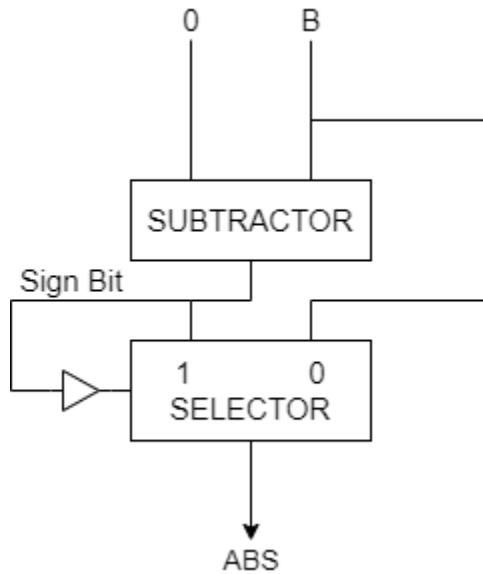
c. Vẽ mạch khối công/trừ từ các khối Full_Adder



Hình 1.12: Mạch khối công/trừ 8 bit

7. Thiết kế khối tính ABS

a. Phân tích thiết kế khối tính ABS

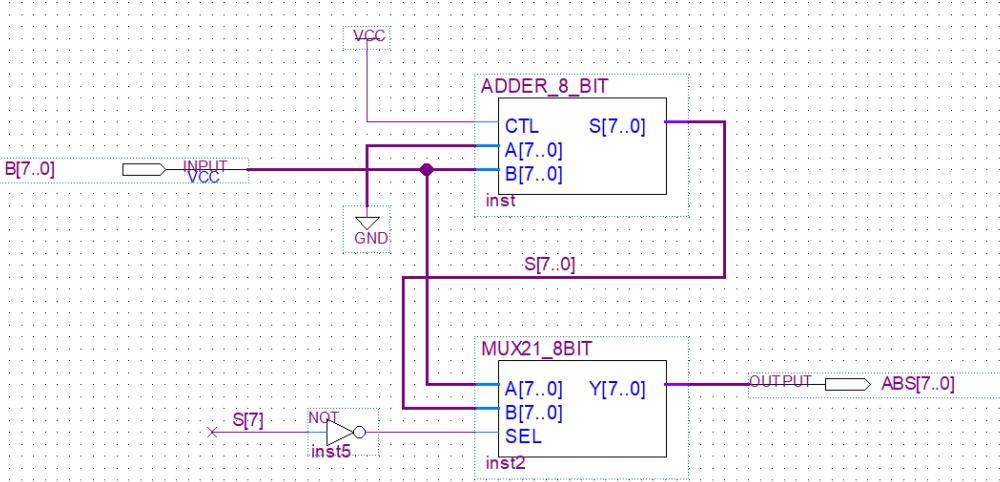


Hình 1.13: Sơ đồ thiết kế khối tính ABS

Khối ABS này được thiết kế theo nguyên lý như sau: lấy số 0 trừ cho input đầu vào B, nếu như đây là số dương, kết quả sẽ ra âm, đồng thời (Sign Bit)' sẽ ra 0. Khối chọn sẽ lấy giá trị của B vì đây đã là số dương.

Ngược lại, nếu B là số âm, thì $0 - B$ sẽ ra số dương, khi đó (Sign Bit)' sẽ ra 1, đồng nghĩa với việc vẫn lấy kết quả của phép tính $0 - B$. Kết quả của cả 2 trường hợp khi xuất ra từ khối chọn lúc nào cũng là một số dương.

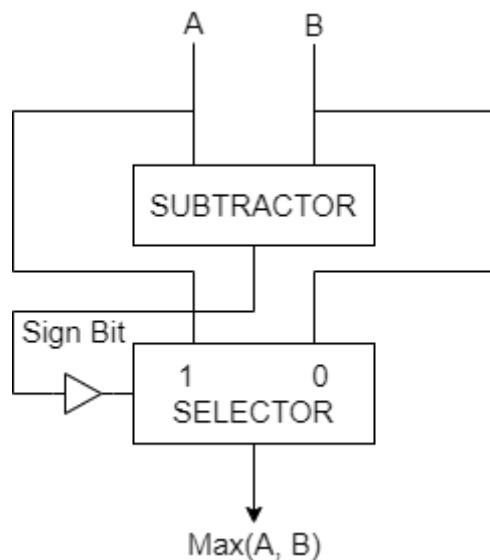
b. Vẽ mạch khối tính ABS



Hình 1.14: Mạch khối tính ABS

8. Thiết kế khối tính MAX

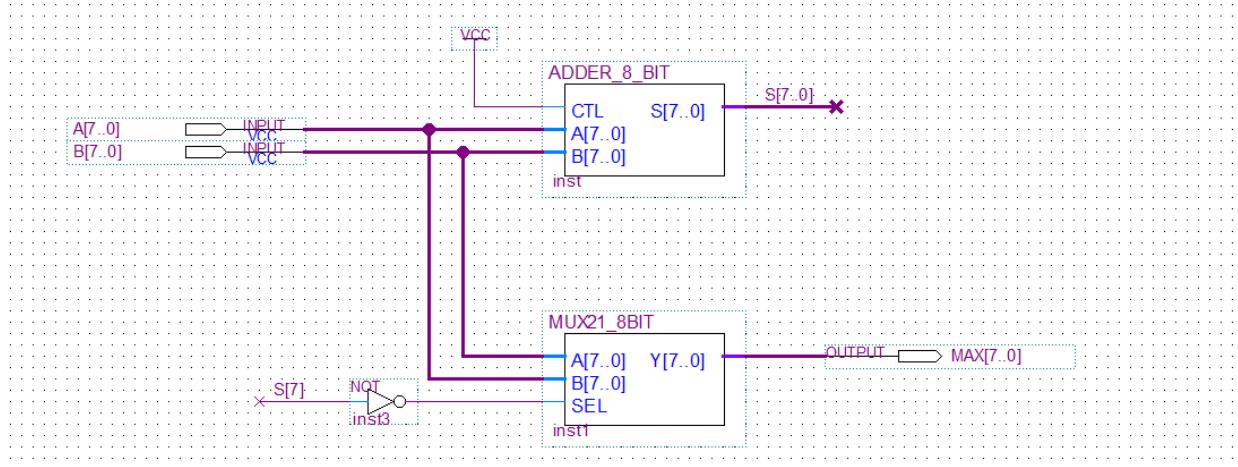
a. Phân tích thiết kế khối tính MAX



Hình 1.15: Sơ đồ thiết kế khối tính MAX

Khối tính MAX được thiết kế theo nguyên lý: Lấy số A trừ số B, nếu kết quả ra dương, nghĩa là số A lớn hơn. Khi đó (Sign Bit) sẽ ra 1, kết quả lấy số A làm số max. Ngược lại nếu số A trừ số B kết quả ra âm, nghĩa là số B lớn hơn. Khi đó (Sign Bit) sẽ ra 0, kết quả lấy số B làm số max.

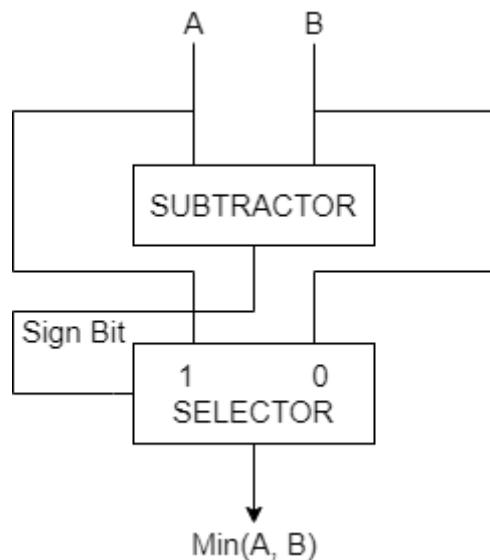
b. Vẽ mạch khối tính MAX



Hình 1.16: Mạch khối tính MAX

9. Thiết kế khối tính MIN

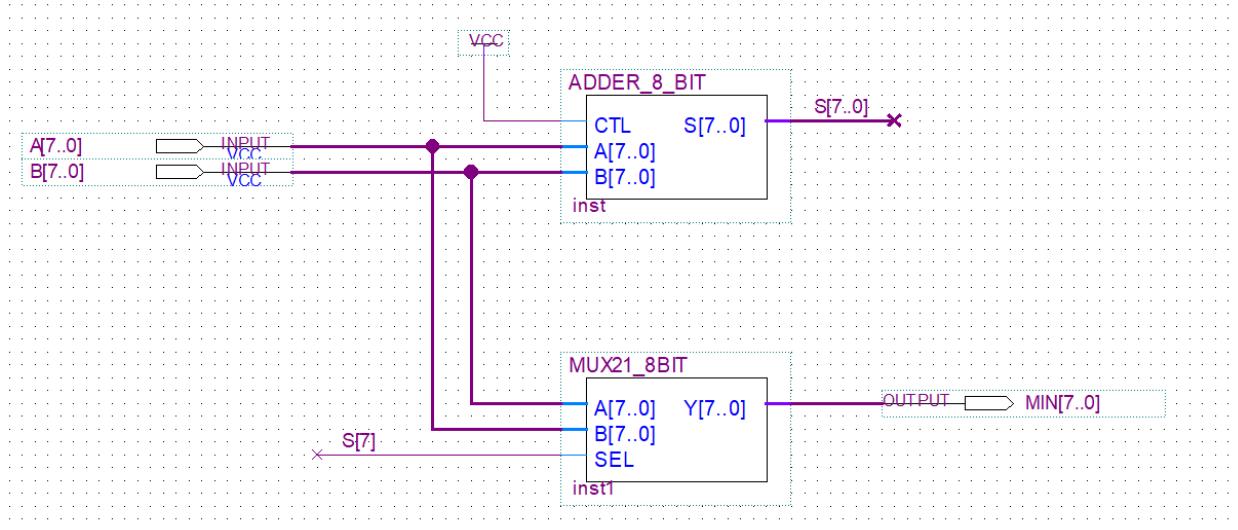
a. Phân tích thiết kế khối tính MIN



Hình 1.17: Sơ đồ thiết kế khối tính MIN

Tương tự cách hoạt động của khối tính MAX, khối tính MIN chỉ khác một chỗ là không có lấy bù Sign Bit để lấy giá trị ngược lại.

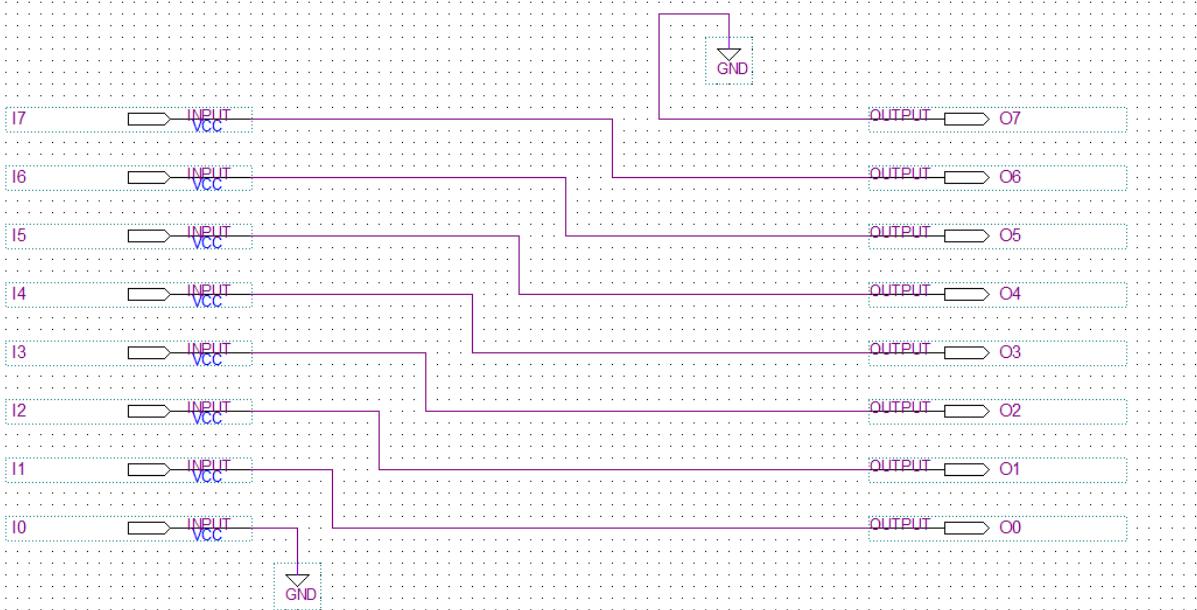
b. Vẽ mạch khối tính MIN



Hình 1.18: Mạch khối tính MIN

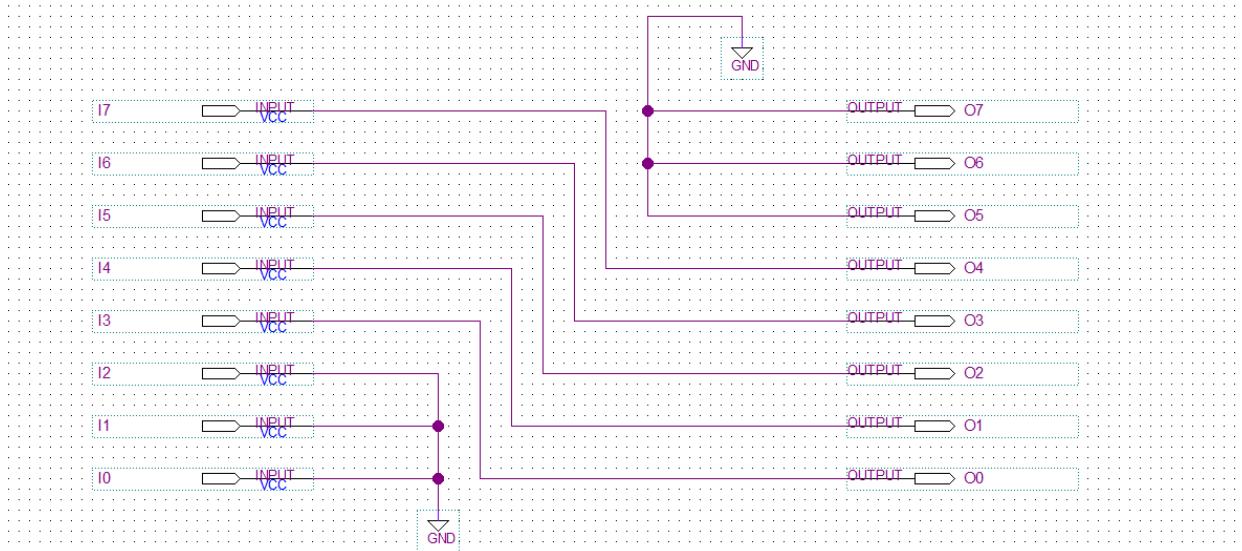
10.Thiết kế khối dịch phải (1 bit, 3 bit)

a. Vẽ mạch khối dịch phải 1 bit



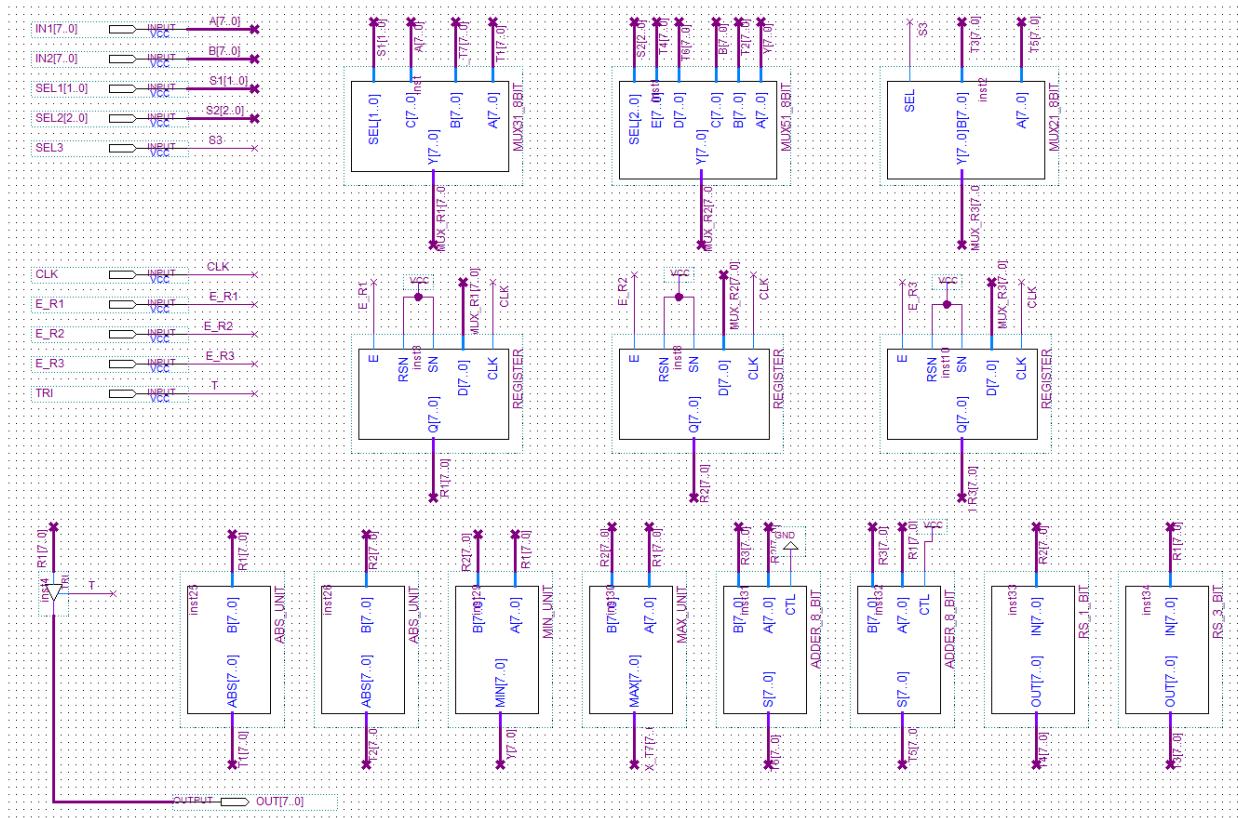
Hình 1.19: Mạch khối dịch phải 1 bit ($>>1$)

b. Vẽ mạch khối dịch phải 3 bit



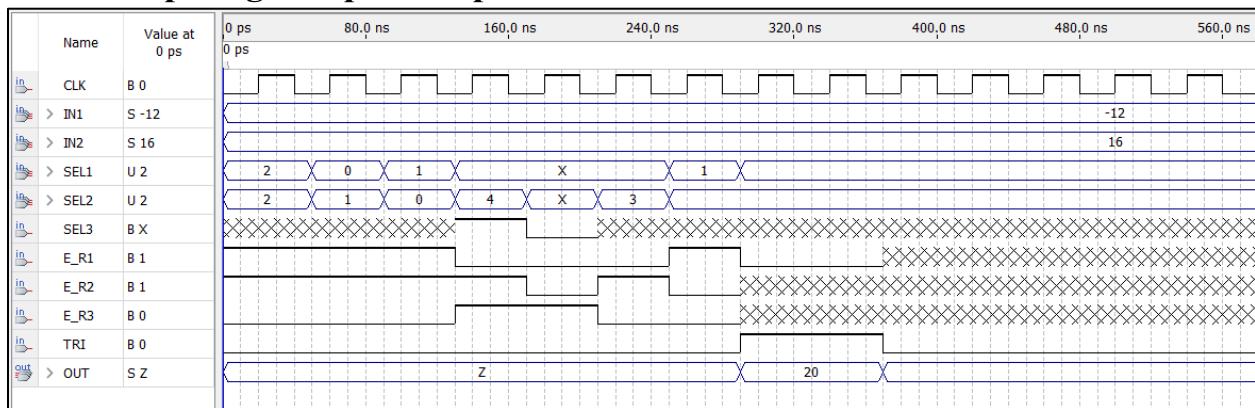
Hình 1.20: Mạch khối dịch phải 3 bit ($>>3$)

11.Tổng hợp thiết kế datapath



Hình 1.21: Tổng hợp datapath theo thiết kế Register Sharing

12.Mô phỏng datapath và phân tích



Hình 1.22: Kết quả chạy mô phỏng datapath

Phân tích:

Ở cạnh lén xung clock lần đầu tiên:

- SEL1 = 2: bộ chọn lấy giá trị a = input và lưu vào R1.
- SEL2 = 2: bộ chọn lấy giá trị b = input và lưu vào R2.
- SEL3 = X: bộ chọn tự do, bởi E_R3 ở lần này không hoạt động.
- E_R1 = 1: thanh ghi R1 hoạt động.
- E_R2 = 1: thanh ghi R2 hoạt động.
- E_R3 = 0: thanh ghi R3 không hoạt động.
- TRI = 0: không cho giá trị xuất ra ngoài ra.

Ở cạnh lén xung clock lần thứ 2:

- SEL1 = 0: bộ chọn lấy giá trị t1 = |a| vừa được tính toán lưu vào R1.
- SEL2 = 1: bộ chọn lấy giá trị t2 = |b| vừa được tính toán lưu vào R2.
- SEL3 = X: bộ chọn tự do, bởi E_R3 ở lần này không hoạt động.
- E_R1 = 1: thanh ghi R1 hoạt động.
- E_R2 = 1: thanh ghi R2 hoạt động.
- E_R3 = 0: thanh ghi R3 không hoạt động.
- TRI = 0: không cho giá trị xuất ra ngoài ra.

Ở cạnh lén xung clock lần thứ 3:

- SEL1 = 1: bộ chọn lấy giá trị x = max(t1, t2) lưu vào R1.
- SEL2 = 0: bộ chọn lấy giá trị y = min(t1, t2) lưu vào R2.
- SEL3 = X: bộ chọn tự do, bởi E_R3 ở lần này không hoạt động.
- E_R1 = 1: thanh ghi R1 hoạt động.
- E_R2 = 1: thanh ghi R2 hoạt động.
- E_R3 = 0: thanh ghi R3 không hoạt động.
- TRI = 0: không cho giá trị xuất ra ngoài ra.

Ở cạnh lén xung clock lần thứ 4:

- SEL1 = X: bộ chọn tự do, bởi E_R1 ở lần này sẽ ngừng hoạt động.
- SEL2 = 4: bộ chọn lấy giá trị t4 = y >> 1 lưu vào R2.
- SEL3 = 1: bộ chọn lấy giá trị t3 = x >> 3 lưu vào R3.
- E_R1 = 0: thanh ghi R1 ngừng hoạt động.
- E_R2 = 1: thanh ghi R2 hoạt động.
- E_R3 = 1: thanh ghi R3 hoạt động.
- TRI = 0: không cho giá trị xuất ra ngoài ra.

Ở cạnh lén xung clock lần thứ 5:

- SEL1 = X: bộ chọn tự do, bởi E_R1 ở lần này sẽ ngừng hoạt động.
- SEL2 = X: bộ chọn tự do, bởi E_R2 ở lần này sẽ ngừng hoạt động.
- SEL3 = 0: bộ chọn lấy giá trị t5 = x - t3 kết quả lưu vào R3.

- $E_R1 = 0$: thanh ghi R1 ngưng hoạt động.
- $E_R2 = 0$: thanh ghi R2 ngưng hoạt động.
- $E_R3 = 1$: thanh ghi R3 hoạt động.
- $TRI = 0$: không cho giá trị xuất ra ngõ ra.

Ở cạnh lén xung clock lần thứ 6:

- $SEL1 = X$: bộ chọn tự do, bởi E_R1 ở lần này sẽ ngừng hoạt động.
- $SEL2 = 3$: bộ chọn lấy $t6 = t4 + t5$ kết quả lưu vào R2.
- $SEL3 = X$: bộ chọn tự do, bởi E_R3 ở lần này sẽ ngừng hoạt động.
- $E_R1 = 0$: thanh ghi R1 ngưng hoạt động.
- $E_R2 = 1$: thanh ghi R2 hoạt động.
- $E_R3 = 0$: thanh ghi R3 ngưng hoạt động.
- $TRI = 0$: không cho giá trị xuất ra ngõ ra.

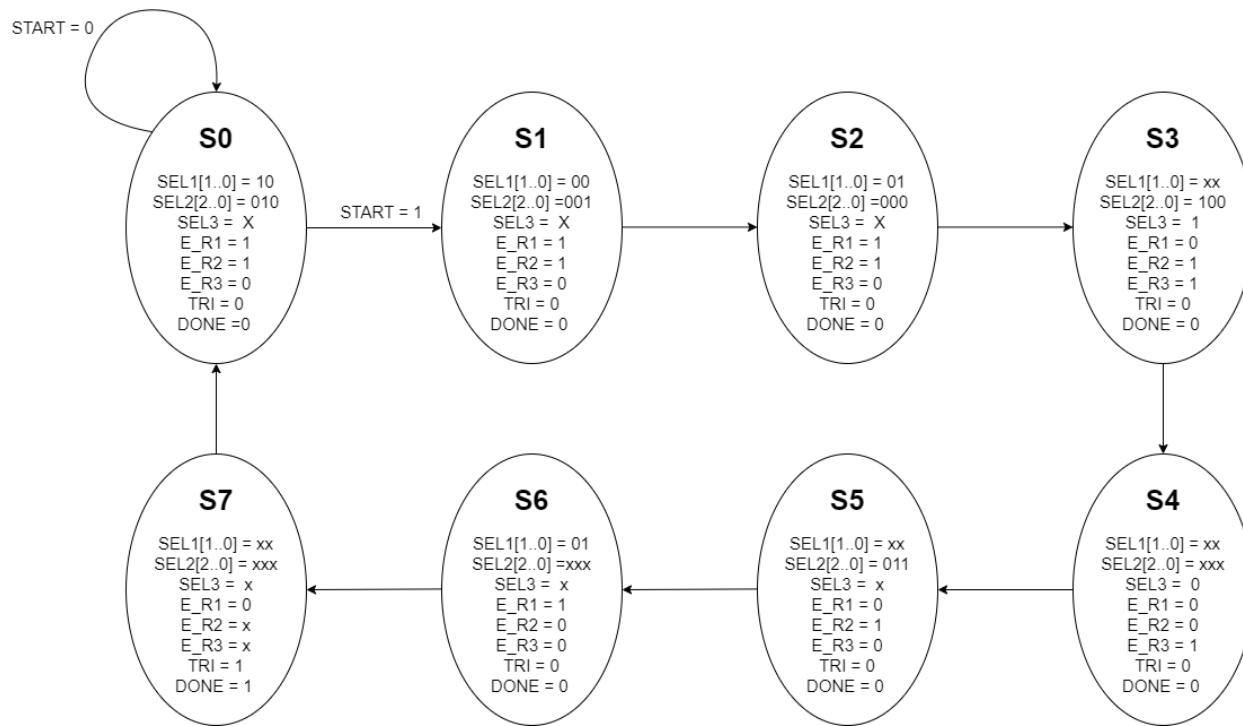
Ở cạnh lén xung clock lần thứ 7:

- $SEL1 = 1$: bộ chọn lấy giá trị $t7 = \max(t6, x)$ lưu vào R1.
- $SEL2 = X$: bộ chọn tự do, bởi E_R2 ở lần này sẽ ngừng hoạt động.
- $SEL3 = X$: bộ chọn tự do, bởi E_R3 ở lần này sẽ ngừng hoạt động.
- $E_R1 = 1$: thanh ghi R1 hoạt động.
- $E_R2 = 0$: thanh ghi R2 ngưng hoạt động.
- $E_R3 = 0$: thanh ghi R3 ngưng hoạt động.
- $TRI = 0$: không cho giá trị xuất ra ngõ ra.

Lúc này, giá trị cần tính đã có thể xuất hiện ở ngõ ra, khi tín hiệu TRI được bật lên mức cao, ngõ ra sẽ lập tức xuất hiện giá trị của kết quả cần tính.

III. THIẾT KẾ KHỐI CONTROLLER

1. Sơ đồ trạng thái



Hình 1.23: Sơ đồ trạng thái theo thiết kế Register Sharing

Các tín hiệu được mô tả như sau:

- SEL1[1..0]: bộ chọn giá trị [x, t1, a] để lưu vào thanh ghi R1.
- SEL2[2..0]: bộ chọn giá trị [y, t2, b, t6, t4] để lưu vào thanh ghi R2.
- SEL3: bộ chọn giá trị [t5, t3] để lưu vào thanh ghi R3.
- E_R1: cho phép ghi hoặc không vào thanh ghi R1.
- E_R2: cho phép ghi hoặc không vào thanh ghi R2.
- E_R3: cho phép ghi hoặc không vào thanh ghi R3.
- TRI: tín hiệu cho phép xuất giá trị ngõ ra.
- DONE: tín hiệu báo hoàn thành.

2. Mã hóa trạng thái

Bảng 1.4: Bảng gán trạng thái theo bìa K-Map

F		Q1Q0			
		00	01	11	10
Q2	0	S0	S1	S2	S3
	1	S7	S6	S5	S4

Bảng 1.5: Bảng mã hóa trạng thái

Trạng thái	Mã hóa
S0	000
S1	001
S2	011
S3	010
S4	110
S5	111
S6	101
S7	100

3. Chọn loại flipflop và lập bảng chuyển trạng thái

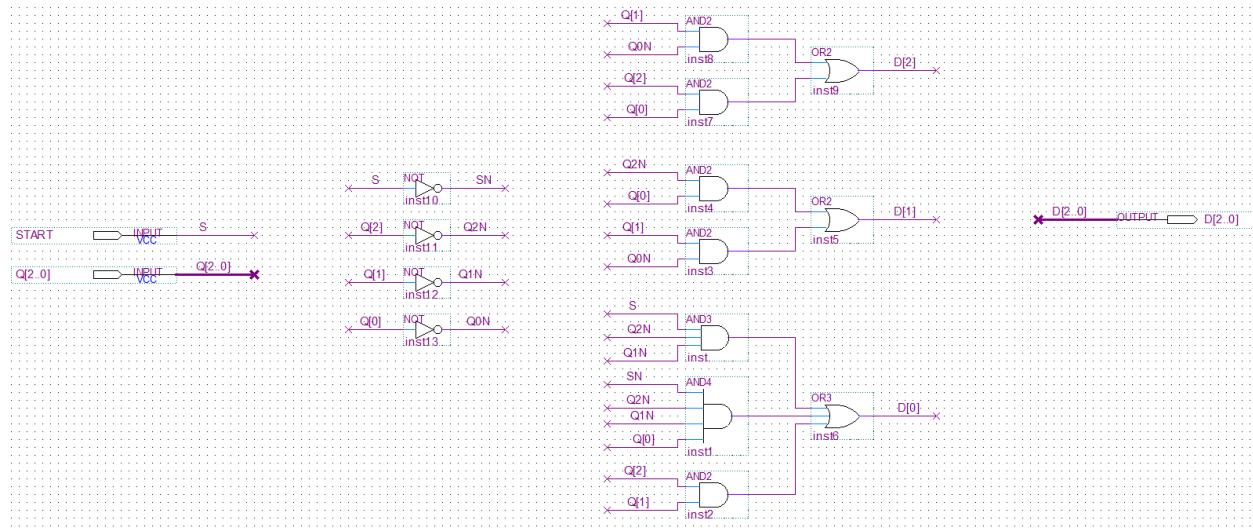
Bảng 1.6: Bảng chuyển trạng thái

TTHT	TTKT		DONE
	START = 0	START = 1	
S0	S0	S1	0
S1	S2	S2	0
S2	S3	S3	0
S3	S4	S4	0
S4	S5	S5	0
S5	S6	S6	0
S6	S7	S7	0
S7	S0	S0	1

Từ bảng chuyển trạng thái, ta lập phương trình ngõ vào các DFF như sau:

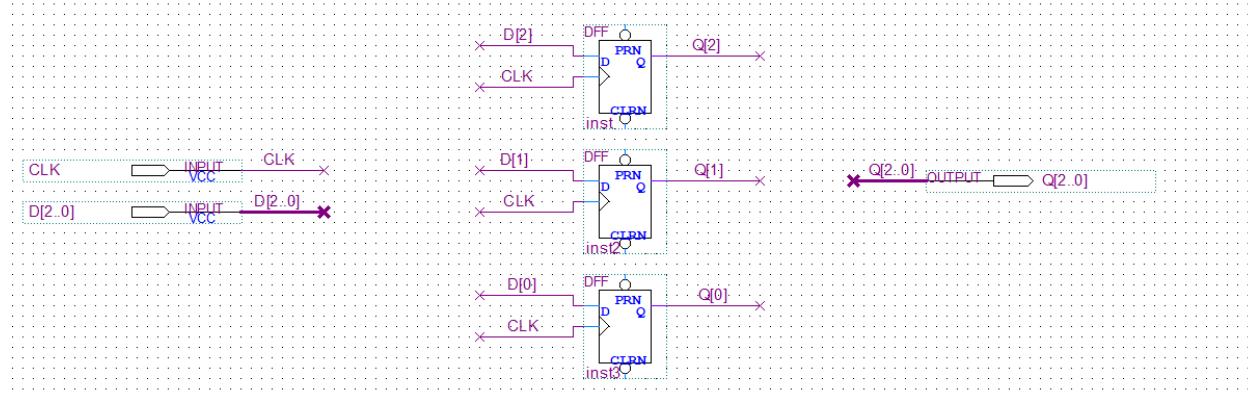
- $D_2 = S_3 + S_4 + S_5 + S_6 = Q_1.Q_0' + Q_2.Q_0$
- $D_1 = S_1 + S_3 + S_4 + S_5 = Q_2'.Q_0 + Q_1.Q_0'$
- $D_0 = S.(S_0 + S_1) + S_4 + S_5 = S.Q_2'.Q_1' + S'.Q_2'.Q_1'.Q_0 + Q_2.Q_1$

4. Thiết kế khối trạng thái kế tiếp



Hình 1.24: Mạch khối trạng thái kế tiếp

5. Thiết kế khối nhớ



Hình 1.25: Mạch khối nhớ

6. Thiết kế khối ngõ ra tạo tín hiệu điều khiển

a. Bảng sự thật ngõ ra tạo tín hiệu điều khiển

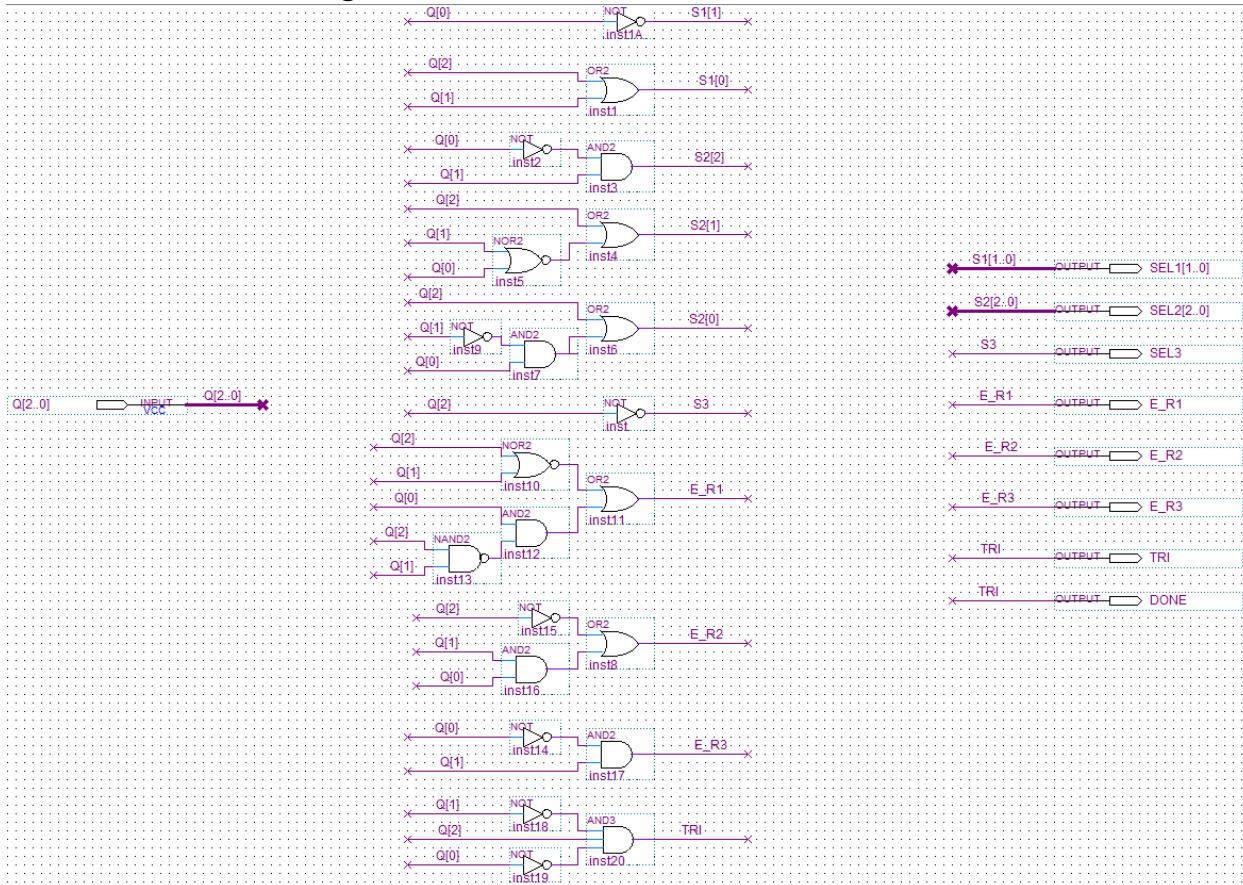
Bảng 1.7: Bảng sự thật ngõ ra tạo tín hiệu điều khiển

TTHT	Mã hóa	SEL1	SEL2	SEL3	ER1	ER2	ER3	TRI
S0	000	10	010	X	1	1	0	0
S1	001	00	001	X	1	1	0	0
S2	011	01	000	X	1	1	0	0
S3	010	XX	100	1	0	1	1	0
S4	110	XX	XXX	0	0	0	1	0
S5	111	XX	011	X	0	1	0	0
S6	101	01	XXX	X	1	0	0	0
S7	100	XX	XXX	X	0	X	X	1

Từ bảng trên, ta lập được các phương trình cho các tín hiệu ngõ ra như sau:

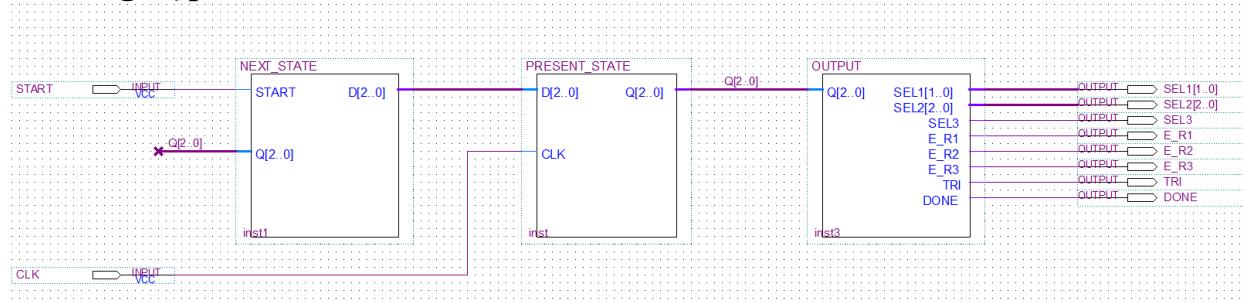
- $SEL1[1] = Q0'$
- $SEL1[0] = Q2 + Q1$
- $SEL2[2] = Q1.Q0'$
- $SEL2[1] = Q2 + (Q1 + Q0)'$
- $SEL2[0] = Q2 + Q1'.Q0$
- $SEL3 = Q2'$
- $ER1 = (Q2 + Q1)' + Q0.(Q2.Q1)'$
- $ER2 = Q2' + Q1.Q0$
- $ER3 = Q1.Q0'$
- $TRI = Q2.Q1'.Q0'$

b. Vẽ mạch khối ngõ ra tạo tín hiệu điều khiển



Hình 1.26: Mạch khối ngõ ra tạo tín hiệu điều khiển

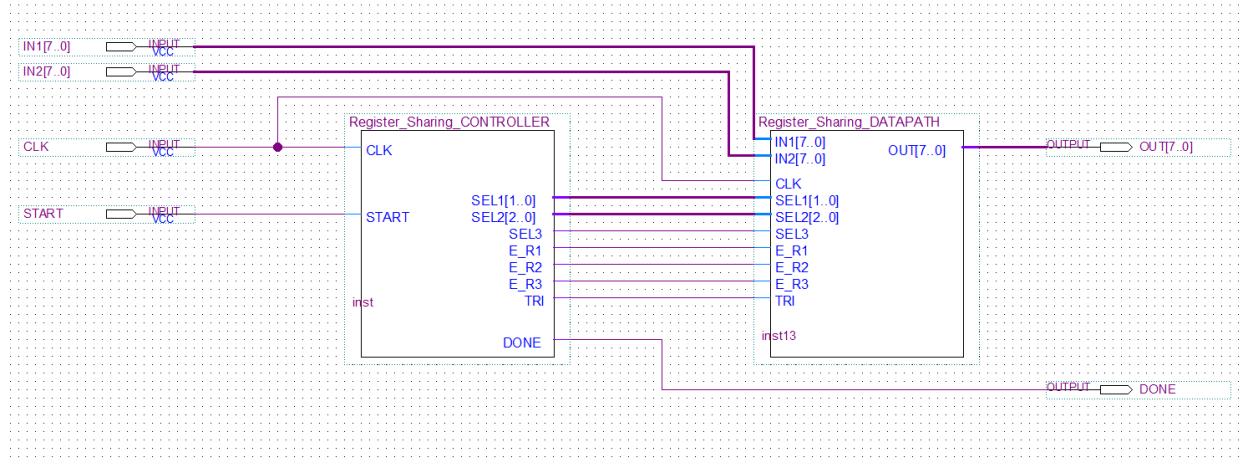
7. Tổng hợp thiết kế controller



Hình 1.27: Tổng hợp thiết kế controller

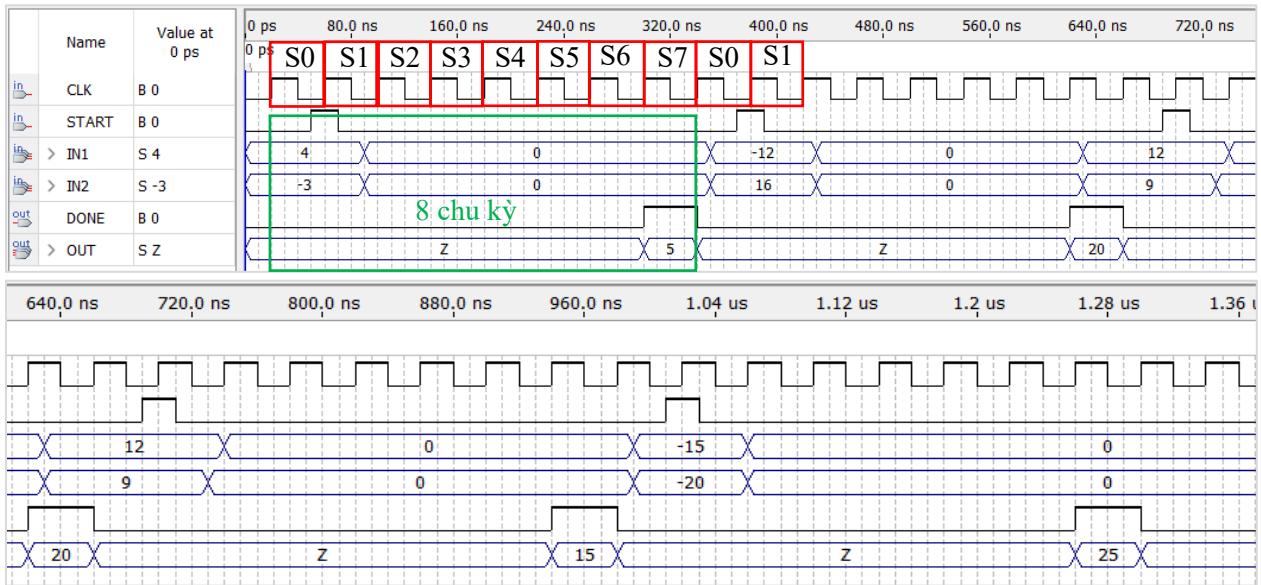
IV. TỔNG HỢP THIẾT KẾ VÀ MÔ PHỎNG

1. Tổng hợp thiết kế



Hình 1.28: Tổng hợp thiết kế Register Sharing

2. Mô phỏng và phân tích



Hình 1.29: Kết quả mô phỏng theo thiết kế Register Sharing

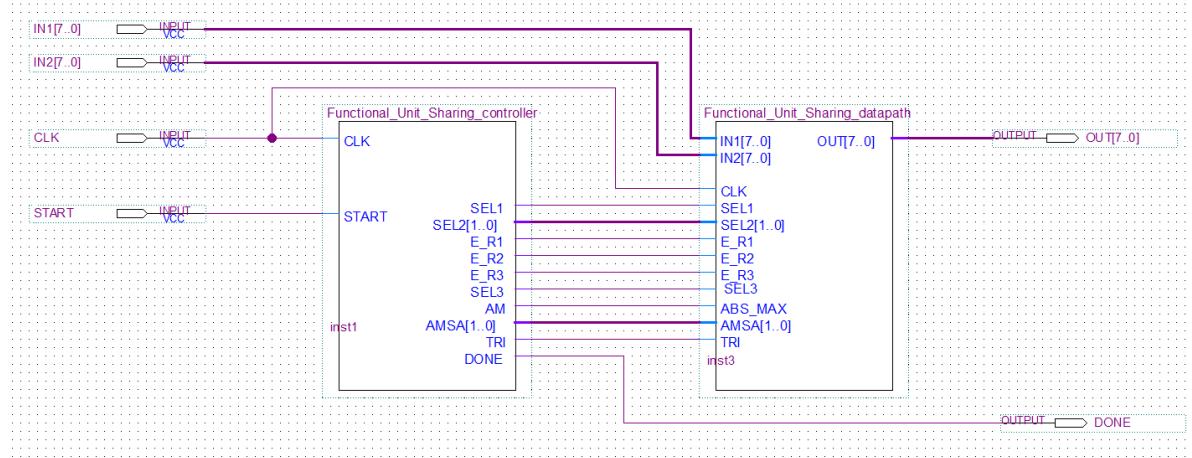
Các trường hợp thử nghiệm là:

- Số A dương, số B âm: $\sqrt{4^2 + (-3)^2} = 5$
- Số A âm, số B dương: $\sqrt{(-12)^2 + 16^2} = 20$
- Số A dương, số B dương: $\sqrt{12^2 + 9^2} = 15$
- Số A âm, số B âm: $\sqrt{(-15)^2 + (-20)^2} = 25$

Sau khi tín hiệu START = 1, trạng thái sẽ chuyển từ S0 sang S1 khi có xung clock kích cạnh lên, đồng thời các tín hiệu điều khiển từ khói controller sẽ được tính toán và chuyển qua khói đường dữ liệu để thực thi các lệnh theo yêu cầu. Đến khi xung clock kích cạnh lên lần thứ 7 (tính từ khi START = 1), lúc này trạng thái hiện tại là S7, tín hiệu báo DONE = 1 và đồng thời giá trị phép tính sẽ được xuất ra ngoài ra OUT. Tổng thời gian thực thi của thiết kế này mất 8 chu kỳ xung clock để hoàn thành.

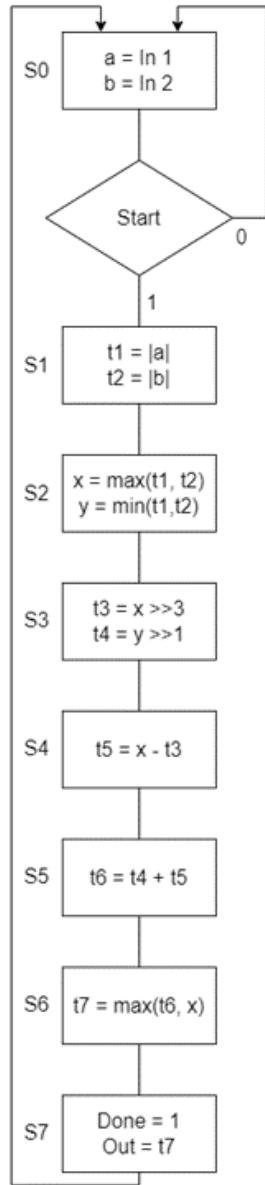
BÀI TẬP 2. FUNCTIONAL - UNIT SHARING

I. TỔNG QUÁT THIẾT KẾ



Hình 2.1: Tổng quát thiết kế Functional – Unit Sharing

Mục tiêu chính của bài toán Functional – Unit Sharing là để tối thiểu hóa số khói chức năng trong khối đường dữ liệu. Trong bất kỳ một trạng thái nào thì khối đường dữ liệu cũng sẽ không thực thi mọi phép toán. Do đó, các phép toán giống nhau sẽ được gom nhóm thành một khối đa chức năng, nó sẽ được sử dụng thường xuyên hơn do đó sẽ tăng tính hữu dụng của mỗi khối.



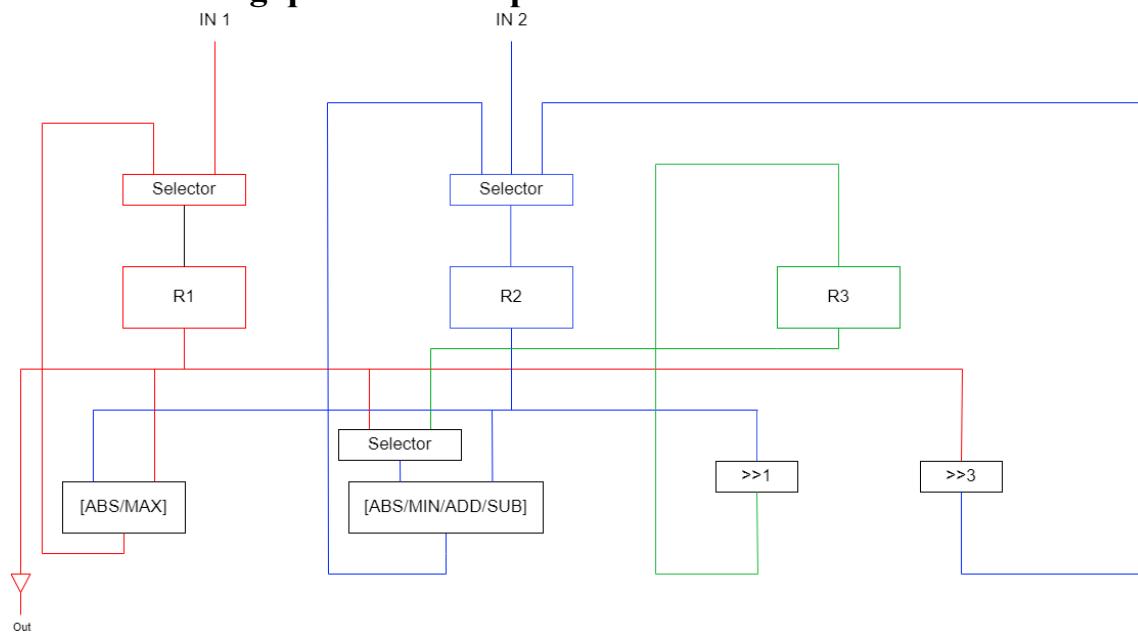
Hình 2.2: Biểu đồ ASM cho bài toán tính xấp xỉ căn bậc hai theo thiết kế Functional - Unit Sharing

Dựa vào biểu đồ phân nhóm cho khối đường dữ liệu, chúng ta tổng hợp được các khối chức năng như sau:

- R1 = [a, t1, x, t7]
- R2 = [b, t2, y, t3, t5, t6]
- R3 = [t4]
- AU1 = [ADD/MIN/SUB/ABS]
- AU2 = [ABS, MAX]
- SH1 = [>>1]
- SH2 = [>>3]

II. THIẾT KẾ KHỐI DATAPATH

1. Mô hình tổng quát khối datapath



Hình 2.3: Sơ đồ thiết kế khối đường dữ liệu của bài toán Functional – Unit Sharing

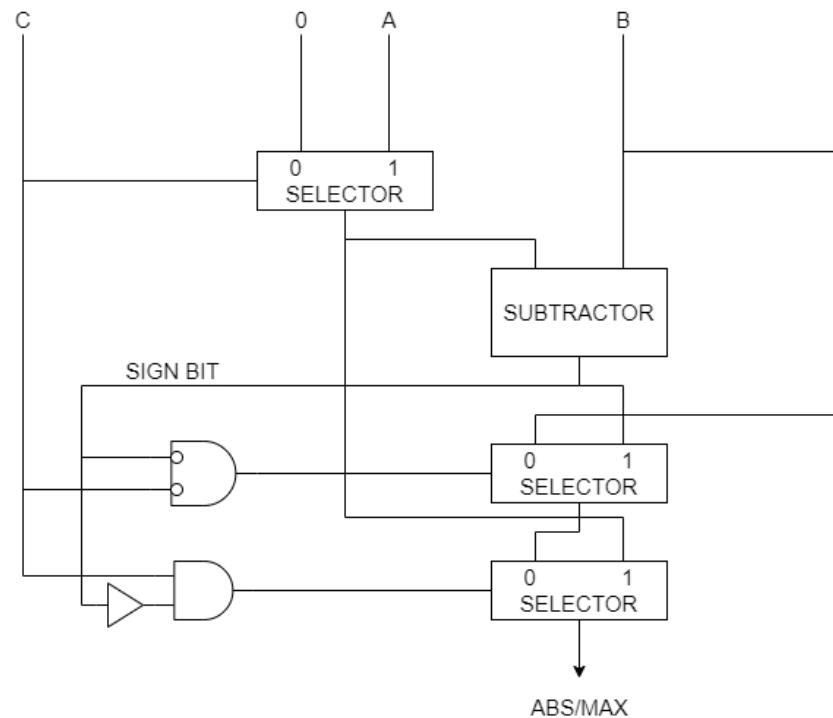
2. Phân tích các thành phần trong khối datapath

Datapath trong bài toán này gồm các thành phần sau:

- Register: R1, R2, R3.
- Bộ chọn: MUX 2to1, MUX 3to1.
- Khối tính ABS/MAX.
- Khối tính ADD/MIN/SUB/ABS.
- Khối dịch phải 1 bit.
- Khối dịch phải 3 bit.

3. Thiết kế khối tính ABS/MAX

a. Phân tích thiết kế khối tính ABS/MAX



Hình 2.4: Sơ đồ thiết kế khối tính ABS/MAX

Khối tính ABS/MAX được thiết kế thông qua sử dụng 3 khối chọn MUX 2to1 và một khối trừ kèm theo các công NOT, AND và bit dấu (SIGN BIT) để thực hiện việc tính toán thông qua tín hiệu điều khiển C. Khi C = 0, khối này sẽ thực hiện tính ABS. Ngược lại, khi C = 1, khối sẽ thực hiện tính MAX(A, B).

b. Bảng mã hóa chức năng

Bảng 2.1: Bảng mã hóa chức năng

C	Chức năng
0	ABS
1	MAX

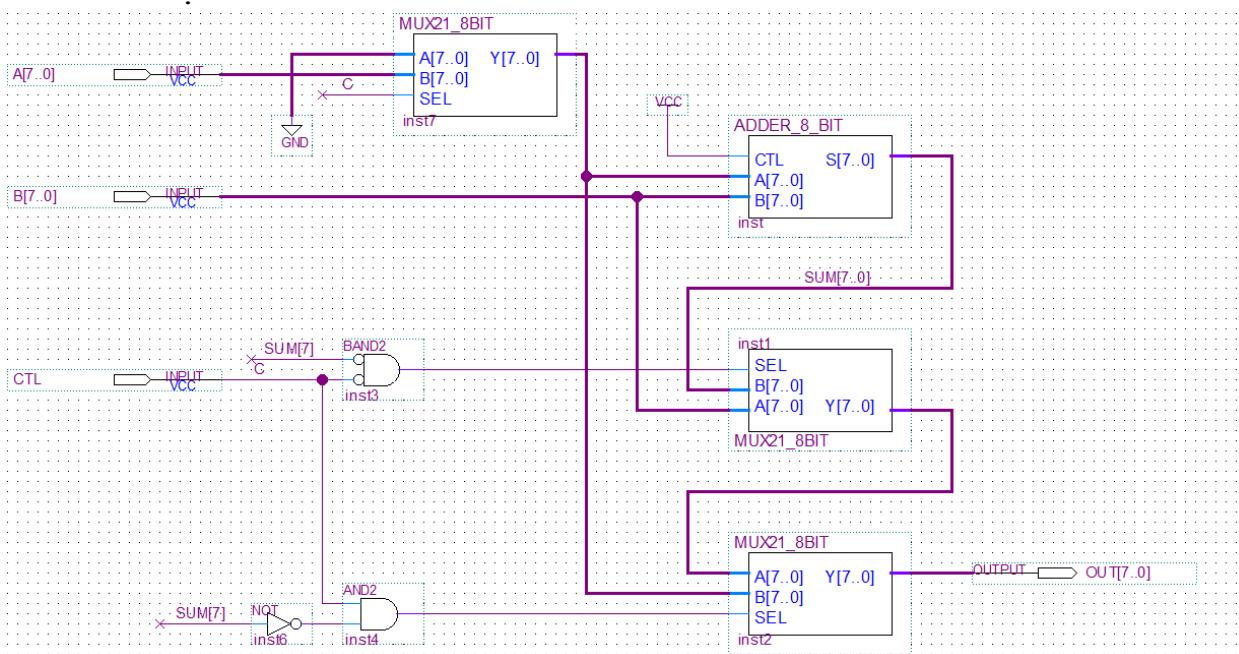
c. Bảng sự thật các chân kết nối

Bảng 2.2: Bảng sự thật các chân kết nối

SignBit	C	SEL0	SEL1
0	0	SignBit' = 1	0
0	1	0	SignBit' = 1
1	0	SignBit' = 0	0
1	1	0	SignBit' = 0

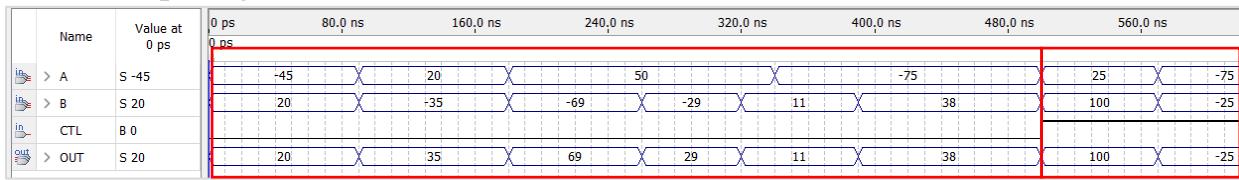
- SEL0 = SignBit'.C'
- SEL1 = SignBit'.C

d. Vẽ mạch khối tính ABS/MAX



Hình 2.5: Mạch khối tính ABS/MAX

e. Mô phỏng khối tính ABS/MAX



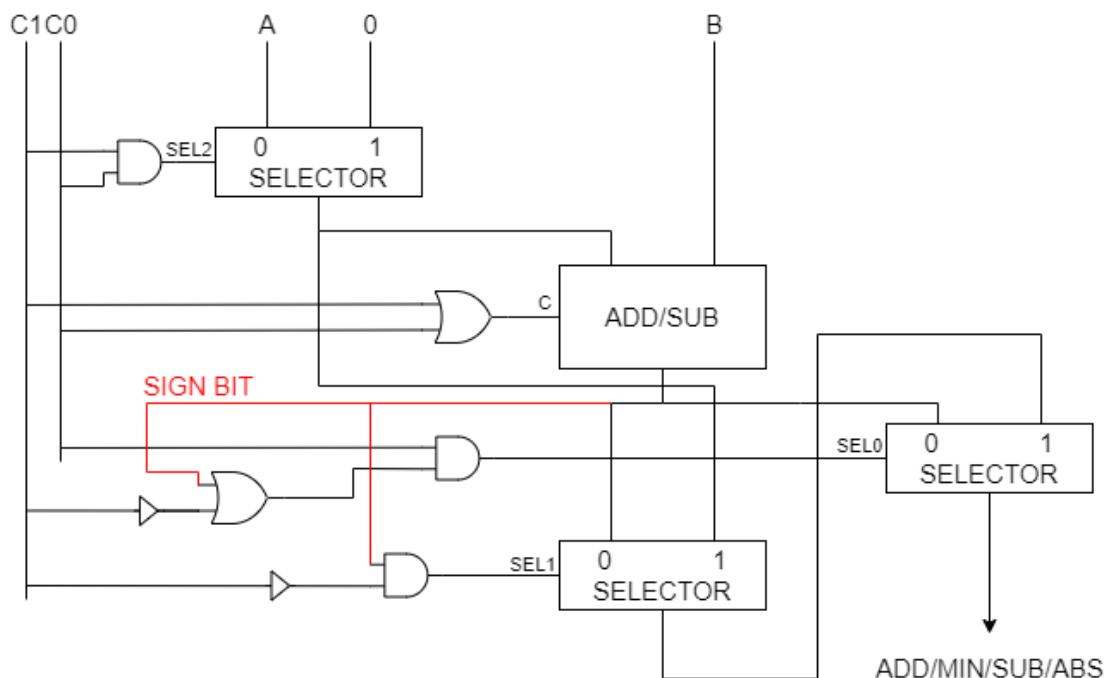
|B|

MAX

Hình 2.6: Kết quả mô phỏng khối tính ABS/MAX

4. Thiết kế khối tính ADD/MIN/SUB/ABS

a. Phân tích thiết kế khối tính ADD/MIN/SUB/ABS



Hình 2.7: Sơ đồ thiết kế khối tính ADD/MIN/SUB/ABS

Khối ABS/MIN/ADD/SUB có 4 chức năng nên cần 3 bộ chọn MUX 2to1 và một khối cộng/trừ đã thiết kế trước đó kèm theo 2 tín hiệu điều khiển C1, C0 để lựa chọn chức năng cần tính toán.

b. Bảng mã hóa chức năng

Bảng 2.3: Bảng mã hóa chức năng khối tính ADD/MIN/SUB/ABS

C1	C0	Chức năng
0	0	ADD
0	1	MIN
1	0	SUB
1	1	ABS

c. Bảng sự thật các chân kết nối

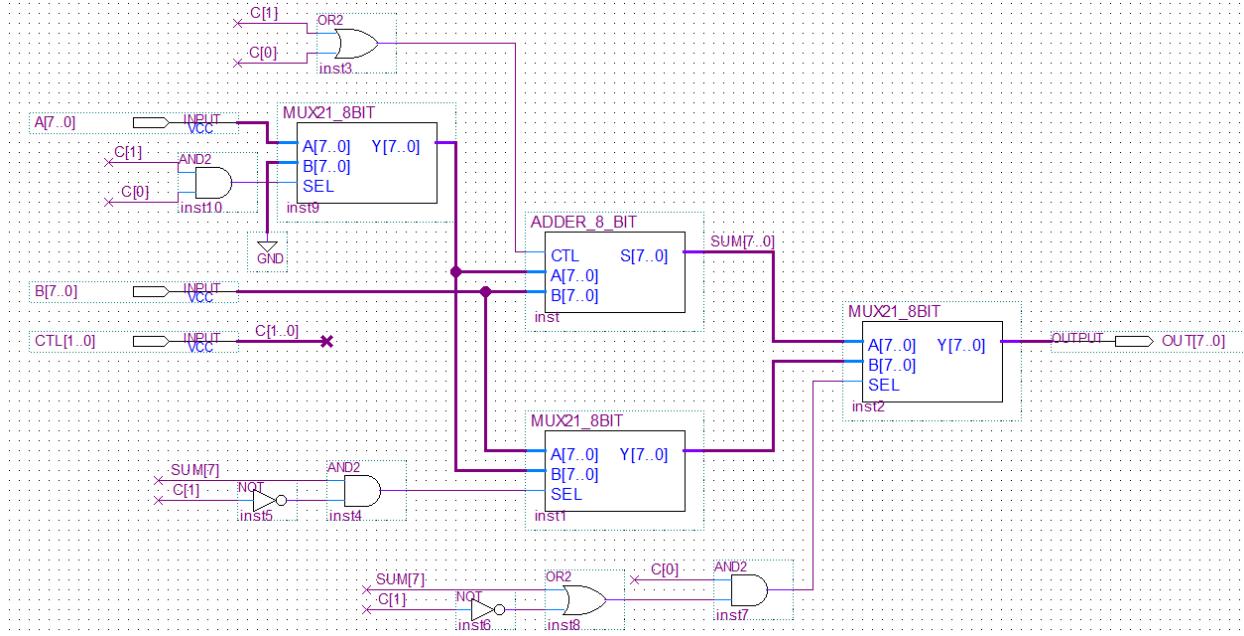
Bảng 2.4: Bảng sự thật các chân kết nối

SignBit	C1	C0	C	SEL 2	SEL1	SEL0
0	0	0	0	0	X	0
0	0	1	1	0	SignBit = 0	1
0	1	0	1	0	X	0
0	1	1	1	1	0	SignBit = 0
1	0	0	0	0	X	0
1	0	1	1	0	SignBit = 1	1
1	1	0	1	0	X	0
1	1	1	1	1	0	SignBit = 1

Qua bảng trên, ta lập được các phương trình tín hiệu cho các ngõ vào lựa chọn cho bộ cộng/trừ và các bộ MUX 2to1 như sau:

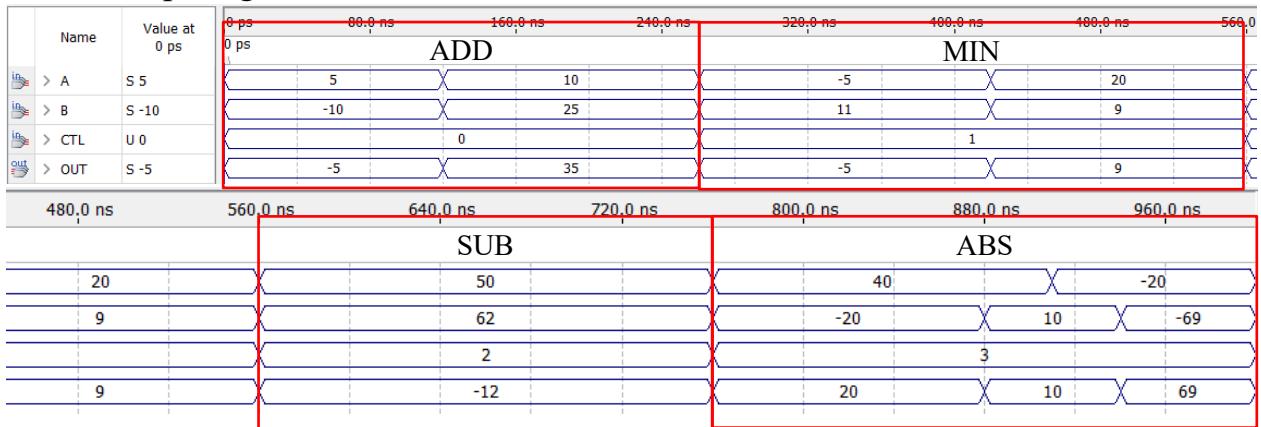
- $C = C1 + C0$
- $SEL2 = C1.C0$
- $SEL1 = SignBit.C1'$
- $SEL0 = C0.(SignBit + C1')$

d. Vẽ mạch khối tính ADD/MIN/SUB/ABS



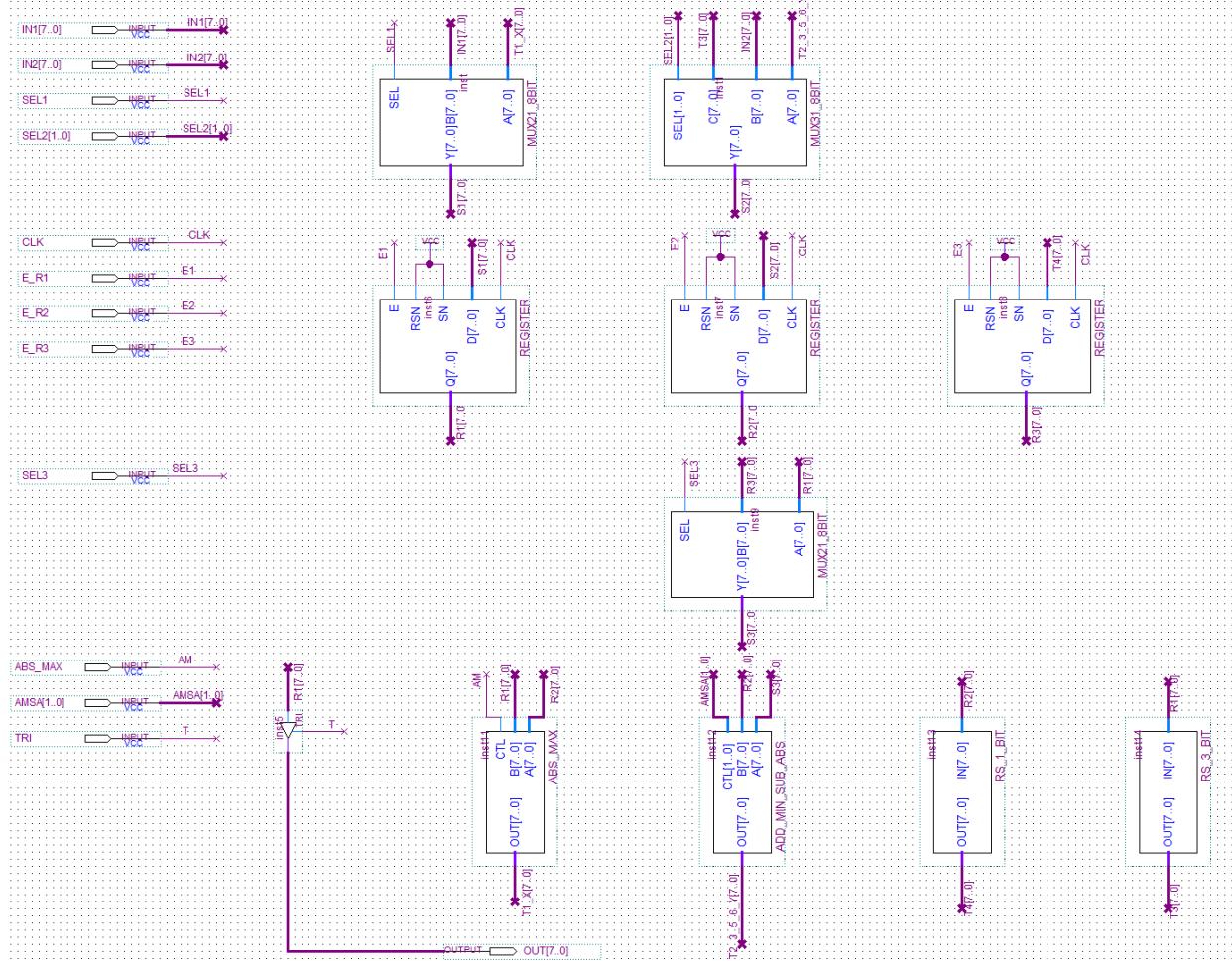
Hình 2.8: Mạch khối tính ADD/MIN/SUB/ABS

e. Mô phỏng khối ADD/MIN/SUB/ABS



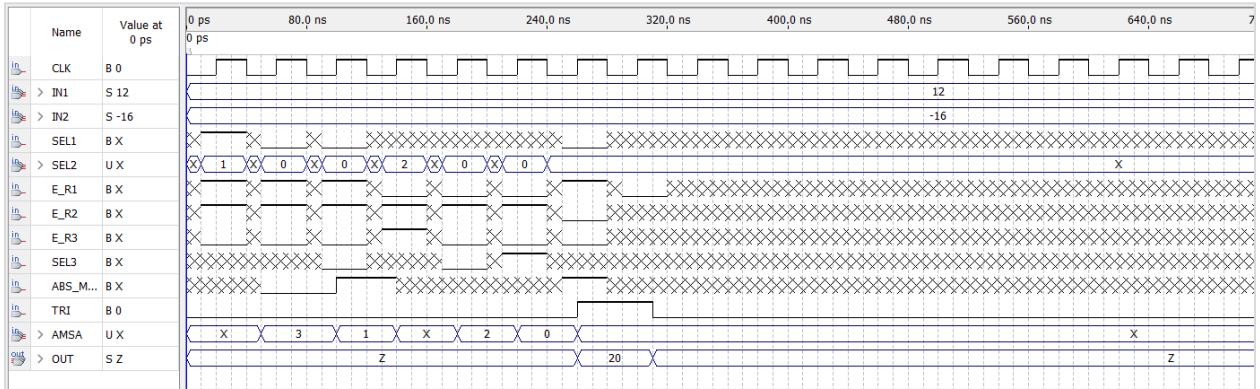
Hình 2.9: Mô phỏng khối tính ADD/MIN/SUB/ABS

5. Tổng hợp thiết kế datapath



Hình 2.10: Tổng hợp datapath theo thiết kế Functional - Unit Sharing

6. Mô phỏng datapath và phân tích



Hình 2.11: Mô phỏng datapath

Phân tích:

Ở cạnh lẻ xung clock lần 1:

- SEL1 = 1: lấy dữ liệu input lưu vào R1.
- SEL2 = 01: lấy dữ liệu input lưu vào R2.
- SEL3 = X: chưa quan tâm đến bộ chọn thứ 3.
- E_R1 = 1: cho phép lưu giá trị vừa lấy từ bộ chọn.
- E_R2 = 1: cho phép lưu giá trị vừa lấy từ bộ chọn.
- E_R3 = 0: thanh ghi R3 chưa hoạt động.
- AM = X: chưa sử dụng khối tính ABS/MAX.
- AMSA = X: chưa sử dụng khối tính ADD/MIN/SUB/ABS.
- TRI = 0: không có phép giá trị xuất ra ngoài.
- OUT = Z: chưa có kết quả phép tính.

Ở cạnh lẻ xung clock lần 2:

- SEL1 = 0: lấy dữ liệu từ khối tính ABS/MAX lưu vào R1.
- SEL2 = 00: lấy dữ liệu từ khối tính ADD/MIN/SUB/ABS lưu vào R2.
- SEL3 = X: chưa quan tâm đến bộ chọn thứ 3.
- E_R1 = 1: cho phép lưu giá trị vừa lấy từ bộ chọn.
- E_R2 = 1: cho phép lưu giá trị vừa lấy từ bộ chọn.
- E_R3 = 0: thanh ghi R3 chưa hoạt động.
- AM = 0: chọn chế độ tính ABS
- AMSA = 11: chọn chế độ tính ABS.
- TRI = 0: không có phép giá trị xuất ra ngoài.
- OUT = Z: chưa có kết quả phép tính.

Ở cạnh lén xung clock lần 3:

- SEL1 = 0: lấy dữ liệu từ khối tính ABS/MAX lưu vào R1.
- SEL2 = 00: lấy dữ liệu từ khối tính ADD/MIN/SUB/ABS lưu vào R2.
- SEL3 = 0: chọn giá trị t1 (lưu trong R1) để tính $y = \text{MIN}(t1, t2)$.
- E_R1 = 1: cho phép lưu giá trị vừa lấy từ bộ chọn.
- E_R2 = 1: cho phép lưu giá trị vừa lấy từ bộ chọn.
- E_R3 = 0: thanh ghi R3 chưa hoạt động.
- AM = 1: chọn chế độ tính MAX
- AMSA = 01: chọn chế độ tính MIN.
- TRI = 0: không có phép giá trị xuất ra ngoài ra.
- OUT = Z: chưa có kết quả phép tính.

Ở cạnh lén xung clock lần 4:

- SEL1 = X: chưa quan tâm đến bộ chọn thứ 1.
- SEL2 = 10: lấy dữ liệu $t3 = x >> 3$ lưu vào R2.
- SEL3 = X: chưa quan tâm đến bộ chọn thứ 3.
- E_R1 = 0: thanh ghi R1 chưa hoạt động.
- E_R2 = 1: cho phép lưu giá trị vừa lấy từ bộ chọn vào R2.
- E_R3 = 1: thanh ghi R3 lưu giá trị $t4 = y >> 1$.
- AM = X: chưa sử dụng
- AMSA = XX: chưa sử dụng.
- TRI = 0: không có phép giá trị xuất ra ngoài ra.
- OUT = Z: chưa có kết quả phép tính.

Ở cạnh lén xung clock lần 5:

- SEL1 = X: chưa quan tâm đến bộ chọn thứ 1.
- SEL2 = 00: lấy dữ liệu từ khối tính ADD/MIN/SUB/ABS lưu vào R2.
- SEL3 = 0: chọn giá trị x từ R1 để tính $t5 = \text{SUB}(x, t3)$.
- E_R1 = 0: thanh ghi R1 chưa hoạt động.
- E_R2 = 1: cho phép lưu giá trị vừa lấy từ bộ chọn vào R2.
- E_R3 = 0: thanh ghi R3 chưa hoạt động.
- AM = X: chưa sử dụng.
- AMSA = 10: chọn chế độ tính SUB.
- TRI = 0: không có phép giá trị xuất ra ngoài ra.
- OUT = Z: chưa có kết quả phép tính.

Ở cạnh lén xung clock lần 6:

- SEL1 = X: chưa quan tâm đến bộ chọn thứ 1.
- SEL2 = 00: lấy dữ liệu từ khối tính ADD/MIN/SUB/ABS lưu vào R2.
- SEL3 = 1: chọn giá trị t4 từ R3 để tính t6 = ADD(t4, t5).
- E_R1 = 0: thanh ghi R1 chưa hoạt động.
- E_R2 = 1: cho phép lưu giá trị vừa lấy từ bộ chọn vào R2.
- E_R3 = 0: thanh ghi R3 chưa hoạt động.
- AM = X: chưa sử dụng.
- AMSA = 00: chọn chế độ tính ADD.
- TRI = 0: không có phép giá trị xuất ra ngoặc ra.
- OUT = Z: chưa có kết quả phép tính.

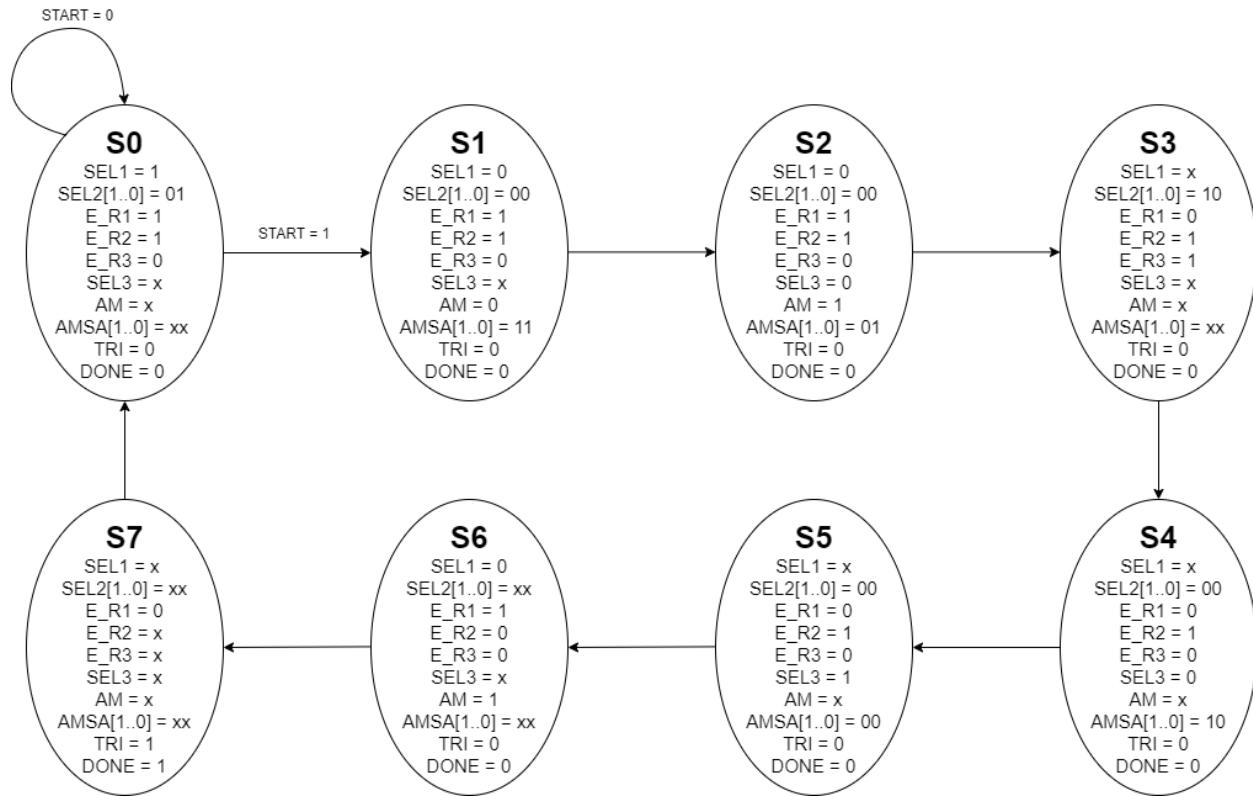
Ở cạnh lén xung clock lần 7:

- SEL1 = 0: lấy dữ liệu từ khối tính ABS/MAX lưu vào R1.
- SEL2 = XX: chưa quan tâm đến bộ chọn thứ 2.
- SEL3 = X: chưa quan tâm đến bộ chọn thứ 3.
- E_R1 = 1: thanh ghi R1 hoạt động để lưu t7 = MAX(t6, x).
- E_R2 = 0: thanh ghi R2 chưa hoạt động.
- E_R3 = 0: thanh ghi R3 chưa hoạt động.
- AM = 1: chọn chế độ tính MAX.
- AMSA = XX: chưa sử dụng.
- TRI = 0: không có phép giá trị xuất ra ngoặc ra.
- OUT = Z: chưa có kết quả phép tính.

Khi tín hiệu TRI = 1, kết quả phép tính lập tức sẽ xuất hiện ở ngoặc ra.

III. THIẾT KẾ KHỐI CONTROLER

1. Sơ đồ trạng thái



Hình 2.12: Sơ đồ trạng thái theo thiết kế Functional – Unit Sharing

Các tín hiệu được mô tả như sau:

- START: tín hiệu bắt đầu quá trình tính toán.
- SEL1: bộ chọn [ABS/MAX, In1] để lưu vào thanh ghi R1.
- SEL2[1..0]: bộ chọn [ADD/MIN/SUB/ABS, In2, >>3] để lưu vào R2.
- SEL3: bộ chọn giá trị lưu trong R1 và R3 để đưa vào khối ABS/MAX.
- E_R1: cho phép ghi hoặc không vào thanh ghi R1.
- E_R2: cho phép ghi hoặc không vào thanh ghi R2.
- E_R3: cho phép ghi hoặc không vào thanh ghi R3.
- AM: chọn chế độ trong khối ABS/MAX.
- AMSA: chọn chế độ trong khối ADD/MIN/SUB/ABS.
- TRI: tín hiệu cho phép/cấm xuất giá trị ngõ ra.
- DONE: tín hiệu hoàn thành.

2. Mã hóa trạng thái

Bảng 2.5: Bảng gán trạng thái theo bìa K-Map

F		D1D0			
		00	01	11	10
D2	0	S0	S1	S2	S3
	1	S7	S6	S5	S4

Bảng 2.6: Bảng mã hóa trạng thái

Trạng thái	Mã hóa
S0	000
S1	001
S2	011
S3	010
S4	110
S5	111
S6	101
S7	100

3. Chọn loại flipflop và lập bảng chuyển trạng thái

Bảng 2.7: Bảng chuyển trạng thái

TTHT	START		DONE
	0	1	
S0	S0	S1	0
S1	S2	S2	0
S2	S3	S3	0
S3	S4	S4	0
S4	S5	S5	0
S5	S6	S6	0
S6	S7	S7	0
S7	S0	S0	1

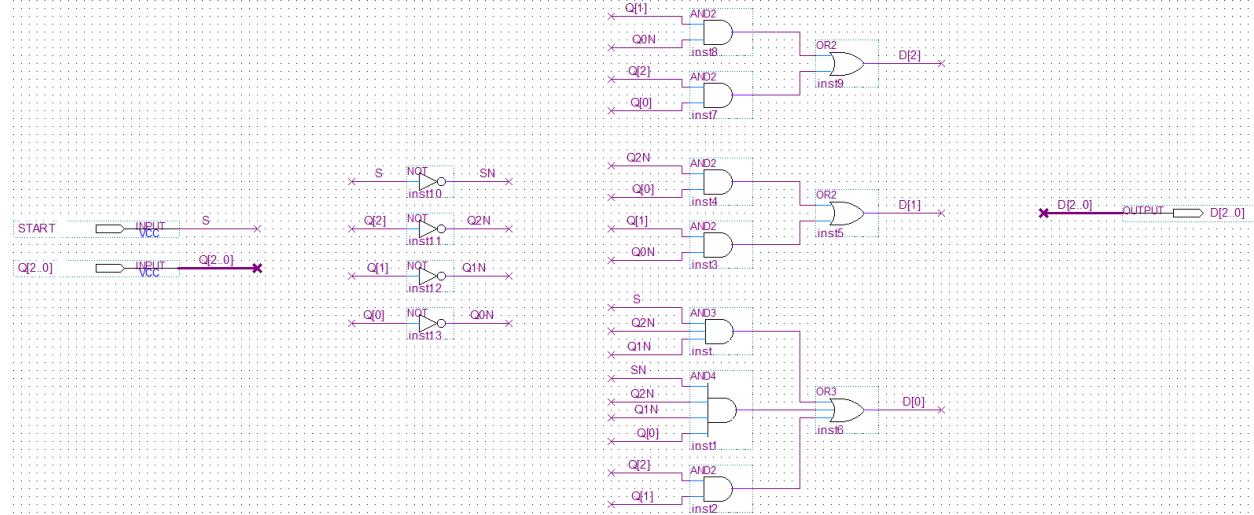
4. Thiết kế khối trạng thái kế tiếp

a. Phương trình chuyển trạng thái

Từ bảng chuyển trạng thái, ta lập các phương trình chuyển trạng thái như sau:

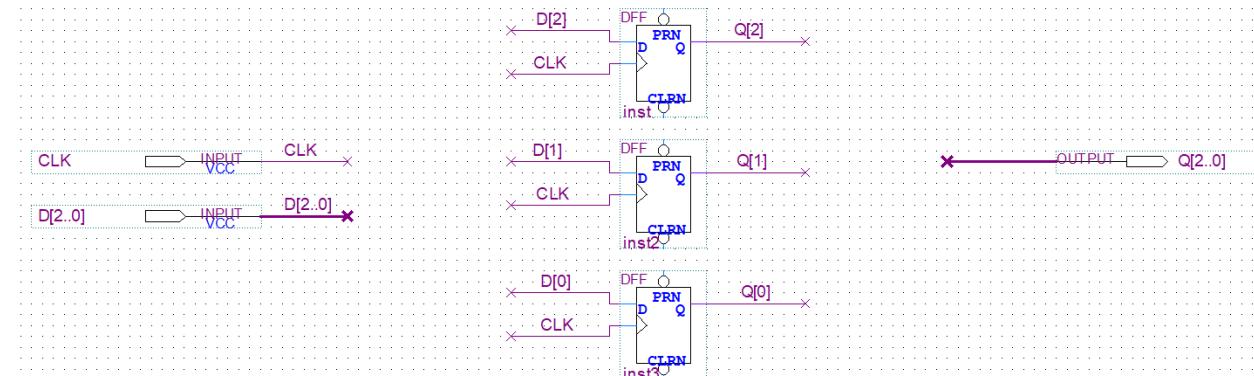
- $D_2(\text{next}) = S_3 + S_4 + S_5 + S_6 = D_1.D_0' + D_2.D_0$
- $D_1(\text{next}) = S_1 + S_2 + S_3 + S_4 = D_2'.D_0 + D_1.D_0'$
- $D_0(\text{next}) = S.(S_0 + S_1) + S_4 + S_5 = S.D_2'.D_1' + S'.D_2'.D_1'.D_0 + D_2.D_1$

b. Vẽ mạch khối trạng thái kế tiếp



Hình 2.13: Mạch khối trạng thái kế tiếp

5. Thiết kế khối nhớ



Hình 2.14: Mạch khối nhớ

6. Thiết kế khối ngõ ra tạo tín hiệu điều khiển

a. Bảng sự thật tín hiệu ngõ ra

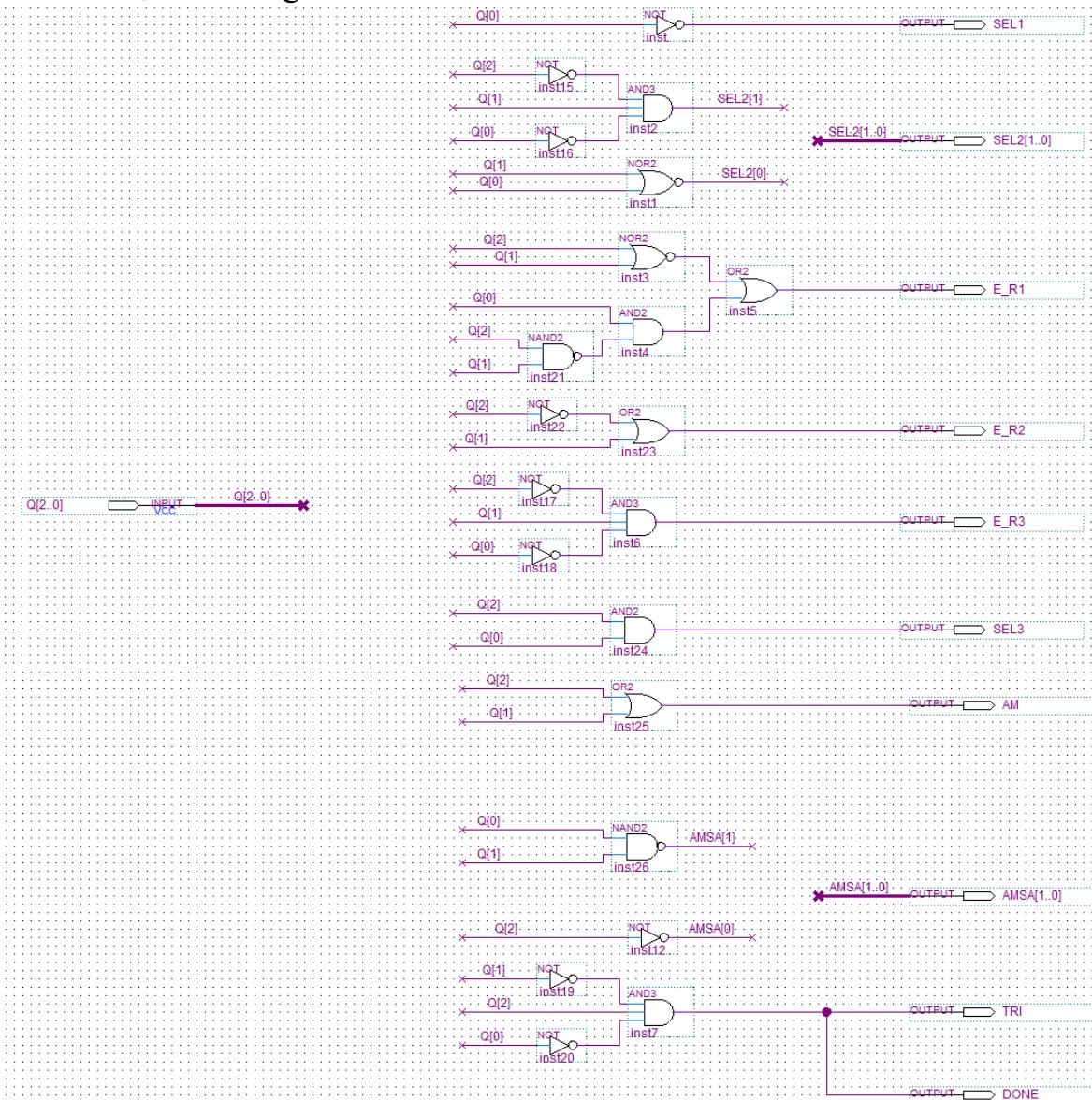
Bảng 2.8: Bảng sự thật tín hiệu ngõ ra

Trạng thái	Mã hóa	SEL1	SEL2	ER1	ER2	ER3	SEL3	AM	AMSA	TRI
S0	000	1	01	1	1	0	X	X	XX	0
S1	001	0	00	1	1	0	X	0	11	0
S2	011	0	00	1	1	0	0	1	01	0
S3	010	X	10	0	1	1	X	X	XX	0
S4	110	X	00	0	1	0	0	X	10	0
S5	111	X	00	0	1	0	1	X	00	0
S6	101	0	XX	1	0	0	X	1	XX	0
S7	100	X	XX	0	X	X	X	X	XX	1

Từ bảng sự thật, ta có phương trình các tín hiệu điều khiển ngõ ra như sau:

- $SEL1 = Q0'$
- $SEL2[1] = Q2'.Q1.Q0'$
- $SEL2[0] = (Q1 + Q0)'$
- $ER1 = (Q2 + Q1)' + Q0.(Q2.Q1)'$
- $ER2 = Q2' + Q1$
- $ER3 = Q2'.Q1.Q0'$
- $SEL3 = Q2.Q0$
- $AM = Q2 + Q1$
- $AMSA[1] = (Q1.Q0)'$
- $AMSA[0] = Q2'$
- $TRI = Q2.Q1'.Q0'$

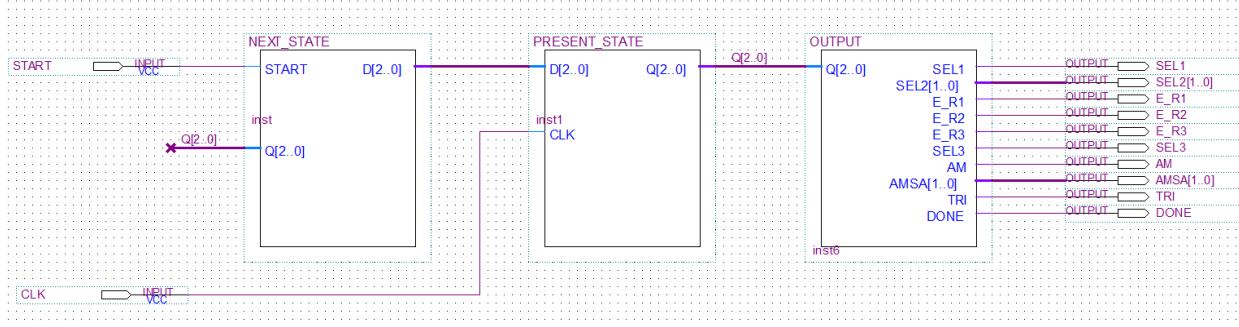
b. Vẽ mạch khói ngõ ra



Hình 2.15: Mạch khói ngõ ra

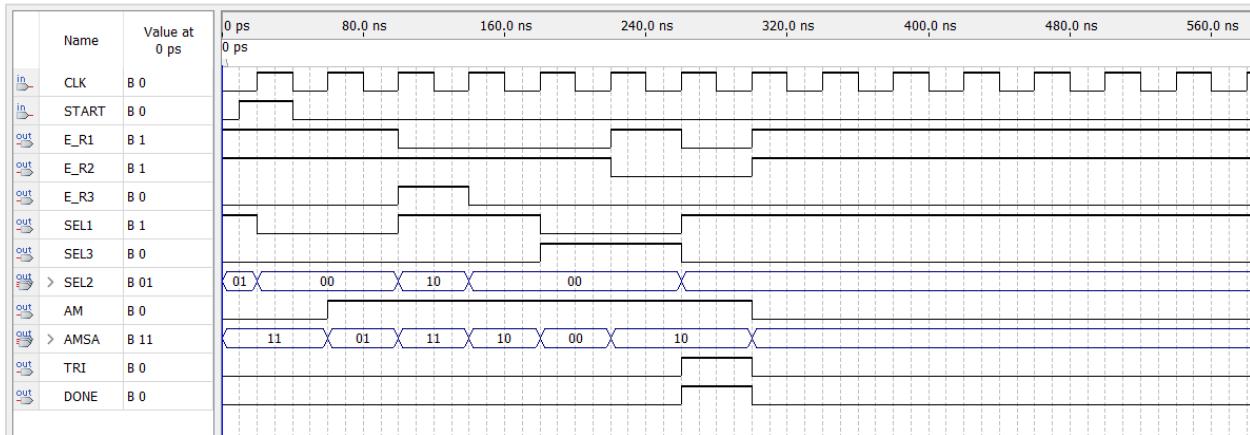
7. TỔNG HỢP THIẾT KẾ controller

a. Vẽ mạch khối controller



Hình 2.16: Tổng hợp thiết kế controller

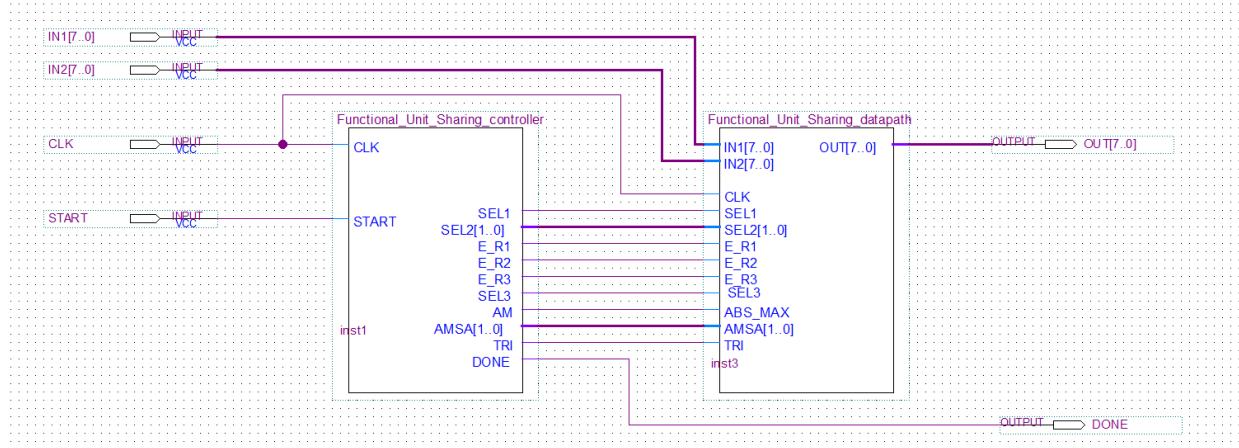
b. Mô phỏng khối controller



Hình 2.17: Mô phỏng khối controller

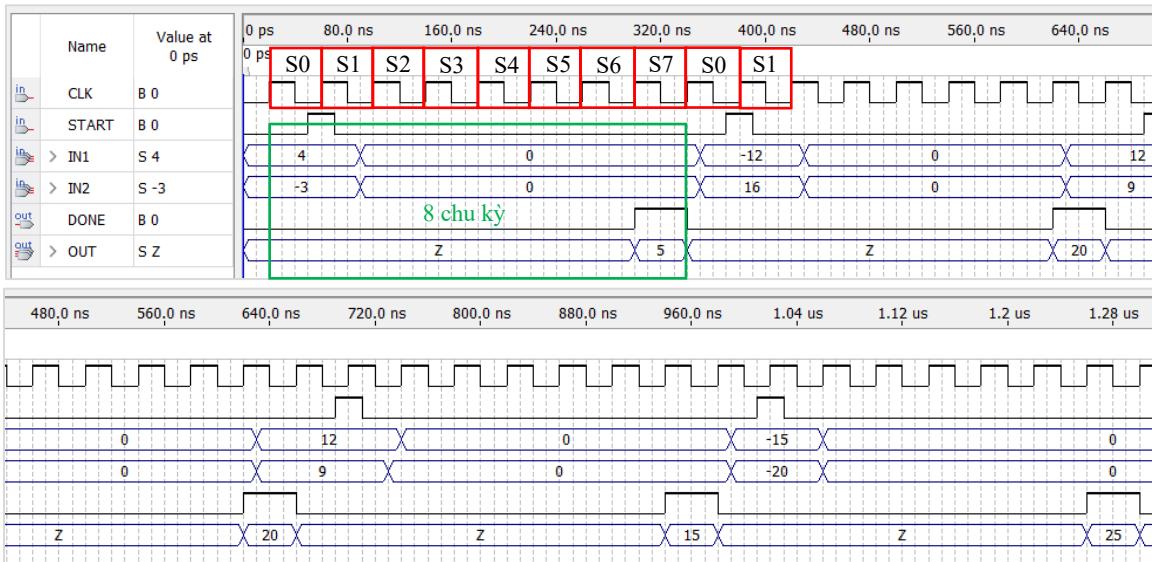
IV. TỔNG HỢP THIẾT KẾ VÀ CHẠY MÔ PHỎNG

1. Tổng hợp thiết kế



Hình 2.18: Tổng hợp thiết kế Functional - Unit Sharing

2. Mô phỏng và phân tích



Hình 2.19: Kết quả mô phỏng theo thiết kế Functional - Unit Sharing

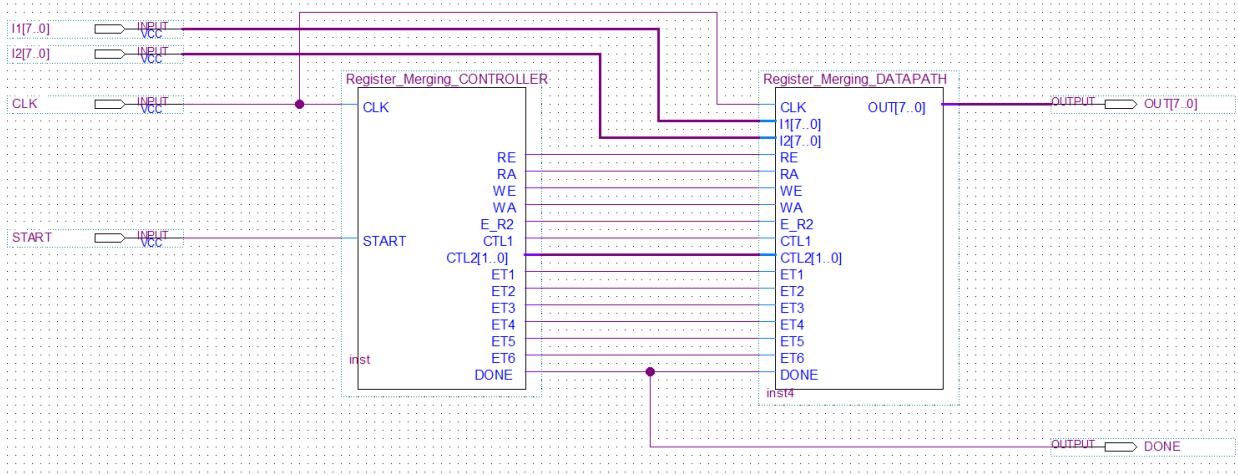
Các trường hợp thử nghiệm là:

- Số A dương, số B âm: $\sqrt{4^2 + (-3)^2} = 5$
- Số A âm, số B dương: $\sqrt{(-12)^2 + 16^2} = 20$
- Số A dương, số B dương: $\sqrt{12^2 + 9^2} = 15$
- Số A âm, số B âm: $\sqrt{(-15)^2 + (-20)^2} = 25$

Sau khi tín hiệu START = 1, trạng thái sẽ chuyển từ S0 sang S1 khi có xung clock kích cạnh lên, đồng thời các tín hiệu điều khiển từ khói controller sẽ được tính toán và chuyển qua khói đường dữ liệu để thực thi các lệnh theo yêu cầu. Đến khi xung clock kích cạnh lên lần thứ 7 (tính từ khi START = 1), lúc này trạng thái hiện tại là S7, tín hiệu báo DONE = 1 và đồng thời giá trị phép tính sẽ được xuất ra ngõ ra OUT. Tổng thời gian thực thi của thiết kế này mất 8 chu kỳ xung clock để hoàn thành.

BÀI TẬP 3. REGISTER MERGING

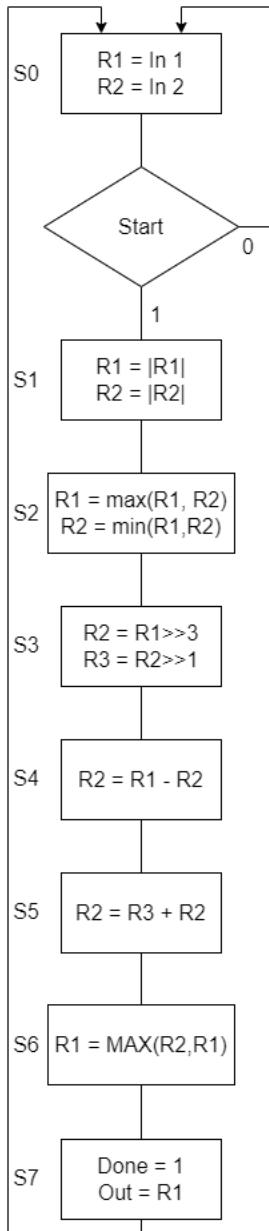
I. TỔNG QUÁT THIẾT KẾ



Hình 3.1: Tổng quát thiết kế Register Merging

Mục tiêu của bài toán Register Merging là gom nhóm các thanh ghi có thời điểm truy cập (đọc và ghi) không bị trùng nhau nhằm sử dụng chung các cổng ngõ vào và ngõ ra giúp làm giảm số đường kết nối trong khối đường dữ liệu bởi số cổng sẽ ít đi. Tuy nhiên, nó cũng làm tăng thời gian truy xuất thanh ghi do phải mất thêm thời gian xử lý bộ giải mã địa chỉ để chọn thanh ghi muốn truy xuất.

Để gom nhóm được các thanh ghi thỏa mãn yêu cầu trên, chúng ta cần xem biểu đồ ASM, thông qua đó kiểm tra thời gian đọc và ghi của các thanh ghi và chọn những thanh ghi tương hợp với nhau để gom nhóm.



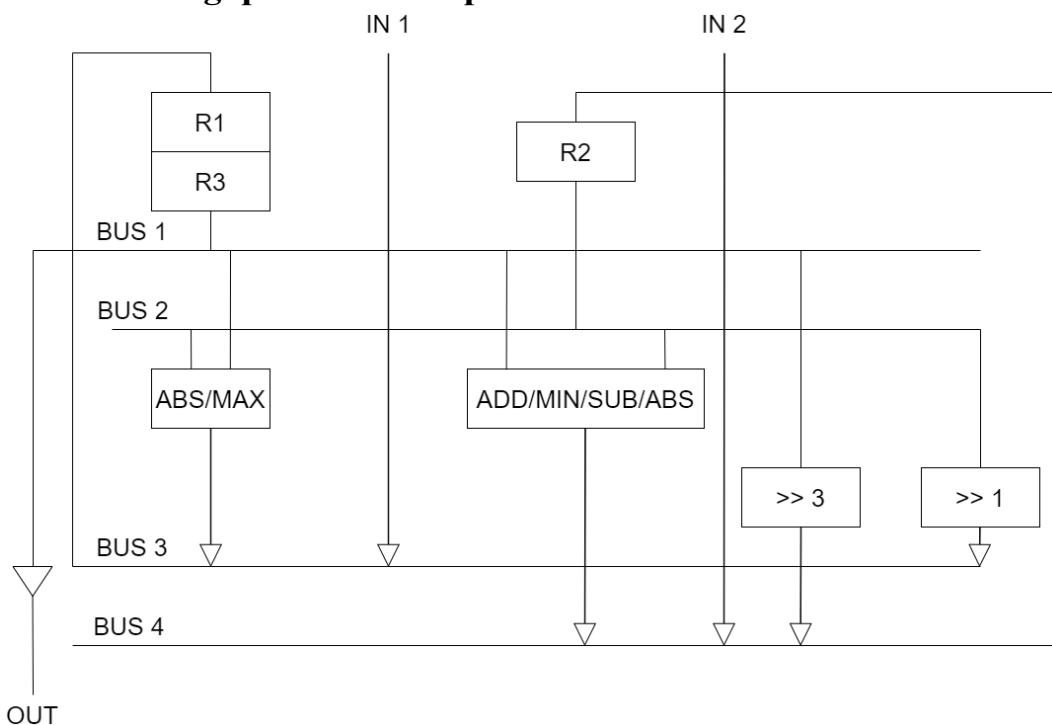
Hình 3.2: Biểu đồ ASM cho bài toán tính xấp xỉ căn bậc hai theo thiết kế Register Merging

	S ₀	S ₁	S ₂	S ₃	S ₄	S ₅	S ₆	S ₇
R ₁	▷	▷	▷	▷	▷	▷	▷	▷
R ₂	▷	▷	▷	▷	▷	▷	▷	▷
R ₃				▷	▷	▷	▷	

Thông qua bảng truy xuất thanh ghi, các thanh ghi R1 và R3 tương hợp vì chúng không được truy cập đồng thời, do đó chúng ta có thể gom nhóm R1 và R3 thành một tập thanh ghi với một cổng đọc và một cổng ghi.

II. THIẾT KẾ KHỐI DATAPATH

1. Mô hình tổng quát khối datapath



Hình 3.3: Sơ đồ thiết kế khối đường dữ liệu thực thi Register Merging

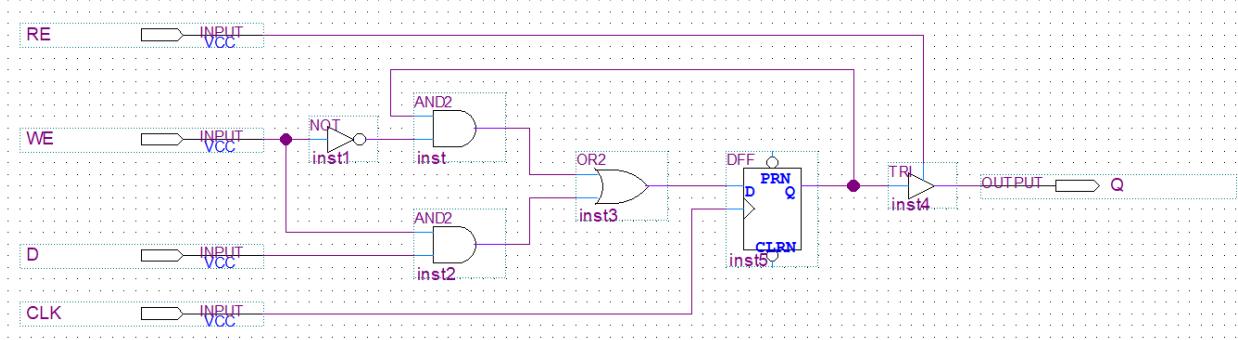
2. Phân tích các thành phần trong khối datapath

Datapath trong bài toán này gồm các thành phần sau:

- Register Files: R1.R3
- Register: R3 (tái sử dụng thiết kế trước đó).
- Khối tính ABS/MAX (tái sử dụng thiết kế trước đó).
- Khối tính ADD/MIN/SUB/ABS (tái sử dụng thiết kế trước đó).

3. Thiết kế Register Files R1.R3

a. Thiết kế Register Files Cell.

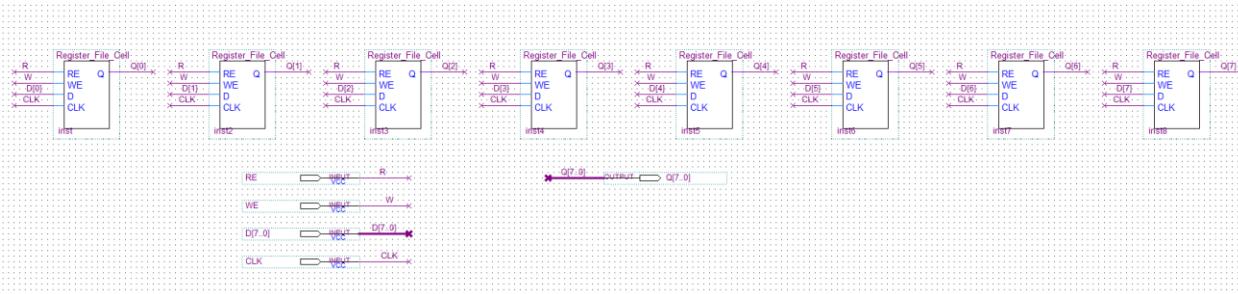


Hình 3.4: Mạch Register Files Cell

Tín hiệu RE (Read Enable): cho phép đọc dữ liệu.

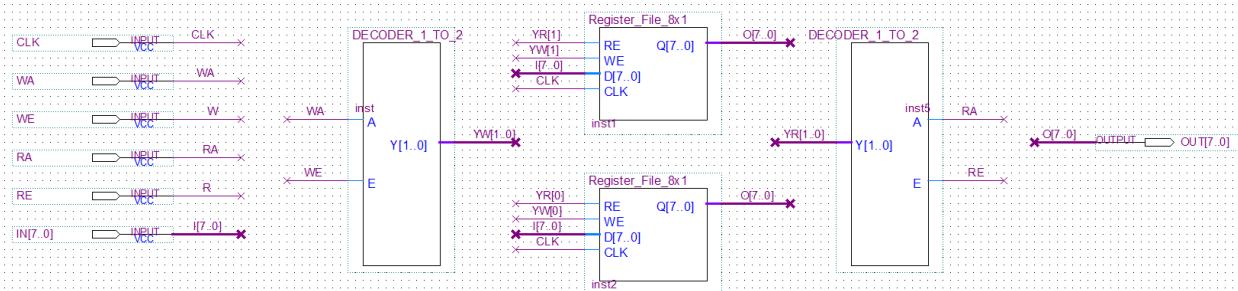
Tín hiệu WE (Write Enable): cho phép ghi dữ liệu.

b. Thiết kế Register Files 8x1 từ Register Files Cell



Hình 3.5: Mạch Register Files 8x1

c. Thiết kế Register Files 8x2 từ Register Files 8x1

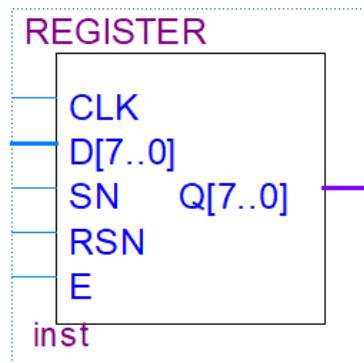


Hình 3.6: Mạch Register Files 8x2

Tín hiệu RA (Read Address): giải mã địa chỉ cần đọc.

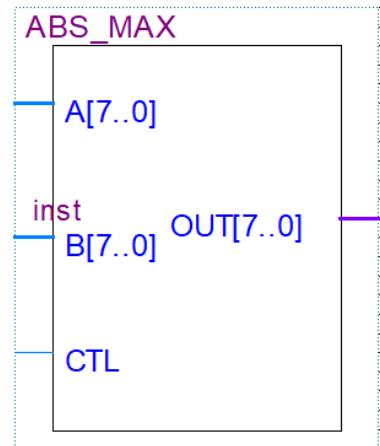
Tín hiệu WA (Write Address): giải mã địa chỉ cần ghi.

4. Tái sử dụng thiết kế Register 8 bit



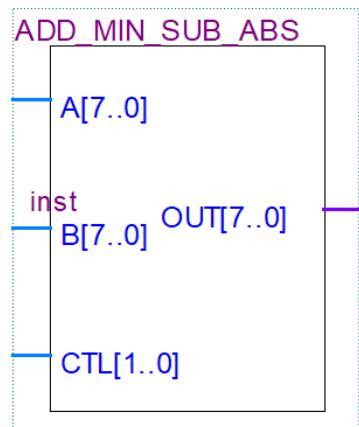
Hình 3.7: Khối REGISTER 8 bit

5. Tái sử dụng thiết kế khối tính ABS/MAX



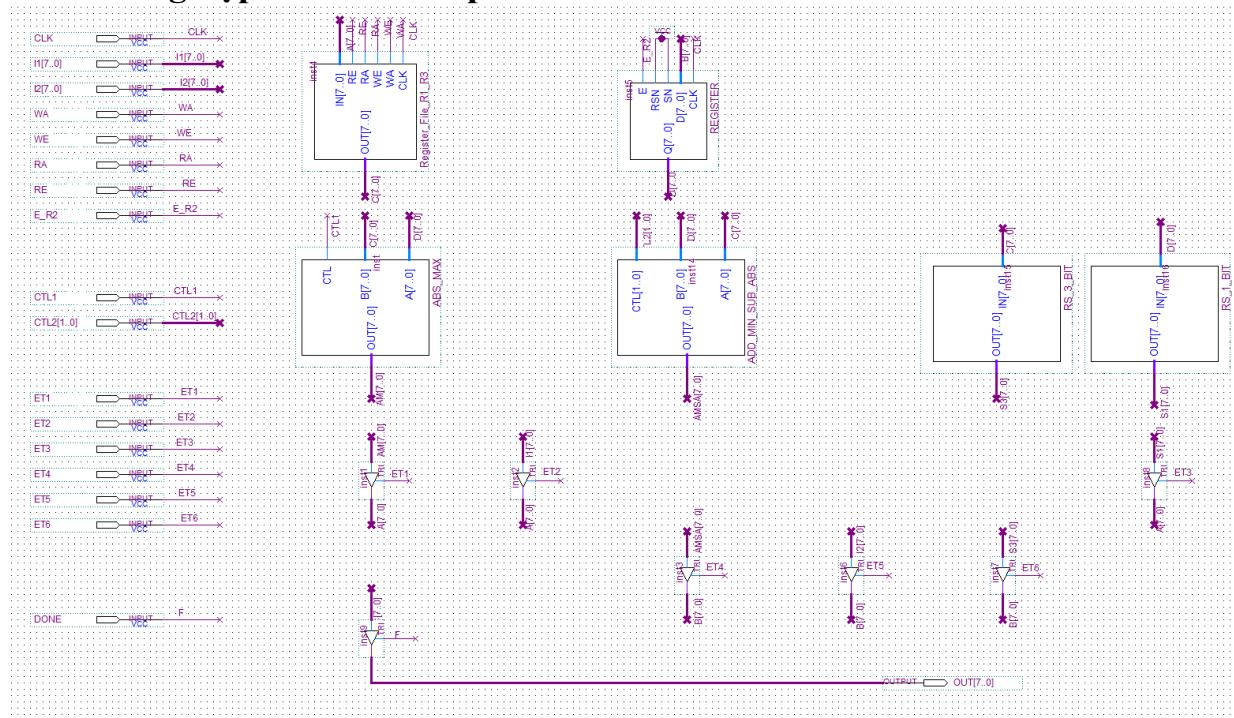
Hình 3.8: Khối tính ABS/MAX

6. Tái sử dụng thiết kế khối tính ADD/MIN/SUB/ABS



Hình 3.9: Khối tính ADD/MIN/SUB/ABS

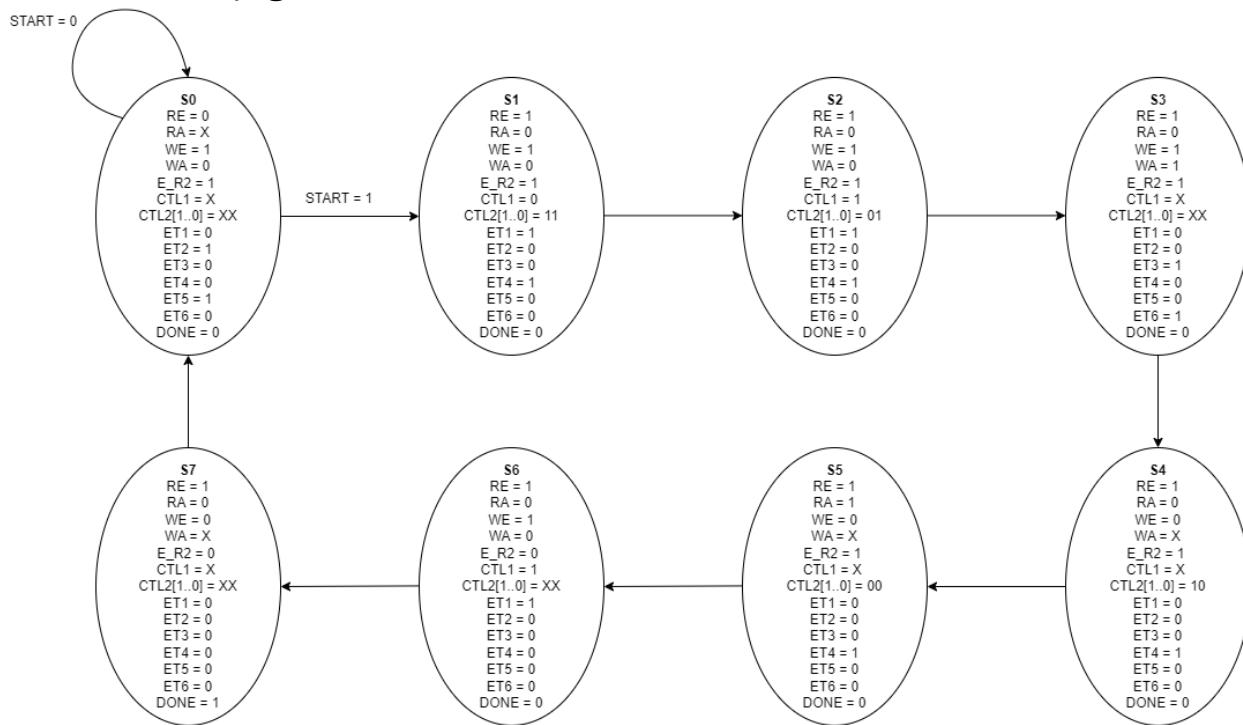
7. Tổng hợp thiết kế datapath



Hình 3.10: Tổng hợp datapath theo thiết kế Register Merging

III. THIẾT KẾ KHÓI CONTROLLER

1. Sơ đồ trạng thái



Hình 3.11: Sơ đồ trạng thái theo thiết kế Register Merging

Các tín hiệu được mô tả như sau:

- START: tín hiệu khởi động quá trình tính toán.
- RE: tín hiệu cho phép đọc từ Register Files R1.R3.
- RA: tín hiệu giải mã địa chỉ để đọc từ Register Files R1.R3.
- WE: tín hiệu cho phép ghi vào Register Files R1.R3.
- WA: tín hiệu giải mã địa chỉ để ghi vào Register Files R1.R3.
- E_R2: tín hiệu cho phép ghi hoặc không vào thanh ghi R2.
- CTL1: tín hiệu chọn chế độ cho khối ABS/MAX.
- CTL2[1..0]: tín hiệu chọn chế độ cho khối ADD/MIN/SUB/ABS.
- ET1: tín hiệu cho phép/cầm xuất giá trị từ khối ABS/MAX.
- ET2: tín hiệu cho phép/cầm xuất giá trị từ input đầu vào 1.
- ET3: tín hiệu cho phép/cầm xuất giá trị từ khối >>1.
- ET4: tín hiệu cho phép/cầm xuất giá trị từ khối ADD/MIN/SUB/ABS.
- ET5: tín hiệu cho phép/cầm xuất giá trị từ input đầu vào 2.
- ET6: tín hiệu cho phép/cầm xuất giá trị từ khối >>3.
- DONE: tín hiệu hoàn thành.

2. Mã hóa trạng thái

Bảng 3.1: Bảng gán trạng thái theo bìa K-Map

F		Q1Q0			
		00	01	11	10
Q2	0	S0	S1	S2	S3
	1	S7	S6	S5	S4

Bảng 3.2: Bảng mã hóa trạng thái

Trạng thái	Mã hóa
S0	000
S1	001
S2	011
S3	010
S4	110
S5	111
S6	101
S7	100

3. Chọn loại flipflop và lập bảng chuyển trạng thái

a. Bảng chuyển trạng thái

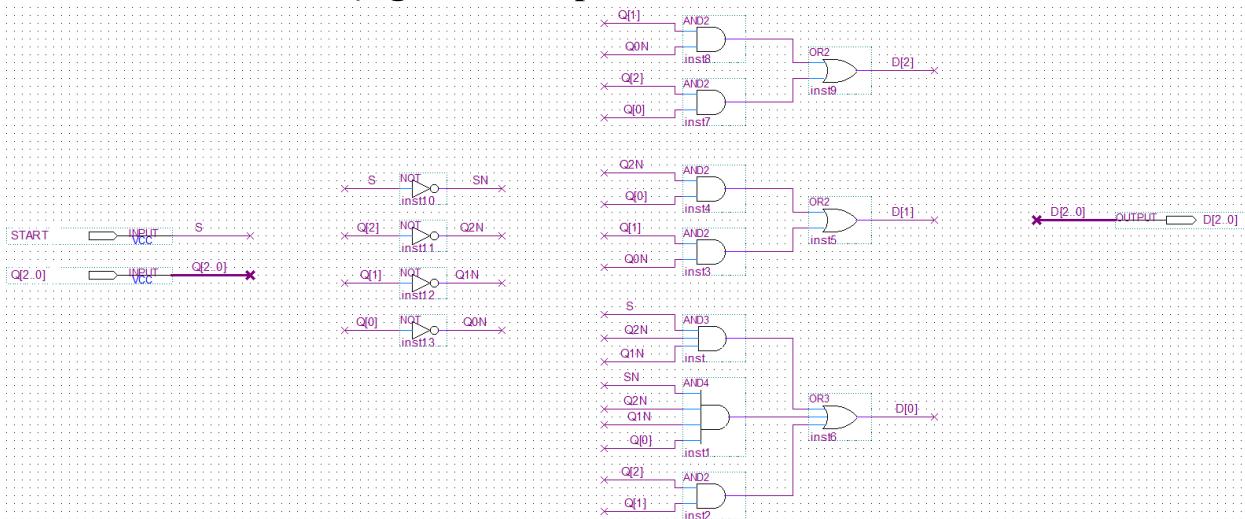
Bảng 3.3: Bảng chuyển trạng thái

TTHT	START		DONE
	0	1	
S0	S0	S1	0
S1	S2	S2	0
S2	S3	S3	0
S3	S4	S4	0
S4	S5	S5	0
S5	S6	S6	0
S6	S7	S7	0
S7	S0	S0	1

b. Phương trình ngõ vào các flipflop

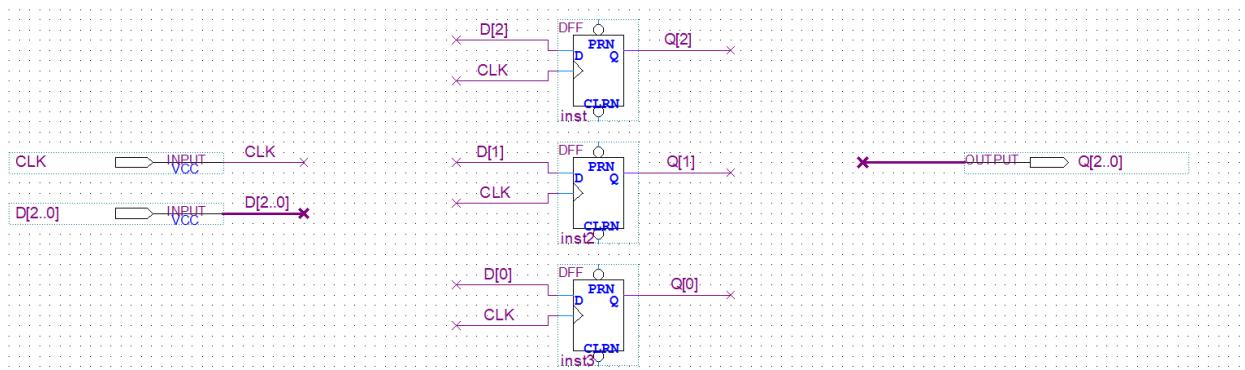
- $D_2 = S_3 + S_4 + S_5 + S_6 = Q_1.Q_0' + Q_2.Q_0$
- $D_1 = S_1 + S_3 + S_4 + S_5 = Q_2'.Q_0 + Q_1.Q_0'$
- $D_0 = S.(S_0 + S_1) + S_4 + S_5 = S.Q_2'.Q_1' + S'.Q_2'.Q_1'.Q_0 + Q_2.Q_1$

4. Thiết kế khối trạng thái kế tiếp



Hình 3.12: Mạch khối trạng thái kế tiếp

5. Thiết kế khối nhớ



Hình 3.13: Mạch khối nhớ

6. Thiết kế khối ngõ ra

a. Bảng sự thật các ngõ ra

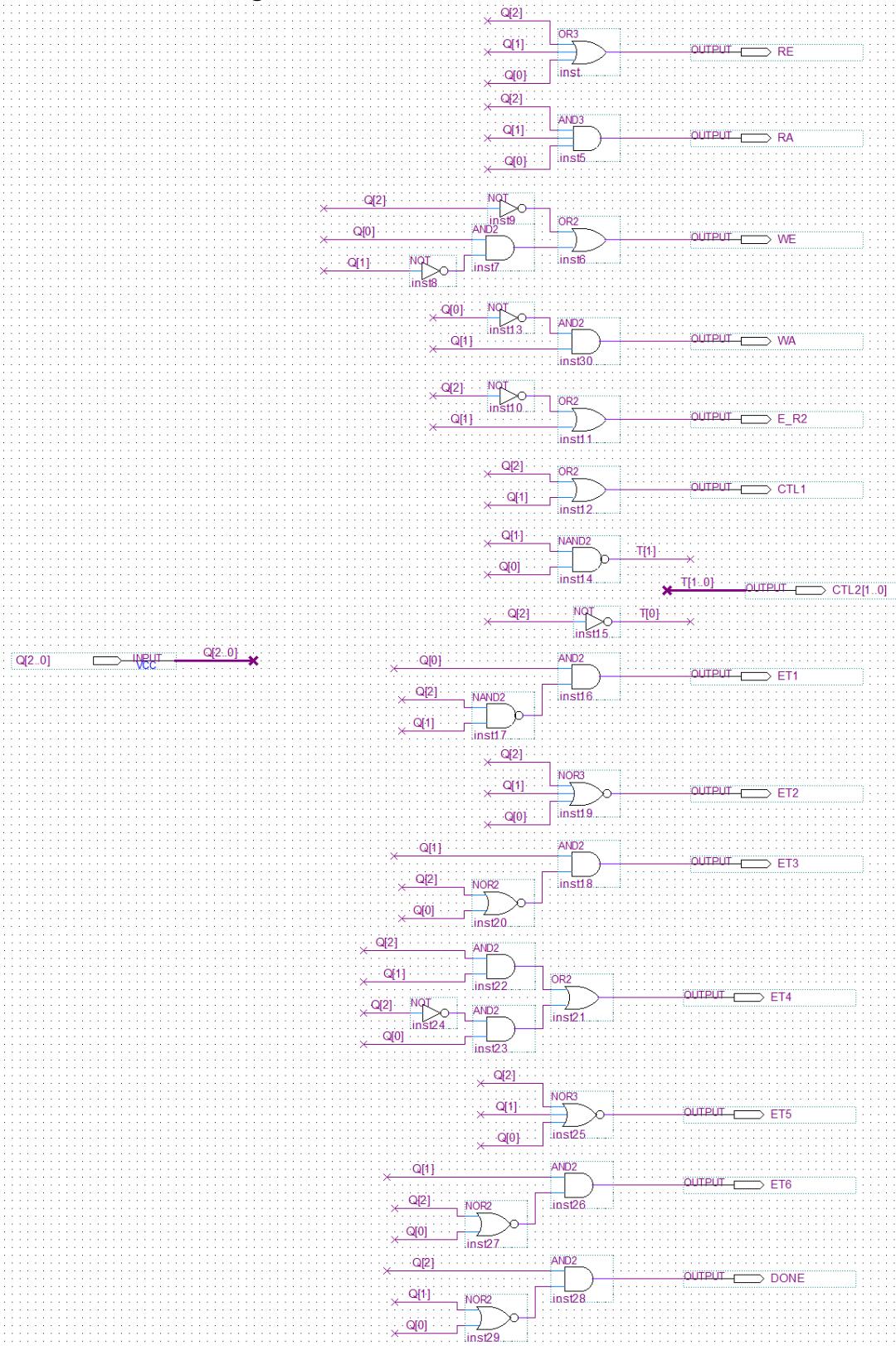
Bảng 3.4: Bảng sự thật các ngõ ra

TH/TT	S0	S1	S2	S3	S4	S5	S6	S7
RE	0	1	1	1	1	1	1	1
RA	X	0	0	0	0	1	0	0
WE	1	1	1	1	0	0	1	0
WA	0	0	0	1	X	X	0	X
ER2	1	1	1	1	1	1	0	0
CTL1	X	0	1	X	X	X	1	X
CTL2[1..0]	XX	11	01	XX	10	00	XX	XX
ET1	0	1	1	0	0	0	1	0
ET2	1	0	0	0	0	0	0	0
ET3	0	0	0	1	0	0	0	0
ET4	0	1	1	0	1	1	0	0
ET5	1	0	0	0	0	0	0	0
ET6	0	0	0	1	0	0	0	0

Từ bảng sự thật, ta tổng hợp được phương trình các ngõ ra như sau:

- RE = Q2 + Q1 + Q0
- RA = Q2.Q1.Q0
- WE = Q2' + Q1'.Q0
- WA = Q1.Q0'
- ER2 = Q2' + Q1
- CTL1 = Q2 + Q1
- CTL2[1] = (Q1.Q0)'
- CTL2[0] = Q2'
- ET1 = Q0.(Q2.Q1)'
- ET2 = (Q2 + Q1 + Q0)'
- ET3 = Q2'.Q1.Q0' = Q1.(Q2 + Q0)'
- ET4 = Q2'.Q0 + Q2.Q1
- ET5 = (Q2 + Q1 + Q0)'
- ET6 = Q2'.Q1.Q0' = Q1.(Q2 + Q0)'
- DONE = Q2.Q1'.Q0' = Q2.(Q1 + Q0)'

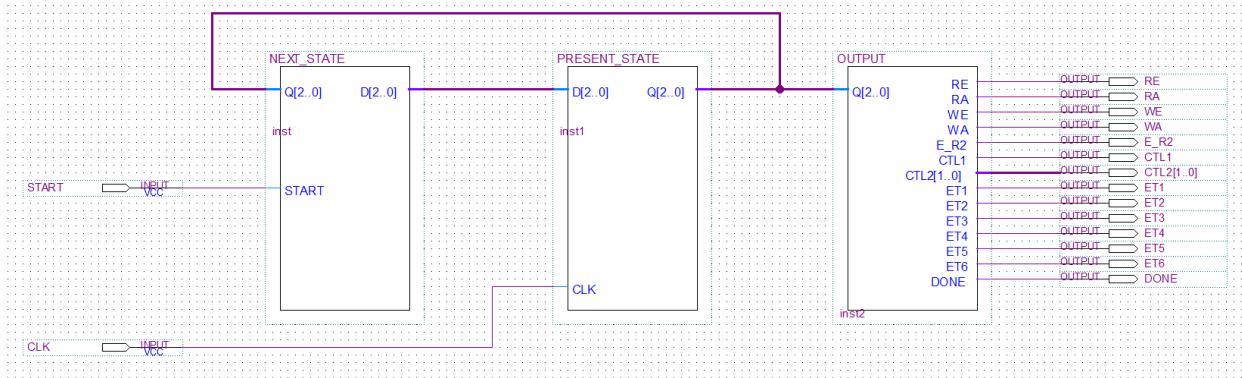
b. Vẽ mạch khôi ngắn ra



Hình 3.14: Mạch khôi ngắn ra

7. TỔNG HỢP THIẾT KẾ controller

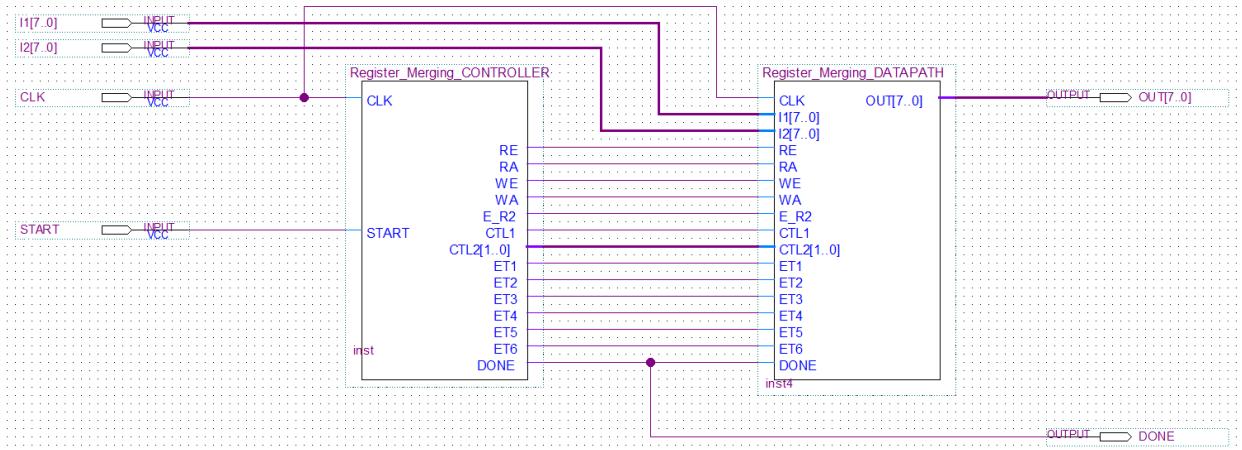
a. Vẽ mạch khối controller



Hình 3.15: Mạch tổng quát khối controller

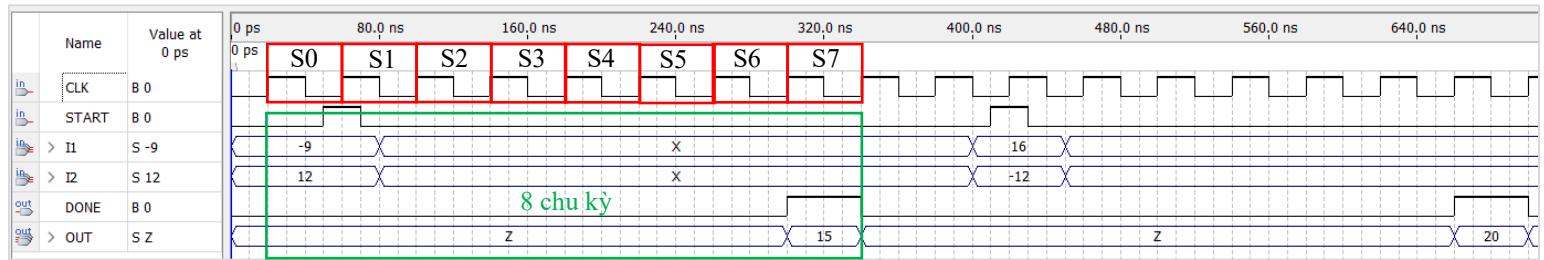
IV. TỔNG HỢP THIẾT KẾ VÀ CHẠY MÔ PHỎNG

1. Tổng hợp thiết kế



Hình 3.16: Tổng hợp thiết kế Register Merging

2. Mô phỏng và phân tích

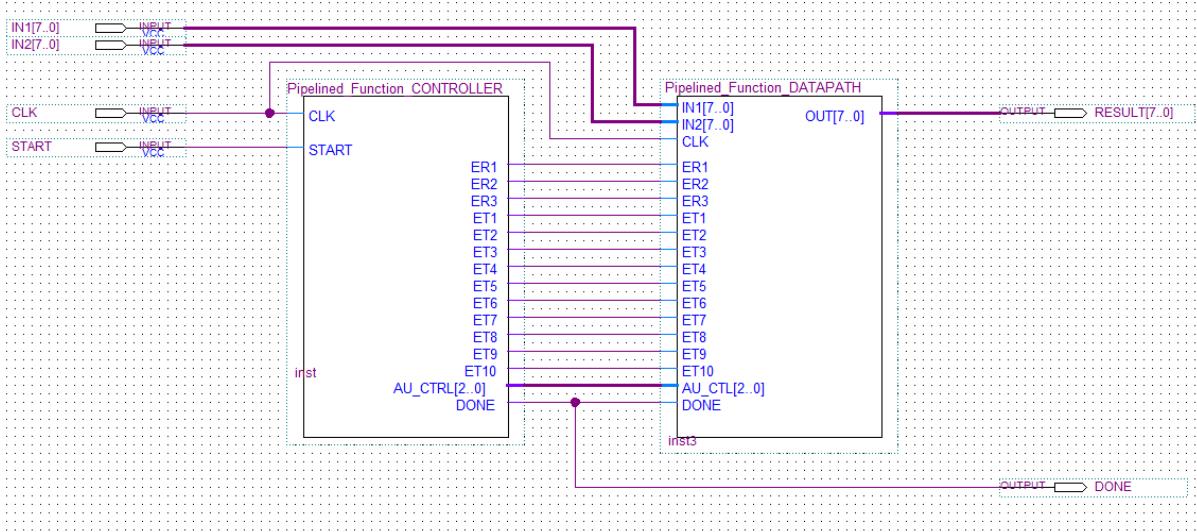


Hình 3.17: Kết quả mô phỏng theo thiết kế Register Merging

- Các trường hợp thử nghiệm là:
 - Số A âm, số B dương: $\sqrt{(-9)^2 + 12^2} = 15$
 - Số A dương, số B âm: $\sqrt{16^2 + (-12)^2} = 20$
- Sau khi tín hiệu START = 1, trạng thái sẽ chuyển từ S0 sang S1 khi có xung clock kích cạnh lên, đồng thời các tín hiệu điều khiển từ khói controller sẽ được tính toán và chuyển qua khói đường dữ liệu để thực thi các lệnh theo yêu cầu. Đến khi xung clock kích cạnh lên lần thứ 7 (tính từ khi START = 1), lúc này trạng thái hiện tại là S7, tín hiệu báo DONE = 1 và đồng thời giá trị phép tính sẽ được xuất ra ngoài ra OUT. Tổng thời gian thực thi của thiết kế này mất 8 chu kỳ xung clock để hoàn thành.

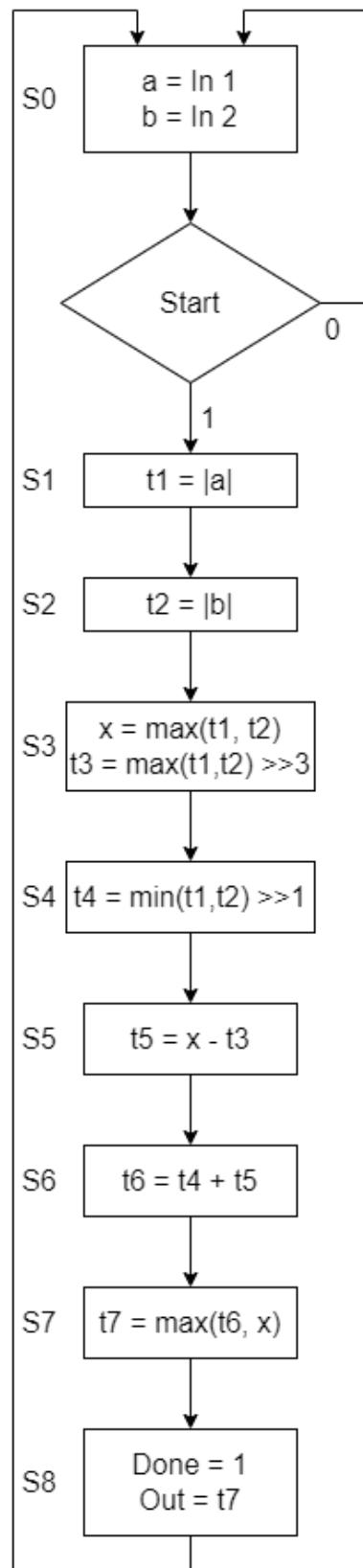
BÀI TẬP 4. PIPELINED FUNCTIONAL UNIT

I. TỔNG QUÁT THIẾT KẾ



Hình 4.1: Tổng quát thiết kế Pipelined Functional Unit

Mục tiêu của bài toán Pipelined Functional Unit nhằm làm tăng tốc độ xử lý khối đường dữ liệu thông qua việc pipeline khối chức năng. Bài toán này sẽ chia các toán hạng thành hai tập tương ứng với hai tầng trong khối chức năng AU. Thời gian để có được kết quả ngõ ra sẽ là hai chu kỳ xung clock. Trong chu kỳ xung clock lần đầu tiên, tầng đầu của khối chức năng sẽ thực thi tạo ra kết quả ngõ ra và lưu vào lại trong mạch chốt; trong chu kỳ xung clock thứ hai, tầng thứ hai của khối chức năng sẽ tiếp tục thực thi tạo ra kết quả cuối cùng, cũng đồng thời trong chu kỳ xung clock thứ hai này, tầng đầu tiên của khối chức năng cũng tiếp tục thực thi tạo ra kết quả ngõ ra cho một tập ngõ vào kế tiếp.



Hình 4.2: Biểu đồ ASM cho bài toán tính xấp xỉ căn bậc hai theo thiết kế AU không pipeline

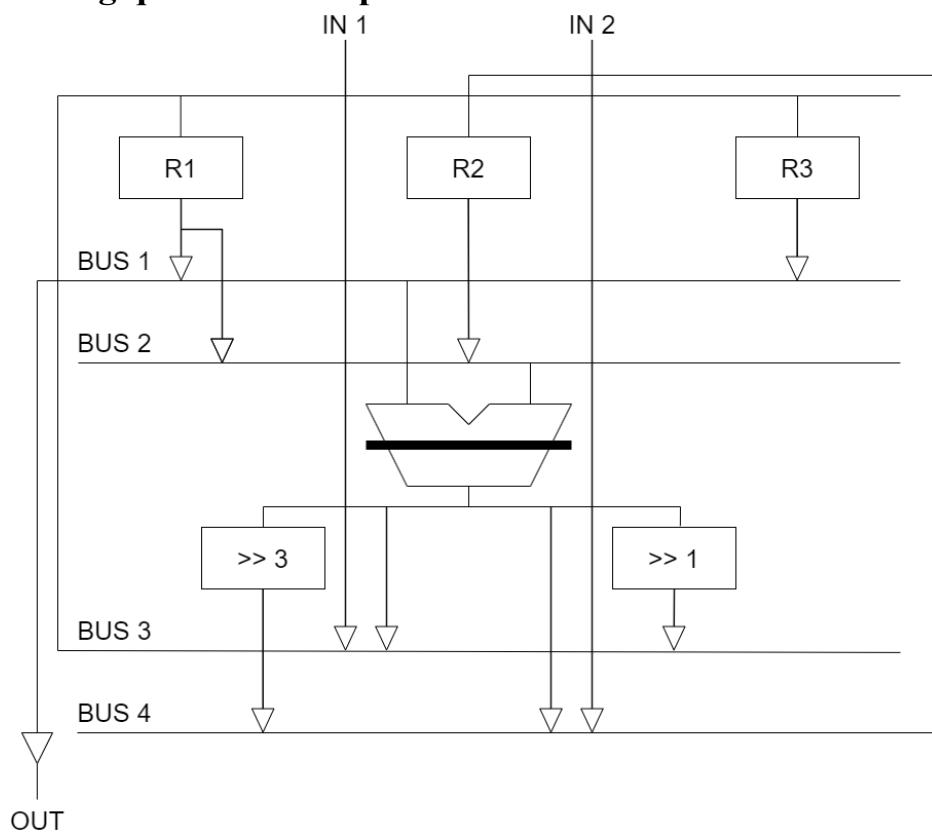
Thông qua biểu đồ ASM thiết kế AU không pipeline được minh họa qua hình 4.2, chúng ta có thể thấy mất 9 chu kỳ xung clock để tính được một giá trị SRA. Mặt khác, chúng ta có thể thiết kế lại khối đường dữ liệu bằng cách thay thế khối AU không pipeline thành khối AU pipeline hai tầng như hình 4.3. Khối đường dữ liệu mới với khối AU pipeline này cần 13 chu kỳ xung clock để tính được giá trị SRA. Các bước tính toán của từng trạng thái của khối mới này được miêu tả thông qua bảng giản đồ thời gian sau:

Bảng 4.1: Bảng giản đồ thời gian với các khối chức năng được pipeline

CV\TT	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12
Đọc R1		a			t_1	t_1	x				x		t_7
Đọc R2			b		t_2	t_2	t_3		t_5		t_6		
Đọc R3									t_4				
AU tầng 1		a	b		max	min	-		+		max		
AU tầng 2			a	b		max	min	-		+		max	
Dịch bit						>>3	>>1						
Ghi vào R1	a		t_1			x						t_7	
Ghi vào R2	b			t_2		t_3		t_5		t_6			
Ghi vào R3							t_4						
Xuất ngoã ra													t_7

II. THIẾT KẾ KHỐI DATAPATH

1. Mô hình tổng quát khối datapath



Hình 4.3: Sơ đồ thiết kế khối đường dữ liệu thực thi Pipeline AU

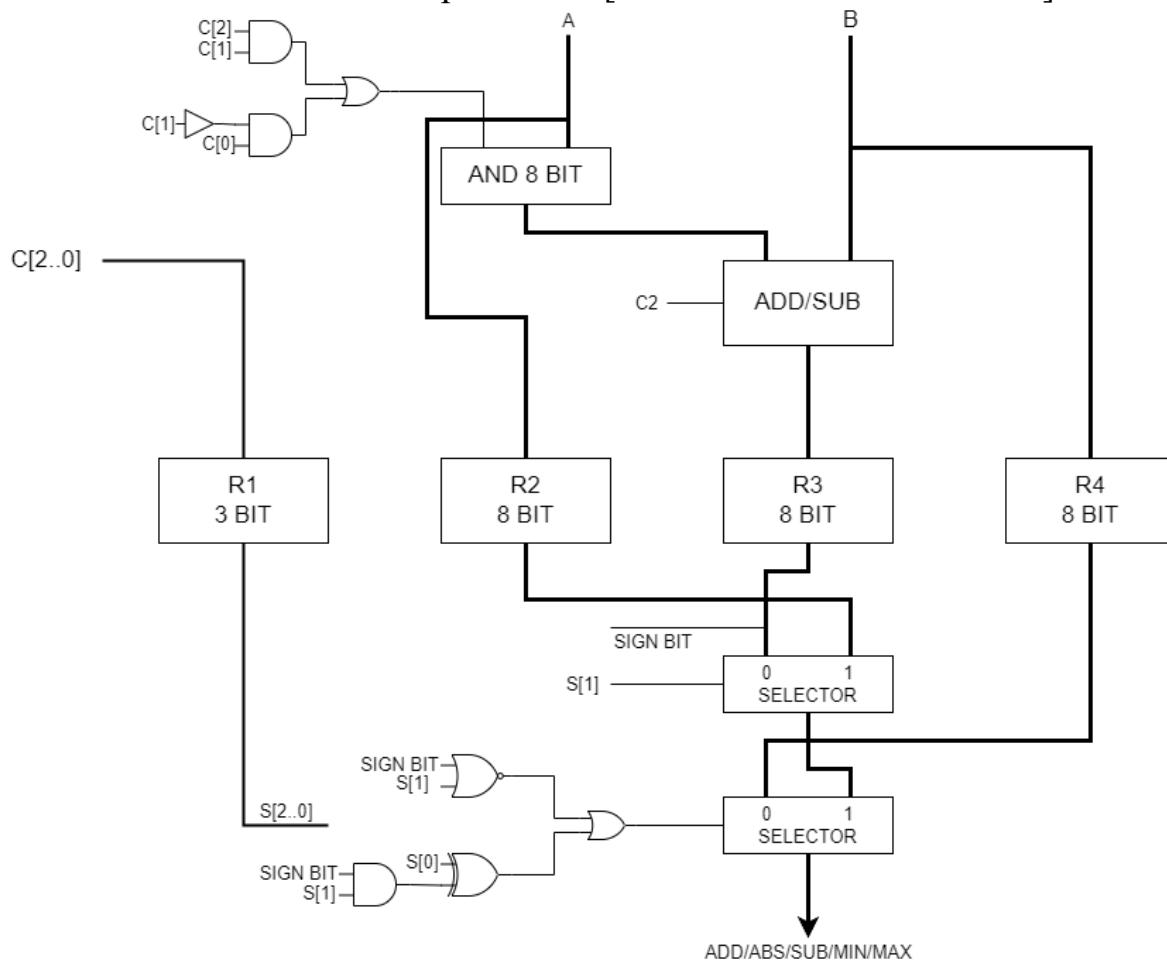
2. Phân tích các thành phần trong khối datapath

Datapath trong bài toán này gồm các thành phần sau:

- Pipeline AU 2 tầng với các chức năng: ADD/ABS/SUB/MIN/MAX.
- Register: R1, R2, R3 (tái sử dụng thiết kế trước).
- Khối dịch phải 1 bit (tái sử dụng thiết kế trước).
- Khối dịch phải 3 bit (tái sử dụng thiết kế trước).
- Tri-State (tái sử dụng thiết kế trước).

3. Thiết kế khối Pipeline AU [ADD/ABS/SUB/MIN/MAX]

a. Phân tích thiết kế khối Pipeline AU [ADD/ABS/SUB/MIN/MAX]



Hình 4.4: Sơ đồ khối AU Pipeline [ADD/ABS/SUB/MIN/MAX]

Ta thực hiện đặt mạch "Chốt" ở giữa AU và các mạch chọn. Ta có thể chọn đặt chốt ngay giữa AU để có chia khối AU sao cho thời gian thực thi mỗi tầng đúng bằng một nửa so với khi thực thi toàn khối AU. Nhưng ở đây để tránh làm bài toán trở nên phức tạp, ta thực hiện chia tầng như bên trên, đổi lại thời gian thực thi của mỗi tầng chỉ "tương đối bằng nhau".

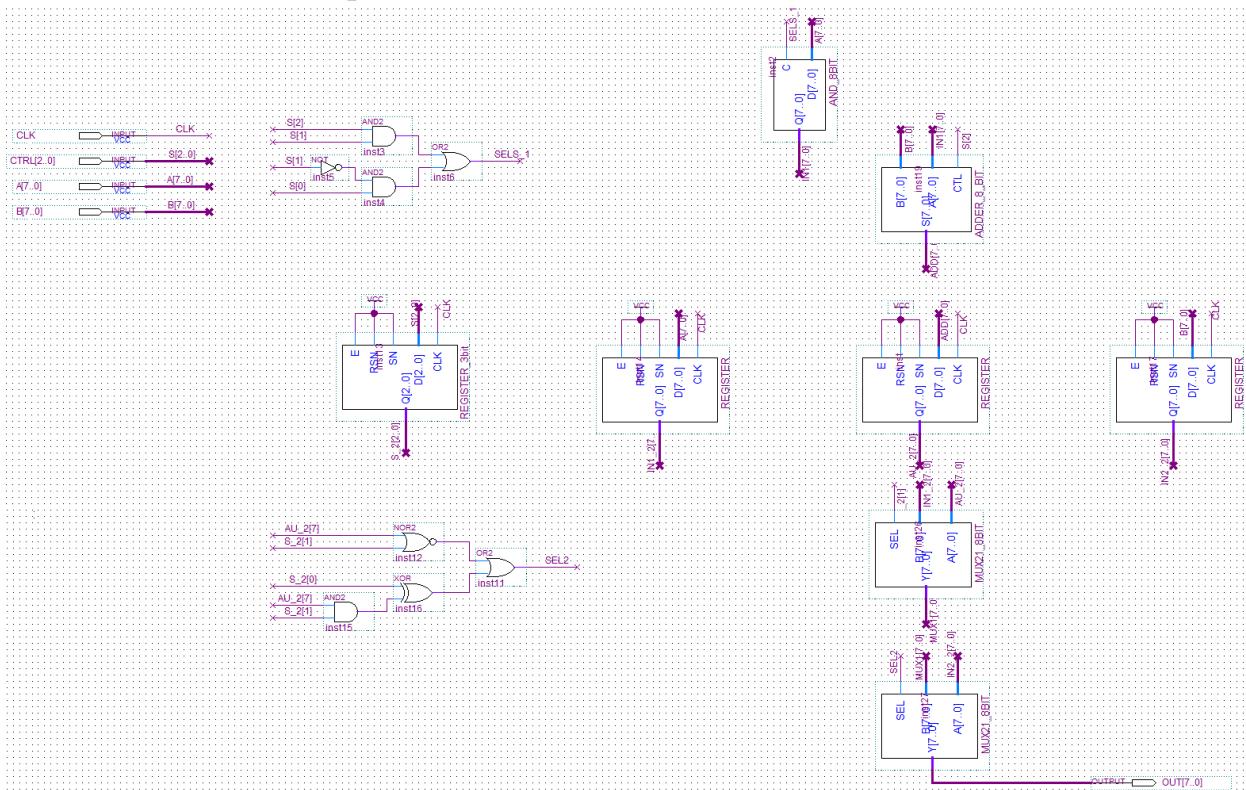
Khối AU pipeline có 5 chức năng khác nhau nên cần có 3 bit để phục vụ cho tín hiệu điều khiển C[2..0]. Thiết kế này sử dụng 4 thanh ghi làm nhiệm vụ như là một mạch chốt phục vụ việc lưu giá trị của dữ liệu. Trong đó thanh ghi R1 là một mạch chốt tín hiệu điều khiển và 3 thanh ghi R2, R3, R4 là các mạch chốt tín hiệu dữ liệu.

b. Bảng sự thật các chức năng khối AU

Bảng 4.2: Bảng sự thật các chức năng khối AU

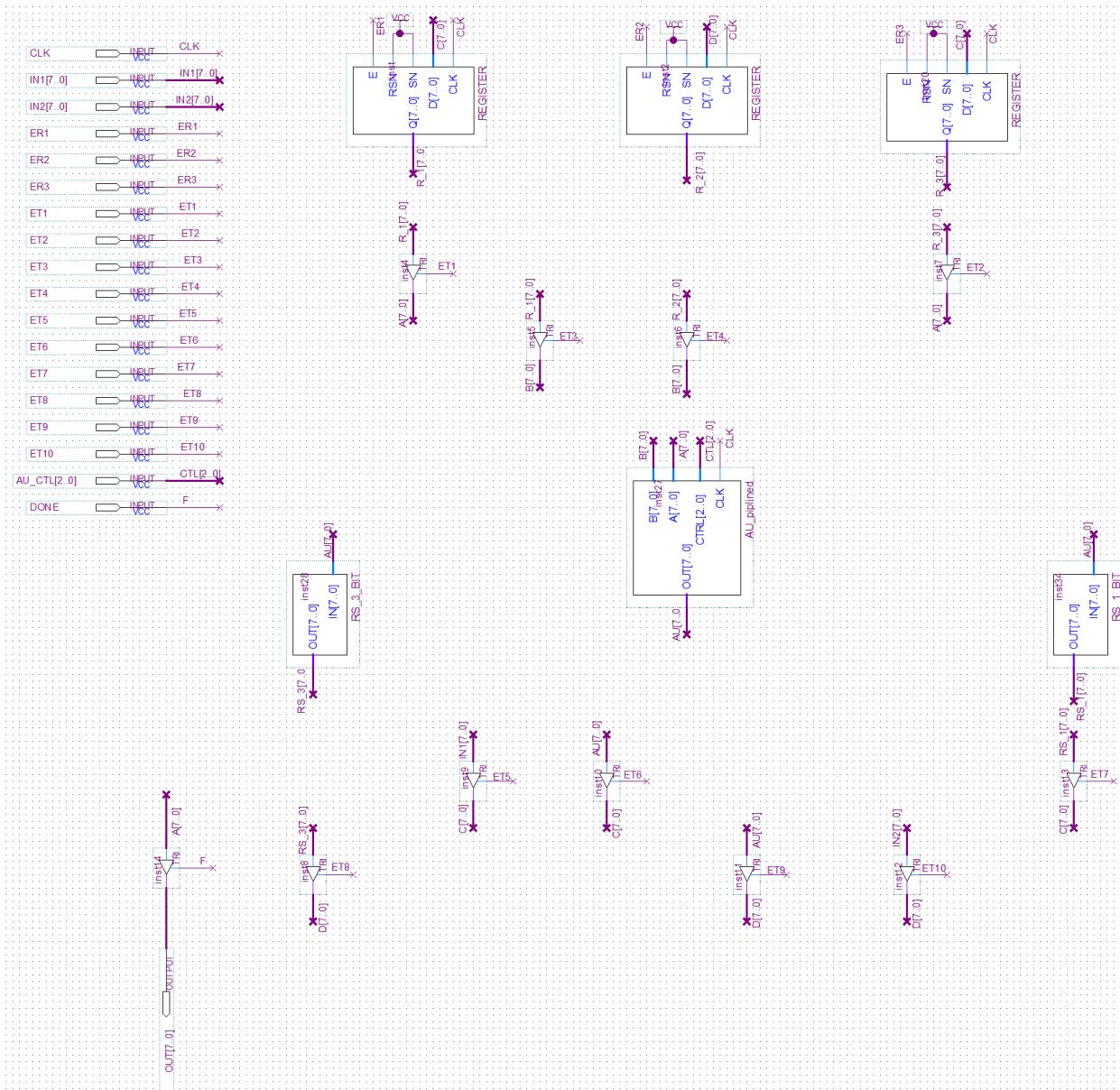
C2	C1	C0	Chức năng
0	0	1	ADD
1	0	0	ABS
1	0	1	SUB
1	1	0	MIN
1	1	1	MAX

c. Vẽ mạch khối Pipeline AU



Hình 4.5: Mạch khối Pipeline AU

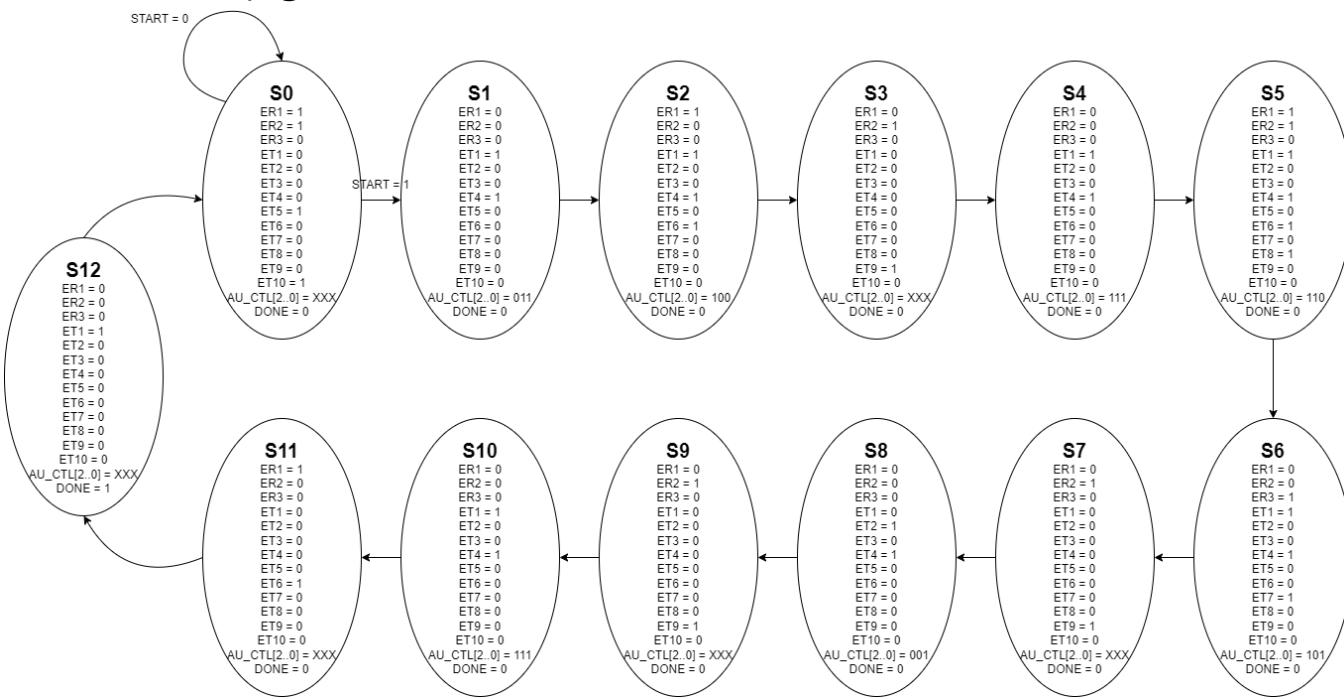
4. Tổng hợp thiết kế datapath



Hình 4.6: Tổng hợp thiết kế khối datapath

III. THIẾT KẾ KHÓI CONTROLLER

1. Sơ đồ trạng thái



Hình 4.7: Sơ đồ trạng thái theo thiết kế Pipelined Functional Unit

Các tín hiệu được mô tả như sau:

- ER1: cho phép ghi hoặc không vào thanh ghi R1.
- ER2: cho phép ghi hoặc không vào thanh ghi R2.
- ER3: cho phép ghi hoặc không vào thanh ghi R3.
- ET1: cho phép truyền hoặc không truyền dữ liệu từ R1 xuống BUS 1.
- ET2: cho phép truyền hoặc không truyền dữ liệu từ R3 xuống BUS 1.
- ET3: cho phép truyền hoặc không truyền dữ liệu từ R1 xuống BUS 2.
- ET4: cho phép truyền hoặc không truyền dữ liệu từ R2 xuống BUS 2.
- ET5: cho phép truyền hoặc không truyền dữ liệu từ IN1 xuống BUS 3.
- ET6: cho phép truyền hoặc không truyền dữ liệu từ khói pipeline AU xuống BUS 3.
- ET7: cho phép truyền hoặc không truyền dữ liệu từ khói >>1 xuống BUS 3.
- ET8: cho phép truyền hoặc không truyền dữ liệu từ khói >>3 xuống BUS 4.
- ET9: cho phép truyền hoặc không truyền dữ liệu từ khói pipeline AU xuống BUS 4.
- ET10: cho phép truyền hoặc không truyền dữ liệu từ IN1 xuống BUS 4.
- AU_CTL[2..0]: chọn chế độ thực thi cho khói pipeline AU.
- DONE: tín hiệu hoàn thành.

2. Mã hóa trạng thái

Bảng 4.3: Bảng gán trạng thái theo bìa K-Map

F		Q1Q0			
		00	01	11	10
Q3Q2	00	S0	S1	S2	S3
	01	S7	S6	S5	S4
	11	S8	S9	S10	S11
	10				S12

Bảng 4.4: Bảng mã hóa trạng thái

Trạng thái	Mã hóa
S0	0000
S1	0001
S2	0011
S3	0010
S4	0110
S5	0111
S6	0101
S7	0100
S8	1100
S9	1101
S10	1111
S11	1110
S12	1010

3. Chọn loại flipflop và lập bảng chuyển trạng thái

Bảng 4.5: Bảng chuyển trạng thái

TTHT	START	
	0	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	S12	S12
S12	S0	S0

Thiết kế này sử dụng các flipflop D làm nhiệm vụ lưu trữ các trạng thái.

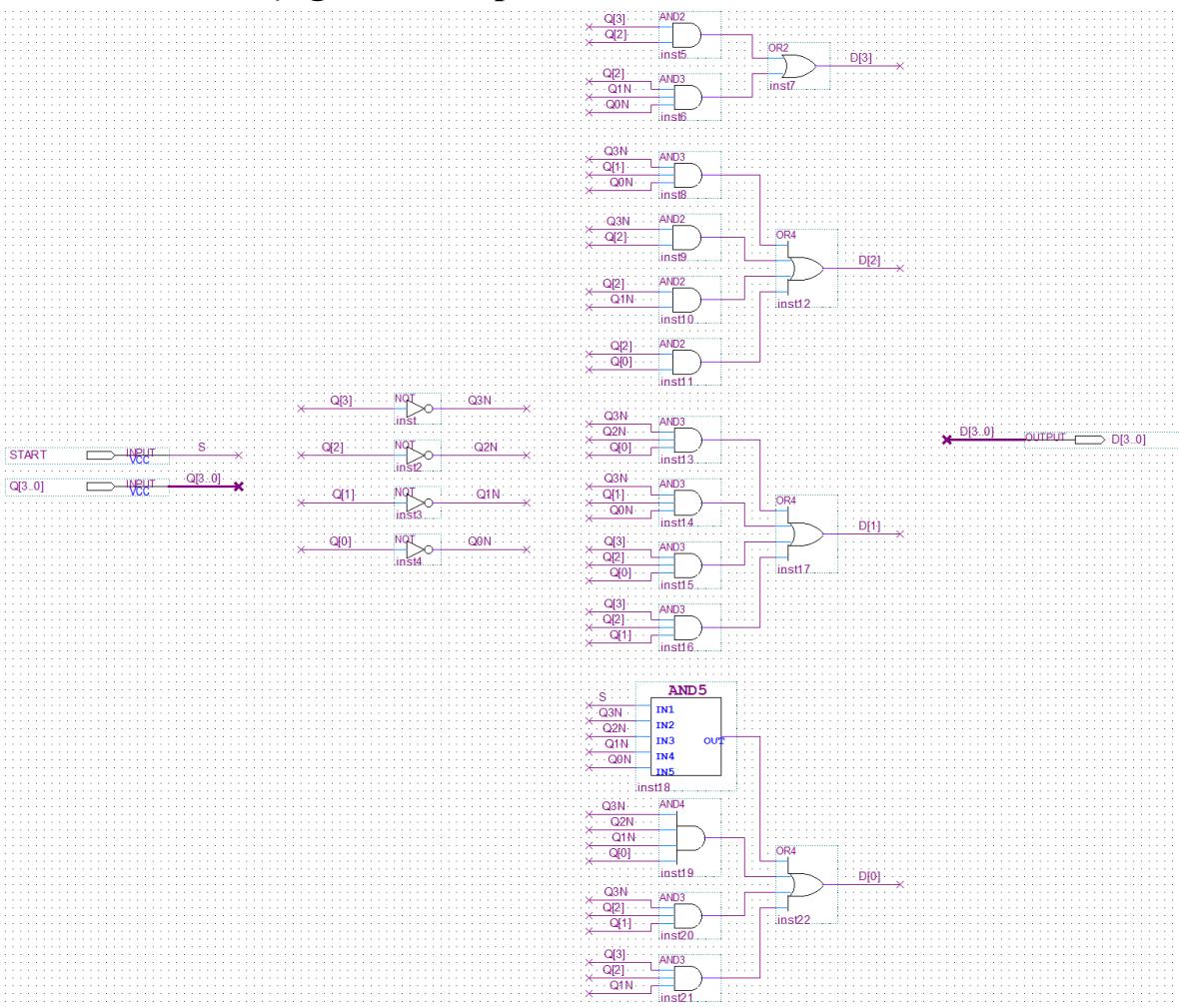
Từ bảng chuyển trạng thái, ta lập các phương trình ngõ vào các flipflop như sau:
 $D_3 = Q_3.Q_2 + Q_2.Q_1'.Q_0'$

$$D_2 = Q_3'.Q_1.Q_0' + Q_3'.Q_2 + Q_2.Q_1' + Q_2.Q_0$$

$$D_1 = Q_3'.Q_2'.Q_0 + Q_3'.Q_1.Q_0' + Q_3.Q_2.Q_0 + Q_3.Q_2.Q_1$$

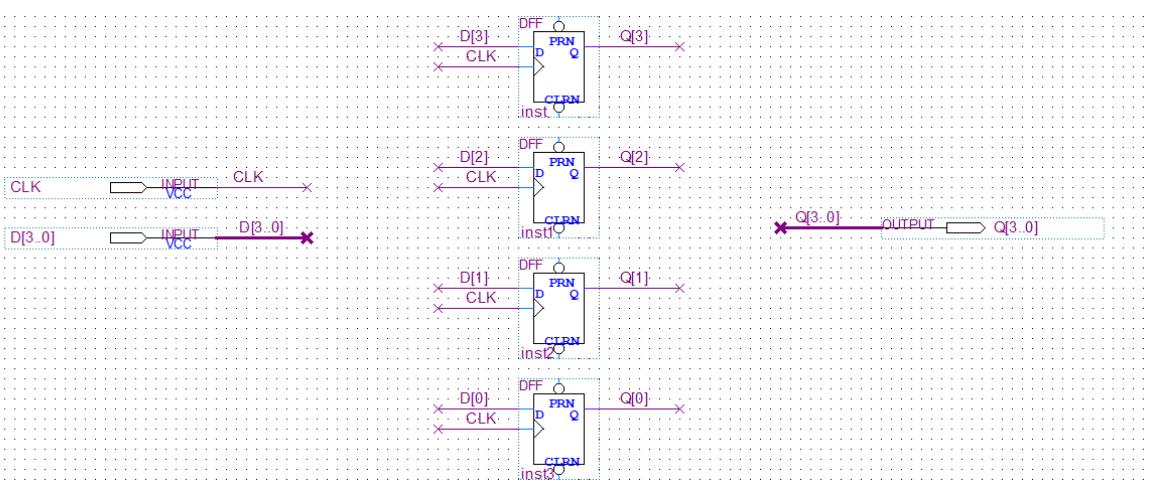
$$D_0 = \text{START}.Q_3'.Q_2'.Q_1'.Q_0' + Q_3'.Q_2'.Q_1'.Q_0 + Q_3'.Q_2.Q_1 + Q_3.Q_2.Q_1'$$

4. Thiết kế khối trạng thái kế tiếp



Hình 4.8: Mạch khối trạng thái kế tiếp

5. Thiết kế khối nhớ



Hình 4.9: Mạch khối nhớ

6. Thiết kế khối ngõ ra

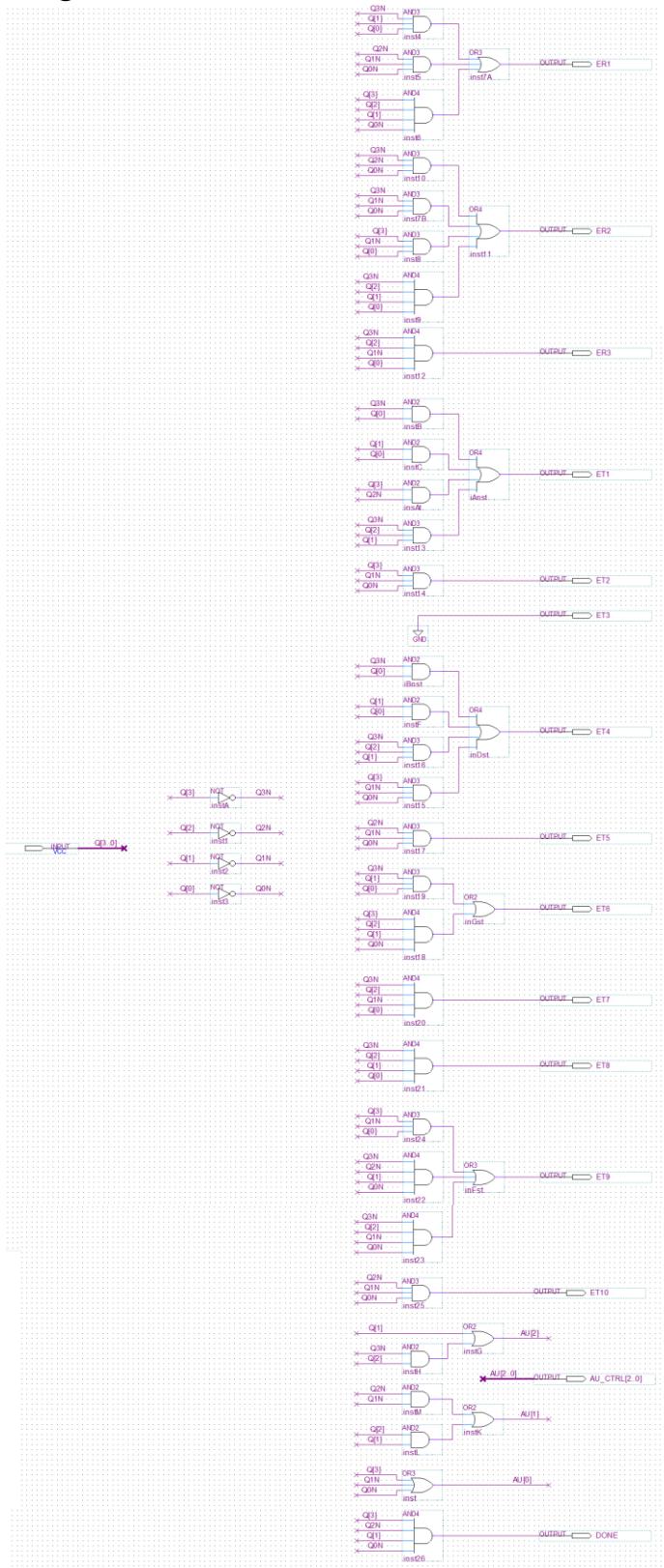
a. Bảng sự thật các ngõ ra

Bảng 4.6: Bảng sự thật các ngõ ra

TH\TT	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12
ER1	1	0	1	0	0	1	0	0	0	0	0	1	0
ER2	1	0	0	1	0	1	0	1	0	1	0	0	0
ER3	0	0	0	0	0	0	1	0	0	0	0	0	0
ET1	0	1	1	0	1	1	1	0	0	0	1	0	1
ET2	0	0	0	0	0	0	0	0	1	0	0	0	0
ET3	0	1	0	0	0	0	0	0	0	0	0	0	0
ET4	0	0	1	0	1	1	1	0	1	0	1	0	0
ET5	1	0	0	0	0	0	0	0	0	0	0	0	0
ET6	0	0	1	0	0	1	0	0	0	0	0	1	0
ET7	0	0	0	0	0	0	1	0	0	0	0	0	0
ET8	0	0	0	0	0	1	0	0	0	0	0	0	0
ET9	0	0	0	1	0	0	0	1	0	1	0	0	0
ET10	1	0	0	0	0	0	0	0	0	0	0	0	0
AU[2]	X	1	1	X	1	1	1	X	0	X	1	X	X
AU[1]	X	0	0	X	1	1	0	X	0	X	1	X	X
AU[0]	X	0	0	X	1	0	1	X	1	X	1	X	X
DONE	0	0	0	0	0	0	0	0	0	0	0	0	1

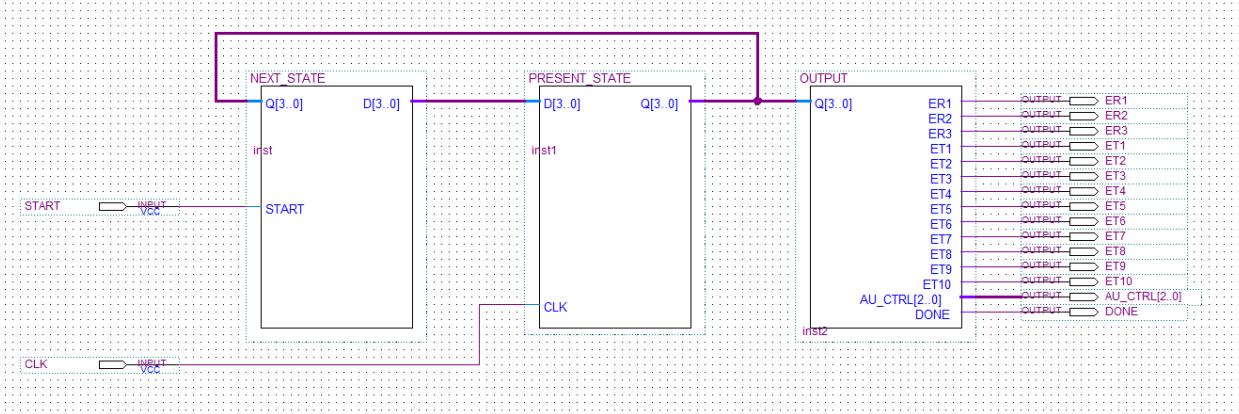
- $ER1 = Q3'.Q1.Q0 + Q2'.Q1'.Q0' + Q3.Q2.Q1.Q0'$
- $ER2 = Q3'.Q2'.Q0' + Q3'.Q1'.Q0' + Q3.Q1'.Q0 + Q3'.Q2.Q1.Q0$
- $ER3 = Q3'.Q2.Q1'.Q0$
- $ET1 = Q3'.Q0 + Q1.Q0 + Q3.Q2' + Q3'.Q2.Q1$
- $ET2 = Q3.Q1'.Q0'$
- $ET3 = Q2'.Q1'.Q0$
- $ET4 = Q3'.Q2.Q0 + Q1.Q0 + Q3'.Q2.Q1 + Q3.Q1'.Q0'$
- $ET5 = Q2'.Q1'.Q0'$
- $ET6 = Q3'.Q1.Q0 + Q3.Q2.Q1.Q0'$
- $ET7 = Q3'.Q2.Q1'.Q0$
- $ET8 = Q3'.Q2.Q1.Q0$
- $ET9 = Q3.Q1'.Q0 + Q3'.Q2'.Q1.Q0' + Q3'.Q2.Q1'.Q0'$
- $ET10 = Q2'.Q1'.Q0'$
- $AU[2] = Q0 + Q3'$
- $AU[1] = Q2.Q1$
- $AU[0] = Q3 + Q0' + Q2.Q1'$
- $DONE = Q3.Q2'.Q1.Q0'$

b. Vẽ mạch khối ngõ ra



Hình 4.10: Mạch khối ngõ ra

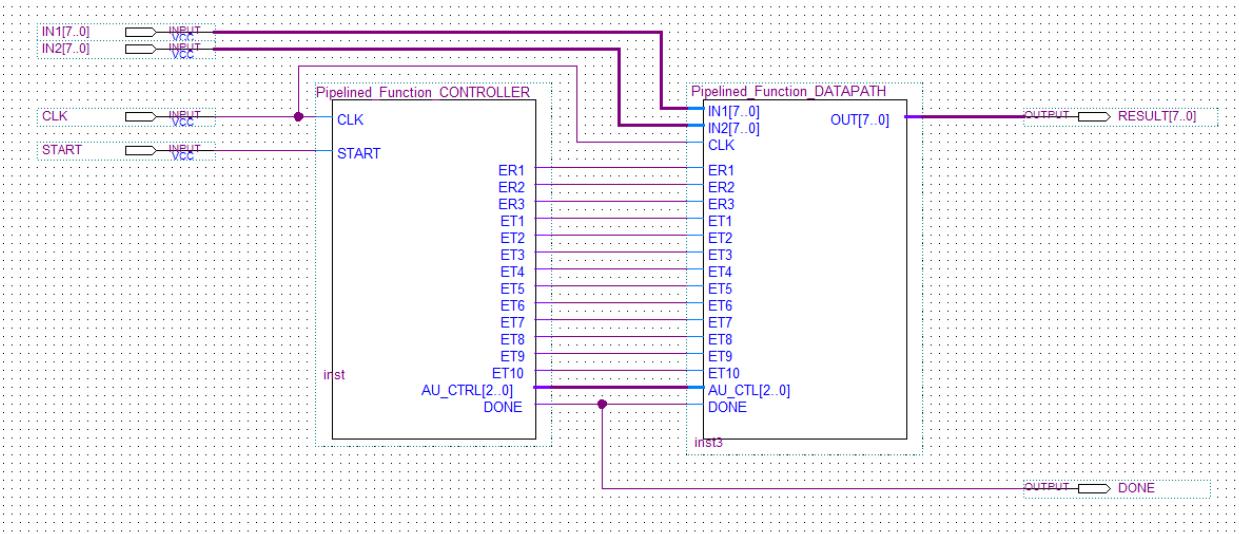
7. Tổng hợp thiết kế controller



Hình 4.11: Tổng hợp thiết kế controller

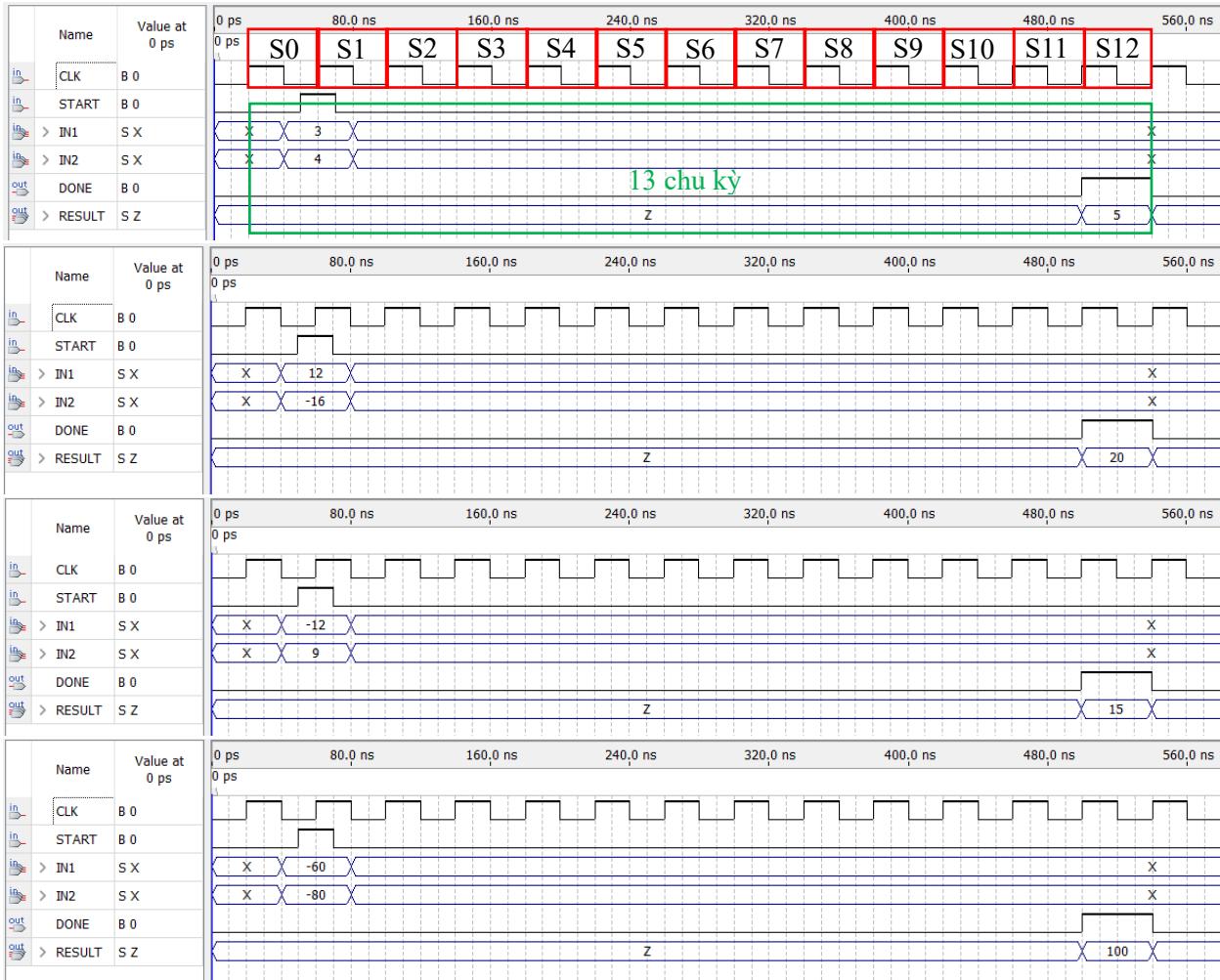
IV. TỔNG HỢP THIẾT KẾ VÀ MÔ PHỎNG

1. Tổng hợp thiết kế



Hình 4.12: Tổng hợp thiết kế Pipelined Functional Unit

2. Mô phỏng và phân tích



Hình 4.13: Kết quả mô phỏng theo thiết kế Pipelined Functional Unit

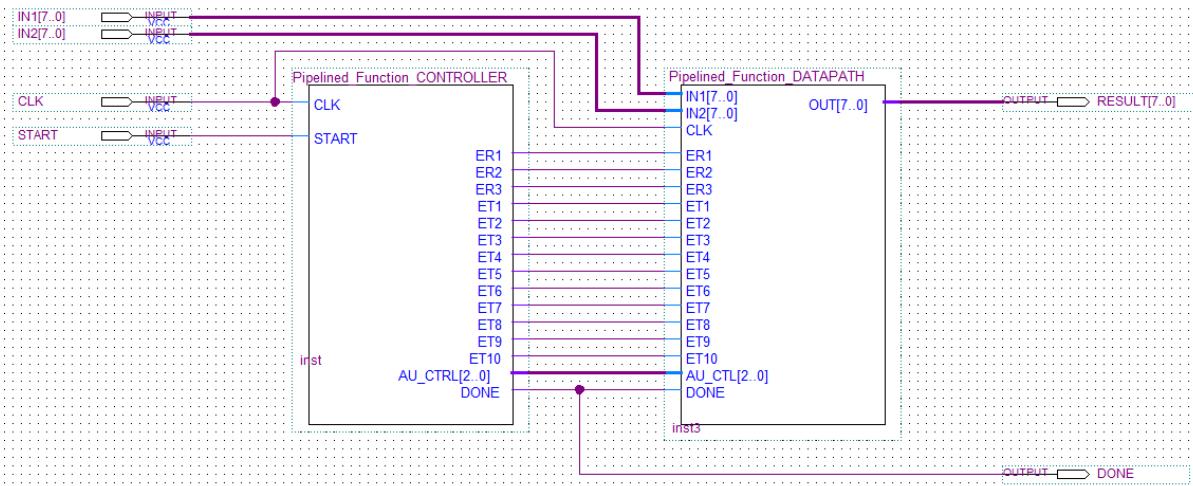
Các trường hợp thử nghiệm là:

- Số A dương, số B dương: $\sqrt{3^2 + 4^2} = 5$
- Số A dương, số B âm: $\sqrt{12^2 + (-16)^2} = 20$
- Số A âm, số B dương: $\sqrt{(-12)^2 + 9^2} = 15$
- Số A âm, số B âm: $\sqrt{(-60)^2 + (-80)^2} = 100$

Đối với việc thực hiện khối datapath với khối AU không pipelined thì cần 9 chu kỳ để thực hiện một phép tính, trong khi datapath với khối AU có pipeline cần 13 chu kỳ để thực hiện tính toán một phép tính. Tuy nhiên, vì đã chia đôi khối AU nên một chu kỳ khi thực hiện pipelined AU chỉ bằng 1/2 chu kỳ khi thực hiện không pipelined AU. Vì vậy, thực chất ta chỉ tiêu tốn 6.5 chu kỳ khi AU pipeline so với 9 chu kỳ ở AU không pipeline.

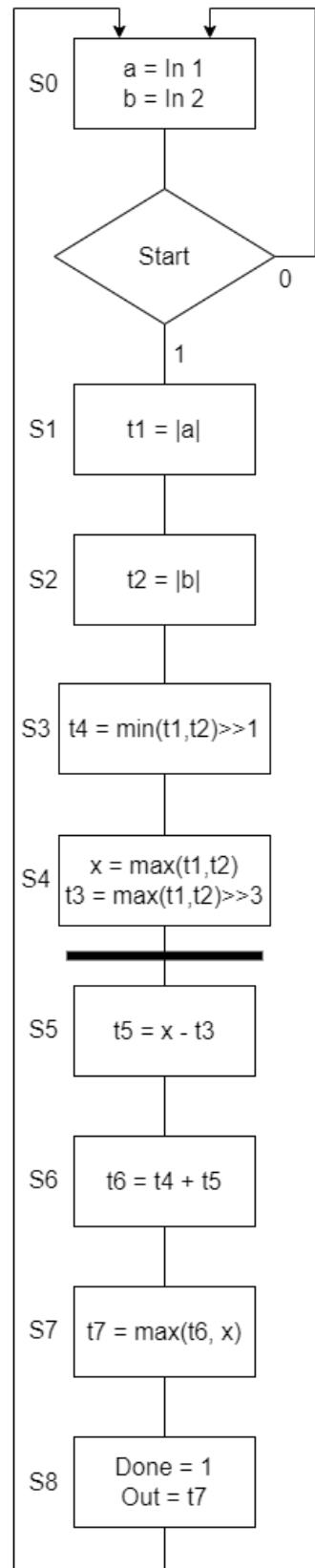
BÀI TẬP 5. DATAPATH PIPELINING

I. TỔNG QUÁT THIẾT KẾ



Hình 5.1: Tổng quát thiết kế Datapath Pipelining

Mục tiêu của thiết kế pipeline đường dữ liệu là cải thiện tốc độ xử lý bằng cách thiết kế pipeline toàn bộ biểu đồ ASM thông qua việc chia toàn bộ biểu đồ ASM ra thành nhiều phần có kích thước bằng nhau và sau đó sử dụng các tầng đường dữ liệu khác nhau để thực thi từng phần đã chia. Với thiết kế này, tất cả các tầng có thể thực thi các tập toán hạng ngõ vào một cách đồng thời, mỗi tầng tạo ra một phần kết quả và kết quả này được sử dụng bởi các tầng đường dữ liệu kế tiếp.



Hình 5.2: Biểu đồ ASM cho bài toán tính xác suất căn bậc hai theo thiết kế Datapath Pipelining

Giản đồ thời gian cho hiện thực pipeline đường dữ liệu được miêu tả qua bảng sau:

Bảng 5.1: Giản đồ thời gian cho việc thực thi pipeline đường dữ liệu

CV/TT	Xử lý cho tập ngõ vào thứ n^{th}					Xử lý cho tập ngõ vào thứ $(n+1)^{\text{th}}$				
	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9
Đọc R1		a		t_1	t_1					
Đọc R2			b	t_2	t_2					
AU tầng 1		$ a $	$ b $	min	max					
Dịch bit				$>>1$	$>>3$					
Ghi vào R1	a	t_1								
Ghi vào R2	b		t_2							
Đọc R3						t_3	t_5	t_6	t_7	
Đọc R4						x		x		
Đọc R5							t_4			
AU tầng 2						-	+	max		
Ghi vào R3				t_3	t_5		t_6	t_7		
Ghi vào R4				x						
Ghi vào R5				t_4						

Tầng khối dữ liệu đầu tiên gồm 2 thanh ghi R1, R2, một khối AU và 2 khối dịch. Tầng khối dữ liệu thứ 2 gồm 3 thanh ghi R3, R4, R5 và một khối AU khác. Cụ thể các chức năng như sau:

Tầng 1:

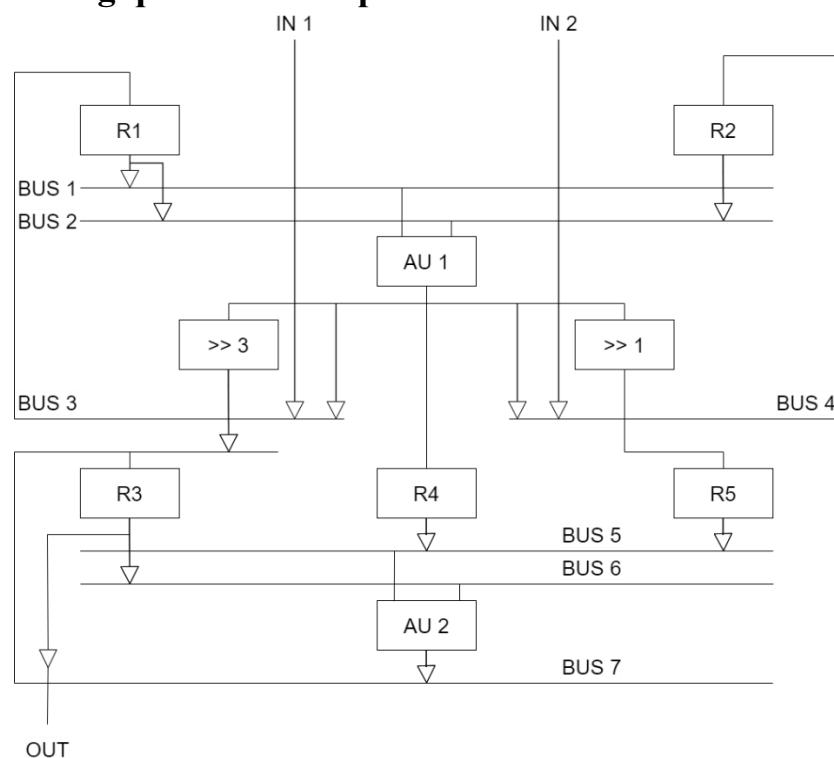
- $R1 = [a, t1]$
- $R2 = [b, t2]$
- $AU1 = [\text{ABS/MIN/MAX}]$
- $>>3$
- $>>1$

Tầng 2:

- $R3 = [t3, t5, t6, t7]$
- $R4 = [x]$
- $R5 = [t4]$
- $AU2 = [\text{ADD/SUB/MAX}]$

II. THIẾT KẾ KHỐI DATAPATH

1. Mô hình tổng quát khối datapath



Hình 5.3: Sơ đồ thiết kế khối đường dữ liệu thực thi Datapath Pipelining

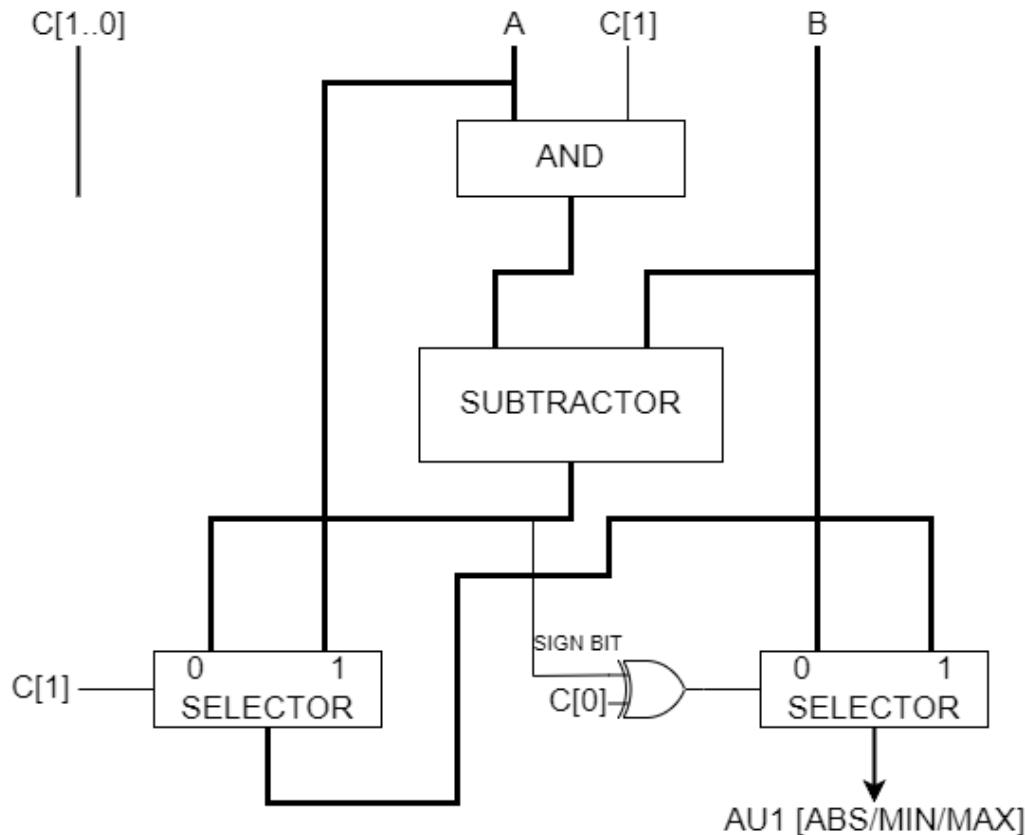
2. Phân tích các thành phần trong khối datapath

Khối datapath trong bài toán này gồm các thành phần sau:

- Register: R1, R2, R3, R4, R5 (tái sử dụng thiết kế trước đó)
- Khối dịch phải 1 bit (tái sử dụng thiết kế trước đó)
- Khối dịch phải 3 bit (tái sử dụng thiết kế trước đó)
- Khối AU 1 (ABS/MIN/MAX)
- Khối AU 2 (ADD/SUB/MAX)

3. Thiết kế khối AU 1 (ABS/MIN/MAX)

a. Phân tích thiết kế khối AU 1



Hình 5.4: Sơ đồ thiết kế khối AU1 [ABS/MIN/MAX]

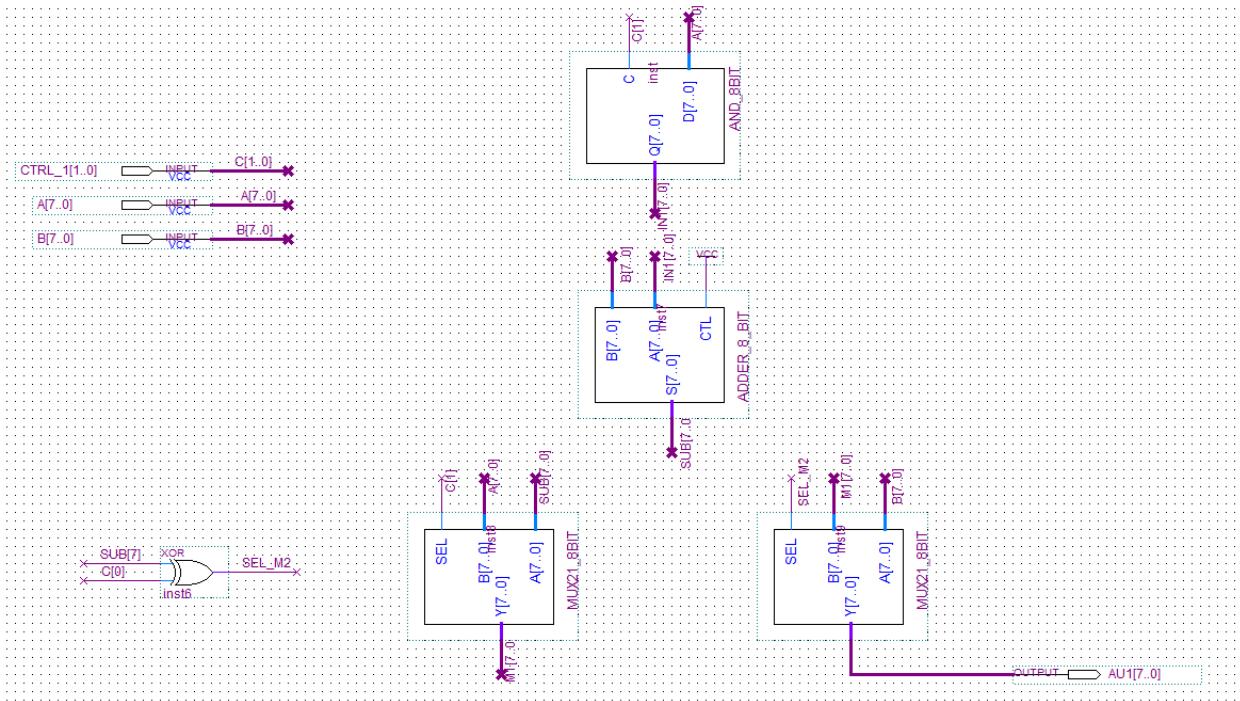
Khối AU1 có 3 chức năng khác nhau nên cần 2 bit tín hiệu điều khiển $C[1..0]$. Thiết kế này sử dụng một khối AND 8 bit, một khối trừ và 2 khối chọn. Các tín hiệu điều khiển $C[1..0]$ được mô tả chức năng cụ thể qua bảng 5.2.

b. Bảng sự thật các tín hiệu điều khiển khối AU 1

Bảng 5.2: Bảng sự thật các tín hiệu điều khiển khối AU 1

CTRL[1..0]	Chức năng
01	ABS
10	MIN
11	MAX

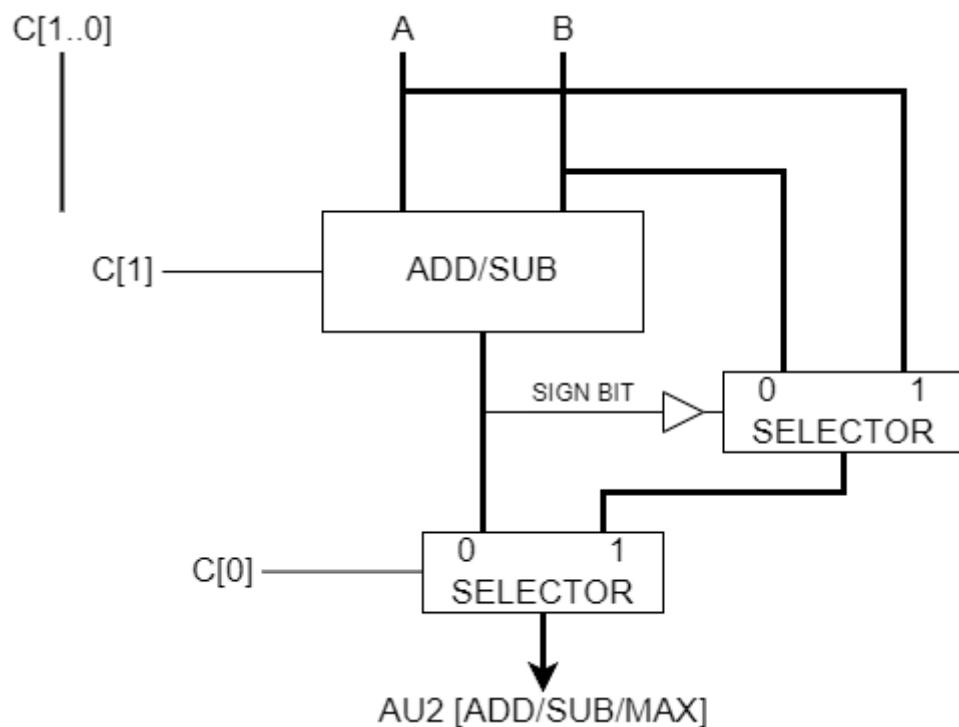
c. Vẽ mạch khối AU 1



Hình 5.5: Mạch khối AU 1

4. Thiết kế khối AU 2 (ADD/SUB/MAX)

a. Phân tích thiết kế khối AU 2



Hình 5.6: Sơ đồ thiết kế khối tính AU2 [ADD/SUB/MAX]

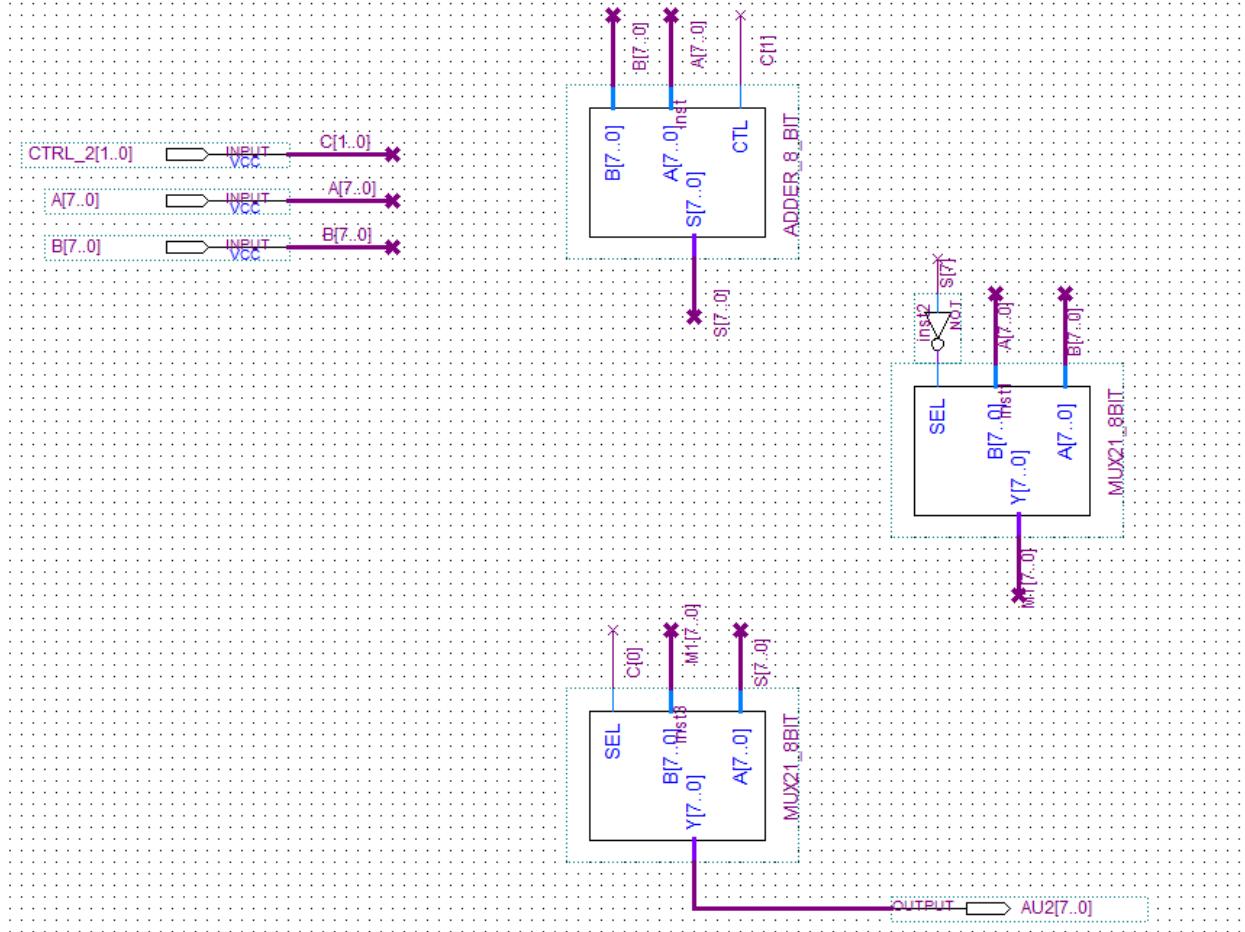
Khối AU2 có 3 chức năng khác nhau nên cần 2 bit tín hiệu điều khiển C[1..0]. Thiết kế này cần một khối tích hợp cộng/trừ và 2 bộ chọn. Các tín hiệu điều khiển C[1..0] được mô tả chức năng cụ thể qua bảng 5.3.

b. Bảng sự thật các tín hiệu điều khiển khối AU 2

Bảng 5.3: Bảng sự thật tín hiệu điều khiển khối AU 1

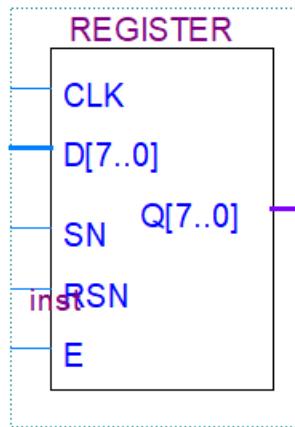
CTRL[1..0]	Chức năng
00	ADD
10	SUB
11	MAX

c. Vẽ mạch khối AU 2



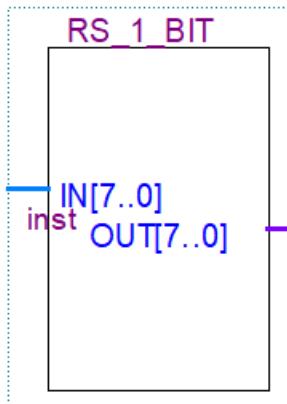
Hình 5.7: Mạch khối AU 2

5. Tái sử dụng các thanh ghi



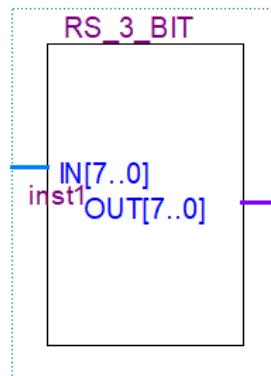
Hình 5.8: Thanh ghi 8 bit

6. Tái sử dụng khối dịch phải 1 bit



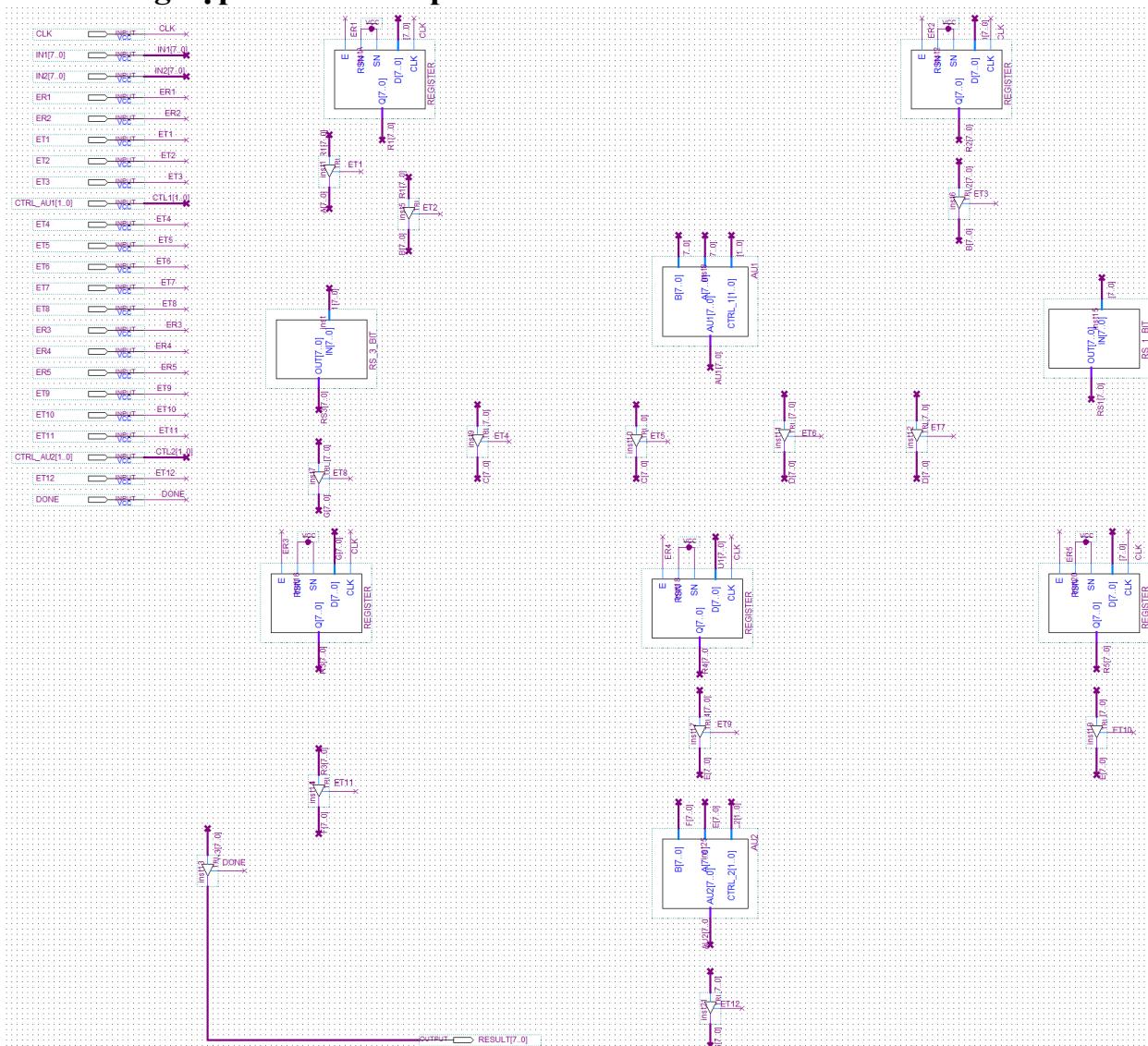
Hình 5.9: Khối dịch phải 1 bit

7. Tái sử dụng khối dịch phải 3 bit



Hình 5.10: Khối dịch phải 3 bit

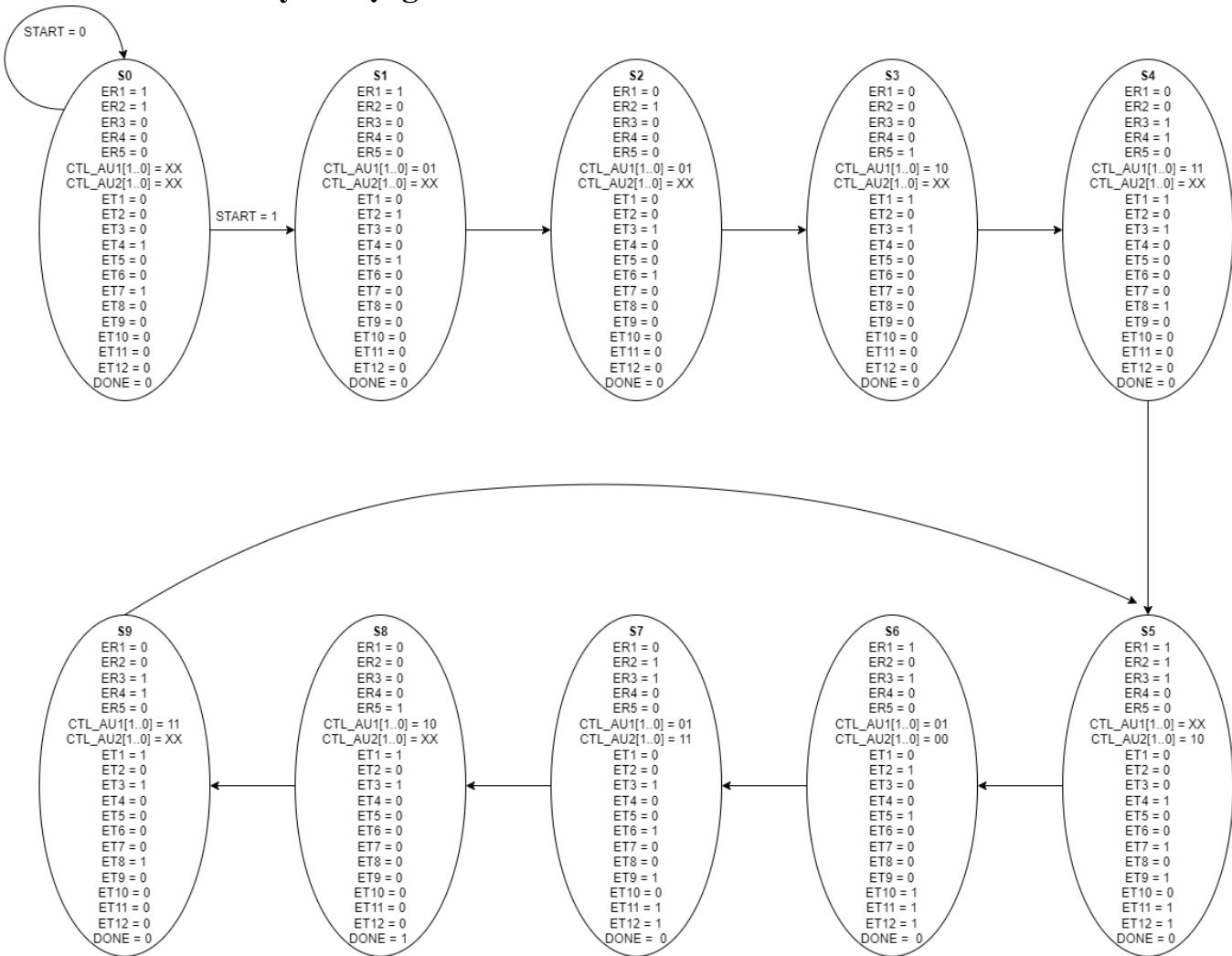
8. Tổng hợp thiết kế datapath



Hình 5.11: Tổng hợp datapath theo thiết kế Datapath Pipelining

III. THIẾT KẾ KHỐI CONTROLLER

1. Sơ đồ chuyển trạng thái



Hình 5.12: Sơ đồ chuyển trạng thái theo thiết kế Datapath Pipelining

Các tín hiệu được mô tả như sau:

- ER1: cho phép ghi hoặc không vào thanh ghi R1.
- ER2: cho phép ghi hoặc không vào thanh ghi R2.
- ER3: cho phép ghi hoặc không vào thanh ghi R3.
- ER4: cho phép ghi hoặc không vào thanh ghi R4.
- ER5: cho phép ghi hoặc không vào thanh ghi R5.
- CTL_AU1[1..0]: tín hiệu chọn chế độ cho khối AU 1.
- CTL_AU2[1..0]: tín hiệu chọn chế độ cho khối AU 2.
- ET1: cho phép hoặc cấm truyền dữ liệu từ R1 xuống BUS 1.
- ET2: cho phép hoặc cấm truyền dữ liệu từ R1 xuống BUS 2.
- ET3: cho phép hoặc cấm truyền dữ liệu từ R3 xuống BUS 2.
- ET4: cho phép hoặc cấm truyền dữ liệu từ IN 1 xuống BUS 3.
- ET5: cho phép hoặc cấm truyền dữ liệu từ AU 1 xuống BUS 3.
- ET6: cho phép hoặc cấm truyền dữ liệu từ AU 1 xuống BUS 4.
- ET7: cho phép hoặc cấm truyền dữ liệu từ IN 2 xuống BUS 4.
- ET8: cho phép hoặc cấm truyền dữ liệu từ >>3 xuống BUS 7.
- ET9: cho phép hoặc cấm truyền dữ liệu từ R4 xuống BUS 5.
- ET10: cho phép hoặc cấm truyền dữ liệu từ R5 xuống BUS 5.
- ET11: cho phép hoặc cấm truyền dữ liệu từ R3 xuống BUS 6.
- ET12: cho phép hoặc cấm truyền dữ liệu từ AU 2 xuống BUS 7.
- DONE: tín hiệu hoàn thành.

2. Mã hóa trạng thái

Bảng 5.4: Bảng gán trạng thái dựa trên bìa K-Map

F		Q1Q0			
		00	01	11	10
Q3Q2	00	S0	S1	S2	S3
	01	S7	S6	S5	S4
	11	S8	S9		
	10				

Bảng 5.5: Bảng mã hóa trạng thái

Trạng thái	Mã hóa
S0	0000
S1	0001
S2	0011
S3	0010
S4	0110
S5	0111
S6	0101
S7	0100
S8	1100
S9	1101

3. Chọn loại flipflop và lập bảng chuyển trạng thái

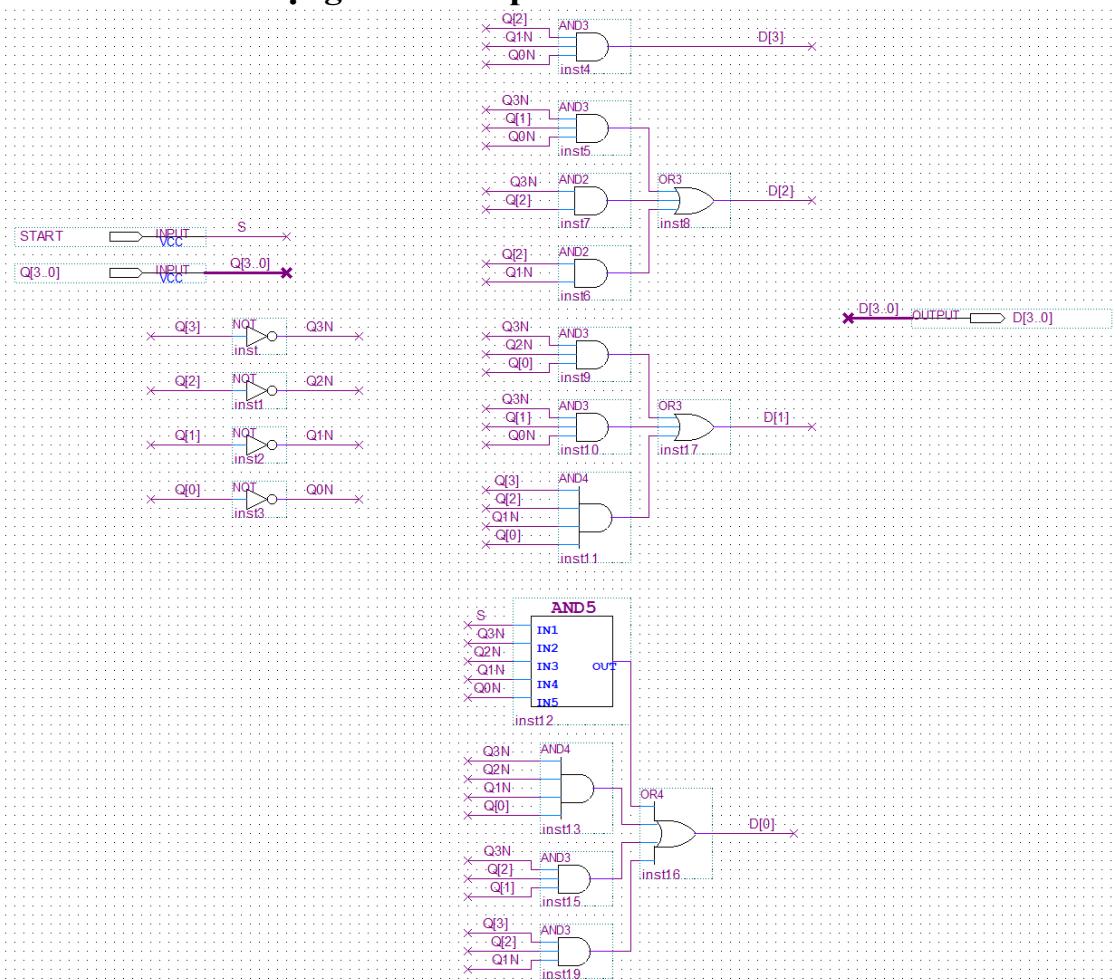
Bảng 5.6: Bảng chuyển trạng thái

TTHT	START	
	0	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	S5	S5

Thiết kế này sử dụng các flipflop D làm nhiệm vụ lưu trữ các trạng thái. Từ bảng chuyển trạng thái, ta lập các phương trình ngõ vào các flipflop D như sau:

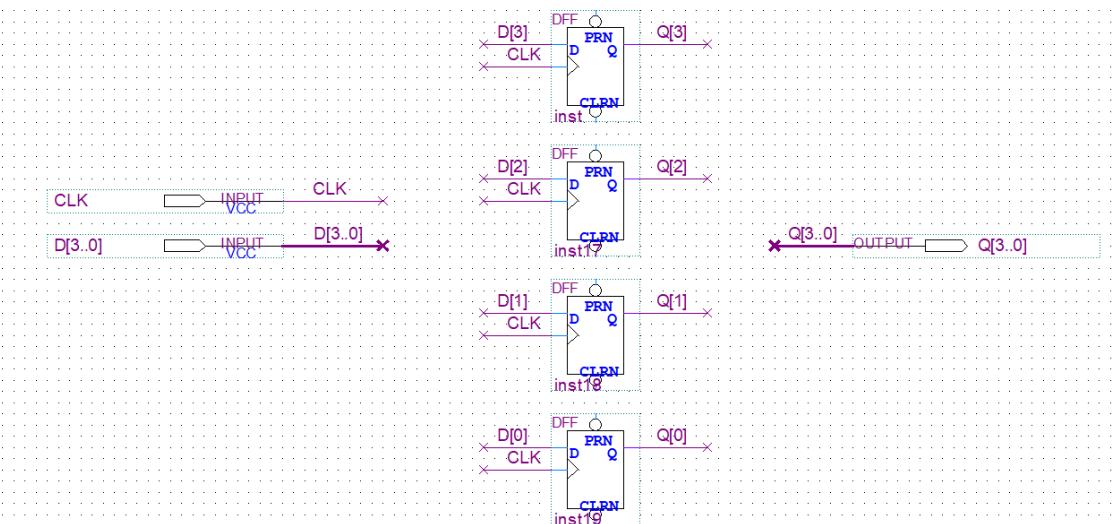
- $D_3 = Q_2.Q_1'.Q_0'$
- $D_2 = Q_3'.Q_1.Q_0' + Q_3'.Q_2 + Q_2.Q_1'$
- $D_1 = Q_3'.Q_2'.Q_0 + Q_3'.Q_1.Q_0' + Q_3.Q_2.Q_1'.Q_0$
- $D_0 = S.Q_3'.Q_2'.Q_1'.Q_0' + Q_3'.Q_2'.Q_1'.Q_0 + Q_3'.Q_2.Q_1 + Q_3.Q_2.Q_1'$

4. Thiết kế khối trạng thái kế tiếp



Hình 5.13: Mạch khối trạng thái kế tiếp

5. Thiết kế khối nhớ



Hình 5.14: Mạch khối nhớ

6. Thiết kế khối ngõ ra

a. Bảng sự thật các tín hiệu ngõ ra

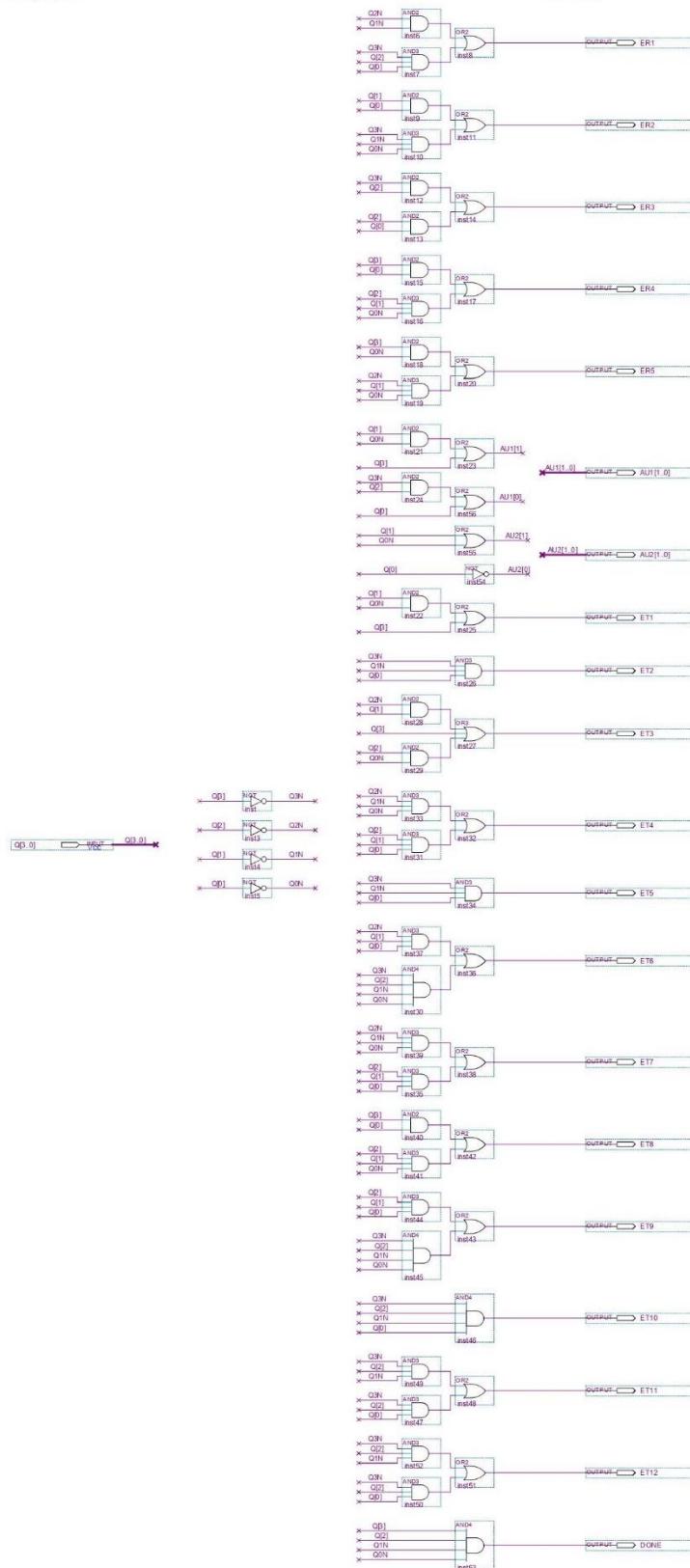
Bảng 5.7: Bảng sự thật các tín hiệu ngõ ra

TH/TT	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9
ER1	1	1	0	0	0	1	1	0	0	0
ER2	1	0	1	0	0	1	0	1	0	0
ER3	0	0	0	0	1	1	1	1	0	1
ER4	0	0	0	0	1	0	0	0	0	1
ER5	0	0	0	1	0	0	0	0	1	0
AU1[1]	X	0	0	1	1	X	0	0	1	1
AU1[0]	X	1	1	0	1	X	1	1	0	1
AU2[1]	X	X	X	X	X	1	0	1	X	X
AU2[0]	X	X	X	X	X	0	0	1	X	X
ET1	0	0	0	1	1	0	0	0	1	1
ET2	0	1	0	0	0	0	1	0	0	0
ET3	0	0	1	1	1	0	0	1	1	1
ET4	1	0	0	0	0	1	0	0	0	0
ET5	0	1	0	0	0	0	1	0	0	0
ET6	0	0	1	0	0	0	0	1	0	0
ET7	1	0	0	0	0	1	0	0	0	0
ET8	0	0	0	0	1	0	0	0	0	1
ET9	0	0	0	0	0	1	0	1	0	0
ET10	0	0	0	0	0	0	1	0	0	0
ET11	0	0	0	0	0	1	1	1	0	0
ET12	0	0	0	0	0	1	1	1	0	0
DONE	0	0	0	0	0	0	0	0	1	0

Từ bảng trên, ta lập được phương trình ngõ ra cho các tín hiệu điều khiển như sau:

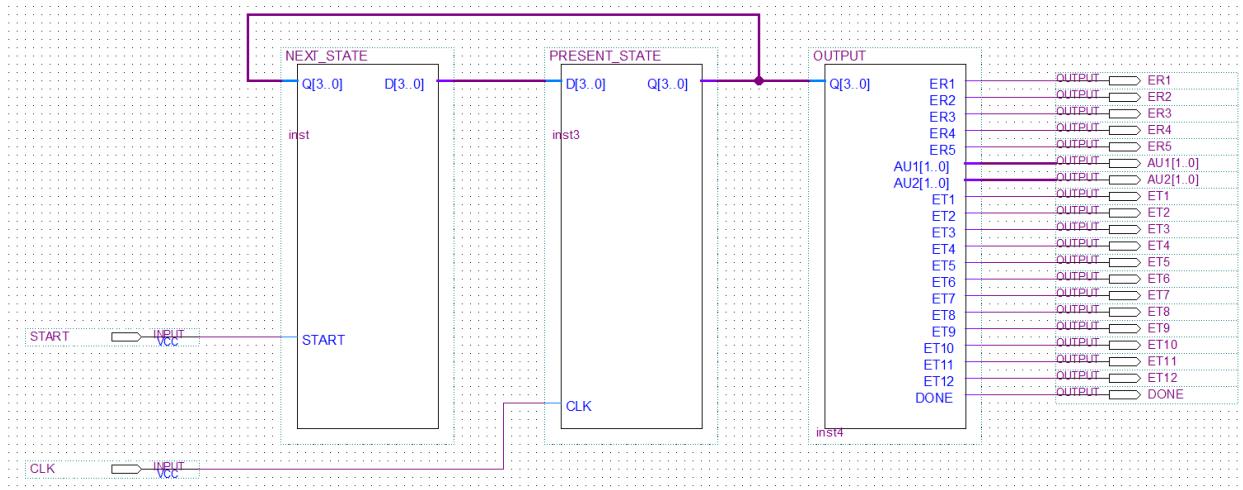
- $ER1 = Q2'.Q1' + Q3'.Q2.Q0$
- $ER2 = Q1.Q0 + Q3'.Q1'.Q0'$
- $ER3 = Q3'.Q2 + Q2.Q0$
- $ER4 = Q3.Q0 + Q2.Q1.Q0'$
- $ER5 = Q3.Q0' + Q2'.Q1.Q0'$
- $AU1[1] = Q1.Q0' + Q3$
- $AU1[0] = Q3'.Q2 + Q0$
- $AU2[1] = Q1 + Q0'$
- $AU2[0] = Q0'$
- $ET1 = Q1.Q0' + Q3$
- $ET2 = Q3'.Q1'.Q0$
- $ET3 = Q2'.Q1 + Q3 + Q2.Q0'$
- $ET4 = Q2'.Q1'.Q0' + Q2.Q1.Q0$
- $ET5 = Q3'.Q1'.Q0$
- $ET6 = Q2'.Q1.Q0 + Q3'.Q2.Q1'.Q0'$
- $ET7 = Q2'.Q1'$
- $ET8 = Q3.Q0 + Q2.Q1.Q0'$
- $ET9 = Q2.Q1.Q0 + Q3'.Q2.Q1'.Q0'$
- $ET10 = Q3'.Q2.Q1'.Q0$
- $ET11 = Q3'.Q2.Q1' + Q3'.Q2.Q0$
- $ET12 = Q3'.Q2.Q1' + Q3'.Q2.Q0$
- $DONE = Q3.Q2.Q1'.Q0'$

b. Vẽ mạch khôi ngũ ra



Hình 5.15: Mạch khôi ngũ ra

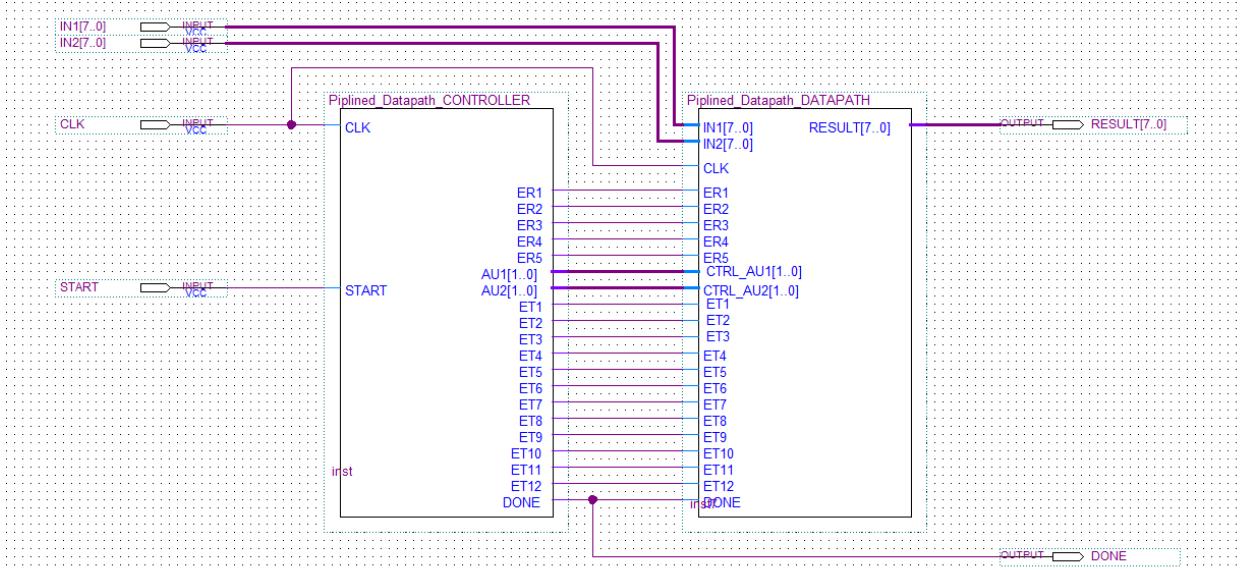
7. Tổng hợp thiết kế controller



Hình 5.16: Tổng hợp thiết kế controller

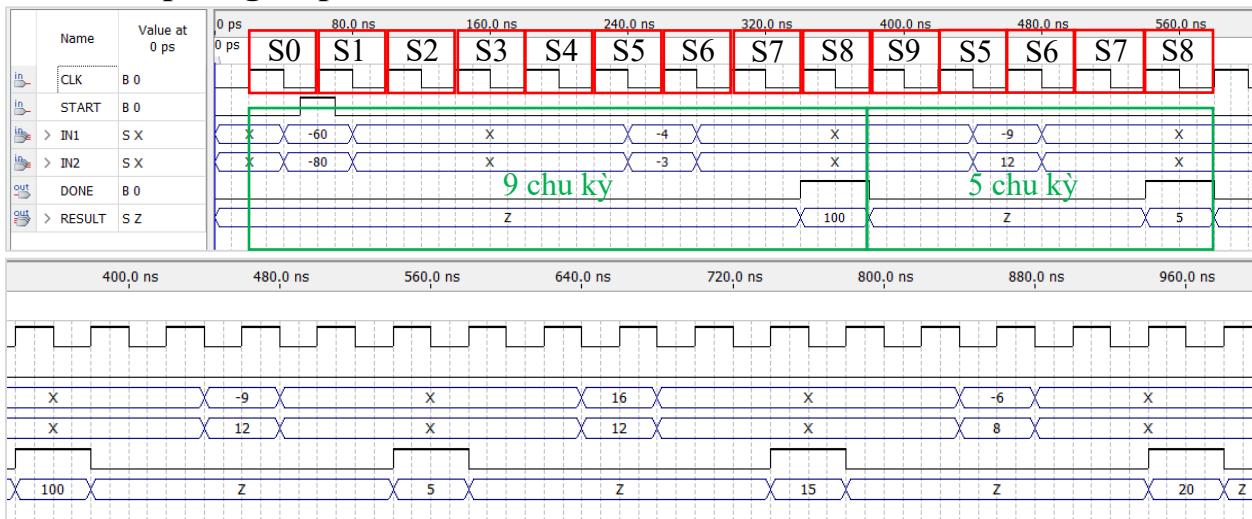
IV. TỔNG HỢP THIẾT KẾ VÀ MÔ PHỎNG

1. Tổng hợp thiết kế



Hình 5.17: Tổng hợp thiết kế Datapath Pipelining

2. Mô phỏng và phân tích



Hình 5.18: Kết quả mô phỏng theo thiết kế Datapath Pipelining

Các trường hợp thử nghiệm là:

- Số A âm, số B âm: $\sqrt{(-60)^2 + (-80)^2} = 100$
- Số A âm, số B dương: $\sqrt{(-4)^2 + (-3)^2} = 5$
- Số A âm, số B dương: $\sqrt{(-9)^2 + (12)^2} = 15$
- Số A dương, số B dương: $\sqrt{(16)^2 + (12)^2} = 20$

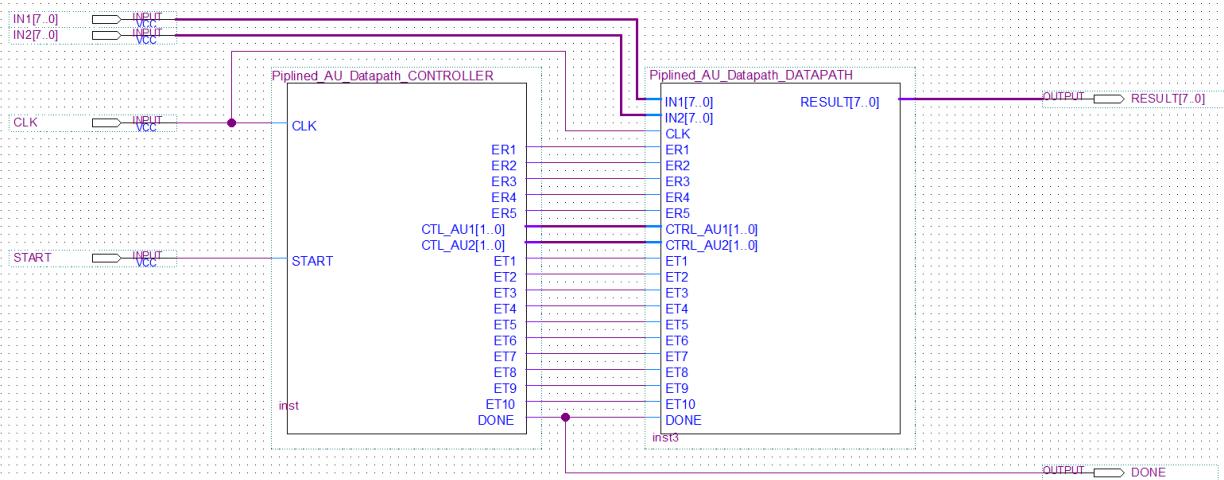
Với datapath được chia làm 2 tầng, các khối chức năng của tầng 1 được sử dụng trong các trạng thái S0 đến S4. Từ các trạng thái S5 đến S9 thì tầng 1 thực hiện xong nhiệm vụ của nó đối với tập lệnh thứ nhất và đang rảnh; tầng 2 lúc này đang thực thi tiếp tục tập lệnh thứ nhất. Lúc này, do tầng 1 rảnh nên chúng ta có thể nạp lệnh tiếp theo vào thời điểm S5, tầng 1 sẽ thực hiện tính toán với tập lệnh tiếp theo này. Khi đến trạng thái S8, kết quả phép tính của tập lệnh thứ nhất đã sẵn sàng xuất ra ngõ ra RESULT, kèm theo đó là tín hiệu báo DONE = 1.

Đối với lệnh đầu tiên, sau 9 chu kỳ ta nhận được kết quả. Kể từ lệnh tiếp theo khi pipeline, chúng ta chỉ mất thêm 5 chu kỳ để nhận được kết quả của lệnh tiếp theo được pipeline.

Tổng số chu kỳ cần thực hiện để có kết quả của phép toán cuối cùng khi thực hiện tính toán pipeline n lệnh là $5n + 4$ chu kỳ.

BÀI TẬP 6. DATAPATH PIPELINE - AU PIPELINE

I. TỔNG QUÁT THIẾT KẾ



Hình 6.1: Tổng quát thiết kế Datapath pipeline – AU pipeline

Mục tiêu của bài toán là kết hợp cả thiết kế pipeline đường dữ liệu và thiết kế pipeline khói chúc năng nhằm cải thiện hơn nữa tốc độ thực thi so với các thiết kế trước đó.

Giản đồ thời gian của thiết kế được mô tả thông qua bảng sau:

Bảng 6.1: Bảng giản đồ thời gian của thiết kế Datapath pipeline – AU pipeline

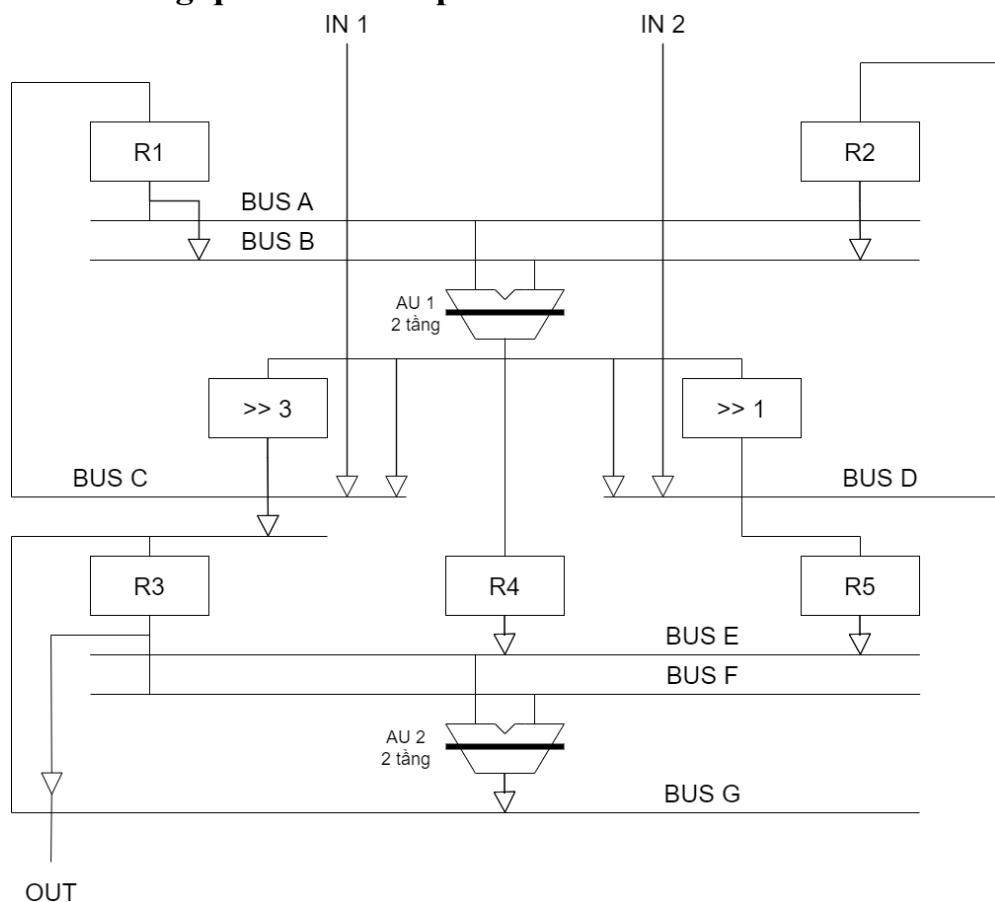
	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13
Đọc R1		a			t_1	t_1								
Đọc R2		b			t_2	t_2								
AU1 tầng 1	a	b			min	max								
AU1 tầng 2		a	b		min	min	max							
Bộ dịch					>>1	>>3								
Ghi R1	a		t_1											
Ghi R2	b			t_2										
Đọc R3							t_3		t_5		t_6		t_7	
Đọc R4							x				x			
Đọc R5								-		t_4				
AU2 tầng 1									-		+	max		
AU2 tầng 2										t_5			t_7	
Ghi R3						t_3					t_6			
Ghi R4						x								
Ghi R5					t_4								t_4	
Out													t_7	

Tầng khói dữ liệu đầu tiên gồm 2 thanh ghi R1, R2, một khói AU1 có pipeline với các chức năng ABS/MIN/MAX và 2 khói dịch. Tầng khói dữ liệu thứ hai gồm 3 thanh ghi R3, R4, R5 và một khói AU2 có pipeline với các chức năng ADD/SUB/MAX. Cụ thể các chức năng như sau:

- Tầng 1:
 - $R1 = [a, t1]$
 - $R2 = [b, t2]$
 - AU1 pipeline 2 tầng = [ABS/MIN/MAX]
 - $>>1$
 - $>>3$
- Tầng 2:
 - $R3 = [t3, t5, t6, t7]$
 - $R4 = [x]$
 - $R5 = [t4]$
 - AU2 pipeline 2 tầng = [ADD/SUB/MAX]

II. THIẾT KẾ KHỐI DATAPATH

1. Mô hình tổng quát khối datapath



Hình 6.2: Sơ đồ thiết kế khối đường dữ liệu thực thi Datapath pipeline - AU pipeline

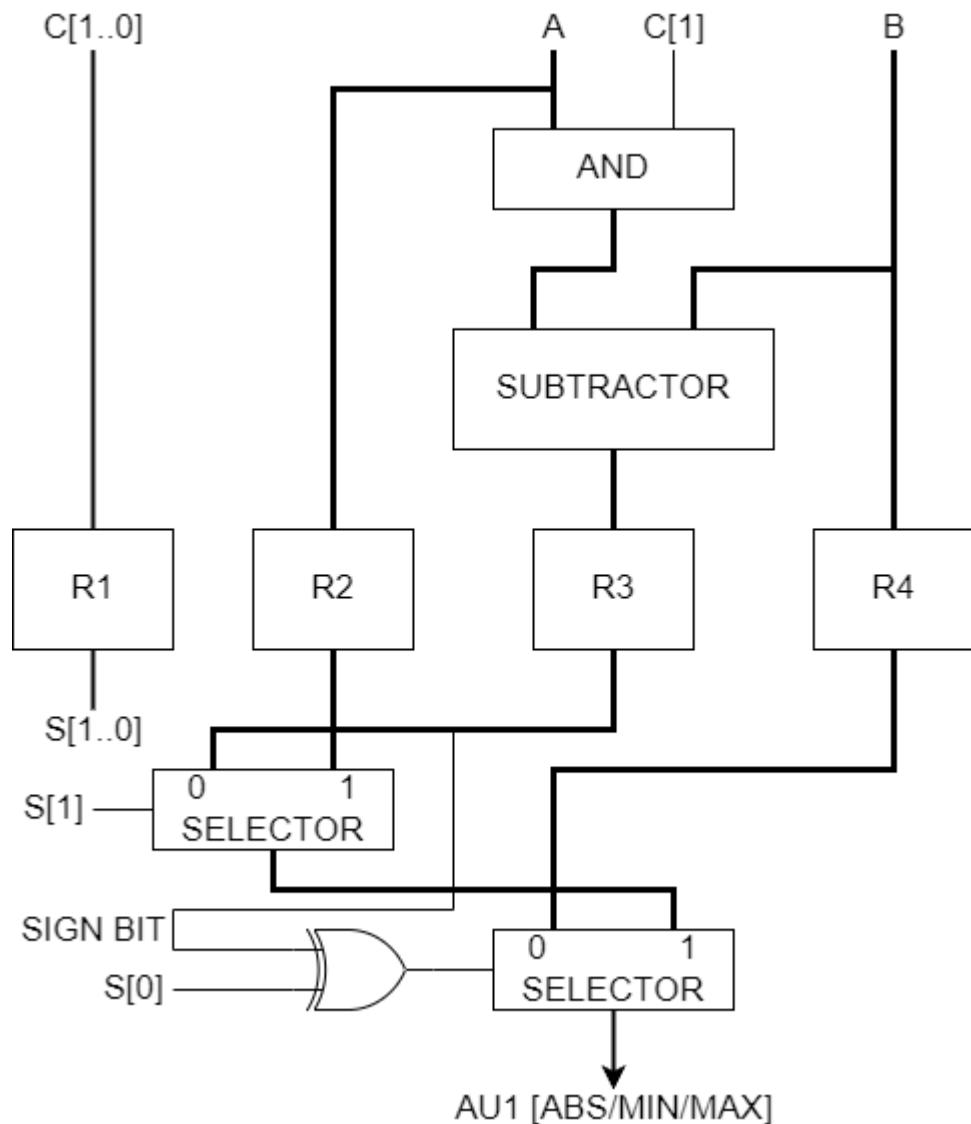
2. Phân tích các thành phần trong khối datapath

Datapath trong bài toán này gồm các thành phần sau:

- Register: R1, R2, R3, R4, R5 (tái sử dụng thiết kế trước đó).
- Bộ dịch phải: 1 bit, 3 bit (tái sử dụng thiết kế trước đó).
- AU1 pipeline 2 tầng với các chức năng: ABS/MIN/MAX.
- AU2 pipeline 2 tầng với các chức năng: ADD/SUB/MAX

3. Thiết kế khối AU 1

a. Phân tích thiết kế AU 1



Hình 6.3: Sơ đồ thiết kế khối tính AU1 [ABS/MIN/MAX] 2 tầng

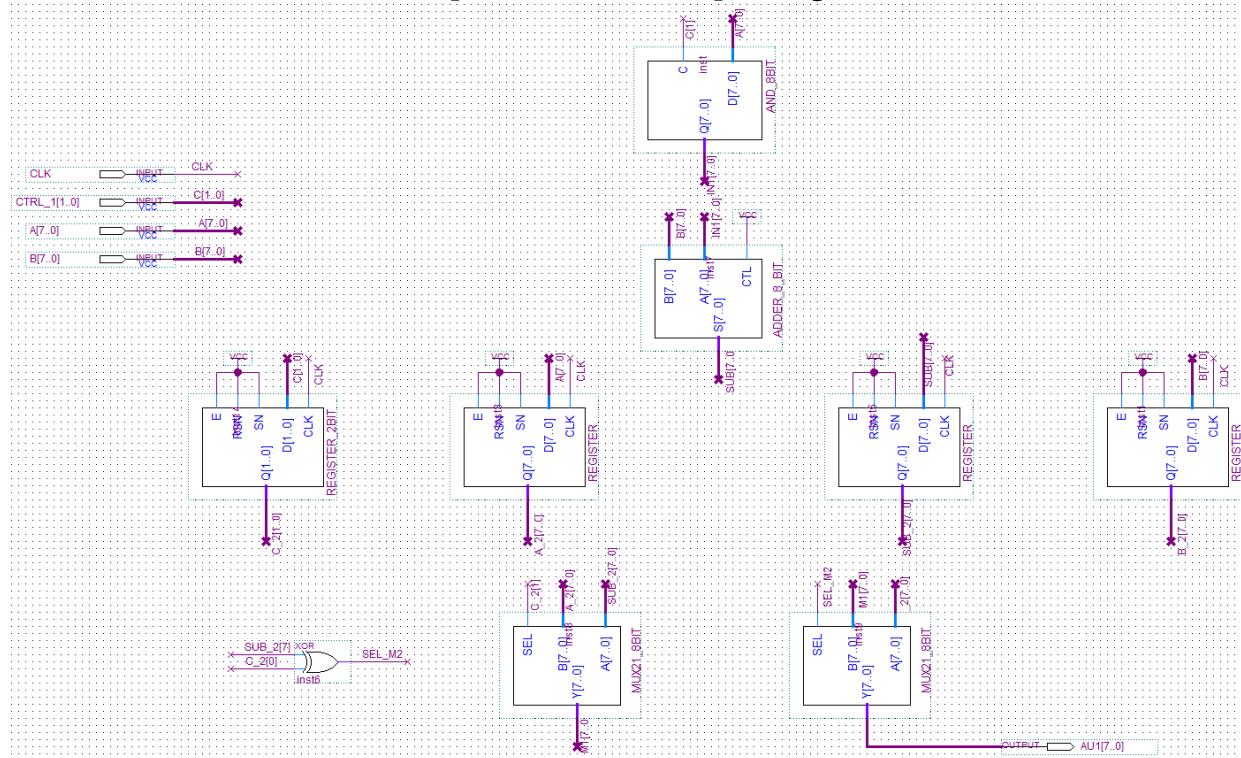
Khối pipeline AU có 3 chức năng nên cần 2 bit điều khiển C[1..0]. Thiết kế này sử dụng 4 thanh ghi làm nhiệm vụ như là một mạch chốt phục vụ việc lưu trữ dữ liệu. Trong đó, thanh ghi R1 là một mạch chốt tín hiệu điều khiển và 3 thanh ghi R2, R3, R4 là các mạch chốt tín hiệu dữ liệu.

b. Bảng sự thật các tín hiệu điều khiển chức năng

Bảng 6.2: Bảng sự thật các tín hiệu điều khiển chức năng AU1 2 tầng

CTL_1 [1..0]	Chức năng
01	ABS
10	MIN
11	MAX

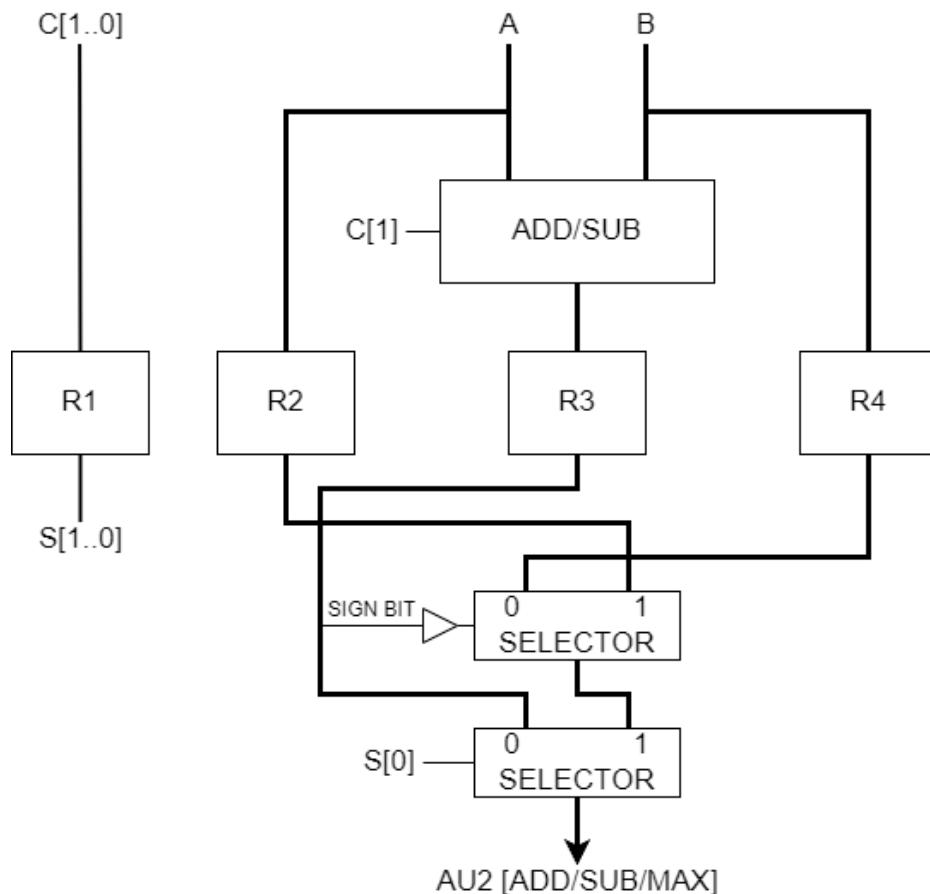
c. Vẽ mạch khối AU1 [ABS/MIN/MAX] 2 tầng



Hình 6.4: Mạch khối tính AU1 [ABS/MIN/MAX] 2 tầng

4. Thiết kế khối AU 2

a. Phân tích thiết kế khối AU 2



Hình 6.5: Sơ đồ khối tính AU 2 [ADD/SUB/MAX] 2 tầng

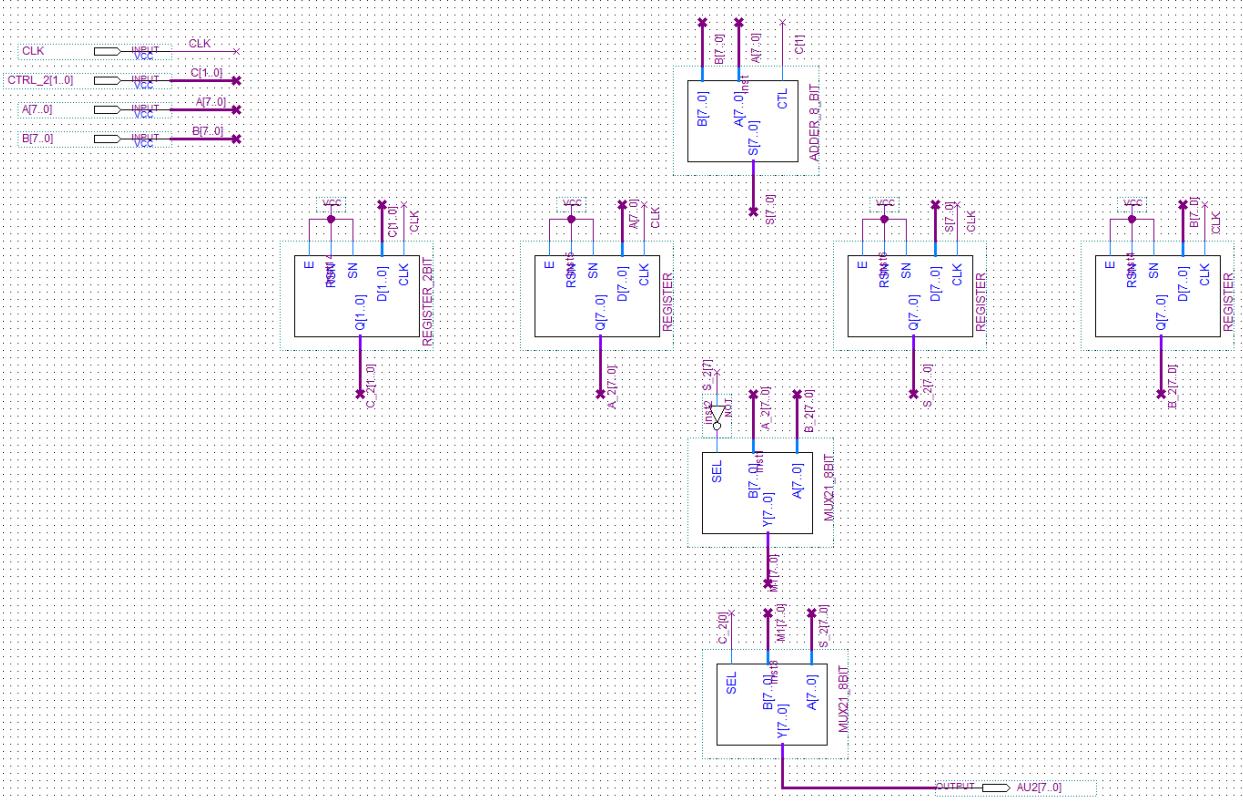
Khối pipeline AU có 3 chức năng nên cần 2 bit điều khiển C[1..0]. Thiết kế này sử dụng 4 thanh ghi làm nhiệm vụ như là một mạch chốt phục vụ việc lưu trữ dữ liệu. Trong đó, thanh ghi R1 là một mạch chốt tín hiệu điều khiển và 3 thanh ghi R2, R3, R4 là các mạch chốt tín hiệu dữ liệu.

b. Bảng sự thật các tín hiệu điều khiển chức năng

Bảng 6.3: Bảng sự thật các tín hiệu điều khiển chức năng AU 2 [ADD/SUB/MAX] 2 tầng

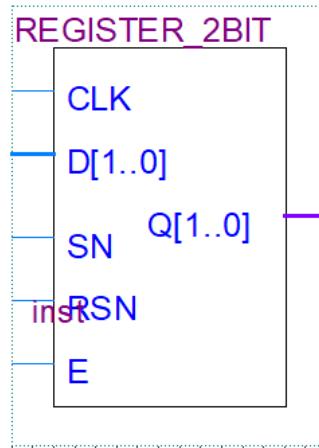
CTL_2 [1..0]	Chức năng
00	ADD
10	SUB
11	MAX

c. Vẽ mạch khối AU 2



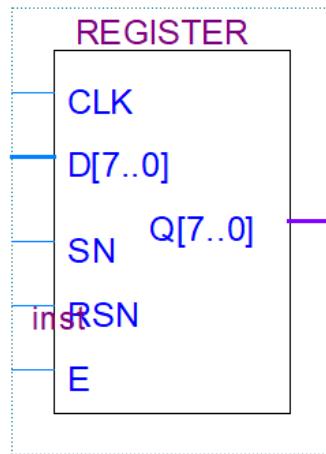
Hình 6.6: Mạch khối AU 2 [ADD/SUB/MAX] 2 tầng

5. Tái sử dụng Register 2 bit



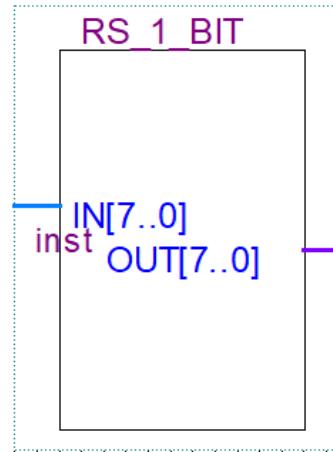
Hình 6.7: Register 2 bit

6. Tái sử dụng Register 8 bit



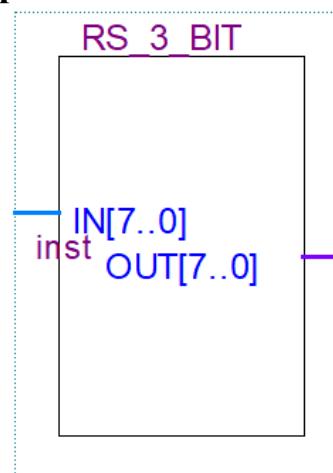
Hình 6.8: Register 8 bit

7. Tái sử dụng khối dịch phải 1 bit



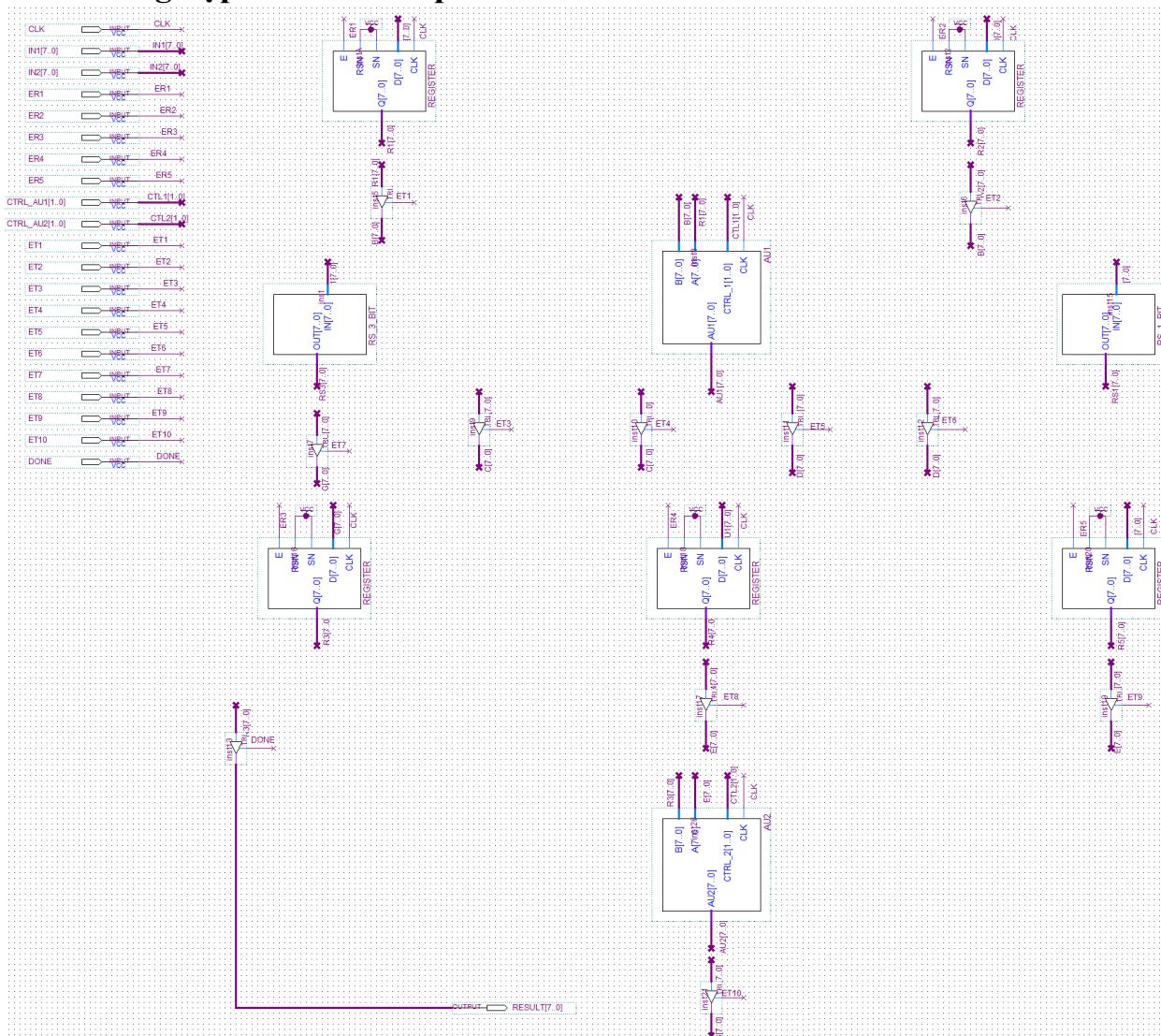
Hình 6.9: Khối dịch phải 1 bit

8. Tái sử dụng khối dịch phải 3 bit



Hình 6.10: Khối dịch phải 3 bit

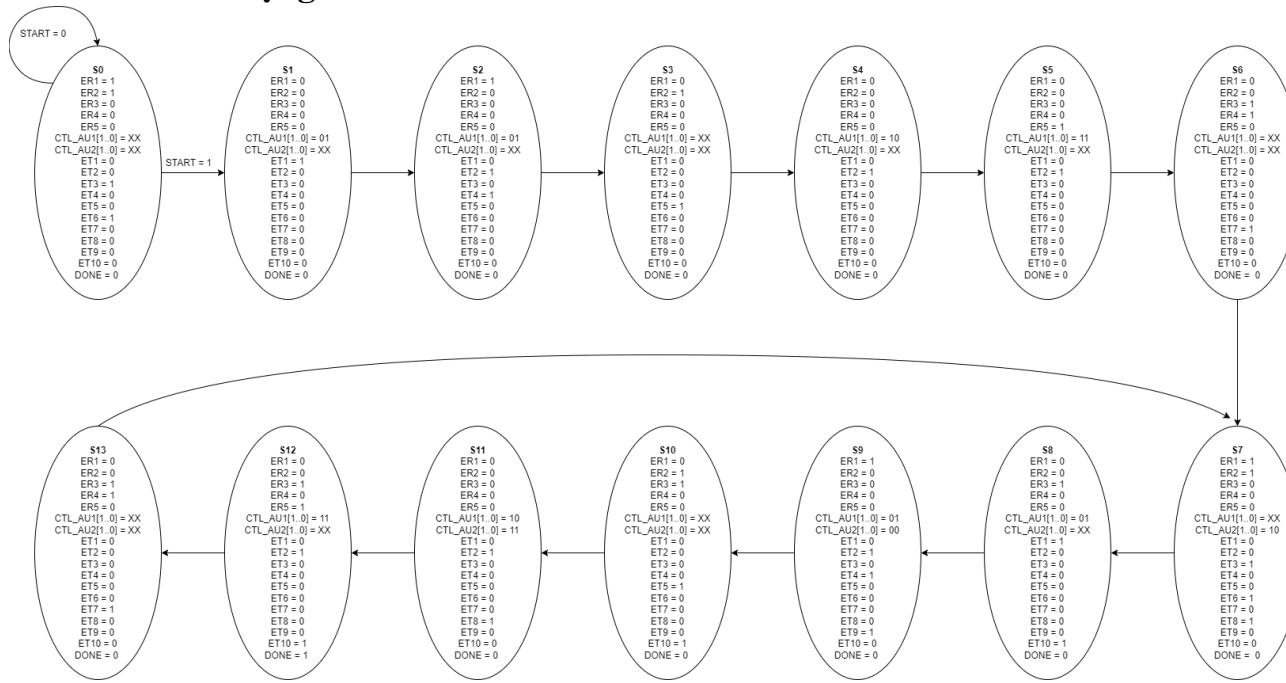
9. Tổng hợp thiết kế datapath



Hình 6.11 Tổng hợp datapath theo thiết kế Datapath pipeline - AU pipeline

III. THIẾT KẾ KHỐI CONTROLLER

1. Sơ đồ trạng thái



Hình 6.12: Sơ đồ trạng thái theo thiết kế Datapath pipeline - AU pipeline

Các tín hiệu được mô tả như sau:

- ER1: cho phép ghi hoặc không vào thanh ghi R1.
- ER2: cho phép ghi hoặc không vào thanh ghi R2.
- ER3: cho phép ghi hoặc không vào thanh ghi R3.
- ER4: cho phép ghi hoặc không vào thanh ghi R4.
- ER5: cho phép ghi hoặc không vào thanh ghi R5.
- CTL_AU1[1..0]: chọn chế độ thực thi cho khối AU1.
- CTL_AU2[1..0]: chọn chế độ thực thi cho khối AU2.
- ET1: cho phép truyền hoặc không truyền dữ liệu từ R1 xuống BUS B.
- ET2: cho phép truyền hoặc không truyền dữ liệu từ R2 xuống BUS B.
- ET3: cho phép truyền hoặc không truyền dữ liệu từ IN1 xuống BUS C.
- ET4: cho phép truyền hoặc không truyền dữ liệu từ AU1 xuống BUS C.
- ET5: cho phép truyền hoặc không truyền dữ liệu từ AU1 xuống BUS D.
- ET6: cho phép truyền hoặc không truyền dữ liệu từ IN2 xuống BUS D.
- ET7: cho phép truyền hoặc không truyền dữ liệu từ >>3 xuống BUS G.
- ET8: cho phép truyền hoặc không truyền dữ liệu từ R4 xuống BUS E.
- ET9: cho phép truyền hoặc không truyền dữ liệu từ R5 xuống BUS E.
- ET10: cho phép truyền hoặc không truyền dữ liệu từ AU2 xuống BUS G.
- DONE: tín hiệu hoàn thành.

2. Mã hóa trạng thái

Bảng 6.4: Bảng gán trạng thái theo bìa K-Map

F		Q1Q0			
		00	01	11	10
Q3Q2	00	S0	S1	S2	S3
	01	S7	S6	S5	S4
	11	S8	S9	S10	S11
	10			S13	S12

Bảng 6.5: Bảng mã hóa trạng thái

Trạng thái	Mã hóa
S0	0000
S1	0001
S2	0011
S3	0010
S4	0110
S5	0111
S6	0101
S7	0100
S8	1100
S9	1101
S10	1111
S11	1110
S12	1010
S13	1011

3. Chọn loại flipflop và lập bảng chuyển trạng thái

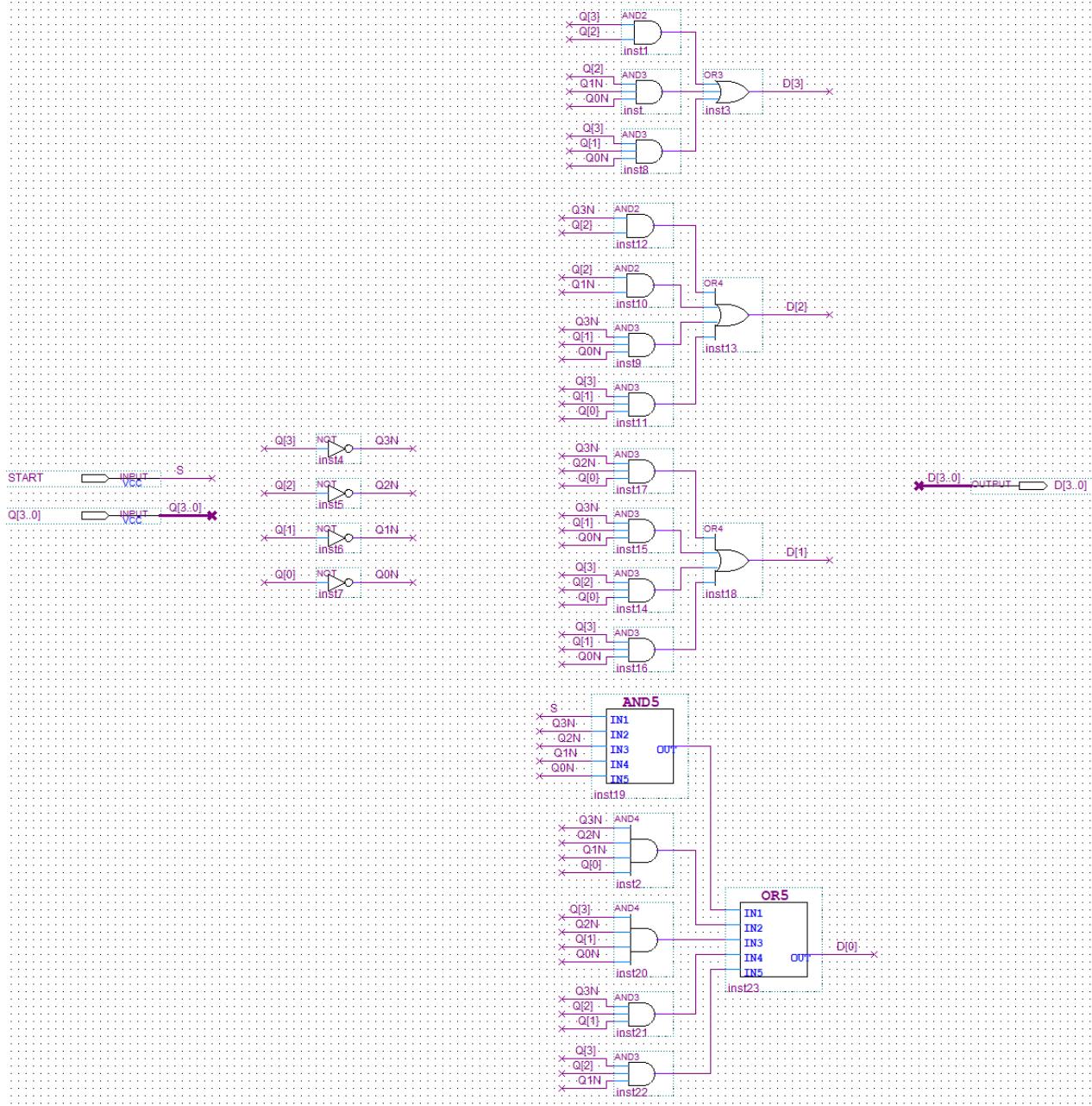
Bảng 6.6: Bảng chuyển trạng thái

TTHT	START	
	0	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	S12	S12
S12	S13	S13
S13	S7	S7

Thiết kế này sử dụng 4 flipflop D làm nhiệm vụ lưu trữ các trạng thái. Từ bảng chuyển trạng thái, ta lập phương trình ngõ vào các flipflop D như sau:

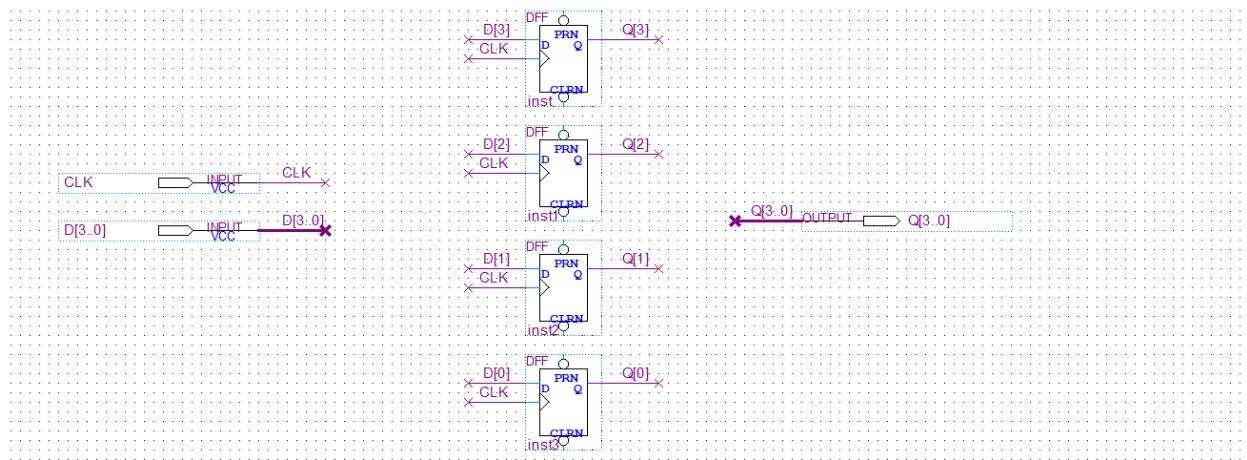
- $D_3 = Q_2.Q_1'.Q_0' + Q_3.Q_2 + Q_3.Q_1.Q_0'$
- $D_2 = Q_3'.Q_1.Q_0' + Q_3'.Q_2 + Q_2.Q_1' + Q_3.Q_1.Q_0$
- $D_1 = Q_3'.Q_2'.Q_0 + Q_3'.Q_1.Q_0' + Q_3.Q_2.Q_0 + Q_3.Q_1.Q_0'$
- $D_0 = S.Q_3'.Q_2'.Q_1'.Q_0' + Q_3'.Q_2'.Q_1'.Q_0 + Q_3'.Q_2.Q_1 + Q_3.Q_2.Q_1'$

4. Thiết kế khối trạng thái kế tiếp



Hình 6.13: Mạch khối trạng thái kế tiếp

5. Thiết kế khối nhớ



Hình 6.14: Mạch khối nhớ

6. Thiết kế khối ngõ ra tạo tín hiệu điều khiển

a. Bảng sự thật các ngõ ra

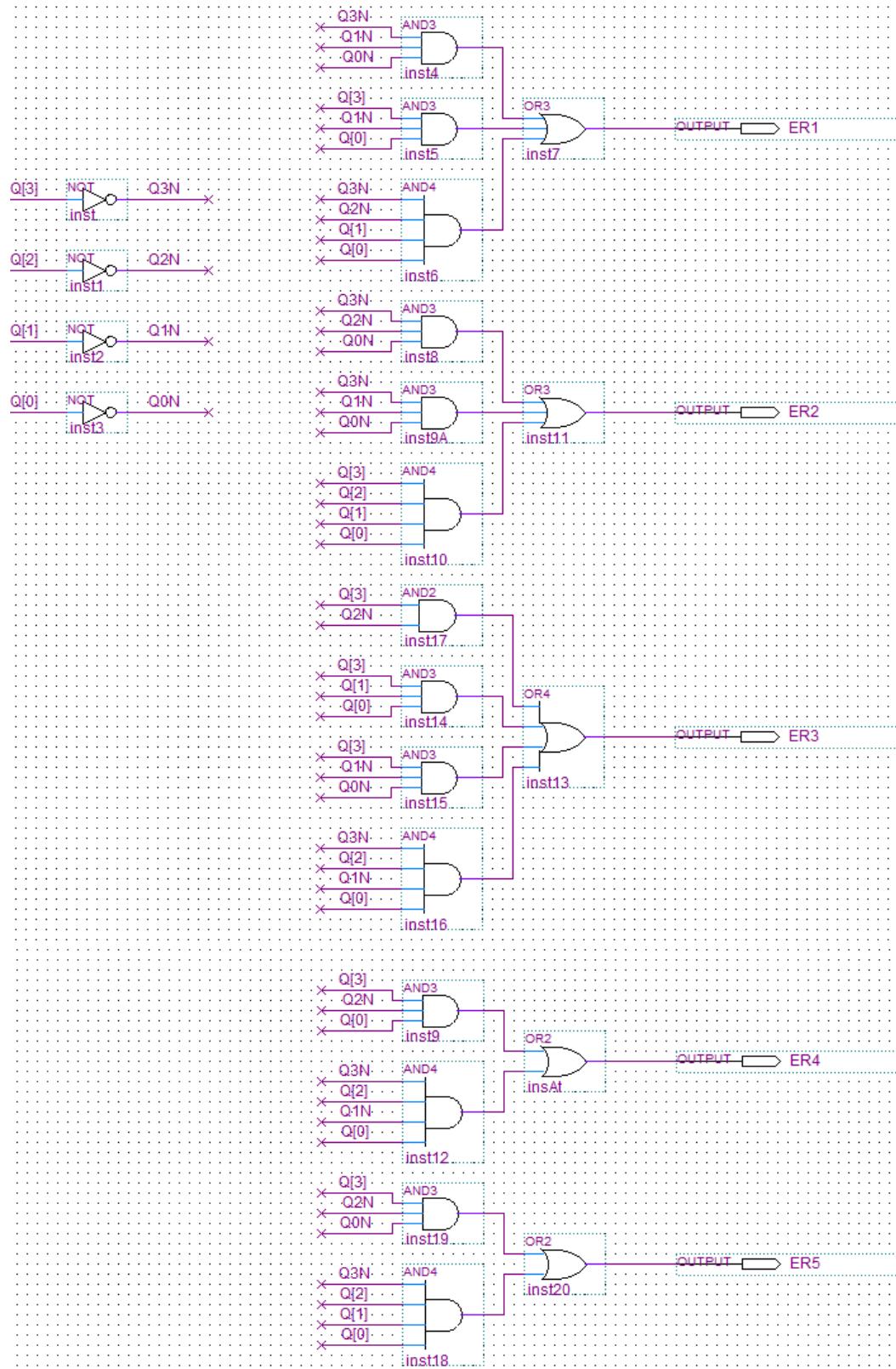
Bảng 6.7: Bảng sự thật các ngõ ra tạo tín hiệu điều khiển khối datapath

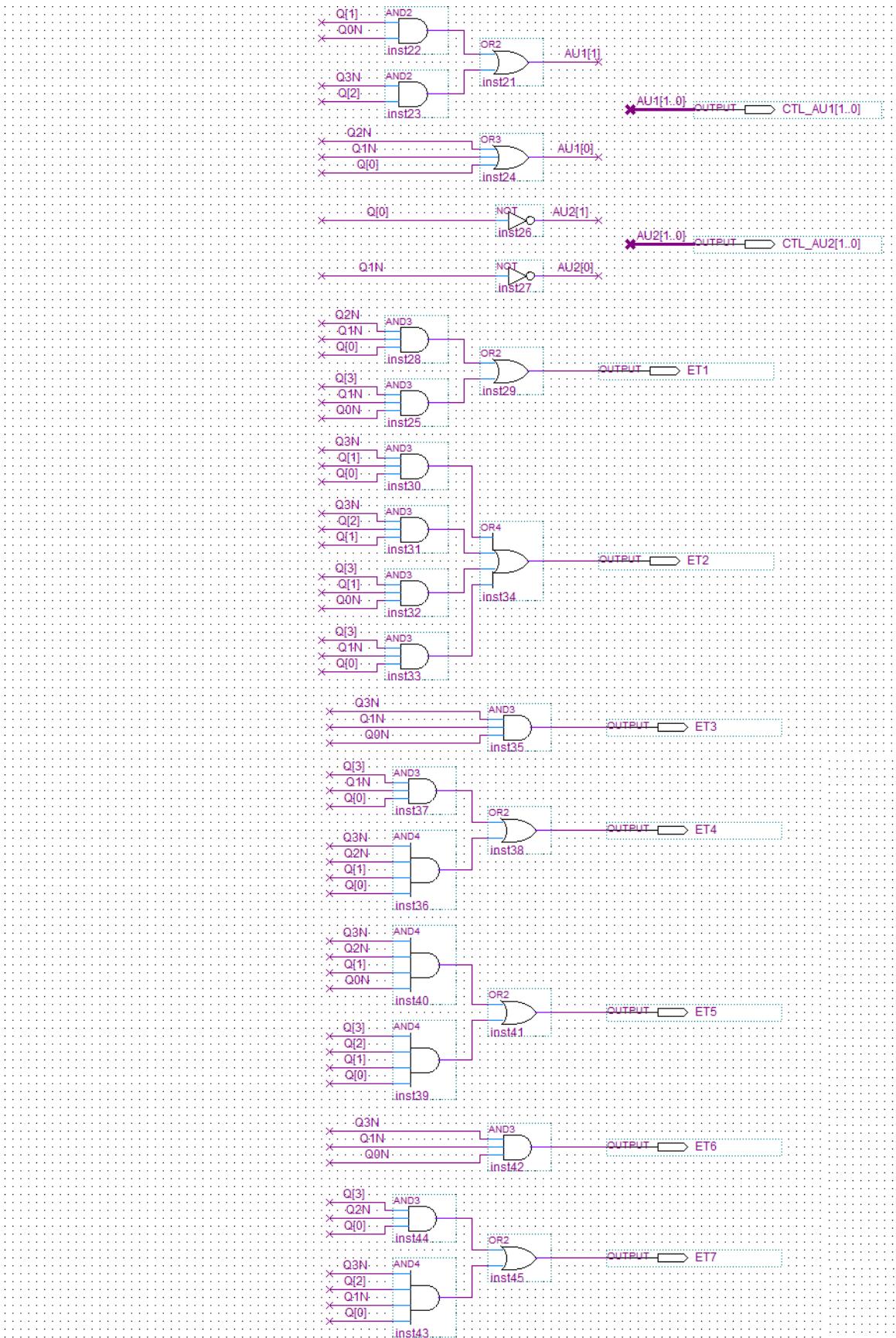
	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13
ER1	1	0	1	0	0	0	0	1	0	1	0	0	0	0
ER2	1	0	0	1	0	0	0	1	0	0	1	0	0	0
ER3	0	0	0	0	0	0	1	0	1	0	1	0	1	1
ER4	0	0	0	0	0	0	1	0	0	0	0	0	0	1
ER5	0	0	0	0	0	1	0	0	0	0	0	0	1	0
AU1[1]	X	0	0	X	1	1	X	X	0	0	X	1	1	X
AU1[0]	X	1	1	X	0	1	X	X	1	1	X	0	1	X
AU2[1]	X	X	X	X	X	X	X	1	X	0	X	1	X	X
AU2[0]	X	X	X	X	X	X	X	0	X	0	X	1	X	X
ET1	0	1	0	0	0	0	0	0	1	0	0	0	0	0
ET2	0	0	1	0	1	1	0	0	0	1	0	1	1	0
ET3	1	0	0	0	0	0	0	1	0	0	0	0	0	0
ET4	0	0	1	0	0	0	0	0	0	1	0	0	0	0
ET5	0	0	0	1	0	0	0	0	0	0	1	0	0	0
ET6	1	0	0	0	0	0	0	1	0	0	0	0	0	0
ET7	0	0	0	0	0	0	1	0	0	0	0	0	0	1
ET8	0	0	0	0	0	0	0	1	0	0	0	1	0	0
ET9	0	0	0	0	0	0	0	0	0	1	0	0	0	0
ET10	0	0	0	0	0	0	0	0	1	0	1	0	1	0
DONE	0	0	0	0	0	0	0	0	0	0	0	1	0	0

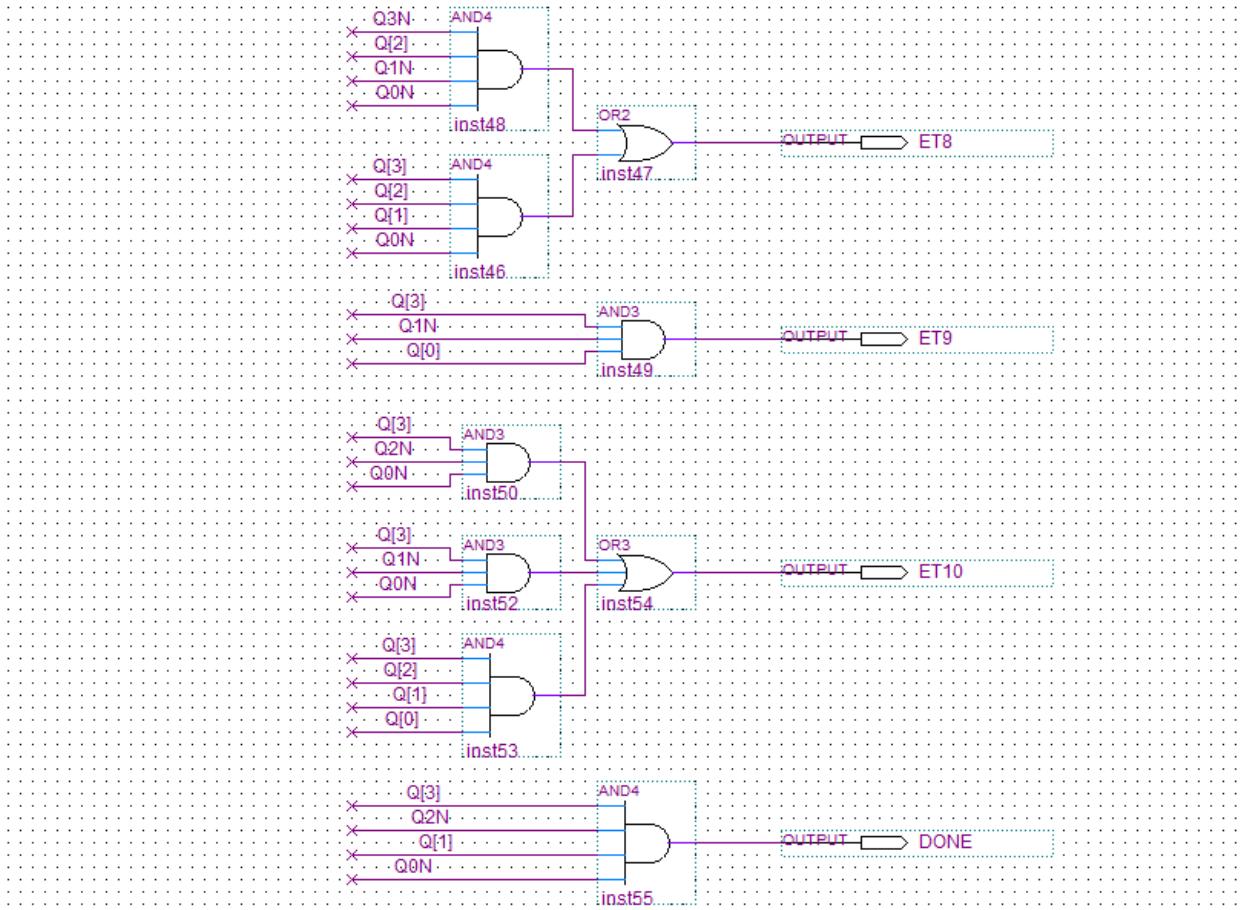
Từ bảng sự thật, ta tổng hợp được phương trình các tín hiệu ngõ ra như sau:

- $ER1 = Q3'.Q1'.Q0' + Q3.Q1'.Q0 + Q3'.Q2'.Q1.Q0$
- $ER2 = Q3'.Q2'.Q0' + Q3'.Q1'.Q0' + Q3.Q2.Q1.Q0$
- $ER3 = Q3.Q2' + Q3.Q1.Q0 + Q3.Q1'.Q0' + Q3'.Q2.Q1'.Q0$
- $ER4 = Q3.Q2'.Q0 + Q3'.Q2.Q1'.Q0$
- $ER5 = Q3.Q2'.Q0' + Q3'.Q2.Q1.Q0$
- $AU1[1] = Q1.Q0' + Q3'.Q2$
- $AU1[0] = Q2'.Q1'.Q0$
- $AU2[1] = Q0'$
- $AU2[0] = Q1$
- $ET1 = Q2'.Q1'.Q0 + Q3.Q1'.Q0'$
- $ET2 = Q3'.Q1.Q0 + Q3'.Q2.Q1 + Q3.Q1.Q0' + Q3.Q1'.Q0$
- $ET3 = Q3'.Q1'.Q0'$
- $ET4 = Q3.Q1'.Q0 + Q3'.Q2'.Q1.Q0$
- $ET5 = Q3'.Q2'.Q1.Q0' + Q3.Q2.Q1.Q0$
- $ET6 = Q3'.Q1'.Q0'$
- $ET7 = Q3.Q2'.Q0 + Q3'.Q2.Q1'.Q0$
- $ET8 = Q3'.Q2.Q1'.Q0' + Q3.Q2.Q1.Q0'$
- $ET9 = Q3.Q1'.Q0'$
- $ET10 = Q3.Q2'.Q0' + Q3.Q1'.Q0' + Q3.Q2.Q1.Q0$
- $DONE = Q3.Q2'.Q1.Q0'$

b. Vẽ mạch khối ngõ ra

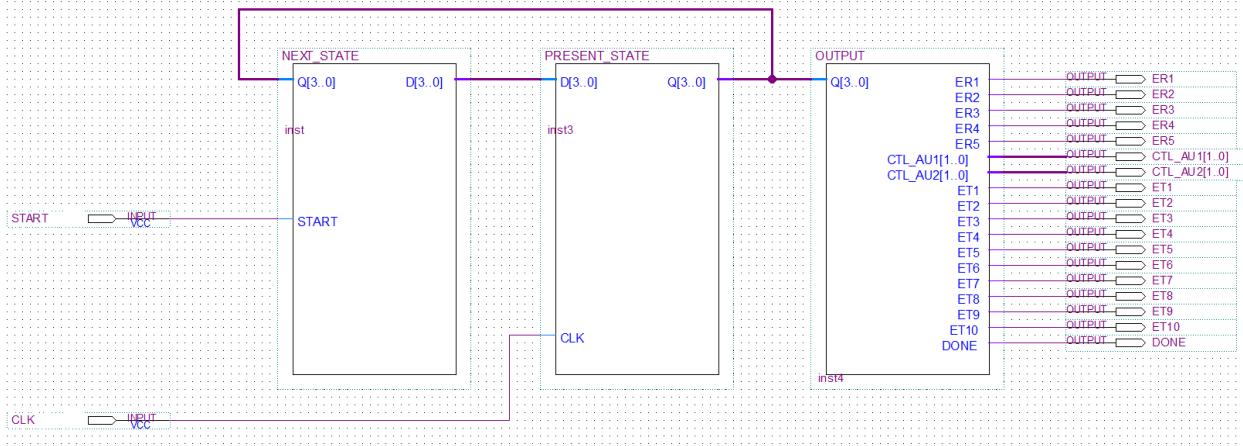






Hình 6.15: Mạch khởi ngoả ra

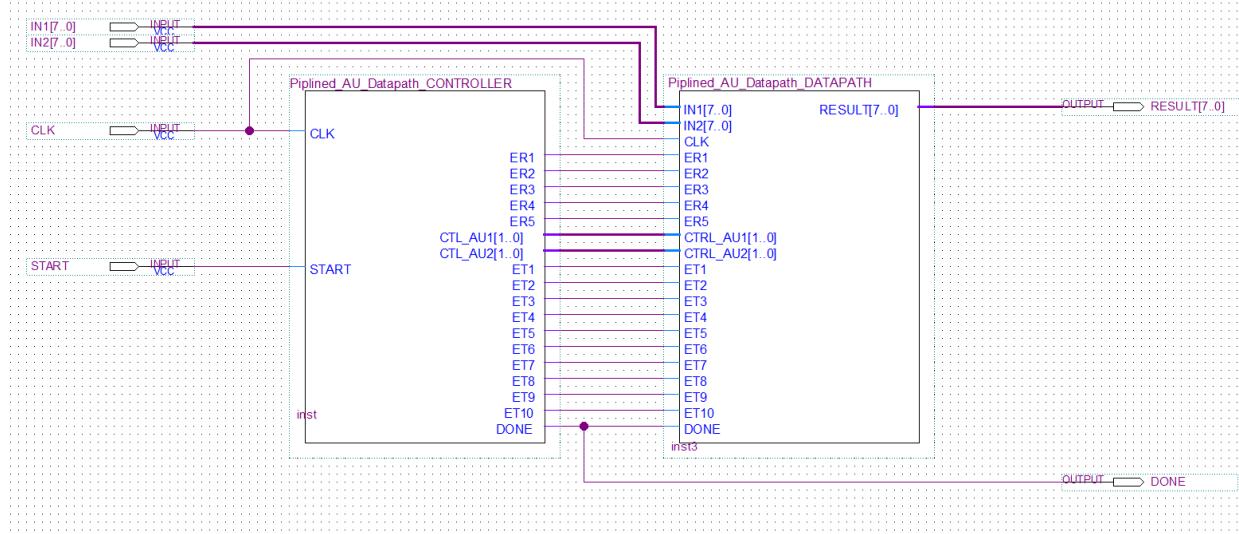
7. Tổng hợp thiết kế controller



Hình 6.16: Mạch khởi controller

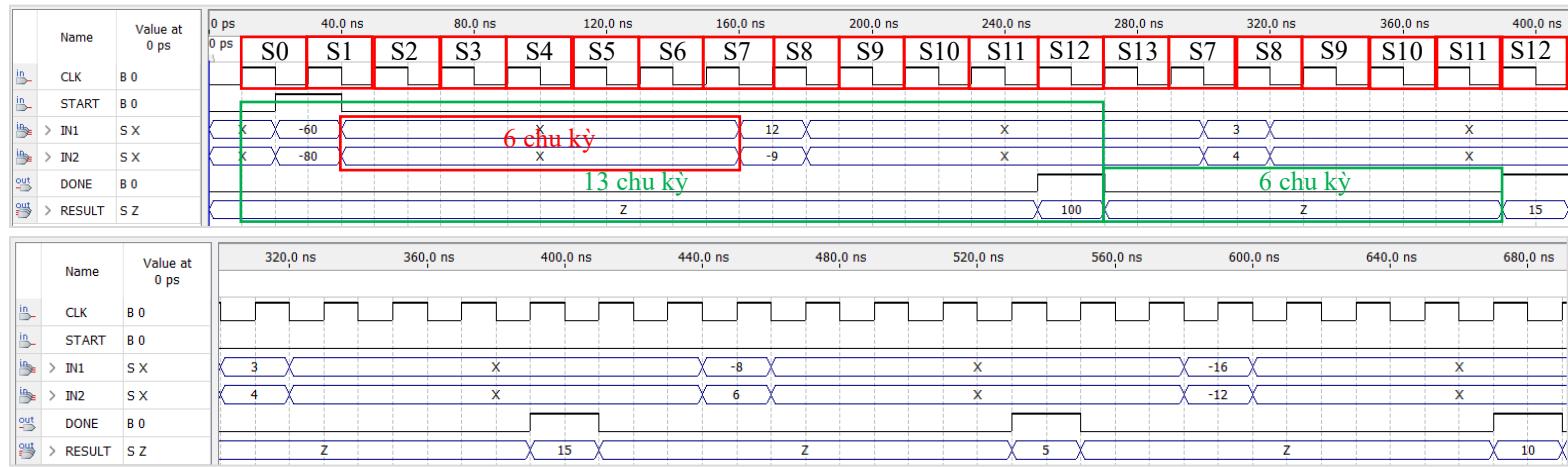
IV. TỔNG HỢP THIẾT KẾ VÀ MÔ PHỎNG

1. Tổng hợp thiết kế



Hình 6.17: Tổng hợp thiết kế Datapath pipeline - AU pipeline

2. Mô phỏng và phân tích



Hình 6.18: Kết quả mô phỏng thiết kế Datapath pipeline - AU pipeline

- Các trường hợp thử nghiệm là:
 - Số A âm, số B âm: $\sqrt{(-60)^2 + (-80)^2} = 100$
 - Số A dương, số B âm: $\sqrt{12^2 + (-9)^2} = 15$
 - Số A dương, số B dương: $\sqrt{3^2 + 4^2} = 5$
 - Số A âm, số B dương: $\sqrt{(-8)^2 + 6^2} = 10$
- Phân tích:
 - Phép toán đầu tiên với 2 input là (-60, -80) sẽ thực hiện tính toán ở tầng 1 từ các trạng thái S1 đến S6 (khi START = 1). Sau đó thì tầng 1 sẽ rảnh, do đó chúng ta có thể nạp tiếp cặp input thứ 2 là (12, -9), lúc này tầng 1 sẽ thực hiện tính toán cặp input thứ 2 đồng thời tầng 2 khi đó sẽ thực hiện tính toán tiếp cặp input đầu tiên. Kết quả của cặp input đầu tiên (-60, -80) sẽ được tính xong ở trạng thái S12, do đó kết quả sẽ được xuất ra ở trạng thái này và tín hiệu báo DONE = 1.
 - Phép toán của cặp input thứ 2 (12, -9) do nạp vào trễ hơn phép toán đầu 6 chu kỳ nên kết quả sẽ có sau kết quả của phép toán đầu 6 chu kỳ.
 - Ta sẽ nhận được kết quả của cặp input đầu tiên sau 13 chu kỳ, các cặp input tiếp theo được pipeline sẽ có kết quả cứ sau mỗi 6 chu kỳ tính từ khi có kết quả của cặp input trước đó.
 - Tổng số chu kỳ cần khi thực hiện tính toán pipeline n lệnh = $7n + 6$.