TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA KỸ THUẬT MÁY TÍNH

BÁO CÁO ĐỒ ÁN

Thiết kế CPU

Môn học: Thiết Kế Luận Lý Số
Lớp: CE118.P12
Giảng viên hướng dẫn: Ths. Tạ Trí Đức

NHÓM 01

Thành viên

Tên	MSSV
Trần Lê Thanh Tùng	22521621
Cao Quang Minh	22520856
Nguyễn Ngô Nhật Toàn	22521491
Đào Tiến Hùng	22520499
Lê Thành Lợi	22520799
Nguyễn Minh Trí	22521521
Nguyễn Trường Anh Kiện	22520707

LỜI CẢM ƠN

Để hoàn thành được đồ án này, nhóm xin chân thành gửi lời cảm ơn đến thầy Tạ Trí Đức. Người đã tận tình giúp đỡ, hỗ trợ nhóm trong quá trình thực hiện đồ án.

Trong quá trình thực hiện đồ án này không tránh khỏi có những thiếu sót. Chúng em mong nhận được những lời góp ý của thầy để nhóm phát triển và hoàn thiện hơn.

Nhóm chúng em chân thành cảm ơn thầy.

Nội dung

Danh sách hình

A. PHÁT TRIỂN TẬP LỆNH

1. Nhóm lệnh RRR (Register – Register – Register)

Sử dụng Opcode 000 và 4-bit Function để định nghĩa các lệnh thao tác giữa các thanh ghi.

Định dạng:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Opcode			Function				R1			R2			R3		

Bảng lệnh RRR:

Opcode	R1	R2	R3	Function	Instruction	Assembly-Code Format	Meaning
000	Rs1	Rs2	Rd	0000	add	add Rs1, Rs2, Rd	Reg[Rd] = Reg[Rs1] + Reg[Rs2]
000	Rs1	000	Rd	0001	inc	inc Rs1, Rd	Reg[Rd] = Reg[Rs1] + 1
000	Rs1	Rs2	Rd	0010	sub	sub Rs1, Rs2, Rd	Reg[Rd] = Reg[Rs1] - Reg[Rs2]
000	Rs1	000	Rd	0011	dec	dec Rs1, Rd	Reg[Rd] = Reg[Rs1] - 1
000	Rs1	Rs2	Rd	0100	and	and Rs1, Rs2, Rd	Reg[Rd] = Reg[Rs1] AND Reg[Rs2]
000	Rs1	Rs2	Rd	0101	or	or Rs1, Rs2, Rd	Reg[Rd] = Reg[Rs1] OR Reg[Rs2]
000	Rs1	Rs2	Rd	0110	xor	xor Rs1, Rs2, Rd	Reg[Rd] = Reg[Rs1] XOR Reg[Rs2]
000	Rs1	Rs2	Rd	0111	nand	nand Rs1, Rs2, Rd	Reg[Rd] = NOT (Reg[Rs1] AND Reg[Rs2])
000	Rs1	000	Rd	1000	shfl	shfl Rs1, Rd	Reg[Rd] = Reg[Rs1] << 1
000	Rs1	000	Rd	1001	shfr	shfr Rs1, Rd	Reg[Rd] = Reg[Rs1] >> 1
000	Rs1	000	Rd	1010	shll	shll Rs1, Rd	Reg[Rd] = Reg[Rs1] << 2
000	Rs1	000	Rd	1011	shrr	shrr Rs1, Rd	Reg[Rd] = Reg[Rs1] >> 2
000	Rs1	000	Rd	1100	slll	slll Rs1, Rd	Reg[Rd] = Reg[Rs1] << 3
000	Rs1	000	Rd	1101	srrr	srrr Rs1, Rd	Reg[Rd] = Reg[Rs1] >> 3
000	Rs1	Rs2	Rd	1110	mul	mul Rs1, Rs2, Rd	Reg[Rd] = Reg[Rs1] * Reg[Rs2]
000	Rs1	Rs2	Rd	1111	div	div Rs1, Rs2, Rd	Reg[Rd] = Reg[Rs1] / Reg[Rs2]

2. Nhóm lệnh RRI (Register – Register - Immediate)

Sử dụng Opcode từ 001 đến 011, thao tác giữa thanh ghi và giá trị hằng số

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Opcode			R1			R2			Immediate						

Bảng lệnh RRI:

Opcode	R1	R2	R3	Instruction	Assembly-Code Format	Meaning
001	Rs1	Rd	Imm	addi	add Rs1, Rd, Imm	Reg[Rd] = Reg[Rs1] + Imm
010	Rs1	Rd	Imm	subi	subi Rs1, Rd, Imm	Reg[Rd] = Reg[Rs1] - Imm
011	Rs1	Rd	Imm	beq	beq Rs1, Rd, Imm	PC = Reg[Rs1] == Reg[Rd] ? Imm : PC + 1

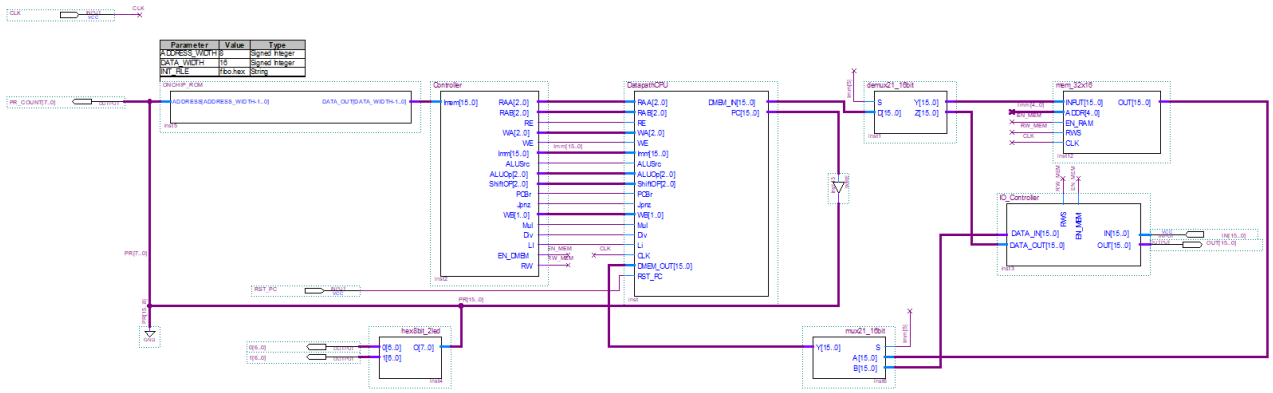
3. Nhóm lệnh RI (Register-Immediate)

Sử dụng Opcode 100 đến 111 cho các lệnh thao tác giá trị hằng.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Opcode			R1			Imm									

Bảng lệnh RI:

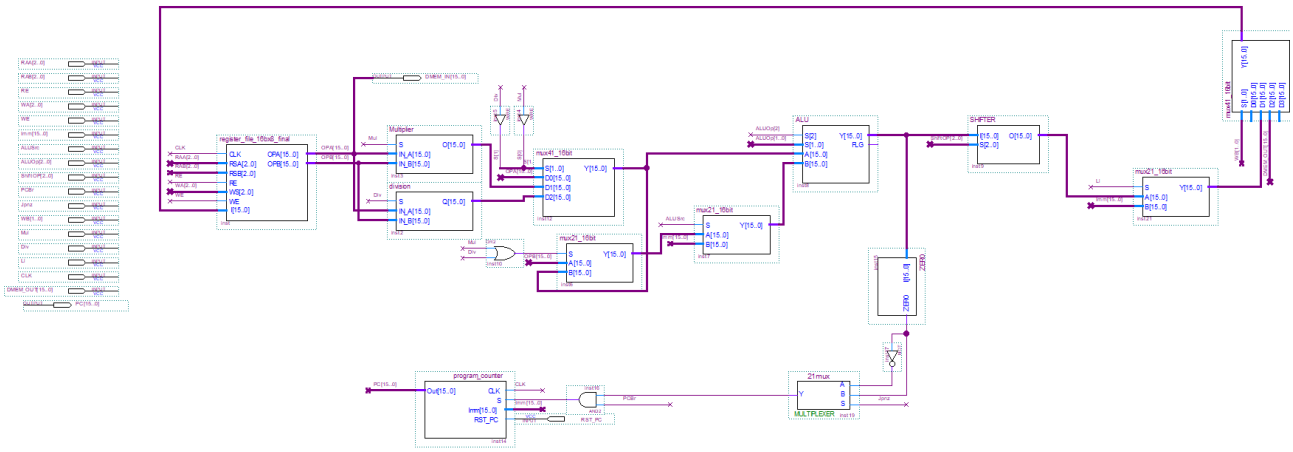
Opcode	R1	Imm	Instruction	Assembly-Code Format	Meaning
100	Rd	Imm	lw	lw Rd, Imm	Reg[Rd] <- Mem[Imm]
101	Rd	Imm	sw	sw Rd, Imm	Reg[Rd] -> Mem[Imm]
110	Rd	Imm	jpnz	jpnz Rd, Imm	PC = Reg[Rd] != 0 ? Imm : PC + 1
111	Rd	Imm	li	li Rd, Imm	Reg[Rd] <- Imm



Hình 1. CPU

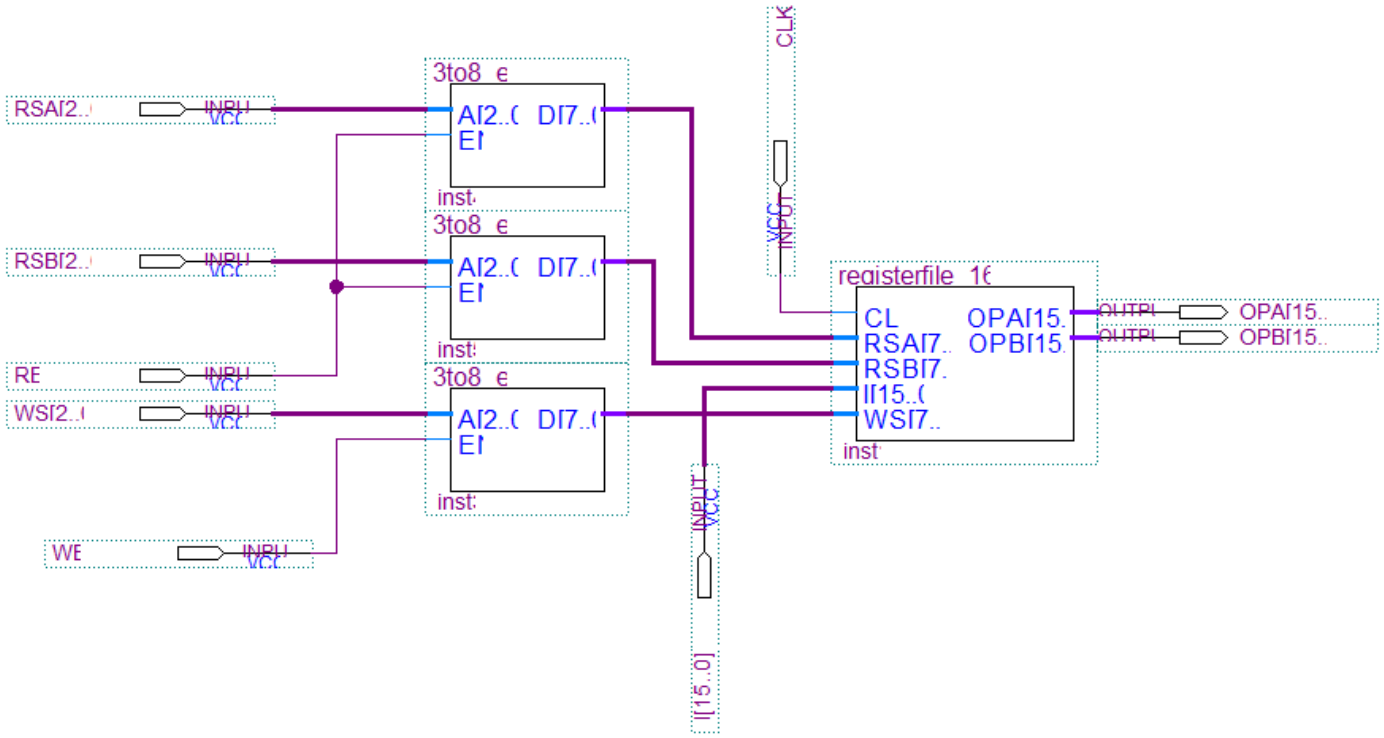
B. THIẾT KẾ DATAPATH

1. Xác định các thành phần của Datapath

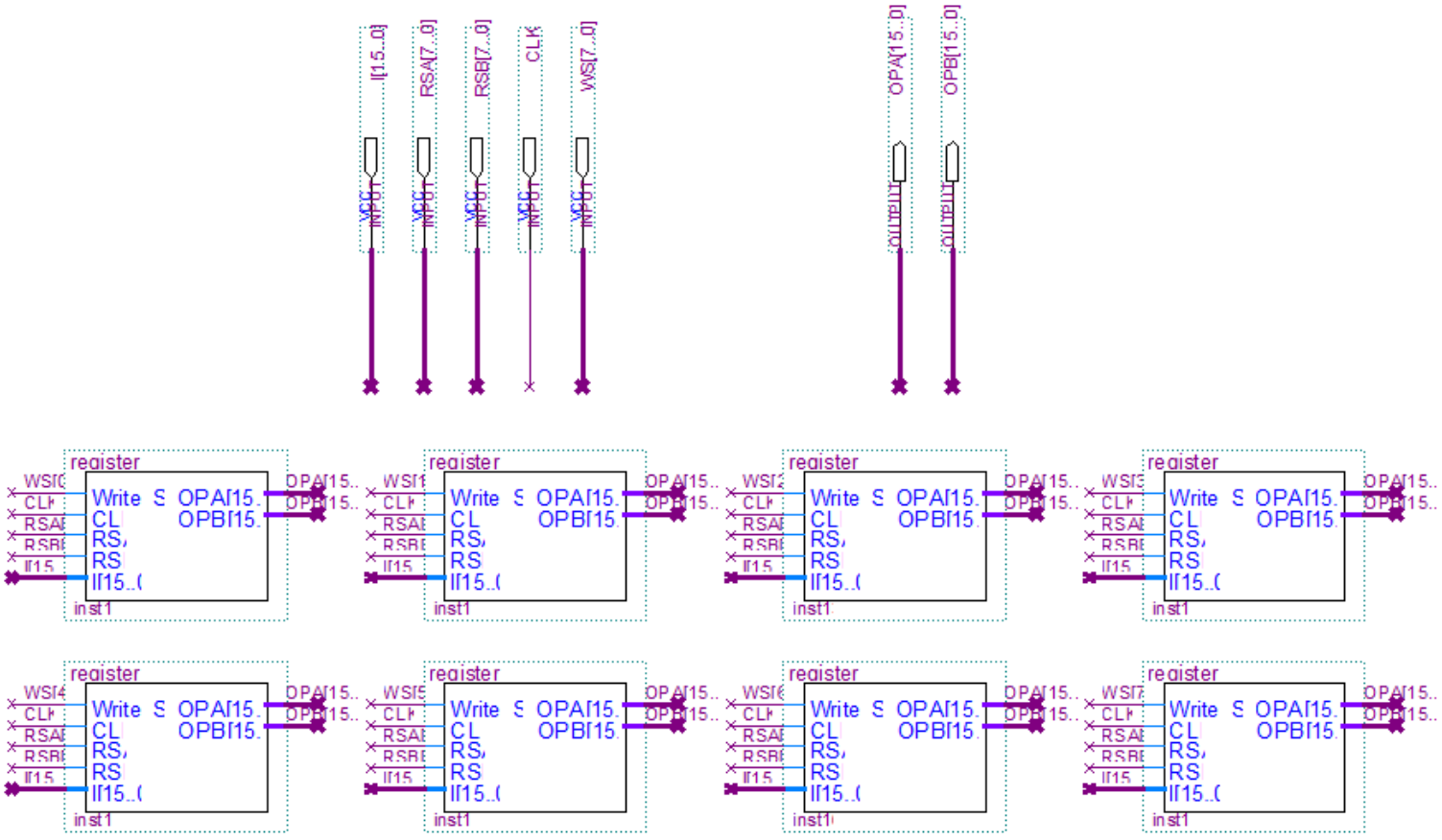


Hình 2. Khối Datapath

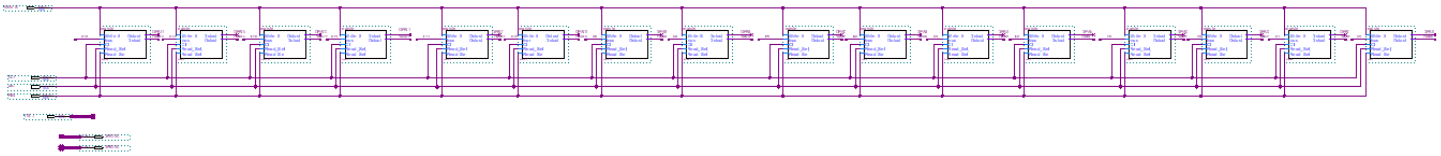
Thanh ghi: 8 thanh ghi 16-bit (R0 - R7).



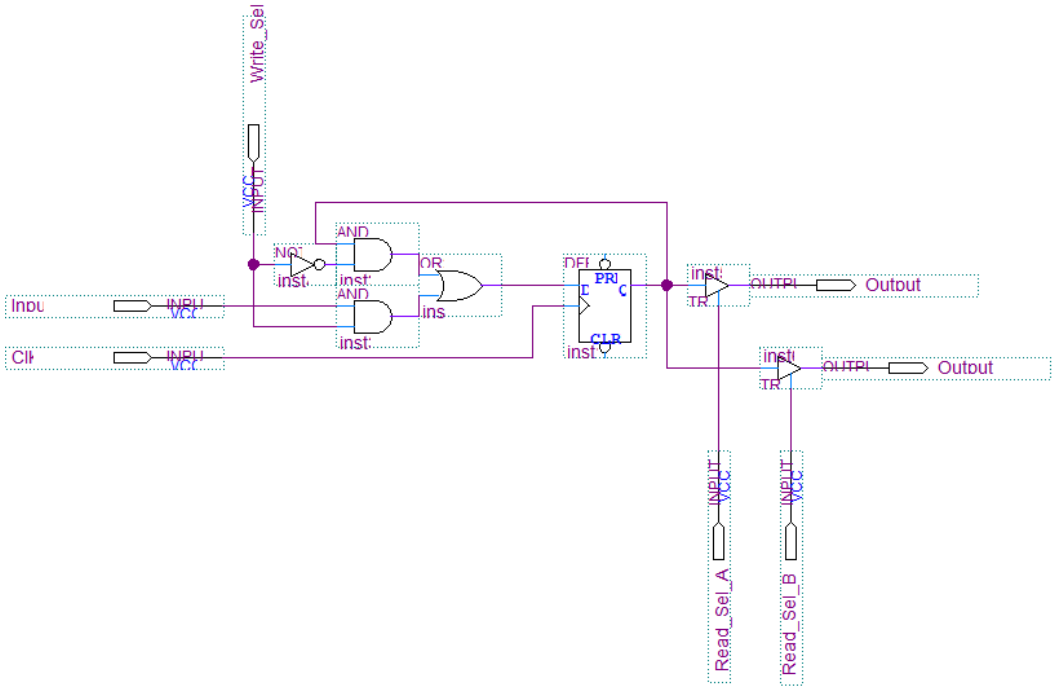
Hình 3. Khối 8 thanh ghi 16-bit với các tín hiệu điều khiển



Hình 4. Khối 8 thanh ghi 16 bit

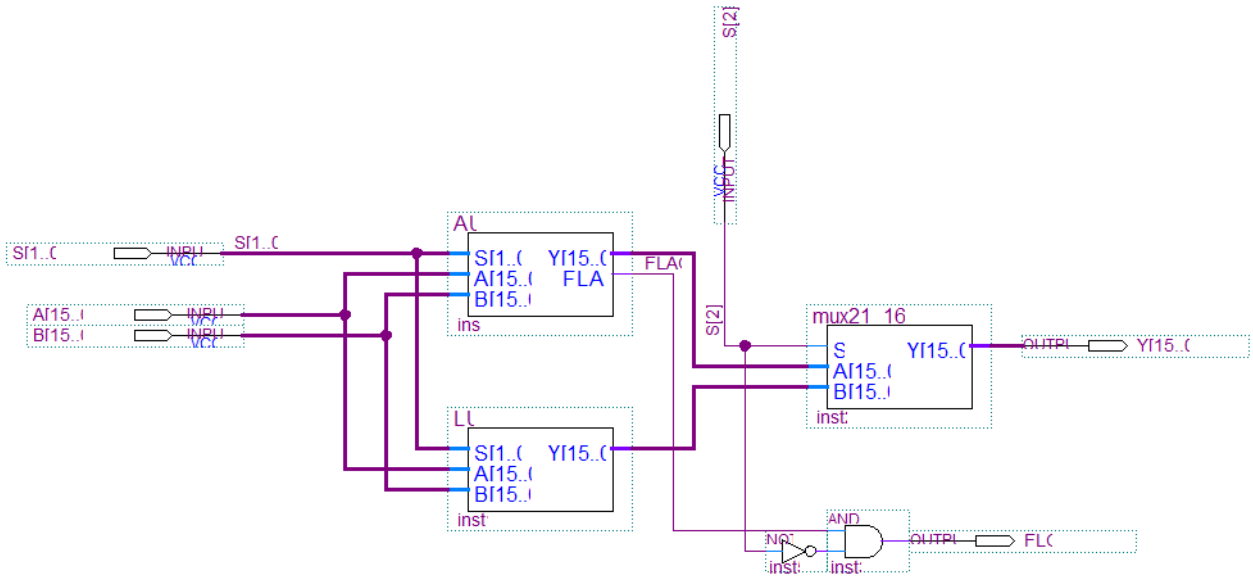


Hình 5. Khối thanh ghi 16 bit

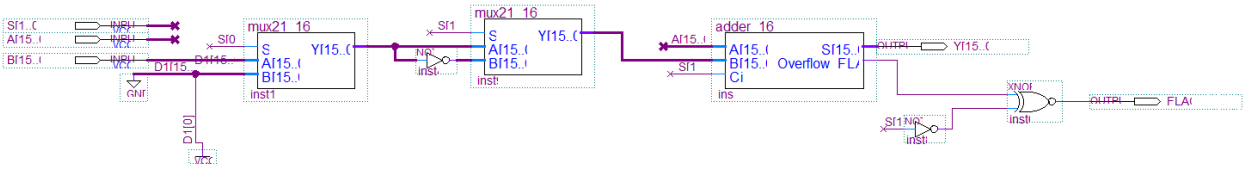


Hình 6. Khối thanh ghi

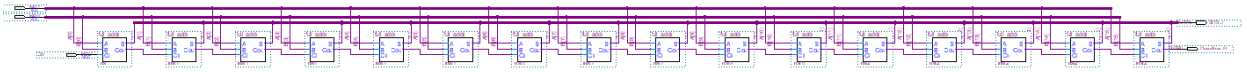
ALU (Arithmetic Logic Unit): Hỗ trợ các phép toán: cộng, cộng 1 bit, trừ, trừ 1 bit, AND, OR, XOR, NAND.



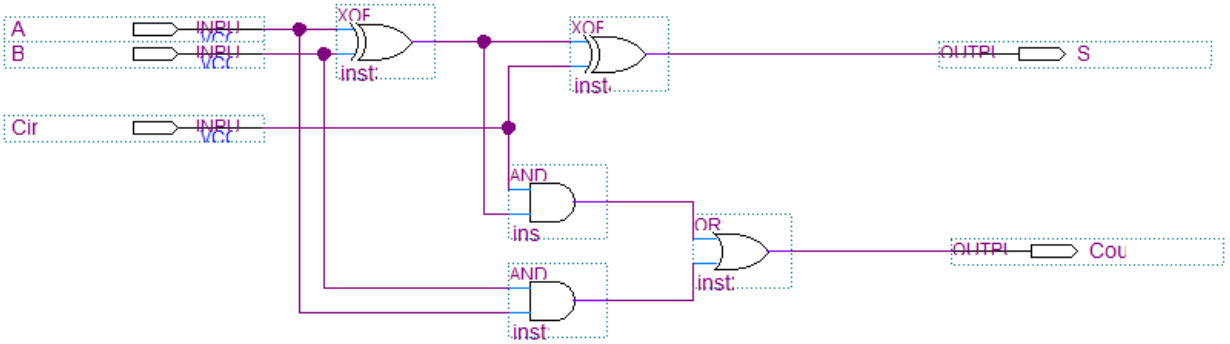
Hình 7. Khối ALU



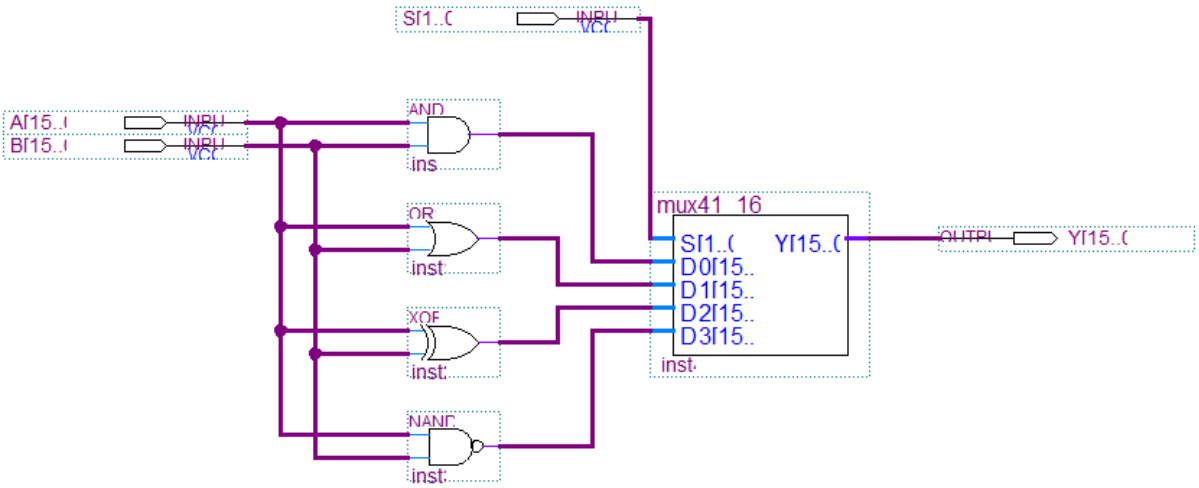
Hình 8. Khối AU



Hình 9. Khối Full Adder 16 bit



Hình 10. Khối Full Adder 1 bit



Hình 11. Khối LU

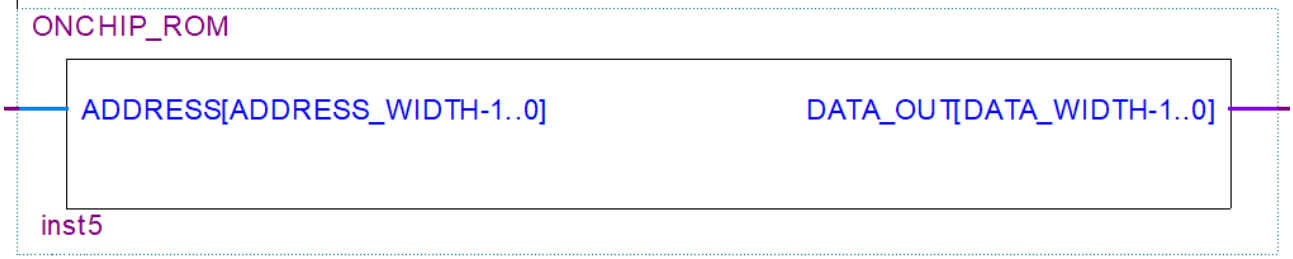
Bộ giải mã tập lệnh (Instruction Decoder): Giải mã Opcode và Function để điều khiển các khối.

Bộ nhớ:

- Bộ nhớ chương trình (64x16-bit).
- Bộ nhớ dữ liệu (32x16-bit).

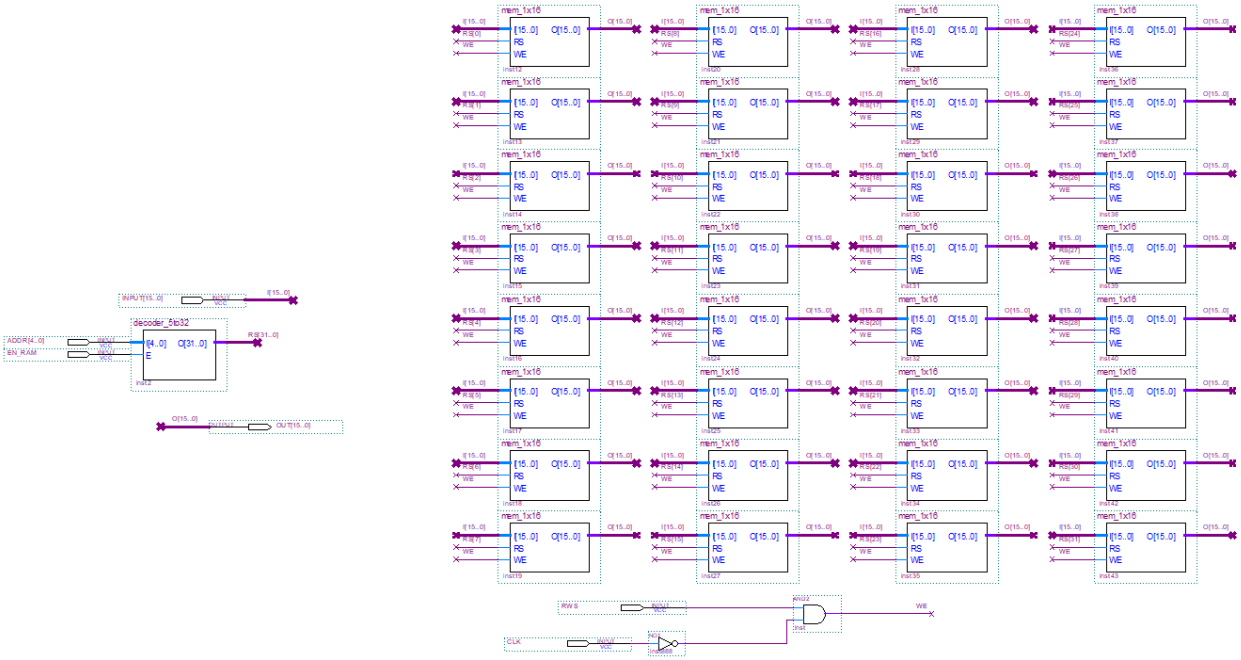
64 x 16 Imem:

Parameter	Value	Type
ADDRESS_WIDTH	8	Signed Integer
DATA_WIDTH	16	Signed Integer
INIT_FILE	fibo.hex	String

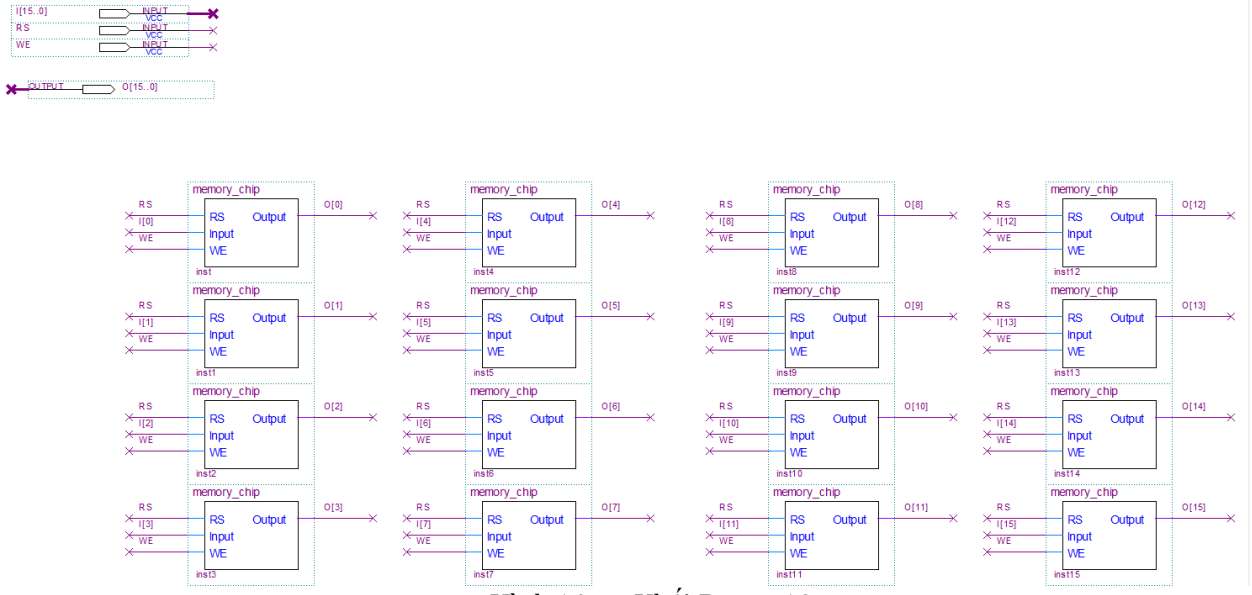


Hình 12. Khối Imem

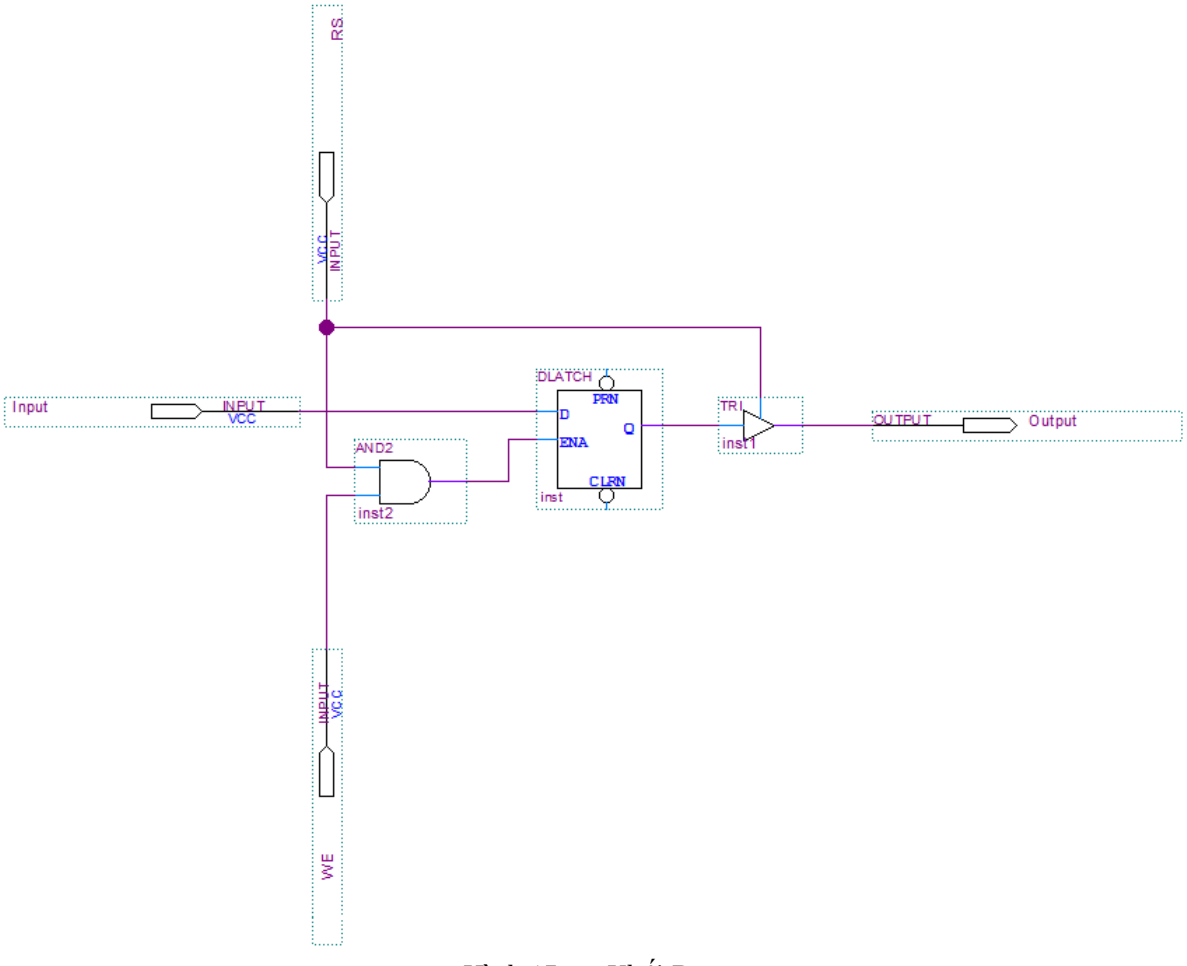
32 x 16 Dmem:



Hình 13. Khối Dmem 32x16

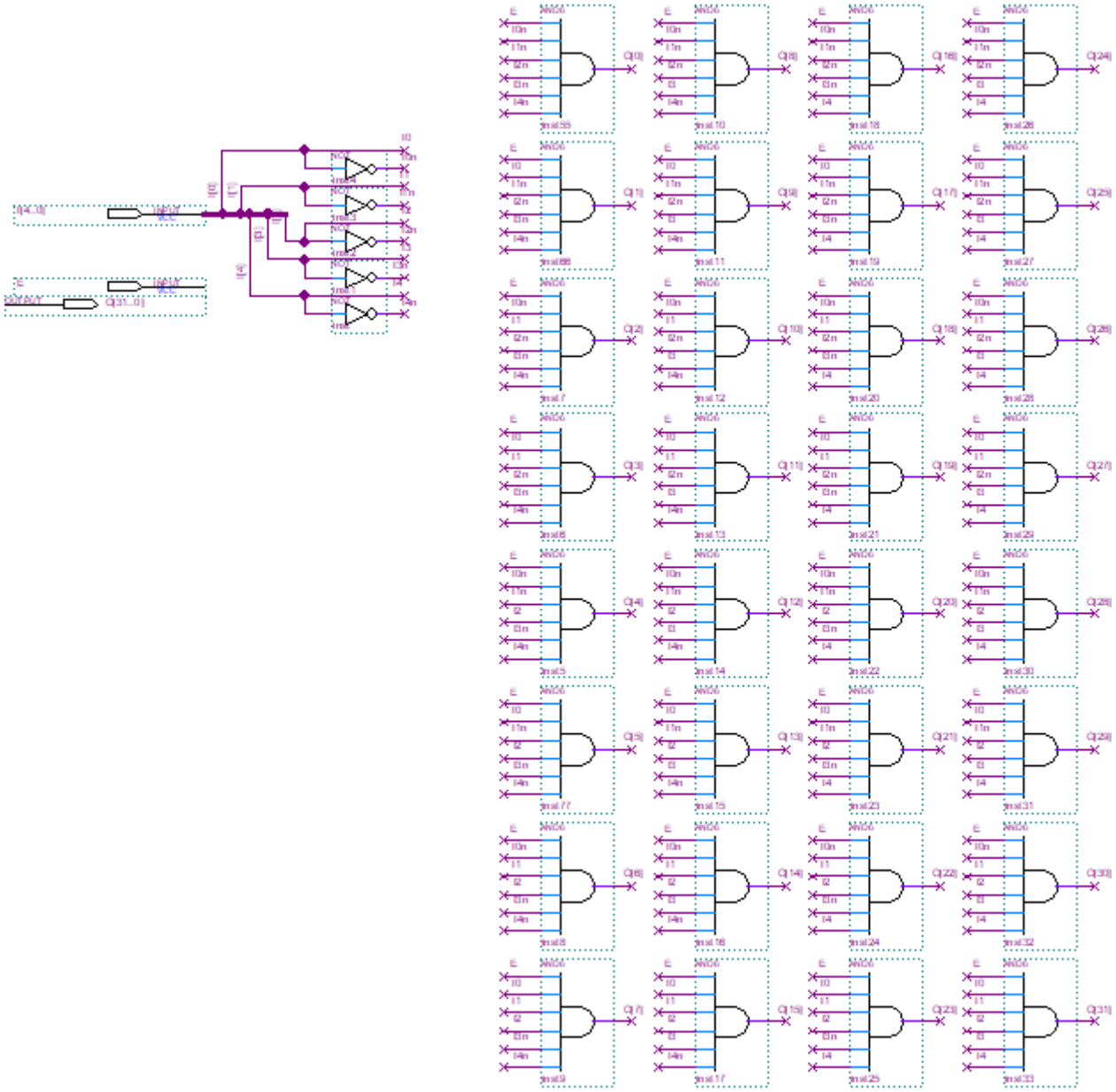


Hình 14. Khối Dmem 16



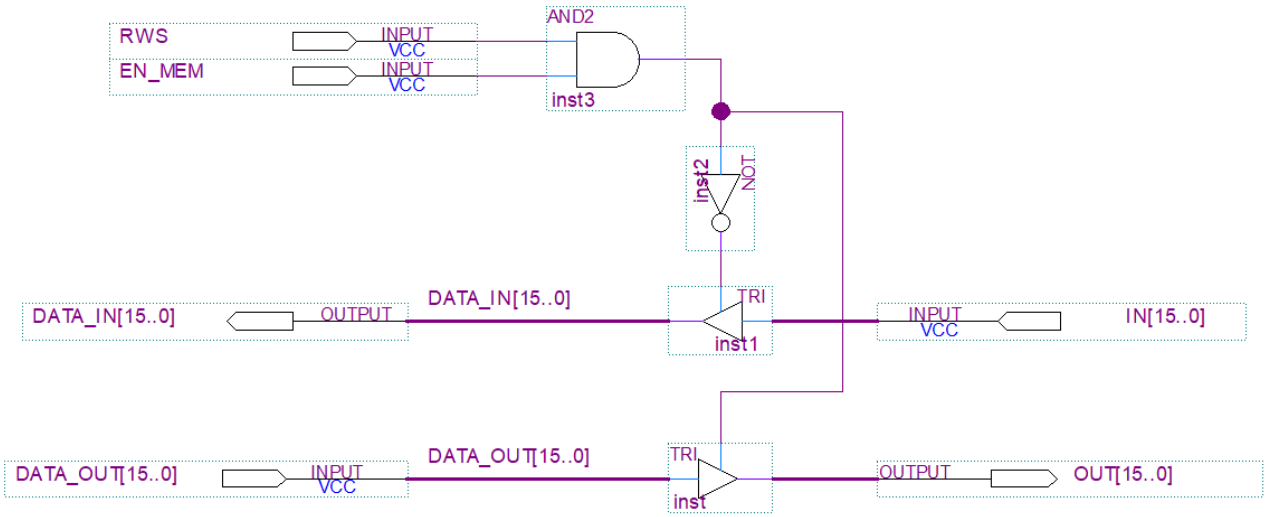
Hình 15. Khối Dmem

Data Address Decoder (DAD):



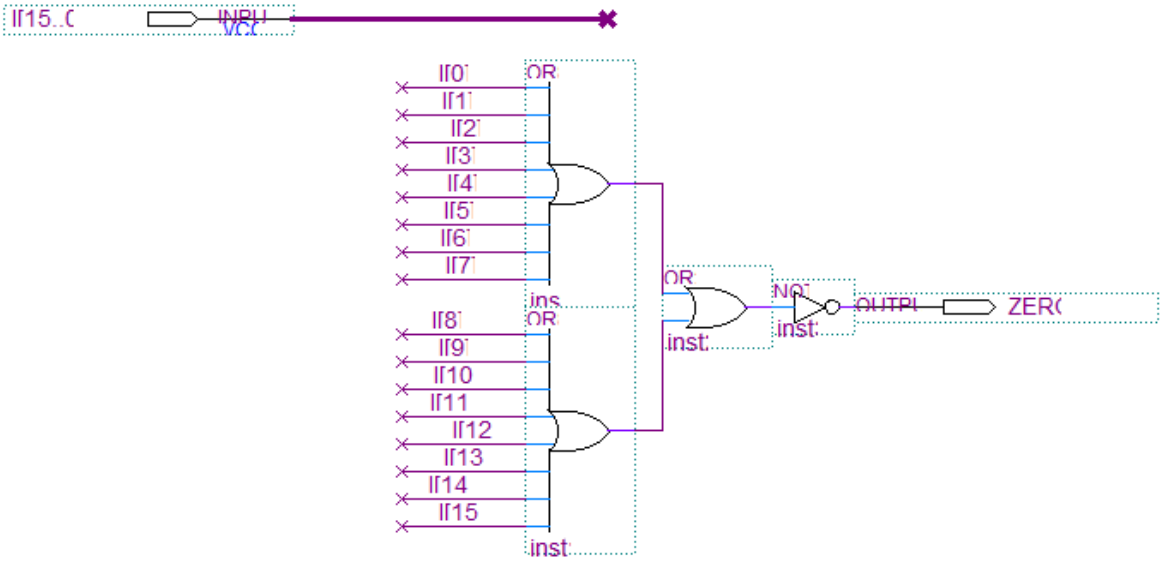
Hình 16. Khối Data Address Decoder

I/O controller:



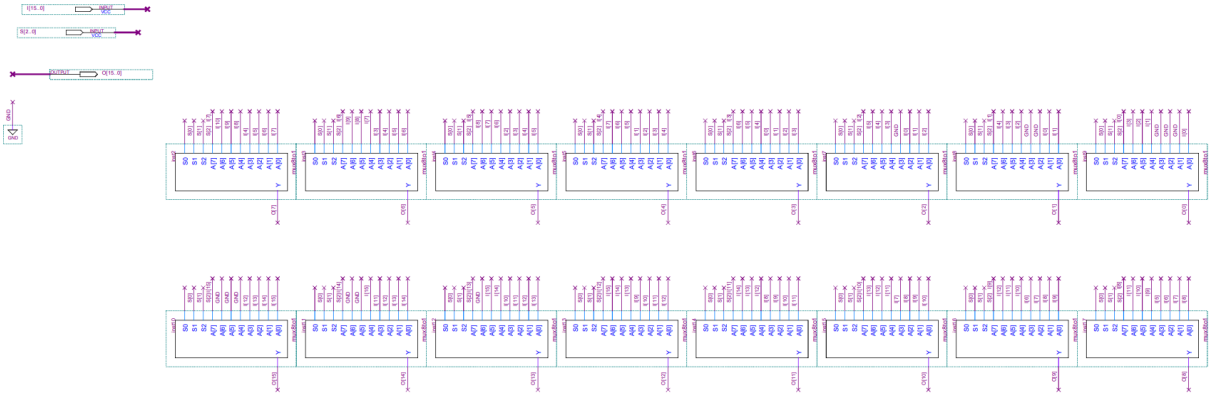
Hình 17. Khối IO controller

Zero Flag:



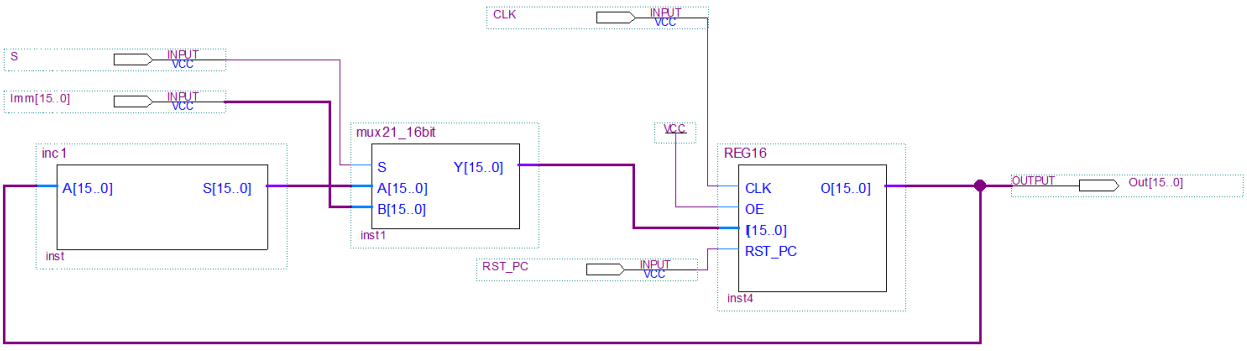
Hình 18. Khối cờ zero

Shifter:

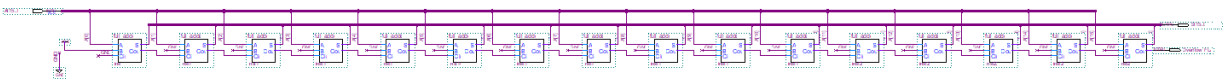


Hình 19. Khối Shifter

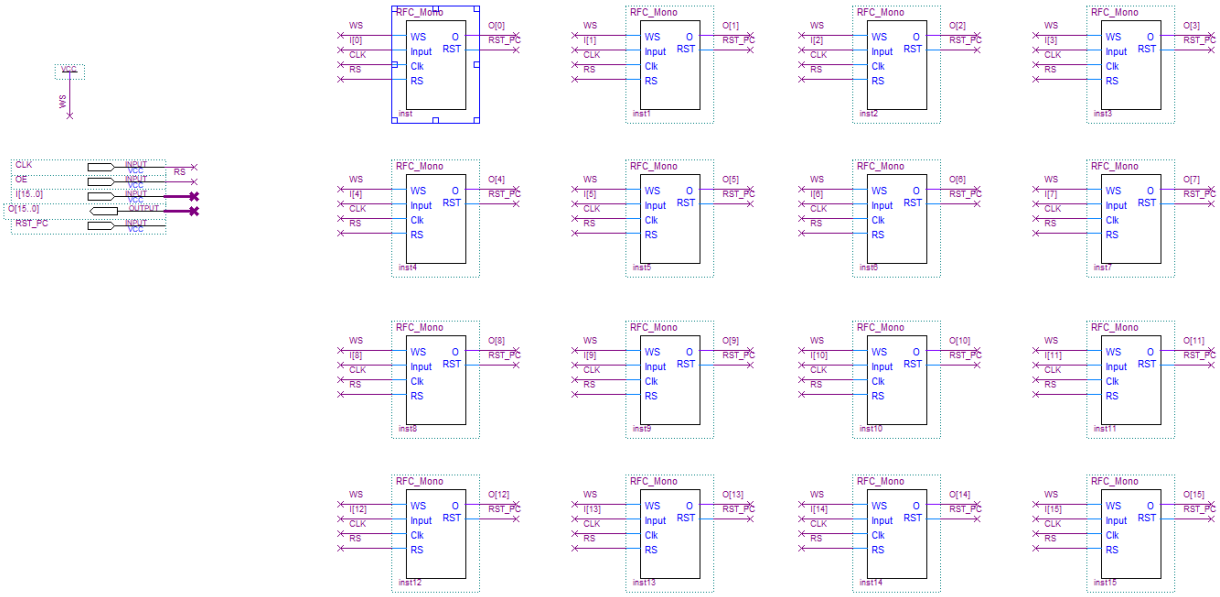
Program counter (PC):



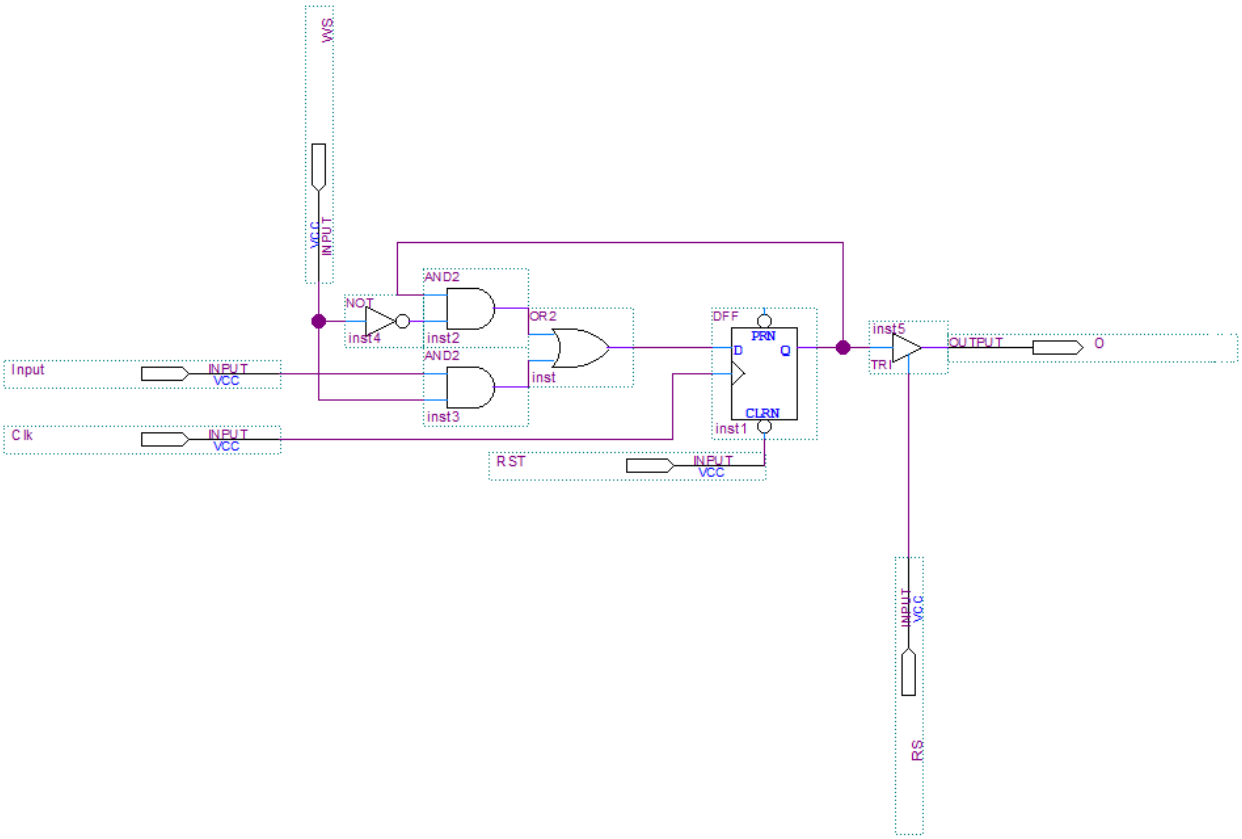
Hình 20. Khối PC



Hình 21. Khối inc 1bit

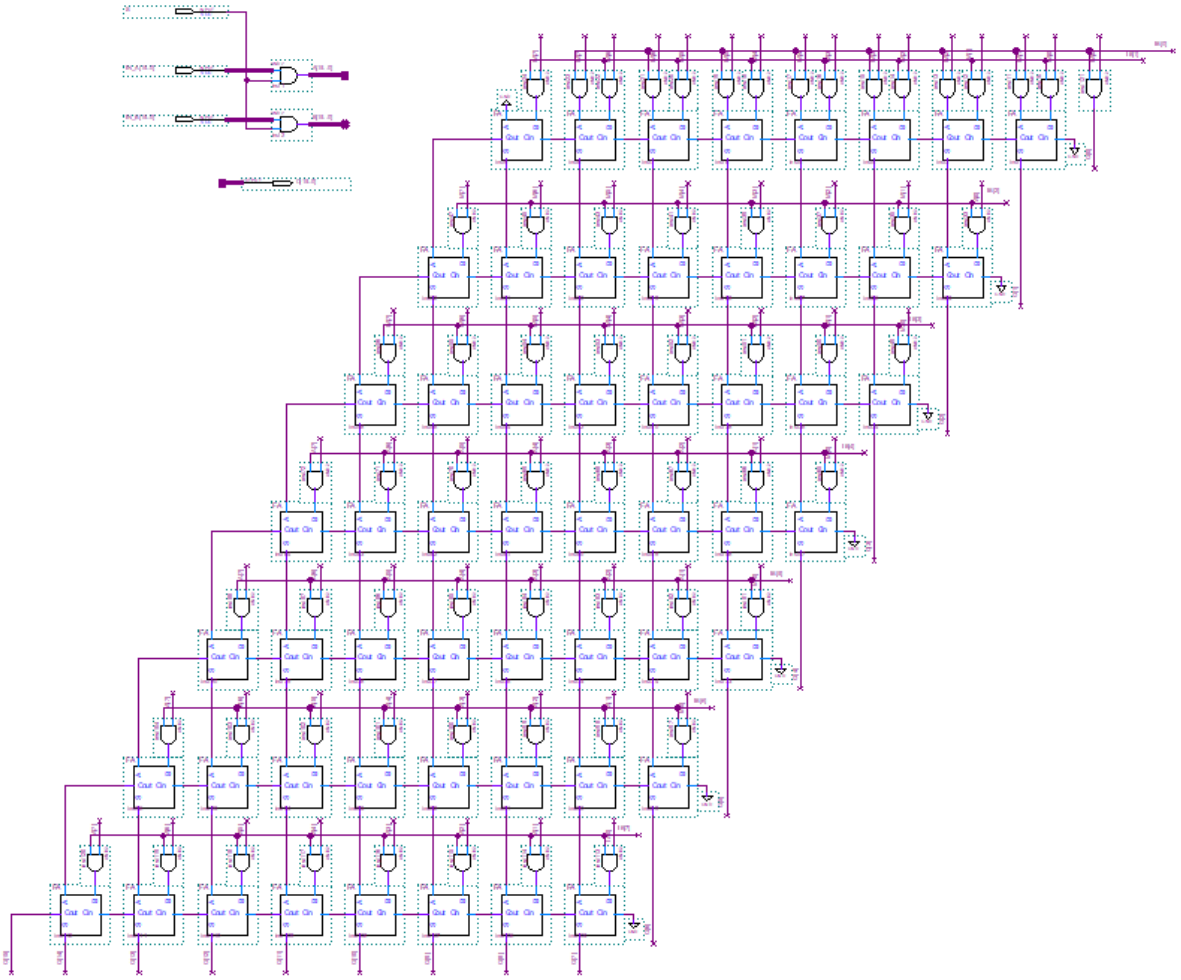


Hình 22. Khối Register file cell 16 bit



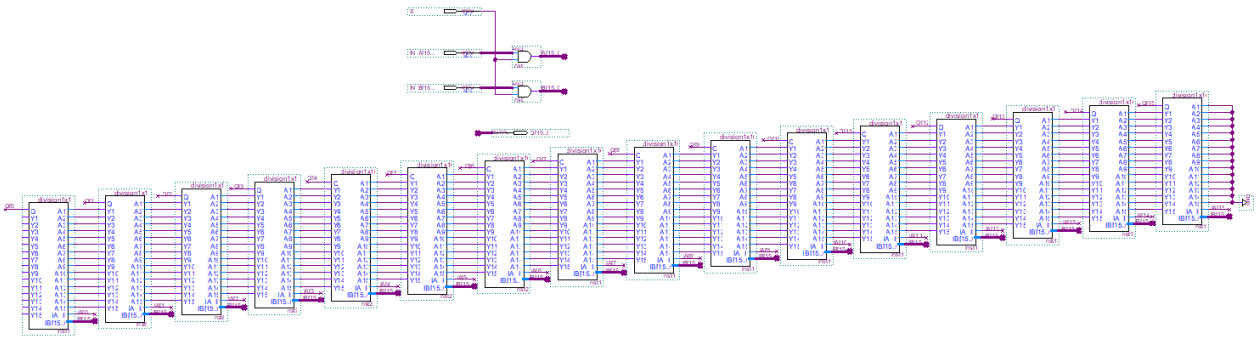
Hình 23. Khối Register file cell

Bộ nhân 8 bit:

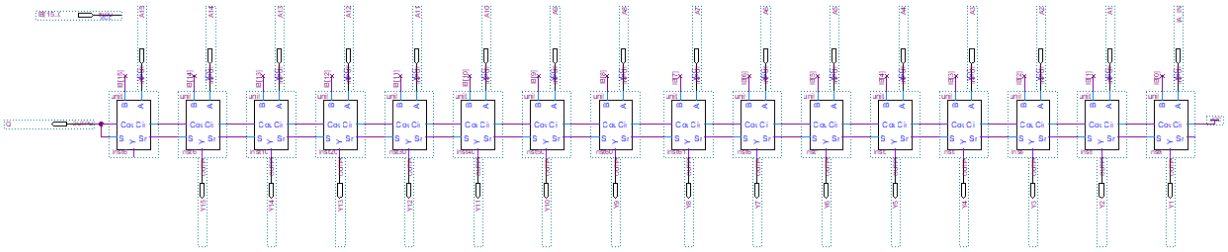


Hình 24. Khối nhân 8 bit

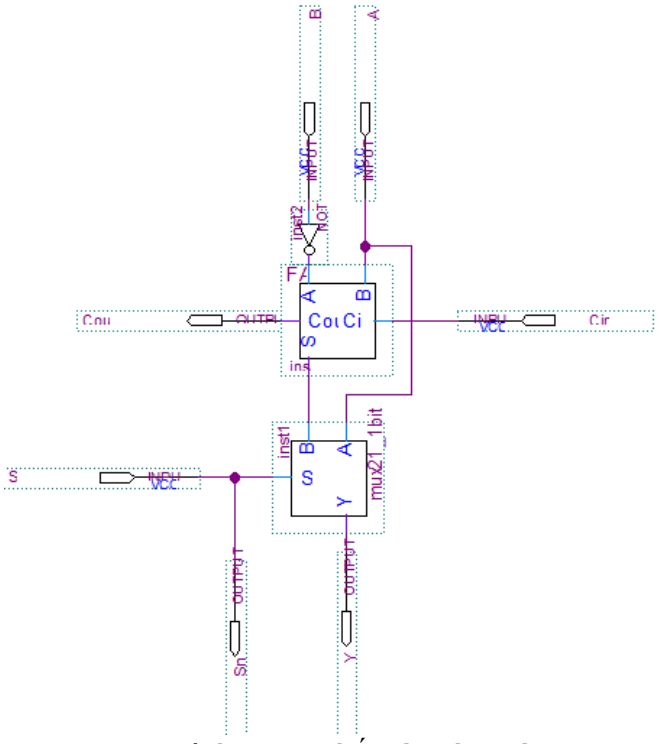
Bộ chia 16 bit:



Hình 25. Khối 16 bit chia 16 bit



Hình 26. Khối 1 bit chia 16 bit



Hình 27. Khối 1 bit chia 1 bit

2. Chức năng cho các khối trong datapath

Khối 8 thanh ghi 16 bit với các tín hiệu điều khiển:

Khối 8 thanh ghi 16-bit với các tín hiệu điều khiển có chức năng ghi và đọc giá trị của 8 thanh ghi khác nhau đồng thời có chức năng đọc giá trị 2 thanh ghi cùng lúc. Với các tín hiệu RSA, RSB, WS (0->7) điều khiển chọn thanh ghi được đọc hoặc ghi giá trị (từ thanh ghi 0 đến thanh ghi 7); các tín hiệu RE, WE điều khiển việc cho phép đọc hay ghi thanh ghi, RE điều khiển việc cho phép đọc thanh ghi và WE điều khiển việc cho phép ghi thanh ghi.

Chức năng các khối con:

- **Khối 8 thanh ghi 16 bit**: Khối 8 thanh ghi 16-bit cũng có chức năng ghi và đọc giá trị của 8 thanh ghi khác nhau đồng thời có chức năng đọc giá trị 2 thanh ghi cùng lúc. Vì không có các tín hiệu điều khiển nên khối 8 thanh ghi 16-bit nên chọn thanh ghi để ghi và đọc theo chuỗi 8 bit (00000001, 00000010, 00000100, 00001000, 00010000, 00100000, 01000000, 10000000) cho thanh ghi 0 đến thanh ghi 7. Bên cạch đó chuỗi 00000000 thực hiện việc không cho phép đọc/ghi giá trị thanh ghi. Khi kết hợp khối 8 thanh ghi 16 bit với các encoder 3to8 thì ta có thể tạo nên khối 8 thanh ghi 16 bit có tín hiệu điều khiển.
- **Khối thanh ghi 16 bit:** Khối thanh ghi 16 bit có chức năng đọc và ghi giá trị của 1 thanh ghi 16 bit. Đây là thành phần để tạo nên khối 8 thanh ghi 16 bit.
- **Khối thanh ghi 1 bit:** Khối thanh ghi có chức năng ghi và đọc giá trị của 1 thanh ghi 1 bit. Đây là thành phần để tạo nên khối thanh ghi 16 bit.

Khối ALU:

Khối ALU có chức năng thực hiện các phép toán và phép logic với hai giá trị đầu vào 16 bit A và B cùng với tín hiệu S lựa chọn phép toán hay phép logic được sử dụng để cho ra kết quả ở đầu ra 16 bit Y. Khối ALU thực hiện 8 phép toán và logic lần lượt là cộng, tăng 1 giá trị, trừ ,trừ 1 giá trị, and, or, xor, nand với tín hiện S từ 0->7. Có kèm cờ báo tràn của các phép tính. Chức năng của các khối con:

- **Khối AU:** Khối AU có chức năng thực hiên các phép toán với hai giá trị đầu vào 16 bit A và B cùng với tín hiệu S lựa chọn phép toán được sử dụng để cho ra kết quả ở đầu ra Y. Khối AU thực hiện các phép toán cộng, cộng 1 giá trị, trừ, trừ 1 giá trị với tín hiện điều khiển S từ 0->3. Đây là 1 trong 2 khối chính trong khối ALU.
- **Khối Full Adder 16 bit:** Khối Full Adder 16 bit có chức năng thực hiện phép cộng 2 số 16 bit. Đây là thành phần chính trong khối AU.

- **Khối Full Adder 1 bit:** Khối Full Adder 1 bit có chức năng cộng 2 số 1 bit. Đây là thành phần của khối Full Adder 16 bit.
- **Khối LU:** Khối LU có chức năng thực hiên các phép logic với hai giá trị đầu vào 16 bit A và B cùng với tín hiệu S lựa chọn phép toán được sử dụng để cho ra kết quả ở đầu ra Y. Khối AU thực hiện các phép toán and, or, xor, nand với tín hiện điều khiển S từ 0->3. Đây là 1 trong 2 khối chính trong khối ALU.

Khối Dmem 32x16:

Khối Dmem 32x16 có chức năng lưu trữ 32 giá trị 16 bit cho phép đọc và ghi các giá trị đó. Từ tín hiệu EN_RAM cho phép sử dụng khối Dmem 32x16, ADDR chọn vị trí thực hiện đọc/ghi giá trị của khối Dmem 32x16, RWS chọn chức năng đọc hay ghi giá trị khối Dmem 32x16. Chức năng của các khối con:

- **Khối Dmem 16:** Khối Dmem 16 có chức năng lưu trữ 1 giá trị 16 bit cho phép đọc và ghi các giá trị đó. Từ tín hiệu WE cho phép ghi giá trị khối Dmem 16, RW chọn chức năng đọc hay ghi giá trị khối Dmem 16. Đây là thành phần để tạo nên khối Dmem32x16.
- **Khối Dmem:** Khối Dmem có chức năng lưu trữ 1 giá trị 1 bit cho phép đọc và ghi các giá trị đó. Từ tín hiệu WE cho phép ghi giá trị khối Dmem, RW chọn chức năng đọc hay ghi giá trị khối Dmem. Đây là thành phần để tạo nên khối Dmem 16.

Khối IO controller:

Khối IO controller cho phép người dùng nhập dữ liệu vào thanh ghi hoặc xuất dữ liệu từ thanh ghi ra cho người dùng theo dõi.

Khối cờ Zero:

Khối cờ Zero có chức năng kiểm tra kết quả đầu ra của khối ALU có bằng 0 hay không. Nếu bằng 0 thì cờ Zero = 1, ngược lại thì cờ Zero = 0.

Khối Shifter:

Khối Shifter có chức năng thực hiện dịch trái hoặc phải giá trị của thanh ghi với tín hiện S. Nếu giá trị của tín hiệu S từ 1 đến 3 thì thực hiện dịch trái 1 bit đến 3 bit tương ứng, còn nếu giá trị của tín hiệu S từ 4 đến 6 thì thực hiện dịch phải 1 bit đến 3 bit tương ứng. Khi tín hiệu S = 0 thì giá trị khi đi qua khối Shifter sẽ không đổi.

Khối PC:

Khối PC thực hiện chức năng đếm trạng thái hiện tại của chương trình, cho biết được CPU đang chạy vị trí nào trong chương trình. Từ tín hiệu S cho biết giá trị tiếp theo của PC (vị trí của câu lệnh thực hiến kế tiếp) bằng PC + 1 hoặc Immediate. Chức năng của các khối con:

- **Khối inc 1 bit:** Khối inc 1 bit có chức năng tăng giá trị của thanh ghi 16 bit lên 1 bit để thực hiện tăng giá trị PC lên 1(PC = PC + 1)
- **Khối Register flie cell 16 bit:** Khối Register file cell 16 bit có chức năng kiểm soát đầu ra của khối PC giới hạn 1 xung clock chỉ có 1 output của PC.
- **Khối Register file cell:** Khối Register file cell có chức năng kiểm soát tín hiệu đầu ra 1 bit. Đây là thành phần cho khối Register file cell 16 bit.

Khối Nhân 8 bit:

Khối nhân 8 bit có chức năng thực hiện phép nhân 2 số 8 bit.

Khối 16 bit chia 16 bit:

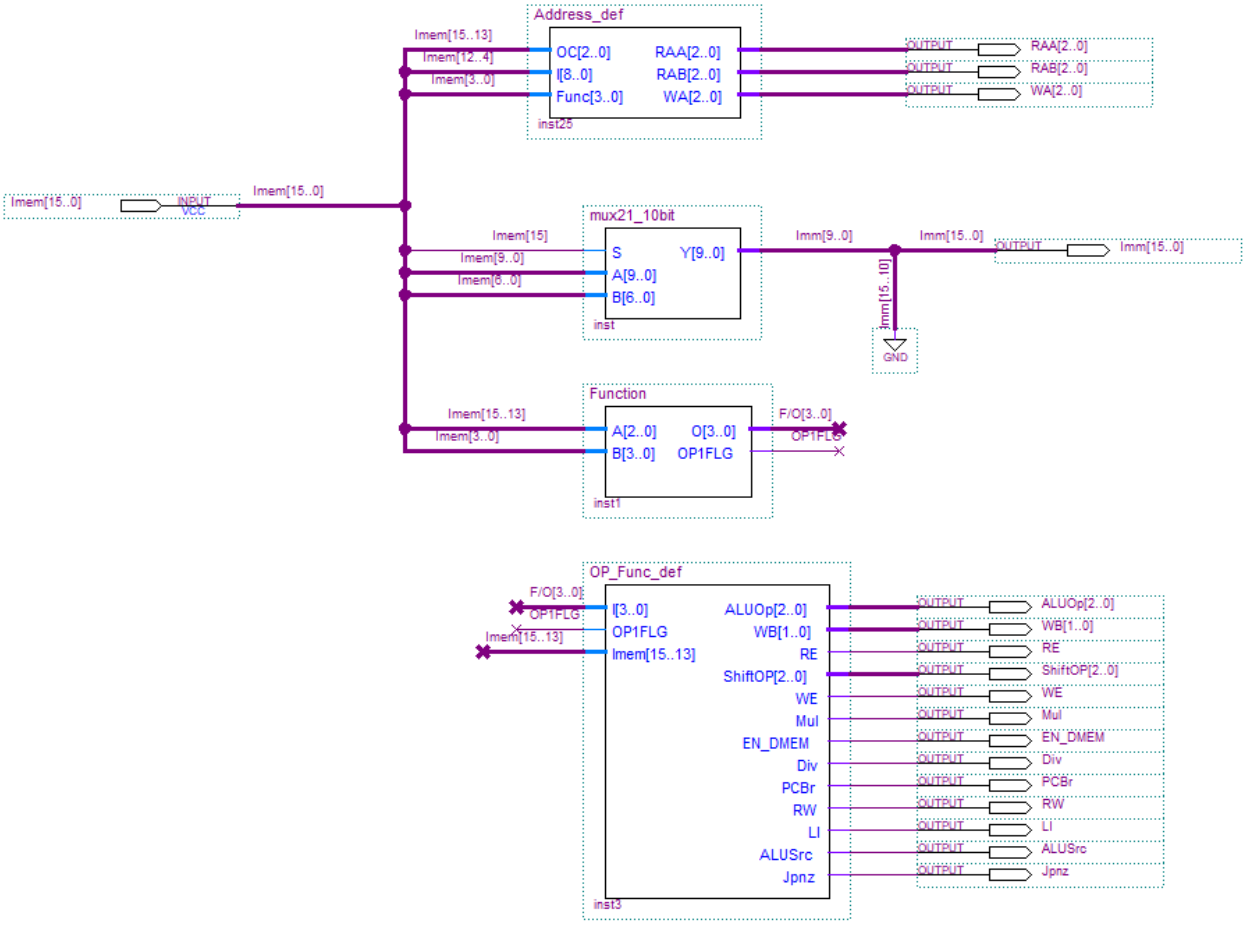
Khối 16 bit chia 16 bit có chức năng thực hiện phép chia 2 số 16 bit. Chức năng các khối con:

- **Khối 1 bit chia 16 bit:** Khối 1 bit chia 16 bit có chức năng thực hiện phép chia số 1 bit cho số 16 bit. Đây là thành phần cho khối 16 bit chia 16 bit.
- **Khối 1 bit chia 1 bit:** Khối 1 bit chia 1 bit có chức năng thực hiện phép chia số 1 bit cho số 1 bit. Đây là thành phần cho khối 1 bit chia 16 bit.

C. THIẾT KẾ CONTROLLER

OC[3]	R1[3]	R2[3]	R2[3]	Func[4]		RE	RAA	RAB	WE	WA	EN	RW	ALUSrc	ALUOp	Shif	PCBr	Jpnz	WB	Mul	Div	li
000	Rs1	Rs2	Rd	0000	add	1	Rs1	Rs2	1	Rd	0	x	0	0	0	0	0	1	0	0	0
000	Rs1	Rs2	Rd	0001	sub	1	Rs1	Rs2	1	Rd	0	x	0	2	0	0	0	1	0	0	0
000	Rs1	Rs2	Rd	0010	inc	1	Rs1	Rs2	1	Rd	0	x	0	1	0	0	0	1	0	0	0
000	Rs1	Rs2	Rd	0011	dec	1	Rs1	Rs2	1	Rd	0	x	0	3	0	0	0	1	0	0	0
000	Rs1	Rs2	Rd	0100	and	1	Rs1	Rs2	1	Rd	0	x	0	4	0	0	0	1	0	0	0
000	Rs1	Rs2	Rd	0101	or	1	Rs1	Rs2	1	Rd	0	x	0	5	0	0	0	1	0	0	0
000	Rs1	Rs2	Rd	0110	xor	1	Rs1	Rs2	1	Rd	0	x	0	6	0	0	0	1	0	0	0
000	Rs1	Rs2	Rd	0111	nand	1	Rs1	Rs2	1	Rd	0	x	0	7	0	0	0	1	0	0	0
000	Rs1	Rs2	Rd	1000	shfl	1	Rs1	Rs1	1	Rd	0	x	0	4	1	0	0	1	0	0	0
000	Rs1	Rs2	Rd	1001	shfr	1	Rs1	Rs1	1	Rd	0	x	0	4	4	0	0	1	0	0	0
000	Rs1	Rs2	Rd	1010	shll	1	Rs1	Rs1	1	Rd	0	x	0	4	2	0	0	1	0	0	0
000	Rs1	Rs2	Rd	1011	shrr	1	Rs1	Rs1	1	Rd	0	x	0	4	5	0	0	1	0	0	0
000	Rs1	Rs2	Rd	1100	slll	1	Rs1	Rs1	1	Rd	0	x	0	4	3	0	0	1	0	0	0
000	Rs1	Rs2	Rd	1101	srrr	1	Rs1	Rs1	1	Rd	0	x	0	4	6	0	0	1	0	0	0
000	Rs1	Rs2	Rd	1110	mul	1	Rs1	Rs2	1	Rd	0	x	0	4	0	0	0	1	1	0	0
000	Rs1	Rs2	Rd	1111	div	1	Rs1	Rs2	1	Rd	0	x	0	4	0	0	0	1	0	1	0
OC[3]	R1[3]	R2[3]	Imm[7]																		
001	Rs1	Rd	Imm		addi	1	Rs1	Rd	1	Rd	0	x	1	0	0	0	0	1	0	0	0
010	Rs1	Rd	Imm		subi	1	Rs1	Rd	1	Rd	0	x	1	2	0	0	0	1	0	0	0
011	Rs1	Rd	Imm		beq	1	Rs1	Rd	0	Rd	0	x	0	2	0	1	0	0	0	0	0
OC[3]	R1[3]	Imm[10]																			
100	Rd	imm			lw	0	Rd	Rd	1	Rd	1	0	0	4	0	0	0	2	0	0	0
101	Rd	imm			sw	1	Rd	Rd	0	Rd	1	1	0	4	0	0	0	0	0	0	0
110	Rd	imm			jpnz	1	Rd	Rd	0	Rd	0	x	0	4	0	1	1	0	0	0	0
111	Rd	imm			li	0	Rd	Rd	1	Rd	0	x	0	0	0	0	0	1	0	0	1

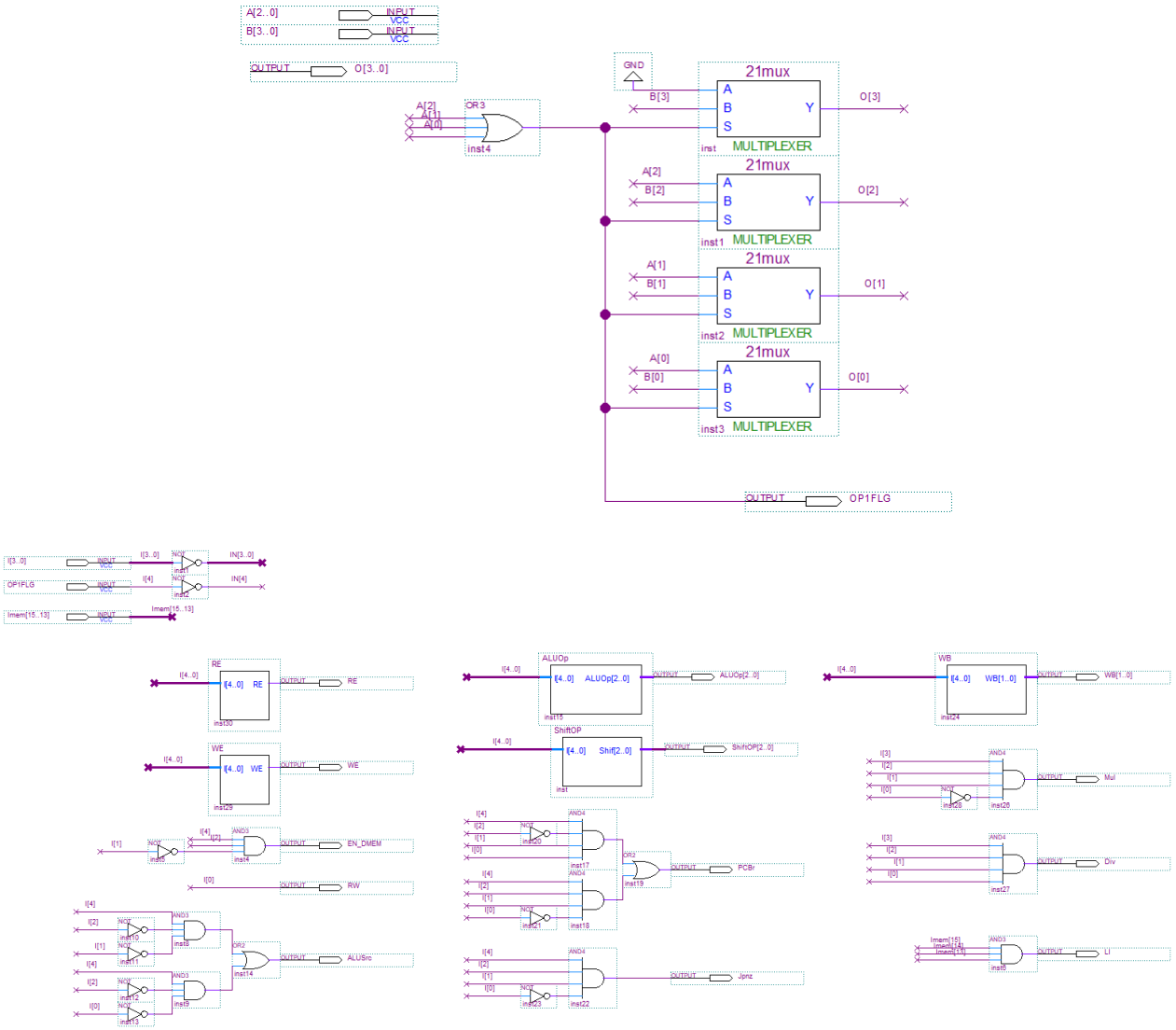
Bảng phân tích tính hiệu đầu ra của controller



Hình 28. Khối Controller

1. Opcode + Function define

Từ Opcode và function của các lệnh cho ra các tín hiện điều khiển các khối ở Datapath thực hiện chức năng tương ứng với các lệnh.



Hình 29. Khối Opcode & Function

- Register file:
  - ReadEnable:
    - 0: Không cho phép đọc giá trị của thanh ghi trong Register file
    - 1: Cho phép đọc giá trị của thanh ghi trong Register file
  - WriteEnable
    - 0: Không cho phép ghi kết quả vào thanh ghi trong Register file
    - 1: Cho phép ghi kết quả vào thanh ghi trong Register file
- ALU:
  - ALUOp:
    - 0: Add

        - 1: Inc
        - 2: Sub
        - 3: Dec
        - 4: And
        - 5: Or
        - 6: Xor
        - 7: Nand
    - ALUSrc:
        - 0: Nguồn thanh ghi B từ Register file
        - 1: Nguồn thanh ghi B từ Immediate
- Dmem:
    - Enable:
        - 0: Không cho phép đọc/ghi giá trị ở Dmem
        - 1: Cho phép đọc/ghi giá trị ở Dmem
    - Read/Write: 0 – Read 1 – Write
- PC:
    - PC <- Imm
        - PCBr = 1, jpnz = 0, ZERO_flag = 1
        - PCBr = 1, jpnz = 1, ZERO_flag = 0
    - PC <- PC + 1
- Write back (Mux 4):
    - 0: No result
    - 1: Result
    - 2: Dmem
- Multiplier(mul):

- 0: Không thực hiện phép nhân
- 1: Thực hiện phép nhân

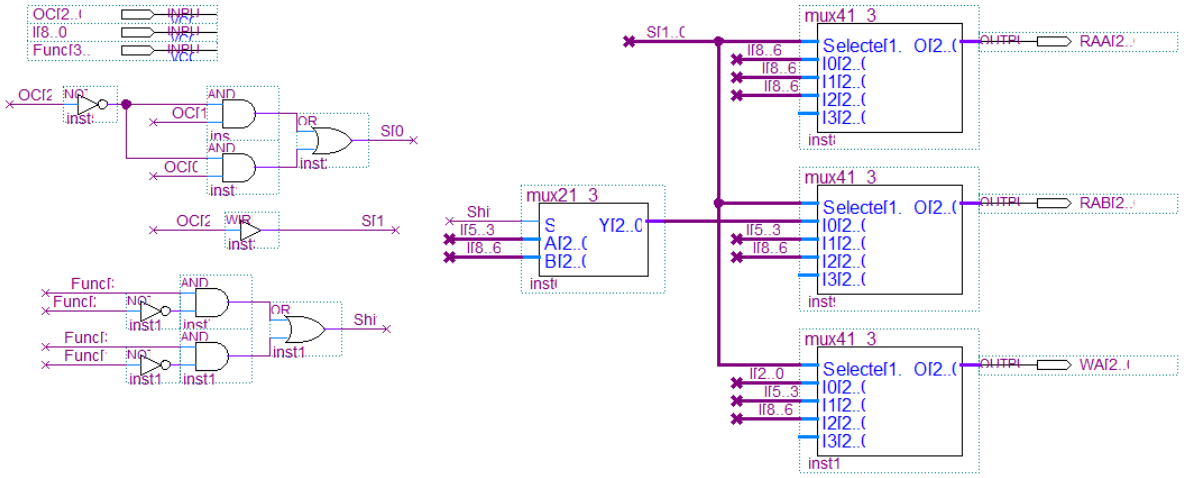
- Division(div):
  - 0: Không thực hiện phép chia
  - 1: Thực hiện phép chia

- Mux 2(li):
  - 0: Kết quả trả về của Result là từ ALU
  - 1: Kết quả trả về của Result là từ Imm

- Shifter:
  - 0: Không thay đổi
  - 1: Shift Left 1bit
  - 2: Shift Left 2bit
  - 3: Shift Left 3bit
  - 4: Shift Right 1bit
  - 5: Shift Right 2bit
  - 6: Shift Right 3bit

2. Address define:

Từ các tập lệnh khác nhau sẽ cho ra ReadAddressA, ReadAddressB, WriteAddress khác nhau cho vào Register file:

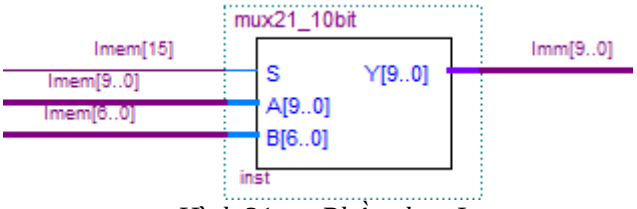


Hình 30. Khối Address define

- RRR:
  - RAA = Rs1
  - RAB = Rs2 (hoặc = Rs1 nếu là lệnh Shifter)
  - WA = Rd
- RRI:
  - RAA = Rs1
  - RAB = Rd
  - WA = Rd
- RI:
  - RAA = Rd
  - RAB = Rd
  - WA = Rd

3. Immediate define

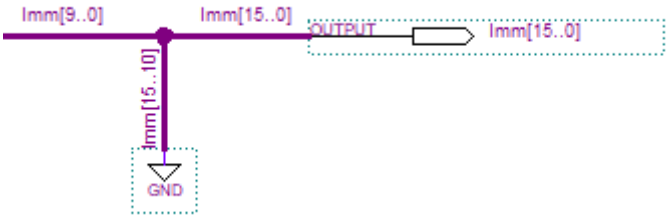
Chọn imm của tập lệnh RRI hay RI.



Hình 31. Phần chọn Imm

4. Immediate extend

Mở rộng imm từ 10 thành 16 bit.



Hình 32. Phần mở rộng bit

D. THIẾT KẾ CHƯƠNG TRÌNH

1. Xử lí phần mềm

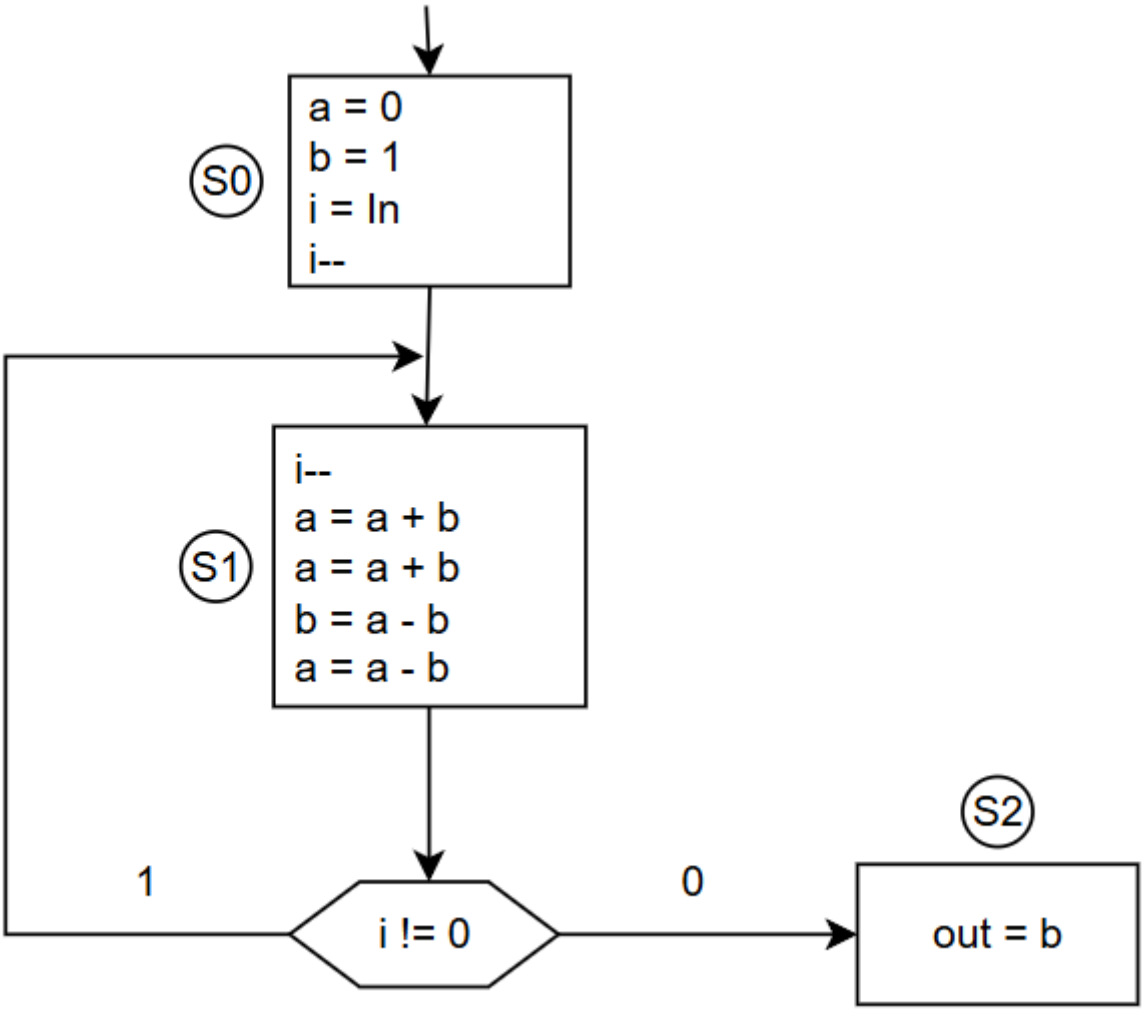
a. Tập lệnh có thể biên dịch bằng phần mềm

```
lw R6, 511
mov R7, 1023
add R1, R2, R0
sub R1, R2, R0
inc R1, R2, R0
dec R1, R2, R0
and R1, R2, R0
or R1, R2, R0
xor R1, R2, R0
nand R1, R2, R0
shfl R1, R1, R0
shfr R1, R1, R0
shll R1, R1, R0
shrr R1, R1, R0
slll R1, R1, R0
srrr R1, R1, R0
mul R1, R2, R0
div R1, R2, R0
addi R1, R0, 100
subi R1, R0, 100
beq R1, R0, 100
lw R0, 100
sw R0, 100
jpnz R0, 100
li R0, 100
not R1, R2
jmp 123
mov R1, R2
not R1, R0
jmp 100
```

b. Xây dựng code asm fibonaci

```
li R0, 0
li R1, 1
lw R7, 32
subi R7, R7, 1
dec R7, R7
add R0, R1, R0
add R0, R1, R0
sub R0, R1, R1
sub R0, R1, R0
jpnz R7, 4
sw R1, 32
jmp 10
```

2. Lưu đồ ASM



Hình 33. Lưu đồ ASM

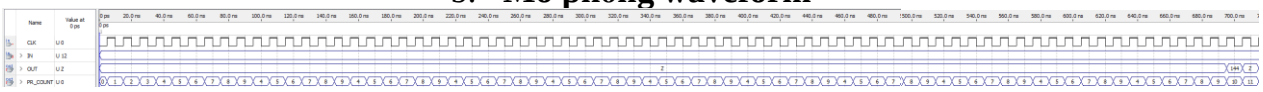
E. KIỂM THỬ

1. Kiểm tra chức năng

a. Nạp code

Sau khi biên dịch file .asm bằng trình biên dịch, kết quả thu được bao gồm ba file: file .bin, file .hex, và file .log. Đầu tiên, file .bin chứa mã nhị phân đã được biên dịch, file .hex chứa mã thập lục phân, và file .log là bản ghi chép quá trình biên dịch, bao gồm các thông tin về lỗi và cảnh báo nếu có. Những file này là kết quả của quá trình biên dịch và sẽ được sử dụng trong các bước tiếp theo để kiểm tra và xác minh thiết kế.

b. Mô phỏng waveform



Hình 34. Mô phỏng waveform chương trình

file này được đặt đúng vị trí để phần mềm có thể nhận diện và sử dụng.

c. Kiểm tra chức năng

Tiến hành kiểm tra chức năng của từng lệnh. Quá trình này là việc kiểm tra từng lệnh trong code đã biên dịch hoạt động đúng như thiết kế. Bằng cách so sánh kết quả đầu ra với kết quả dự kiến. Mục đích là để đảm bảo rằng mỗi lệnh đều hoạt động chính xác.

2. Kiểm tra tài nguyên

	Resource	Usage
1	Estimated Total logic elements	538
2		
3	Total combinational functions	490
4	⌄ Logic element usage by number of LUT inputs	
1	-- 4 input functions	339
2	-- 3 input functions	125
3	-- <=2 input functions	26
5		
6	⌄ Logic elements by mode	
1	-- normal mode	490
2	-- arithmetic mode	0
7		
8	⌄ Total registers	72
1	-- Dedicated logic registers	72
2	-- I/O registers	0
9		
10	I/O pins	56
11	Embedded Multiplier 9-bit elements	0
12	Maximum fan-out node	ONCHIP_ROM:inst5|my_mem~3
13	Maximum fan-out	76
14	Total fan-out	2053
15	Average fan-out	3.32

Hình 35. Kết quả của chạy kiểm tra tài nguyên

3. Kiểm tra định thời

a. Trường hợp chưa nạp code

	Fmax	Restricted Fmax	Clock Name	Note
1	533.33 MHz	450.05 MHz	CLK	limit due to minimum period restriction (max I/O toggle rate)

	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay
1	-0.875	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst2|inst1	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst7|inst1	CLK	CLK	1.000	0.000	1.911
2	-0.814	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst3|inst1	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst7|inst1	CLK	CLK	1.000	0.000	1.850
3	-0.717	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst1|inst1	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst7|inst1	CLK	CLK	1.000	0.000	1.753
4	-0.571	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst|inst1	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst7|inst1	CLK	CLK	1.000	0.000	1.607
5	-0.510	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst|inst1	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst3|inst1	CLK	CLK	1.000	0.000	1.546
6	-0.487	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst2|inst1	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst4|inst1	CLK	CLK	1.000	0.000	1.523
7	-0.483	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst2|inst1	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst6|inst1	CLK	CLK	1.000	0.000	1.519
8	-0.481	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst2|inst1	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst5|inst1	CLK	CLK	1.000	0.000	1.517
9	-0.454	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst6|inst1	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst7|inst1	CLK	CLK	1.000	0.000	1.490
10	-0.436	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst5|inst1	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst7|inst1	CLK	CLK	1.000	0.000	1.472
11	-0.426	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst3|inst1	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst4|inst1	CLK	CLK	1.000	0.000	1.462
12	-0.422	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst3|inst1	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst6|inst1	CLK	CLK	1.000	0.000	1.458
13	-0.420	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst3|inst1	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst5|inst1	CLK	CLK	1.000	0.000	1.456
14	-0.331	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst4|inst1	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst7|inst1	CLK	CLK	1.000	0.000	1.367
15	-0.329	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst1|inst1	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst4|inst1	CLK	CLK	1.000	0.000	1.365
16	-0.325	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst1|inst1	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst6|inst1	CLK	CLK	1.000	0.000	1.361
17	-0.323	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst1|inst1	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst5|inst1	CLK	CLK	1.000	0.000	1.359
18	-0.231	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst1|inst1	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst2|inst1	CLK	CLK	1.000	0.000	1.267
19	-0.204	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst|inst1	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst2|inst1	CLK	CLK	1.000	0.000	1.240
20	-0.201	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst|inst1	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst1|inst1	CLK	CLK	1.000	0.000	1.237
21	-0.183	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst|inst1	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst4|inst1	CLK	CLK	1.000	0.000	1.219
22	-0.179	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst|inst1	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst6|inst1	CLK	CLK	1.000	0.000	1.215
23	-0.177	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst|inst1	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst5|inst1	CLK	CLK	1.000	0.000	1.213
24	-0.119	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst4|inst1	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst5|inst1	CLK	CLK	1.000	0.000	1.155
25	-0.118	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst4|inst1	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst6|inst1	CLK	CLK	1.000	0.000	1.154
26	-0.070	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst5|inst1	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst6|inst1	CLK	CLK	1.000	0.000	1.106
27	-0.058	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst1|inst1	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst3|inst1	CLK	CLK	1.000	0.000	1.094
28	0.043	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst2|inst1	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst3|inst1	CLK	CLK	1.000	0.000	0.993
29	0.379	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst|inst1	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst|inst1	CLK	CLK	1.000	0.000	0.657
30	0.379	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst7|inst1	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst7|inst1	CLK	CLK	1.000	0.000	0.657
31	0.379	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst3|inst1	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst3|inst1	CLK	CLK	1.000	0.000	0.657
32	0.379	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst4|inst1	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst4|inst1	CLK	CLK	1.000	0.000	0.657
33	0.379	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst6|inst1	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst6|inst1	CLK	CLK	1.000	0.000	0.657
34	0.379	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst5|inst1	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst5|inst1	CLK	CLK	1.000	0.000	0.657
35	0.379	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst2|inst1	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst2|inst1	CLK	CLK	1.000	0.000	0.657
36	0.379	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst1|inst1	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst1|inst1	CLK	CLK	1.000	0.000	0.657

Hình 36. Kiểm tra định thời chưa nạp code

b. Trường hợp đã nạp code tính 12 số Fibonacci đầu tiên

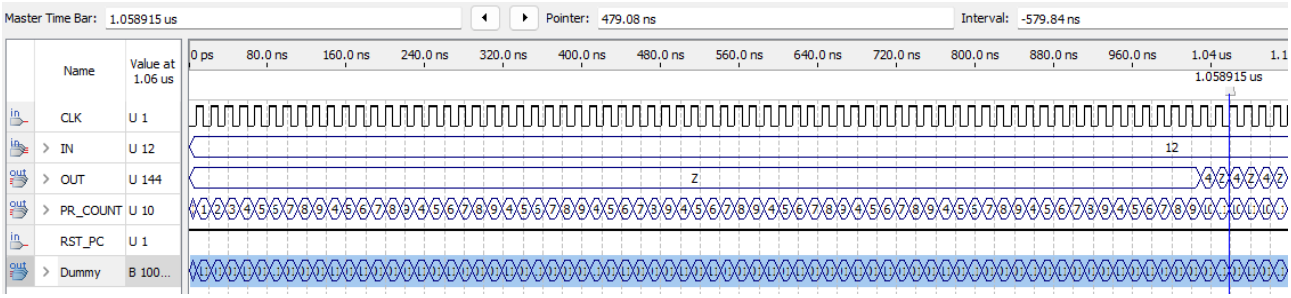
	Fmax	Restricted Fmax	Clock Name	Note
1	67.52 MHz	67.52 MHz	CLK	

Hình 37. Fmax

	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay
1	-13.811	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst6|inst1	DatapathCPU:inst|register_file_16bx8_final:inst|reg..._16x8:inst1|register_2:inst17|register:inst12|inst1	CLK	CLK	1.000	-0.447	14.400
2	-13.806	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst6|inst1	DatapathCPU:inst|register_file_16bx8_final:inst|reg..._16x8:inst1|register_2:inst16|register:inst12|inst1	CLK	CLK	1.000	-0.447	14.395
3	-13.654	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst6|inst1	DatapathCPU:inst|register_file_16bx8_final:inst|reg..._16x8:inst1|register_2:inst17|register:inst10|inst1	CLK	CLK	1.000	-0.441	14.249
4	-13.646	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst6|inst1	DatapathCPU:inst|register_file_16bx8_final:inst|reg..._16x8:inst1|register_2:inst12|register:inst11|inst1	CLK	CLK	1.000	-0.441	14.241
5	-13.644	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst6|inst1	DatapathCPU:inst|register_file_16bx8_final:inst|reg..._16x8:inst1|register_2:inst16|register:inst10|inst1	CLK	CLK	1.000	-0.447	14.233
6	-13.643	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst6|inst1	DatapathCPU:inst|register_file_16bx8_final:inst|reg..._16x8:inst1|register_2:inst11|register:inst10|inst1	CLK	CLK	1.000	-0.447	14.232
7	-13.614	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst6|inst1	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst2|inst1	CLK	CLK	1.000	-0.238	14.412
8	-13.613	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst6|inst1	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst3|inst1	CLK	CLK	1.000	-0.238	14.411
9	-13.610	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst6|inst1	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst1|inst1	CLK	CLK	1.000	-0.238	14.408
10	-13.589	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst6|inst1	DatapathCPU:inst|register_file_16bx8_final:inst|reg..._16x8:inst1|register_2:inst17|register:inst11|inst1	CLK	CLK	1.000	-0.448	14.177
11	-13.586	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst6|inst1	DatapathCPU:inst|register_file_16bx8_final:inst|reg..._16x8:inst1|register_2:inst16|register:inst11|inst1	CLK	CLK	1.000	-0.447	14.175
12	-13.585	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst6|inst1	DatapathCPU:inst|register_file_16bx8_final:inst|reg..._16x8:inst1|register_2:inst11|register:inst12|inst1	CLK	CLK	1.000	-0.447	14.174
13	-13.562	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst7|inst1	DatapathCPU:inst|register_file_16bx8_final:inst|reg..._16x8:inst1|register_2:inst17|register:inst12|inst1	CLK	CLK	1.000	-0.447	14.151
14	-13.557	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst7|inst1	DatapathCPU:inst|register_file_16bx8_final:inst|reg..._16x8:inst1|register_2:inst16|register:inst12|inst1	CLK	CLK	1.000	-0.447	14.146
15	-13.487	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst6|inst1	DatapathCPU:inst|register_file_16bx8_final:inst|regi...le_16x8:inst1|register_2:inst17|register:inst9|inst1	CLK	CLK	1.000	-0.448	14.075
16	-13.483	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst6|inst1	DatapathCPU:inst|register_file_16bx8_final:inst|regi...le_16x8:inst1|register_2:inst16|register:inst9|inst1	CLK	CLK	1.000	-0.447	14.072
17	-13.466	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst4|inst1	DatapathCPU:inst|register_file_16bx8_final:inst|reg..._16x8:inst1|register_2:inst17|register:inst12|inst1	CLK	CLK	1.000	-0.447	14.055
18	-13.461	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst4|inst1	DatapathCPU:inst|register_file_16bx8_final:inst|reg..._16x8:inst1|register_2:inst16|register:inst12|inst1	CLK	CLK	1.000	-0.447	14.050
19	-13.447	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst2|inst1	DatapathCPU:inst|register_file_16bx8_final:inst|reg..._16x8:inst1|register_2:inst17|register:inst12|inst1	CLK	CLK	1.000	-0.209	14.274
20	-13.442	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst2|inst1	DatapathCPU:inst|register_file_16bx8_final:inst|reg..._16x8:inst1|register_2:inst16|register:inst12|inst1	CLK	CLK	1.000	-0.209	14.269
21	-13.429	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst6|inst1	DatapathCPU:inst|register_file_16bx8_final:inst|reg..._16x8:inst1|register_2:inst12|register:inst10|inst1	CLK	CLK	1.000	-0.447	14.018
22	-13.429	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst6|inst1	DatapathCPU:inst|register_file_16bx8_final:inst|reg..._16x8:inst1|register_2:inst12|register:inst13|inst1	CLK	CLK	1.000	-0.447	14.018
23	-13.428	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst6|inst1	DatapathCPU:inst|register_file_16bx8_final:inst|reg..._16x8:inst1|register_2:inst11|register:inst13|inst1	CLK	CLK	1.000	-0.447	14.017
24	-13.405	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst7|inst1	DatapathCPU:inst|register_file_16bx8_final:inst|reg..._16x8:inst1|register_2:inst17|register:inst10|inst1	CLK	CLK	1.000	-0.441	14.000
25	-13.404	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst6|inst1	DatapathCPU:inst|register_file_16bx8_final:inst|reg..._16x8:inst1|register_2:inst17|register:inst13|inst1	CLK	CLK	1.000	-0.447	13.993
26	-13.403	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst6|inst1	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst6|inst1	CLK	CLK	1.000	0.000	14.439
27	-13.403	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst6|inst1	DatapathCPU:inst|register_file_16bx8_final:inst|reg..._16x8:inst1|register_2:inst16|register:inst13|inst1	CLK	CLK	1.000	-0.447	13.992
28	-13.397	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst6|inst1	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst7|inst1	CLK	CLK	1.000	0.000	14.433
29	-13.397	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst6|inst1	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst5|inst1	CLK	CLK	1.000	0.000	14.433
30	-13.397	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst7|inst1	DatapathCPU:inst|register_file_16bx8_final:inst|reg..._16x8:inst1|register_2:inst12|register:inst11|inst1	CLK	CLK	1.000	-0.441	13.992
31	-13.396	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst6|inst1	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst4|inst1	CLK	CLK	1.000	0.000	14.432
32	-13.395	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst7|inst1	DatapathCPU:inst|register_file_16bx8_final:inst|reg..._16x8:inst1|register_2:inst16|register:inst10|inst1	CLK	CLK	1.000	-0.447	13.984
33	-13.394	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst7|inst1	DatapathCPU:inst|register_file_16bx8_final:inst|reg..._16x8:inst1|register_2:inst11|register:inst10|inst1	CLK	CLK	1.000	-0.447	13.983
34	-13.383	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst6|inst1	DatapathCPU:inst|register_file_16bx8_final:inst|regi...le_16x8:inst1|register_2:inst12|register:inst8|inst1	CLK	CLK	1.000	-0.455	13.964
35	-13.382	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst6|inst1	DatapathCPU:inst|register_file_16bx8_final:inst|regi...le_16x8:inst1|register_2:inst17|register:inst8|inst1	CLK	CLK	1.000	-0.455	13.963
36	-13.365	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst7|inst1	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst2|inst1	CLK	CLK	1.000	-0.238	14.163
37	-13.364	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst7|inst1	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst3|inst1	CLK	CLK	1.000	-0.238	14.162
38	-13.361	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst7|inst1	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst1|inst1	CLK	CLK	1.000	-0.238	14.159
39	-13.340	DatapathCPU:inst|program_counter:inst14|REG16:inst4|RFC_Mono:inst5|inst1	DatapathCPU:inst|register_file_16bx8_final:inst|reg..._16x8:inst1|register_2:inst17|register:inst12|inst1	CLK	CLK	1.000	-0.447	13.929

Hình 38. Kiểm tra định thời đã nạp code

4. Kiểm tra hiệu năng



Hình 39. Chạy mô phỏng waveform với Fmax

Thời gian thành chương trình là 1.058915 µs

5. Kiểm tra công suất

a. Phân tích công suất

PowerPlay Power Analyzer Status	Successful - Tue Dec 10 10:25:59 2024
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition
Revision Name	final_project
Top-level Entity Name	CPU
Family	Cyclone II
Device	EP2C35F672C6
Power Models	Final
Total Thermal Power Dissipation	115.56 mW
Core Dynamic Thermal Power Dissipation	0.00 mW
Core Static Thermal Power Dissipation	79.94 mW
I/O Thermal Power Dissipation	35.62 mW
Power Estimation Confidence	Low: user provided insufficient toggle rate data

Hình 40. Kết quả phân tích công suất

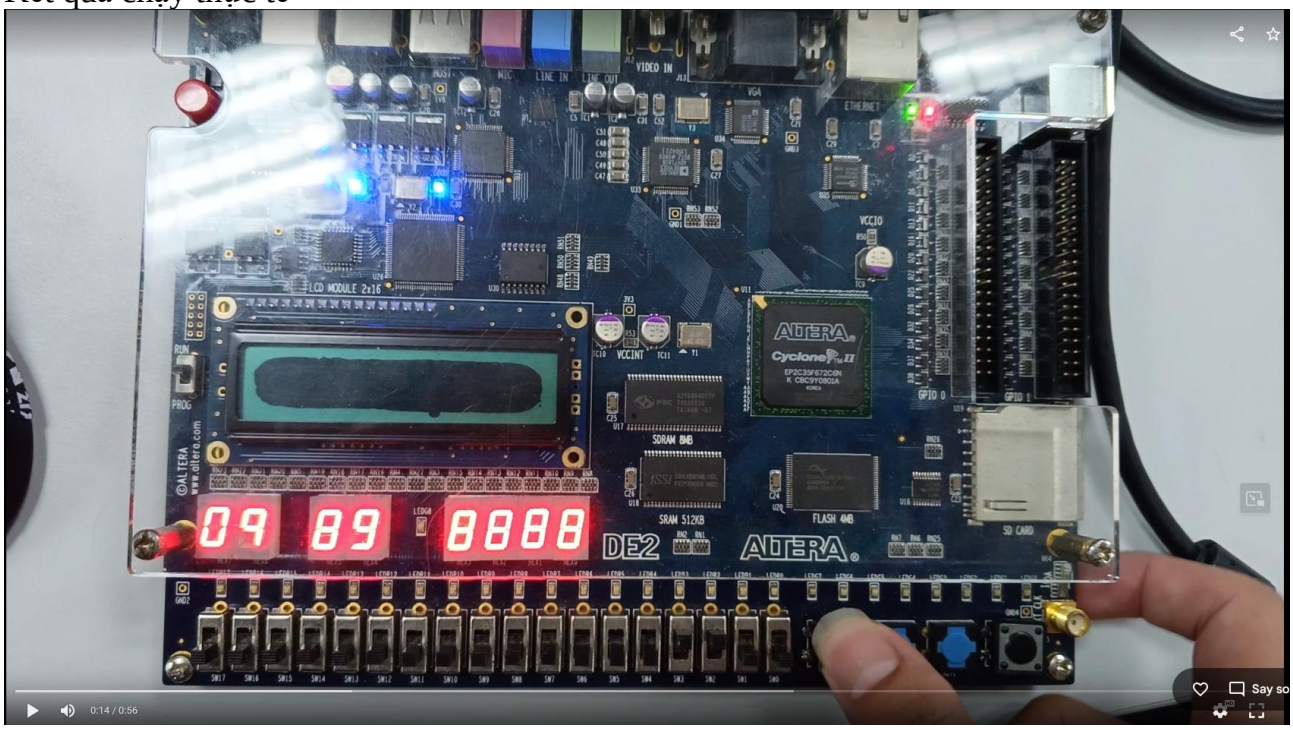
b. Report công suất

Công suất tổng cộng là 115.56 mW

F. TỔNG HỢP KẾT QUẢ

	Kết quả
Tài nguyên	538 thành phần logic
Tần số tối đa	533.33 MHz
Định thời	67.52 MHz
Công suất	115.56 mW
Hiệu năng	1.058915 µs

Kết quả chạy thực tế



Hình 41. Thực hiện trên kit ALTERA DE2

Link video demo

G. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN CỦA ĐỀ TÀI

1. Kết luận

Thiết kế thành công CPU 16 bit hiện thực các tập lệnh đơn giản và nhóm đã lập trình được chương trình tính toán được số fibonaci thứ n, lên tới số thứ 24.

2. Hướng phát triển

Áp dụng pipeline vào CPU, mở rộng tập lệnh, mở rộng số lượng bit CPU có thể xử lí từ 16bit thành 32bit.

HẾT.