Họ và tên: Giản Thanh Sang
MSSV: 22581238

BÁO CÁO FINAL LAB CE118.P13

Thiết kế CPU

1. Tập lệnh

- Lệnh cộng: thực hiện việc cộng 2 toán hạng nguồn và lưu kết quả vào một toán hạng đích
- Lệnh trừ: thực hiện việc trừ 2 toán hạng nguồn và lưu kết quả vào một toán hạng đích
- Lệnh AND: thực hiện việc AND từng bit của 2 toán hạng nguồn và lưu vào một toán hạng đích
- Lệnh OR: thực hiện việc OR từng bit của 2 toán hạng nguồn và lưu vào toán hạng vào một toán hạng đích
- Lệnh NAND: thực hiện việc NAND từng bit của 2 toán hạng nguồn và lưu vào toán hạng đích
- Lệnh dịch trái: thực hiện việc dịch trái toán hạng nguồn với một lượng dịch được xác định bởi toán hạng nguồn và lưu kết quả vào toán hạng đích
- Lệnh dịch phải: thực hiện việc dịch phải toán hạng nguồn với một lượng dịch được xác định bởi toán hạng nguồn và lưu kết quả vào toán hạng đích
- Lệnh nạp: thực hiện việc tìm nạp dữ liệu từ một địa chỉ bộ nhớ được xác định bởi toán hạng nguồn tới một thanh ghi (toán hạng đích)
- Lệnh lưu: thực hiện việc lưu dữ liệu được lưu trong một thanh ghi(toán hạng nguồn) tới một địa chỉ bộ nhớ được xác định bởi một toán hạng nguồn khác.
- Lệnh nhảy có điều kiện: thực hiện kiểm tra một điều kiện nào đó giữa các thanh ghi (các toán hạng nguồn), nếu điều kiện thỏa mãn thì PC sẽ nhảy tới một địa chỉ mới được tính theo công thức PC = PC + độ dời. Trong đó độ dời cũng được xem là toán hạng nguồn.

2. Định dạng lệnh

a. Định dạng lệnh R:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OPCODE					DR			SA			SB			X	X

Định dạng lệnh R có 3 trường ADDRESS được sử dụng cho các lệnh tính toán với dữ liệu được thực hiện toàn bộ trên các thanh ghi của Register File bao gồm: DR(các bit 10 - 9 - 8) cho toán hạng đích, SA(các bit 7 - 6 - 5 cho toán hạng nguồn thứ nhất, SB(các bit 4 - 3 - 2) cho toán hạng nguồn thứ 2). Hai bit còn lại không sử dụng

b. Định dạng lệnh I:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OPCODE					DR			SA			CONSTANT				

Định dạng lệnh I được sử dụng cho các lệnh sau:

- Lệnh truyền dữ liệu giữa bộ nhớ và thanh ghi: DR(các bit 10 - 9 - 8) cho toán hạng là thanh ghi được sử dụng để lưu dữ liệu xuống bộ nhớ(lệnh nạp) hoặc thanh ghi chứa dữ liệu được nạp từ bộ nhớ(lệnh lưu), SA(các bit 7 - 6 - 5) cho toán hạng nguồn làm thanh ghi nền để tính toán địa chỉ bộ nhớ và CONSTANT(các bit 4 - 3 - 2 - 1 - 0) cho toán hạng là hằng số để tính toán địa chỉ bộ nhớ.
- Lệnh tính toán với dữ liệu giữa thanh ghi và một hằng số: DR(các bit 10 - 9 - 8) cho toán hạng đích là thanh ghi, SA(các bit 7 - 6 - 5) cho toán hạng nguồn đầu tiên, CONSTANT(các bit 4 - 3 - 2 - 1 - 0) cho toán hạng nguồn thứ 2 là hằng số.
- Lệnh nhảy có điều kiện: SA(các bit 7 - 6 - 5) cho toán hạng nguồn đầu tiên là thanh ghi, DR(các bit 10 - 9 - 8) cho toán hạng nguồn thứ 2 là thanh ghi, CONSTANT(4 - 3 - 2 - 1 - 0) là một hằng số để tính toán địa chỉ mới của thanh ghi PC nếu điều kiện thỏa mãn.

3. Chu kỳ thực thi lệnh

- Bước 1: tìm nạp lệnh từ bộ nhớ IMEM. Địa chỉ bộ nhớ lệnh được tìm nạp được lưu trong thanh ghi PC là địa chỉ lệnh kế tiếp được thực thi, sau khi tìm nạp lệnh thành công thì PC tự động tăng giá trị để trỏ đến lệnh tiếp theo.
- Bước 2: Giải mã lệnh, xác định thao tác thực thi và địa chỉ hiệu dụng để lấy dữ liệu tính toán.
- Bước 3: Thực thi thao tác trên các toán hạng nguồn đã được xác định ở bước 2

- Bước 4: Lưu kết quả của thao tác đã thực thi trong bước 3. Sau đó quay lại bước 1.

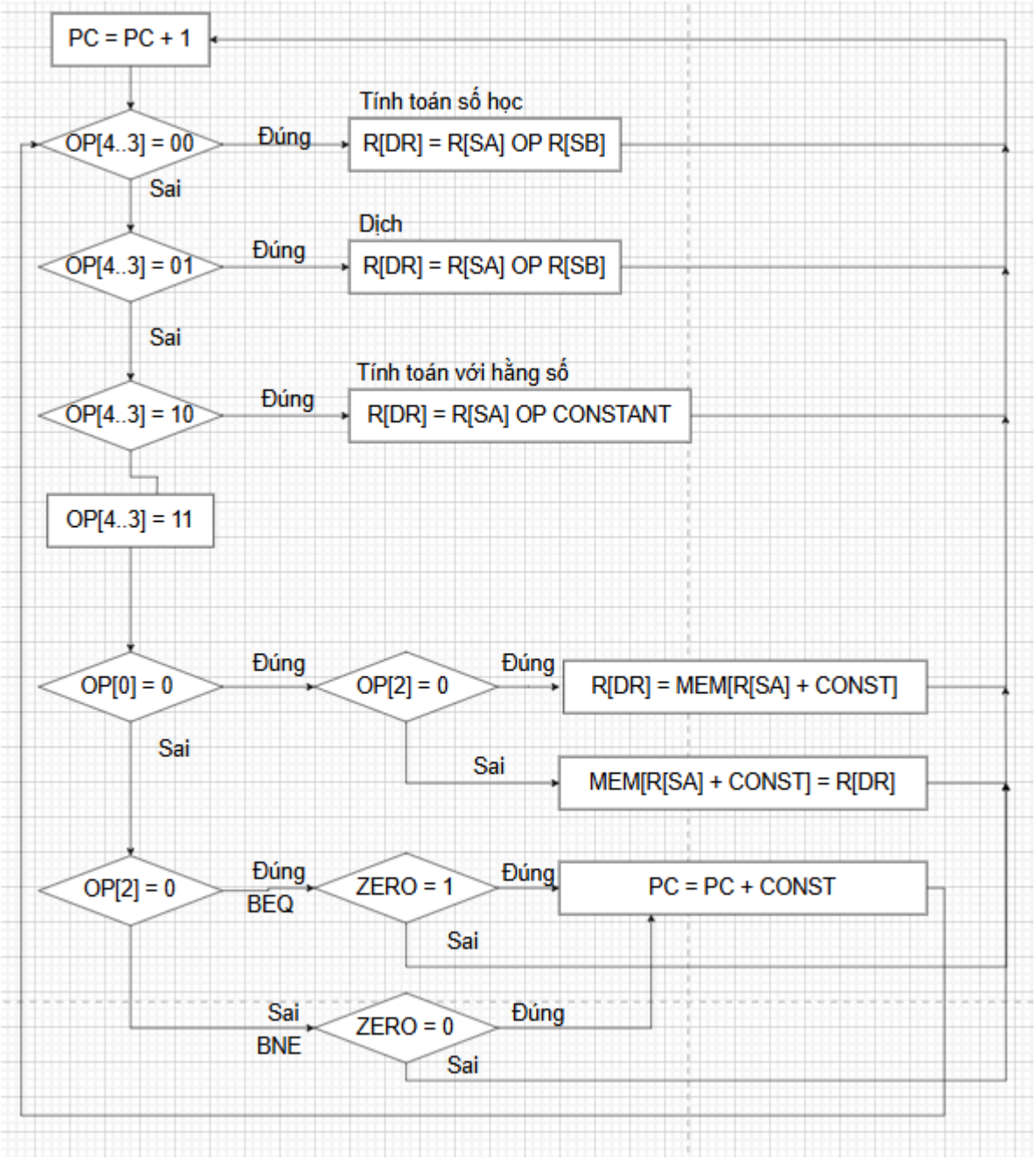
4. Thiết kế tập lệnh

Thao tác	Tên gợi nhớ	OPCODE	Hoạt động	Định dạng lệnh
Cộng	ADD	00 000	R[DR] = R[SA] +R[SB]	R
Tăng 1	INC	00 001	R[DR] = R[SA] + 1	R
Trừ	SUB	00 010	R[DR] = R[SA] -R[SB]	R
Giảm 1	DEC	00 011	R[DR] = R[SA] -1	R
AND	AND	00 100	R[DR] = R[SA] & R[SB]	R
OR	OR	00 101	R[DR] = R[SA] | R[SB]	R
NAND	NAND	00 110	R[DR] = R[SA] NAND R[SB]	R
XOR	XOR	00 111	R[DR] = R[SA] XOR R[SB]	R
Dich phải	SR	01 000	R[DR] = R[SA] >> R[SB]	R
Dịch trái	SL	01 001	R[DR] = R[SA] << R[SB]	R
Dịch phải hằng số	SRI	01 000	R[DR] = R[SA] >> Const	I
Dịch trái hằng số	SLI	01 001	R[DR] = R[SA] << Const	I
Cộng hằng số	ADDI	10 000	R[DR] = R[SA] + Const	I
Trừ hằng số	SUBI	10 010	R[DR] = R[SA] - Const	I
AND hằng số	ANDI	10 100	R[DR] = R[SA] & Const	I
OR hằng số	ORI	10 101	R[DR] = R[SA] | Const	I
NAND hằng số	NANDI	10 110	R[DR] = R[SA] NAND Const	I
XOR hằng số	XORI	10 111	R[DR] = R[SA] XOR Const	I
Nạp	LW	11 010	R[DR] = MEM[R[SA] + Const]	I
Lưu	SW	11 110	MEM[R[SA] + Const] = R[DR]	I
Nhảy nếu bằng	BEQ	11 011	if(R[DR] == R[SA]) PC = PC + Const	I
Nhảy nếu khác	BNE	11 111	if(R[DR] != R[SA])	I

			PC = PC + Const	

- Đối với lệnh BEQ khi thực thi sẽ kiểm tra R[DR] == R[SA] hay không sẽ dẫn đến một bộ so sánh bằng khi thiết kế khối dữ liệu, nên ta có thể chuyển lệnh BEQ thành một phép trừ R[DR] - R[SA] và chúng ta kiểm tra kết quả tính toán có bằng 0 hay không. Tương tự như lệnh BEQ, chúng ta có thể chuyển lệnh BNE thành một phép trừ R[DR] - R[SA] và chúng ta chỉ cần kiểm tra kết quả của phép tính có bằng 0 hay không.

5. Lập sơ đồ tập lệnh

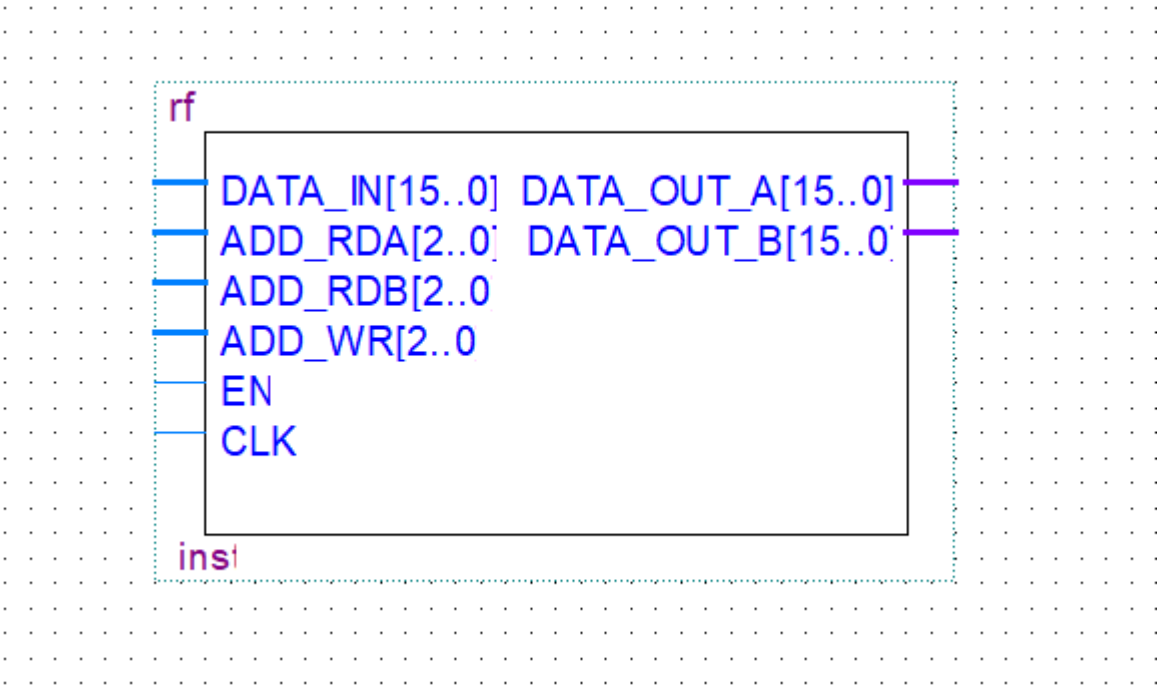


- Việc PC = PC + 1 là do IMEM mà chúng mà chúng ta định địa chỉ với mỗi word là 16 bit.

6. Xác định các thành phần phần cứng cần thiết

a. *Register File*

Theo tập lệnh đã thiết kế thì chúng ta cần một Register File chứa 8 thanh ghi, mỗi thanh ghi rộng 16 bit. Có 2 cổng đọc, 1 cổng ghi

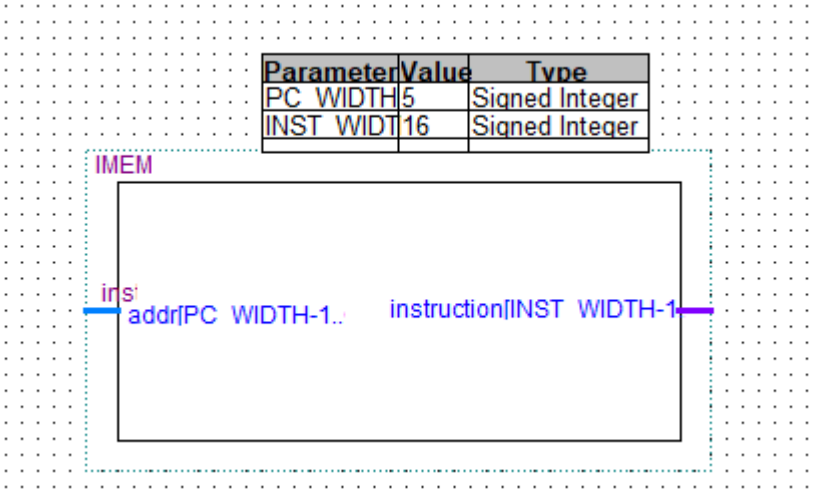


Các tín hiệu của RF:

DATA_IN[15..0]: dữ liệu ghi vào RF
ADD_RDA[2..0]: địa chỉ đọc cổng A
ADD_RDB[2..0]: địa chỉ đọc cổng B
ADD_WR[2..0]: địa chỉ ghi dữ liệu vào RF
DATA_OUT_A[15..0]: đọc giá trị từ RF ra cổng A
DATA_OUT_B[15..0]: đọc giá trị từ RF ra cổng B
EN: tín hiệu cho phép RF ghi dữ liệu
CLK: tín hiệu xung clock

b. *IMEM*

Thiết kế IMEM có kích thước 32x16 bit, bộ nhớ chứa lệnh thực thi



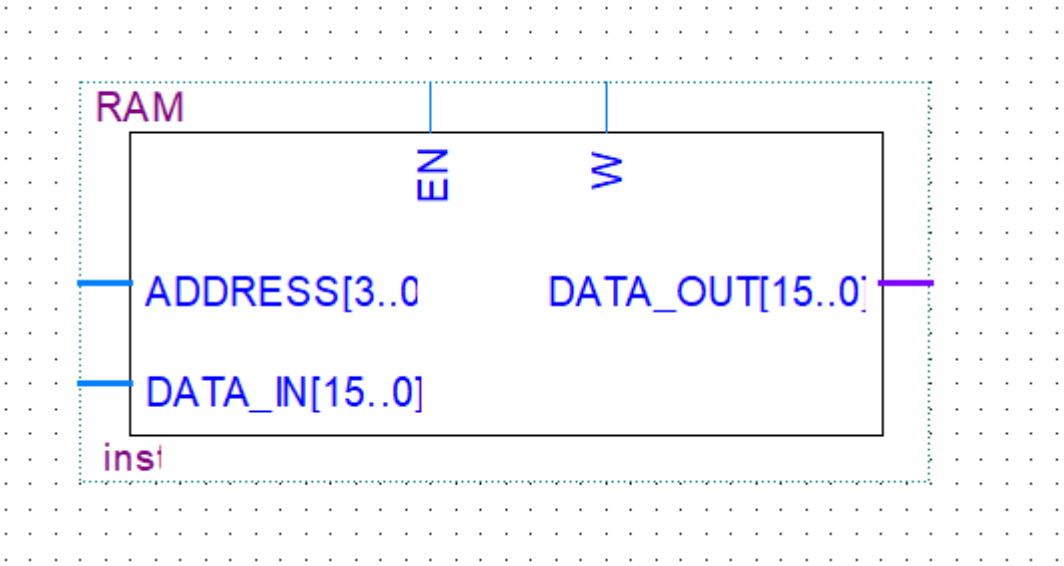
Các tín hiệu của IMEM:

- addr[PC_WIDTH - 1]: địa chỉ trong IMEM có độ rộng 5 bit
- instruction[INST_WIDTH - 1]: dữ liệu được lưu trong IMEM, có độ rộng 16 bit

IMEM có thể lưu được tối đa 32 lệnh.

c. *DMEM*

Thiết kế DMEM có kích thước 16x16 bit



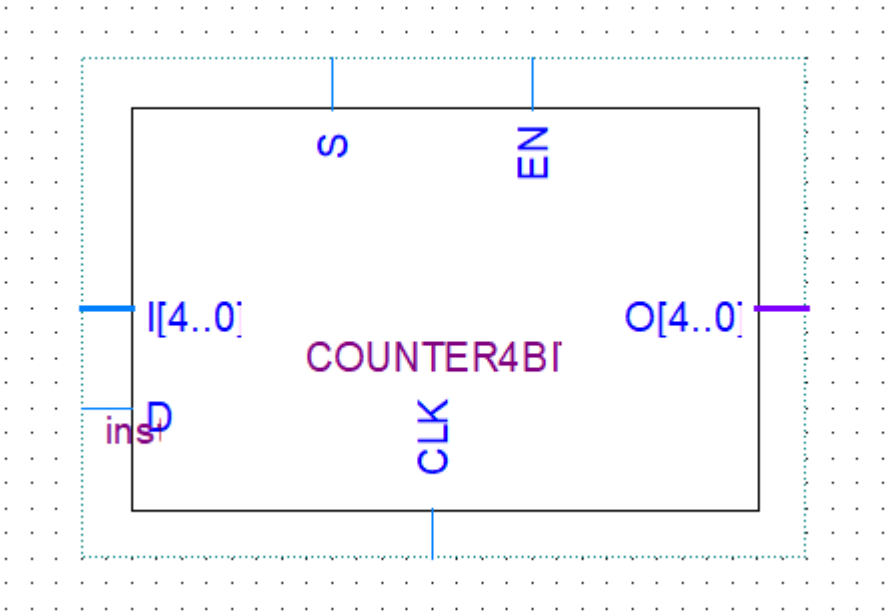
Các tín hiệu của DMEM:

- ADDRESS[3..0]: địa chỉ xác định dữ liệu được lưu vào ô nhớ nào trong DMEM
- DATA_IN[15..0]: dữ liệu để ghi vào DMEM
- DATA_OUT[15..0]: dữ liệu được đọc từ DMEM

EN	W	Chức năng
1	0	đọc dữ liệu
1	1	ghi dữ liệu

d. PC

Thanh ghi PC, vì IMEM có 32 word, nên thanh ghi PC sẽ rộng tối thiểu 5 bit.

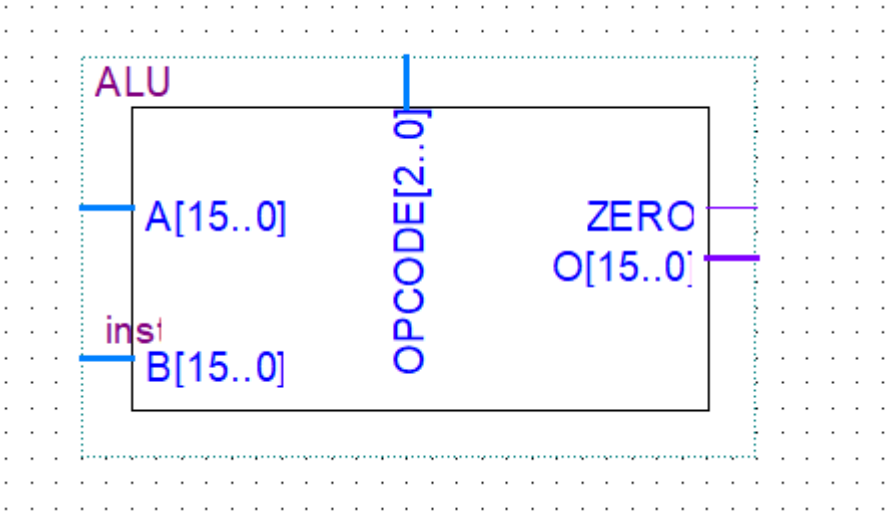


Các tín hiệu của PC:

- I[4..0]: dữ liệu được nạp vào PC, thường cho các lệnh nhảy
- O[4..0]: giá trị của thanh ghi PC
- D(=1): cho phép thanh ghi đếm lên

EN	S	Chức năng
1	0	PC = PC + 1
1	1	PC = PC + I[4..0]

e. ALU

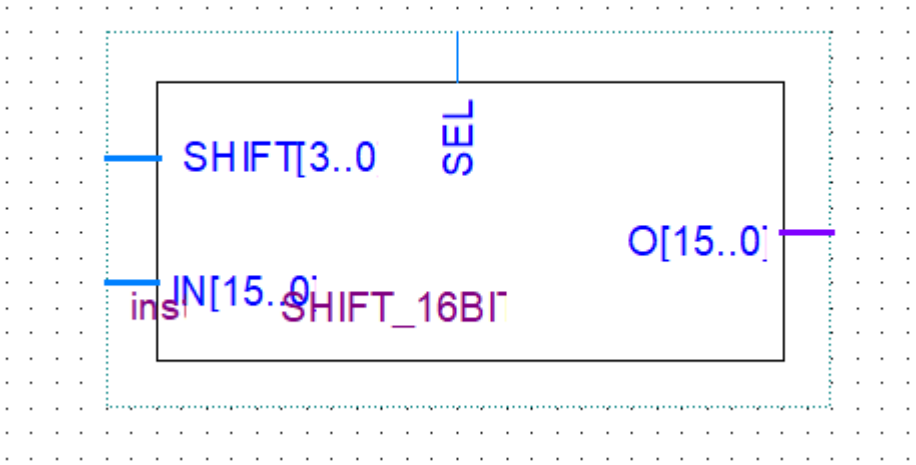


Các tín hiệu của ALU:

- A[15..0]: giá trị của toán hạng thứ nhất
- B[15..0]: giá trị của toán hạng thứ 2
- O[15..0]: giá trị của kết quả
- ZERO: cho biết kết quả của ALU có bằng 0 hay không
- OPCODE[2..0]: chọn chức năng cho ALU

OPCODE[2..0]	Chức năng
000	Cộng
001	Cộng 1
010	Trừ
011	Trừ 1
100	AND
101	OR
110	NAND
111	XOR

f. Bộ dịch

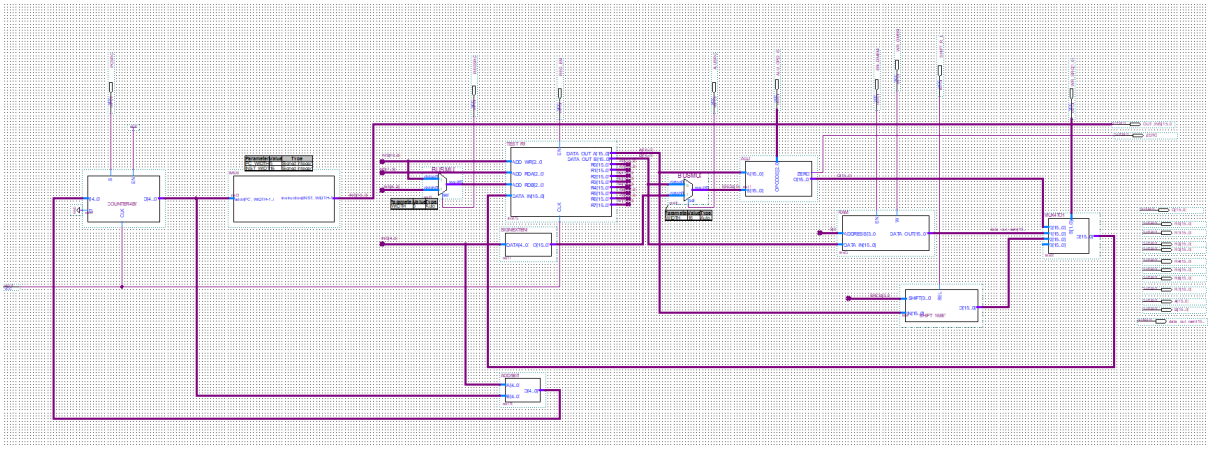


Các tín hiệu của bộ dịch:

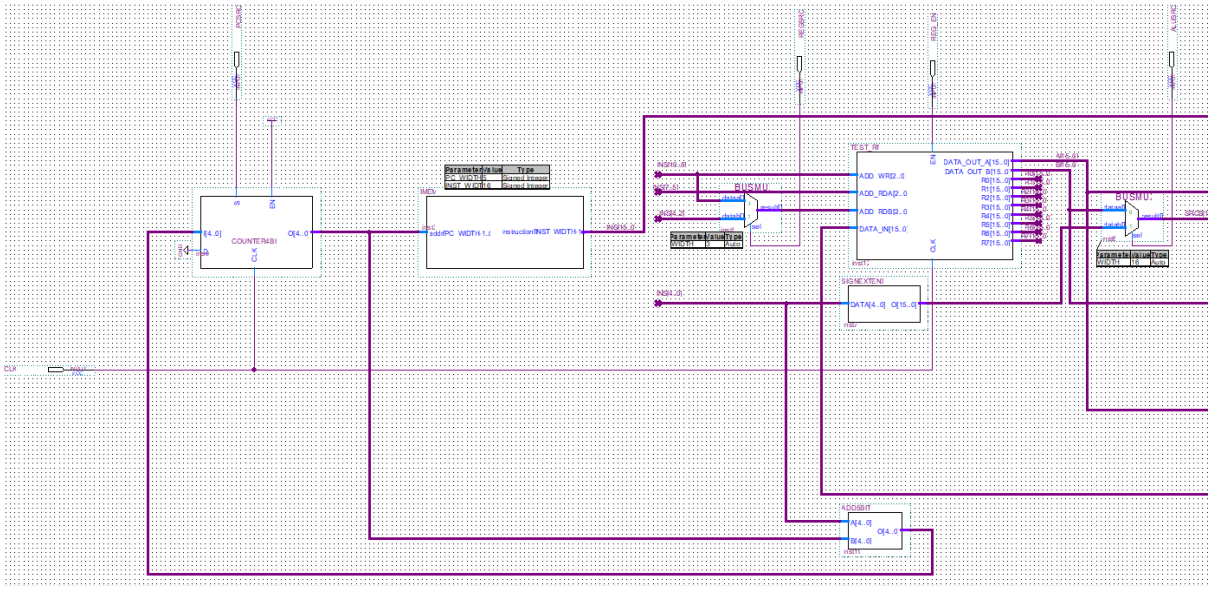
- IN[15..0]: toán hạng nguồn đầu tiên
- SHIFT[3..0]: toán hạng cho biết IN[15..0] được dịch bao nhiêu bit, tối đa 15 bit
- O[15..0]: kết quả sau khi dịch
- SEL: cho phép chọn dịch phải hay trái (0: dịch phải; 1: dịch trái)

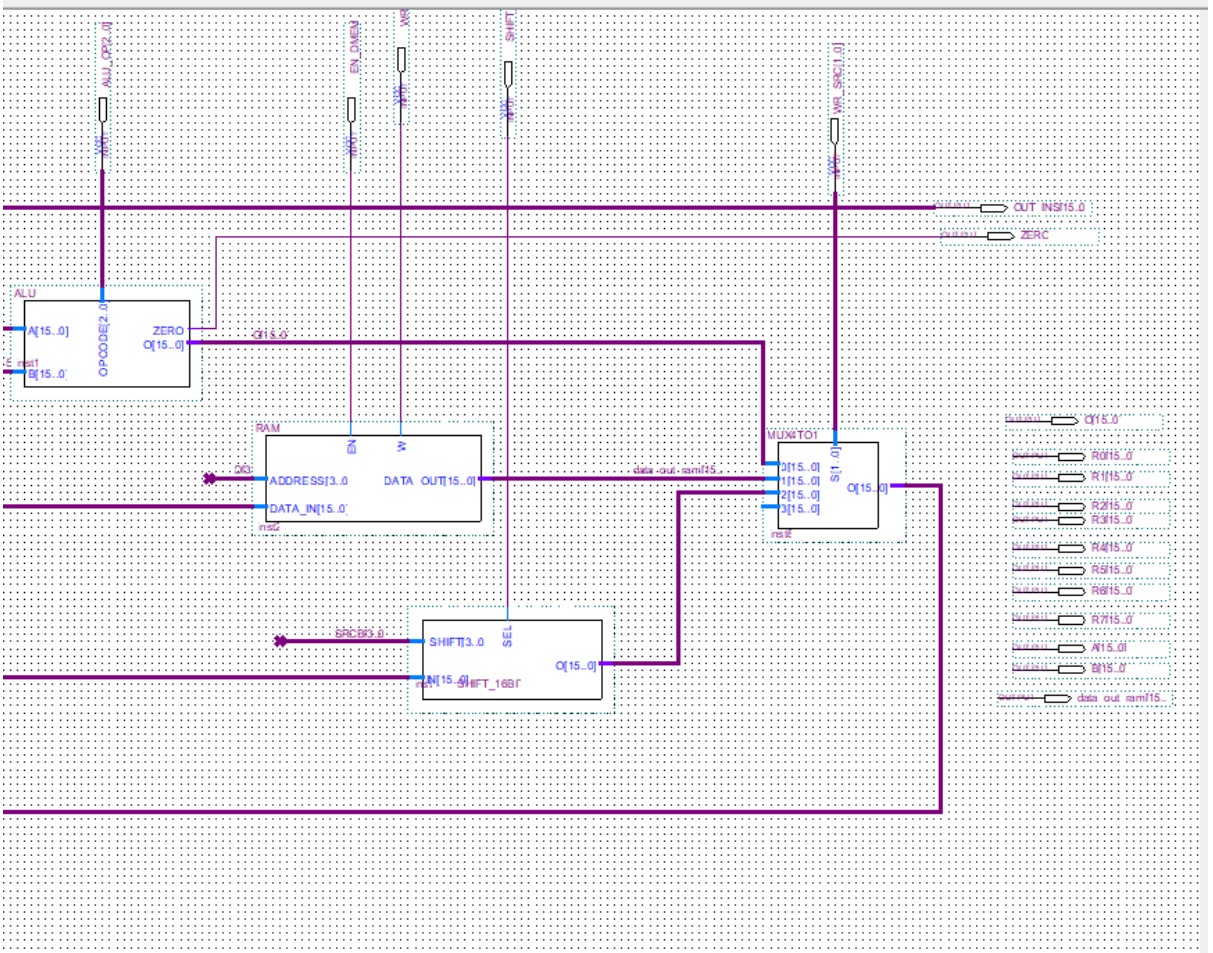
7. Thiết kế khối dữ liệu

7.1. Khối dữ liệu



Phóng to





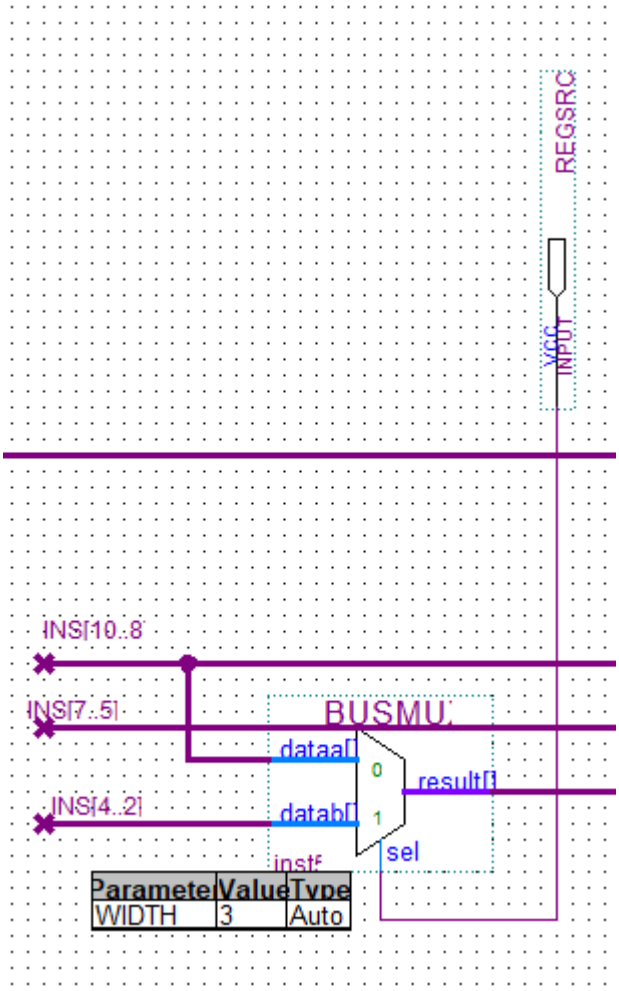
Các tín hiệu điều khiển trong khối dữ liệu:

- PC_SRC: chọn giá trị cập nhật cho thanh ghi PC(PC = PC + 1 hoặc PC = PC + CONSTANT)
- REG_SRC: xác định toán hạng nguồn cho cổng B của RF

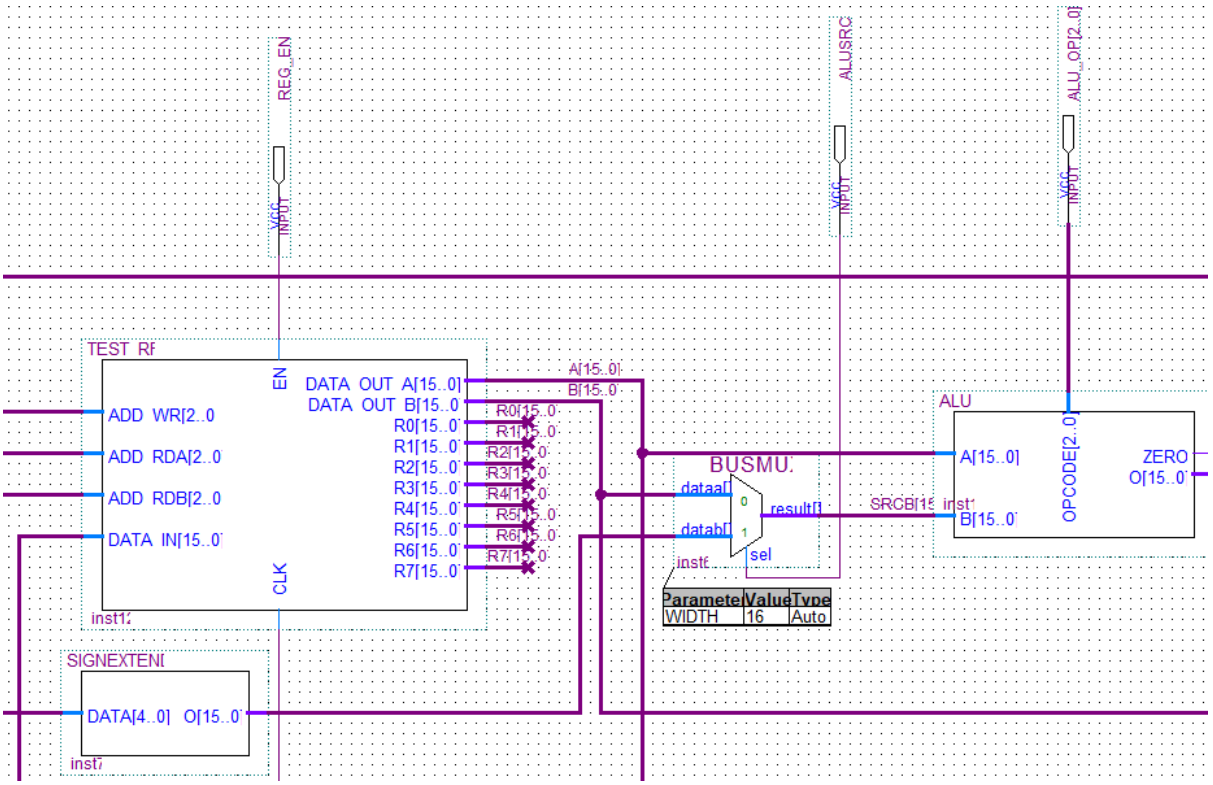
- REG_EN: tín hiệu EN của RF
- ALU_SRC: xác định toán hạng của ALU, có thể là toán hạng là thanh ghi hoặc hằng số
- EN_MEM: tín hiệu EN của DMEM
- WR_MEM: tín hiệu cho phép đọc/ghi của DMEM
- SHIFT_R_L: tín hiệu xác định phép dịch phải/trái
- WR_SRC[1..0]: chọn tín hiệu để ghi vào RF, có thể lấy kết quả từ ALU, DMEM hoặc SHIFT_R_L

Giải thích:

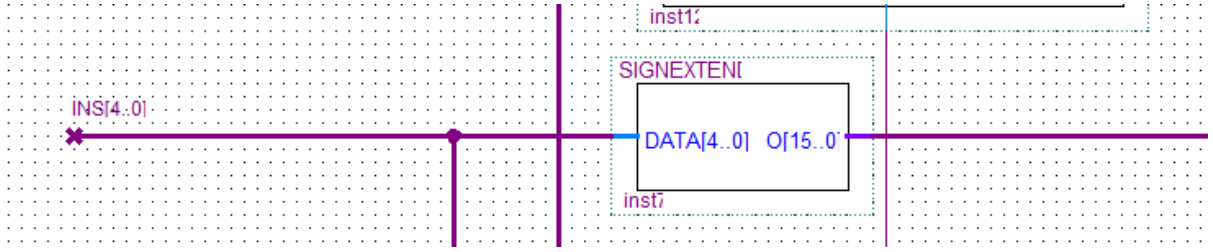
- Đối với các lệnh như lệnh SW, BEQ, BNE thì RD là địa chỉ toán hạng nguồn, nên ta phải thêm 1 MUX2 3 bit để xác định RD.



- Đối với ALU cũng tương tự như thế, ta có các lệnh với toán hạng thứ nhất là thanh ghi và toán hạng thứ 2 là thanh ghi hoặc hằng số nên ta cần MUX2 để xác định.

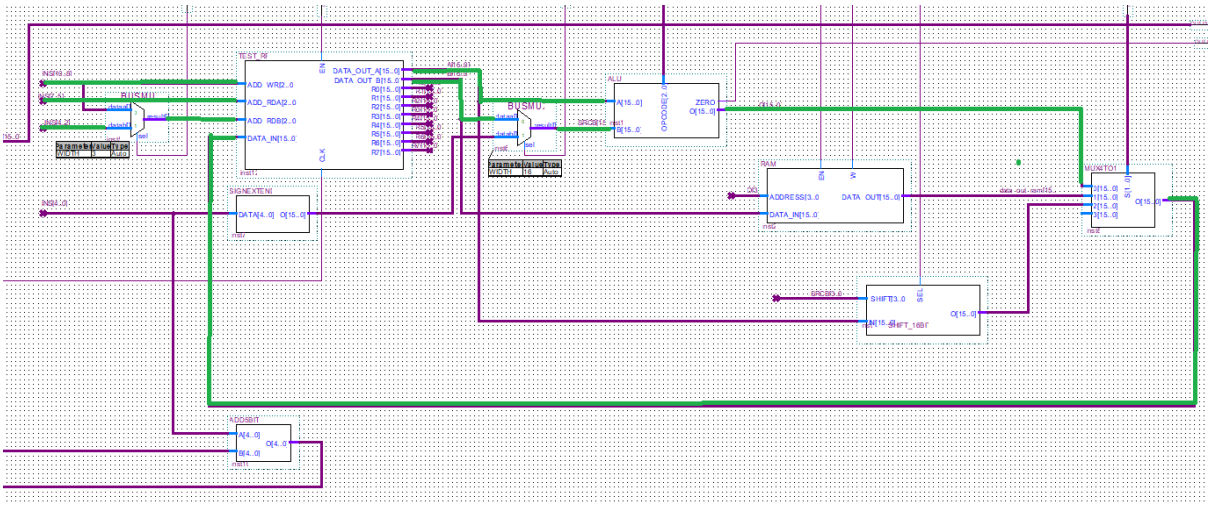


- CONSTANT[4..0]: là hằng số có dấu nhưng độ rộng là 5 bit, trong khi ALU có các toán hạng là 16 bit nên ta phải mở rộng CONSTANT thành số 16 bit qua bộ SIGNEXTEND



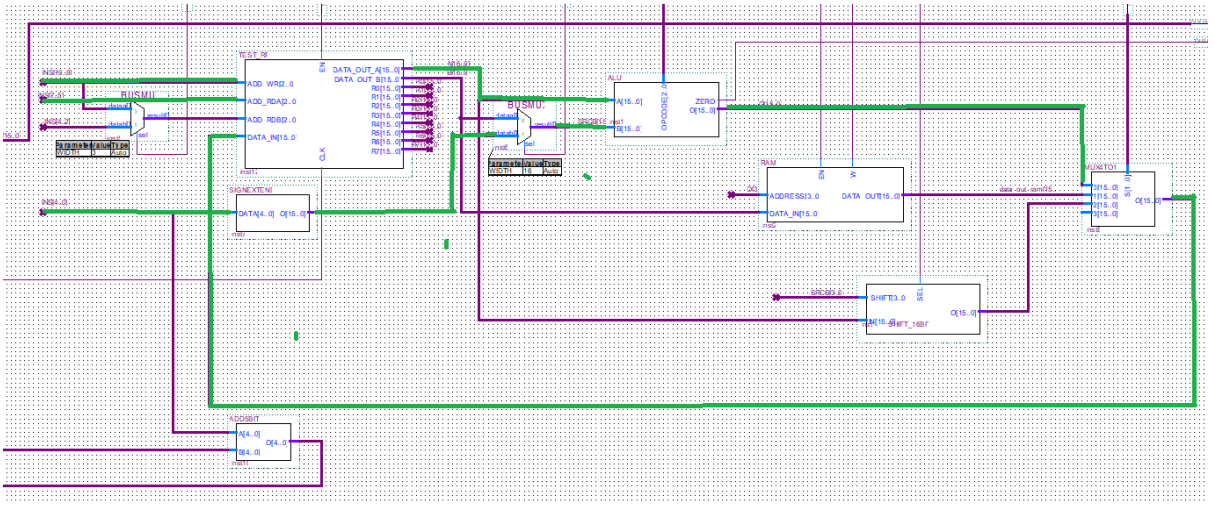
7.2. Đường đi các lệnh trong khối dữ liệu

a. *Lệnh R*



Lệnh R: ta có 1 toán hạng đích là thanh ghi và 2 toán hạng nguồn đều là thanh ghi nên ta vẽ được đường đi dữ liệu của lệnh R trong khối dữ liệu.

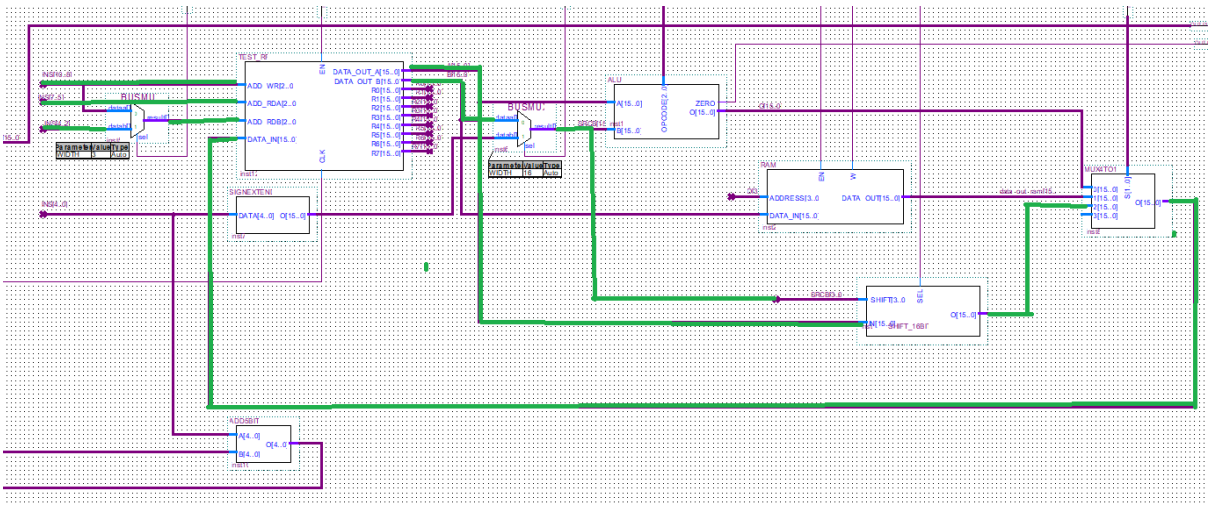
b. *Đối với các lệnh I*



Lệnh I: ta có 1 toán hạng đích là thanh ghi và 1 toán hạng nguồn là thanh ghi và toán hạng nguồn còn lại là hằng số nên ta vẽ được đường đi dữ liệu của lệnh I trong khối dữ liệu.

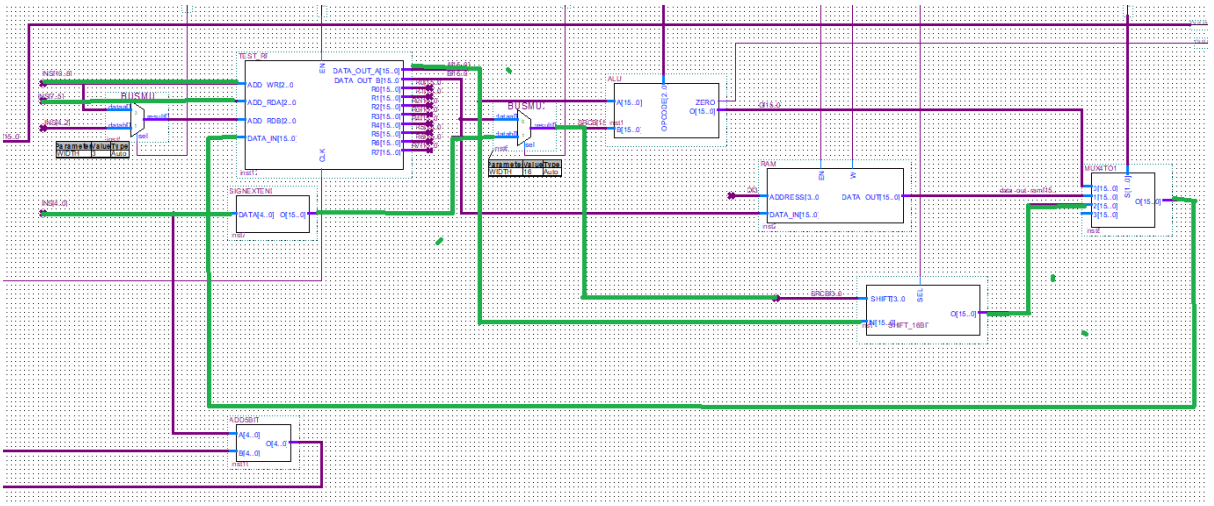
c. *Đối với lệnh SHIFT*

- Lệnh dịch với toán hạng đích là thanh ghi và 2 toán hạng nguồn là 2 thanh ghi.



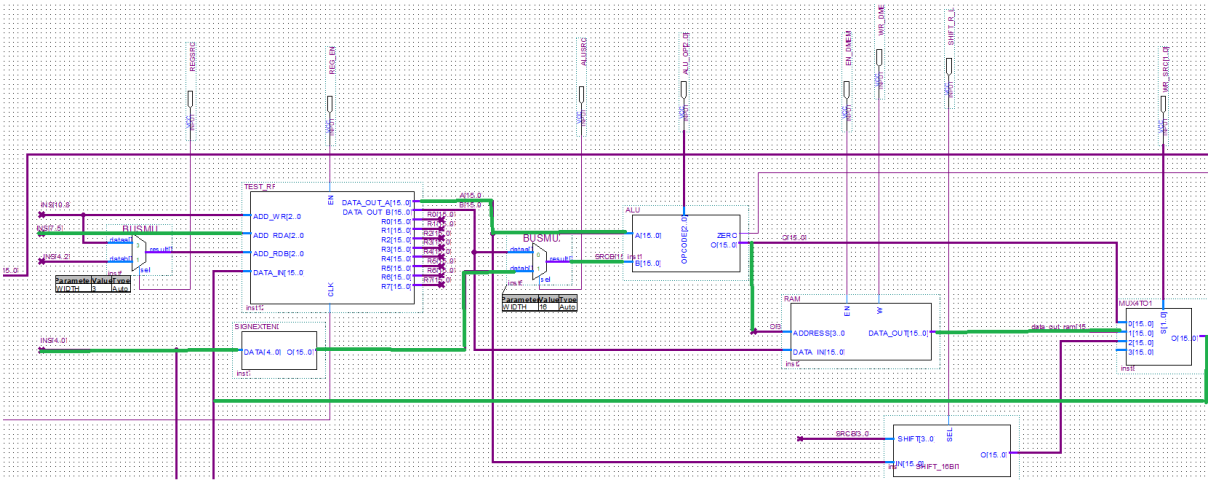
Địa chỉ thanh ghi nguồn SA là giá trị dịch và địa chỉ thanh ghi nguồn SB là số lượng bit dịch, từ đó ta vẽ được đường đi dữ liệu với lệnh dịch trên.

- Lệnh dịch với 2 toán hạng nguồn là 1 thanh ghi và 1 hằng số



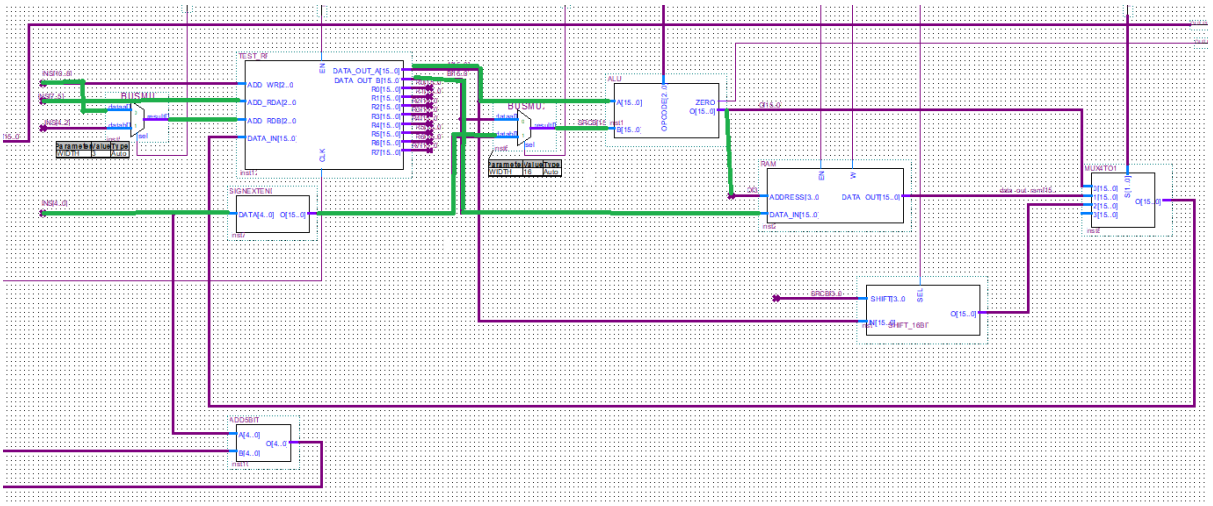
Khác với lệnh dịch với 2 toán hạng nguồn là thanh ghi, lệnh dịch hằng số này với thanh ghi SA là giá trị dịch, CONSTANT là hằng số, thể hiện số bit cần dịch. Và ta vẽ được đường đi dữ liệu đối với lệnh dịch này.

d. Đối với lệnh LW



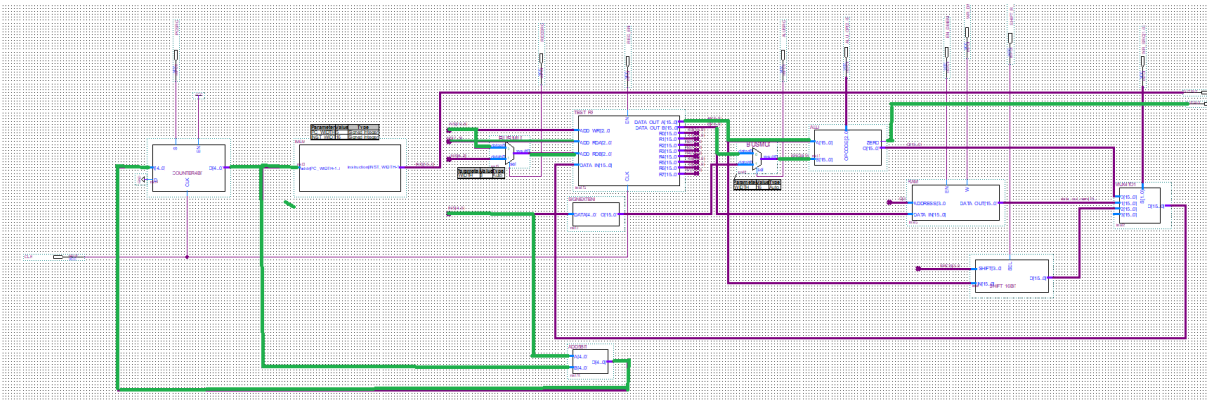
Địa chỉ đọc địa chỉ của thanh ghi nguồn SA để lấy địa chỉ nền, sau đó cộng với OFFSET là CONSTANT để xác định địa chỉ được lưu trong DMEM, khi có được địa chỉ lưu trong DMEM ta tiến hành ghi giá trị vào RF

e. Đối với lệnh SW



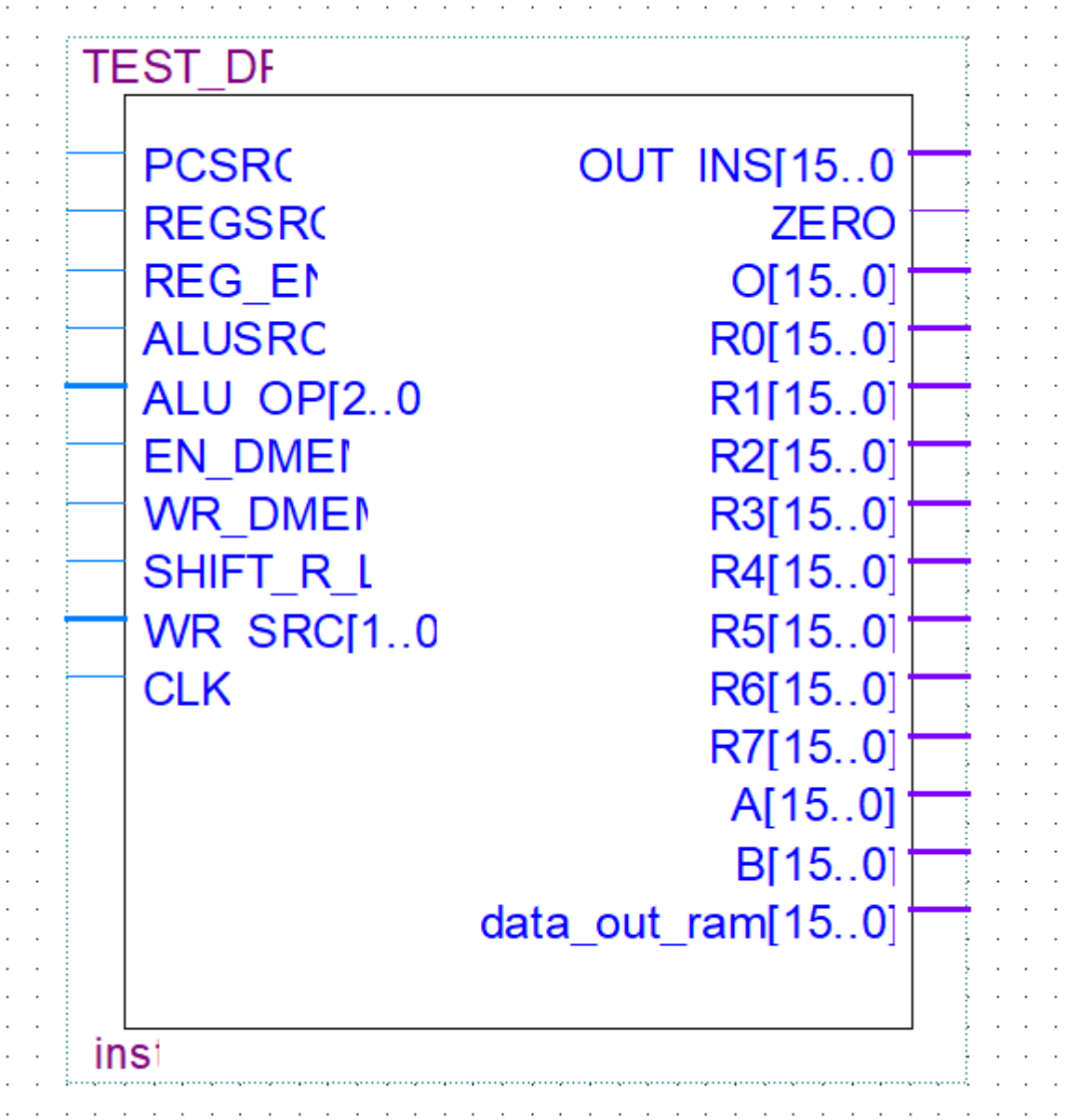
Thanh ghi RD lúc này là thanh ghi nguồn và giá trị được đọc ra từ cổng B, địa chỉ lưu được tính bằng các cộng giá trị thanh ghi SA và OFFSET, sau đó lưu giá trị từ thanh ghi RD và DMEM.

f. Đối với lệnh BEQ, BNE



Khi thực thi lệnh nhảy thì trong ALU sẽ thực hiện phép trừ để so sánh giá trị của 2 thanh ghi nguồn RD và SA, sau đó kiểm tra cờ ZERO để đưa ra quyết định cho lệnh nhảy.

Đóng gói khối dữ liệu



Các tín hiệu đầu ra của khối dữ liệu:

- ZERO: cờ xác định giá trị khi tính toán kết quả của ALU có bằng 0 hay không
- R1 - R7: giá trị của các thanh ghi trong RF
- OUT_INS: cho biết lệnh đang được thực thi trong datapath
  data_out_ram: kiểm tra giá trị của DMEM khi thực thi lệnh LW

8. Thiết kế khối điều khiển

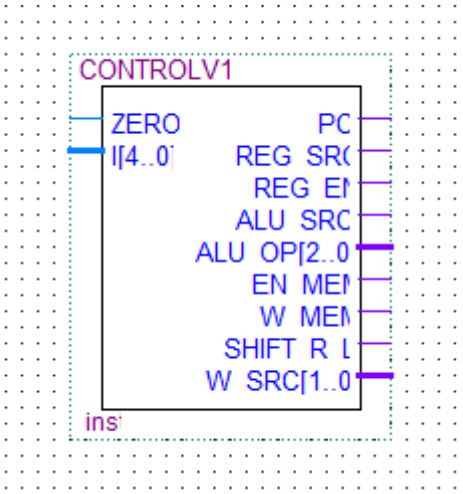
Bảng thiết kế tín hiệu điều khiển

Thao tác	INPUT		OUTPUT								
	INS[15..11]	ZERO	PC_SRC	REG_SRC	REG_EN	ALU_SRC	OPCODE	EM_MEM	W_MEM	SHIFT_R_L	WR
Cộng	00 000	X	0	1	1	0	000	0	0	X	00
Cộng 1	00 001	X	0	1	1	0	001	0	0	X	00
Trừ	00 010	X	0	1	1	0	010	0	0	X	00
Trừ 1	00 011	X	0	1	1	0	011	0	0	X	00
AND	00 100	X	0	1	1	0	100	0	0	X	00
OR	00 101	X	0	1	1	0	101	0	0	X	00
NAND	00 110	X	0	1	1	0	110	0	0	X	00
XOR	00 111	X	0	1	1	0	111	0	0	X	00
Dịch phải	01 000	X	0	1	1	0	XXX	0	0	0	10
Dịch trái	01 001	X	0	1	1	0	XXX	0	0	1	10
Dịch phải(hằng số)	01 010	X	0	X	1	1	XXX	0	0	0	10
Dịch trái(hằng số)	01 011	X	0	X	1	1	XXX	0	0	1	10
Cộng hằng số	10 000	X	0	X	1	1	000	0	0	X	00
Trừ hằng số	10 010	X	0	X	1	1	010	0	0	X	00
AND hằng số	10 100	X	0	X	1	1	100	0	0	X	00
OR hằng số	10 101	X	0	X	1	1	101	0	0	X	00
NAND hằng số	10 110	X	0	X	1	1	110	0	0	X	00
XOR hằng số	10 111	X	0	X	1	1	111	0	0	X	00
LW	11 010	X	0	X	1	1	000	1	0	X	01
SW	11 110	X	0	0	0	1	000	1	1	X	XX
BEQ	11 011	0	0	0	0	0	010	0	X	X	XX
BNE	11 111	0	1	0	0	0	010	0	X	X	XX
BEQ	11 011	1	1	0	0	0	010	0	X	X	XX
BNE	11 111	1	0	0	0	0	010	0	X	X	XX

Biểu thức rút gọn(INS[15..11] ⇔ I[4..0]):

- PC_SRC = I4I3I2’I1I0ZERO + I4I3I2I1I0ZERO’
- REG_SRC = I4’I3’ + I4’I2’I1’
- REG_EN = I4’ + I3’ + I2’I0’
- ALU_SRC = I4’I3I2’I1 + I4I3’ + I4I1I0’
- OPCODE[2] = I3’I2
- OPCODE[1] = I3’I1 + I4I1I0
- OPCODE[0] = I3’I0
- EM_MEM = I4I3I1I0’
- W_MEM = I4I3I2I1I0’
- SHIFT_R_L = I4’I3I2’I0
- WR_SRC[1] = I4’I3I2’I0
- WR_SRC[0] = I4I3I2’I1I0’

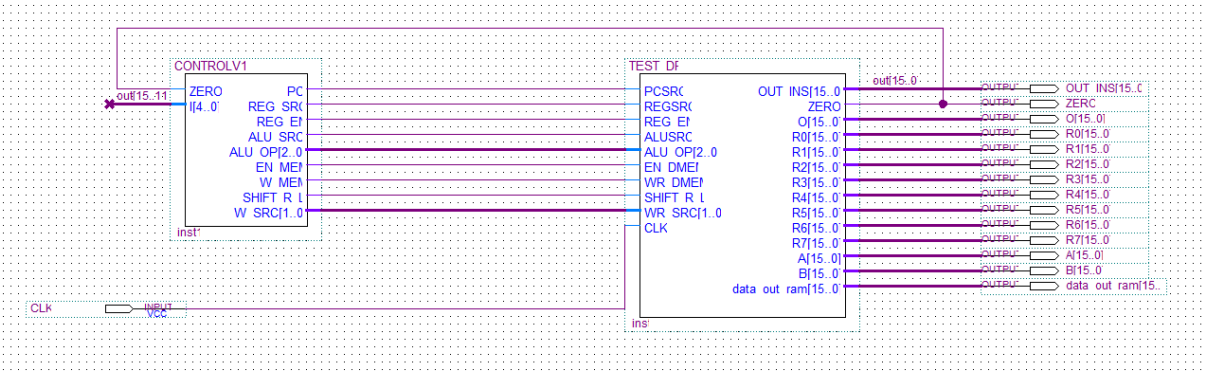
Đóng gói khối điều khiển



- ZERO: cờ ZERO được gửi từ khối dữ liệu
- I[4..0]: tương đương với INS[15..11] của IMEM
- PC_SRC: chọn giá trị cập nhật cho thanh ghi PC(PC = PC + 1 hoặc PC = PC + CONSTANT)
- REG_SRC: xác định toán hạng nguồn cho cổng B của RF
- REG_EN: tín hiệu EN của RF
- ALU_SRC: xác định toán hạng của ALU, có thể là toán hạng là thanh ghi hoặc hằng số
- EN_MEM: tín hiệu EN của DMEM
- WR_MEM: tín hiệu cho phép đọc/ghi của DMEM
- SHIFT_R_L: tín hiệu xác định phép dịch phải/trái
- WR_SRC[1..0]: chọn tín hiệu để ghi vào RF, có thể lấy kết quả từ ALU, DMEM hoặc SHIFT_R_L

9. Hoàn thiện CPU, nạp code, mô phỏng

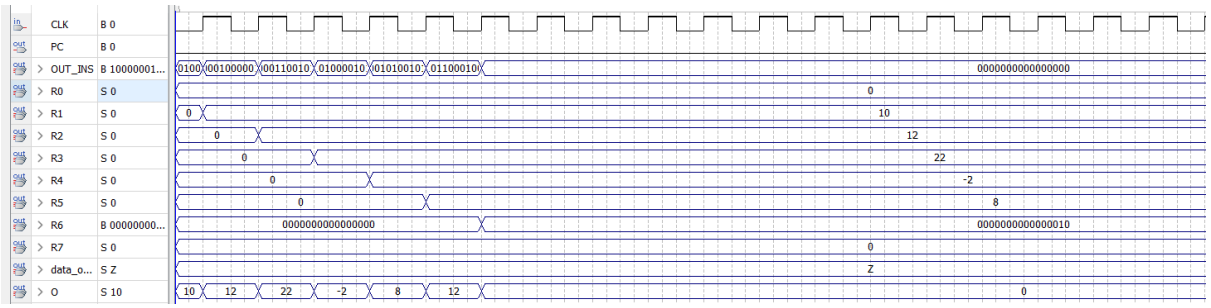
Top_Level



Mô phỏng:

a. *Mã máy một số lệnh cơ bản:*

10000 001 000 01010 // R1 = R0 + 10
10000 010 000 01100 // R2 = R0 + 12
00000 011 001 010 00 // R3 = R1 + R2
00010 100 001 010 00 // R4 = R1 - R2
00100 101 001 010 00 // R5 = R1 and R2
01000 110 001 00010 // R6 = R1 >> 2



Bắt đầu khi CLK tích cực cạnh lên ta tiến hành nạp lệnh:

- CLK tích cực cạnh lên thứ 1: R1 = 10, đúng với lệnh R1 = R0 + 10
- CLK tích cực cạnh lên thứ 2: R2 = 12, đúng với lệnh R2 = R0 + 12
- CLK tích cực cạnh lên thứ 3: R3 = 22, đúng với lệnh R3 = R1 + R2
- CLK tích cực cạnh lên thứ 4: R4 = -2, đúng với lệnh R4 = R1 - R2
- CLK tích cực cạnh lên thứ 5: R5 = (1010 & 1100) = 1000(8), đúng với lệnh R5 = R1 and R2

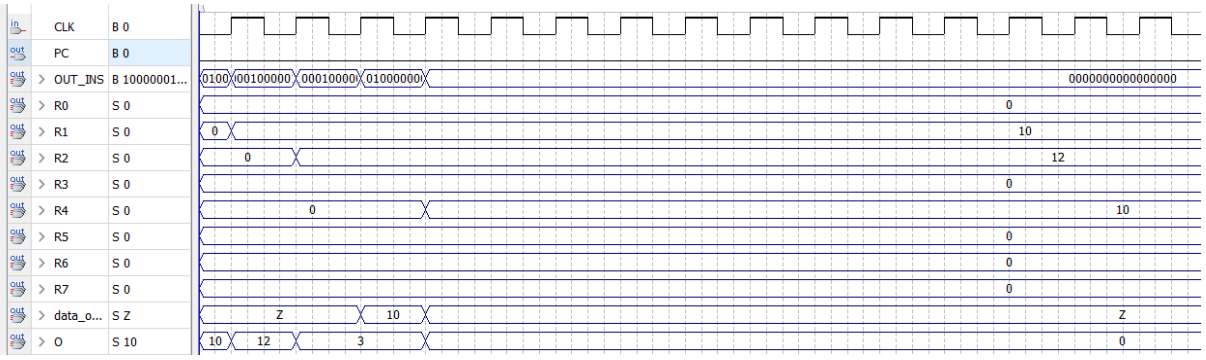
- CLk tích cực cạnh lên thứ 6: R6 = (1010 >> 2) = 10, dịch phải 2 bit, đúng với lệnh R6 = R1 >> 2

Nhận xét: các lệnh cơ bản tính toán trong ALU và bộ dịch là chính xác.

b. *Mô phỏng lệnh LW, SW:*

Mã máy:

```
10000 001 000 01010  // R1 = R0 + 10
10000 010 000 01100  // R2 = R0 + 12
11 110 001 000 00011 // SW R1, 3(R0)
11 010 100 000 00011 // LW R4, 3(R0)
```



- CLK tích cực cạnh lên thứ 1: R1 = 10, đúng với lệnh R1 = R0 + 10
- CLK tích cực cạnh lên thứ 2: R2 = 12, đúng với lệnh R2 = R0 + 12
- CLK tích cực cạnh lên thứ 3: lưu giá trị tại thanh ghi R1 vào ô nhớ có địa chỉ là 3 trong DMEM
- CLK tích cực cạnh lên thứ 4: nạp giá trị tại ô nhớ có địa chỉ là 3 trong DMEM ra R4

Nhận xét: kết quả mô phỏng của lệnh LW và SW là chính xác

c. *Mô phỏng bài toán cụ thể*

Bài toán: nhập vào số nguyên dương n, tính số fibonacci thứ n.

Mã giả bài toán:

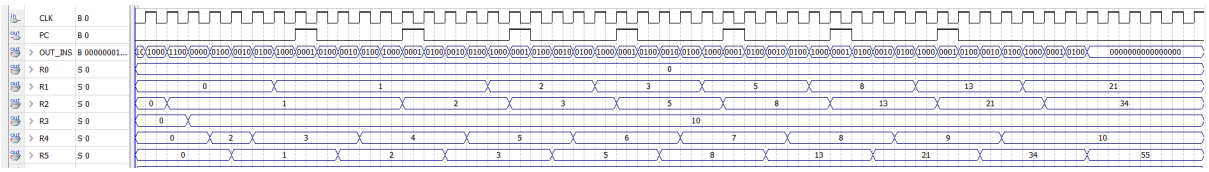
```
add R1, R0, R0    // fibo(0)
addi R2, R0,1     // fibo(1)
addi R3, R0, n    // nhập vào n
addi R4, R0, 2    // bắt đầu tính fibo(2)
add R5, R1, R2    // fibo(n) = fibo(n-1) + fibo(n-2)
inc R4            // R4 = R4 + 1
```

```
add R1, R0, R2     // cập nhật giá trị của fibo(n-2)
add R2, R0, R5     // cập nhật giá trị của fibo(n-1)
beq R4, R3, (-4)   // kiểm tra điều kiện lặp, nếu chưa tính đến n thì quay lại vòng lặp
add R5, R1, R2     // khi lặp n lần thì lần thứ n chưa được tính, nên ta cần tính thêm lần thứ n sau khi thoát khỏi vòng lặp.
```

Dựa vào mã giả, biên dịch ra mã máy của bài toán:

```
0000000100000000
1000101000000000
1000001100001010
1000010000000010
0000010100101000
0000110010000000
0000000100001000
0000001000010100
1111110001111100
0000010100101000
```

Kết quả mô phỏng:



- Khi R4(n) = 10 thì R5 = 55, cho thấy kết quả mô phỏng chính xác
- Những lần PC lên 1 tức là lệnh nhảy quay lại thực hiện vòng lặp, đến R4 = 10 thì không thỏa mãn điều kiện nên ta xác định được Fibo(10) = 55 là chính xác.

Nhận xét: kết quả mô phỏng đúng, bộ vi xử lý có thể lập trình để xử lý một số bài toán cơ bản

10. Tổng hợp

Kiểm tra tài nguyên

Analysis & Synthesis Resource Usage Summary

	Resource	Usage
1	Estimated Total logic elements	88
2		
3	Total combinational functions	88
4	⌄ Logic element usage by number of LUT inputs	
1	-- 4 input functions	43
2	-- 3 input functions	39
3	-- <=2 input functions	6
5		
6	⌄ Logic elements by mode	
1	-- normal mode	88
2	-- arithmetic mode	0
7		
8	⌄ Total registers	23
1	-- Dedicated logic registers	23
2	-- I/O registers	0
9		
10	I/O pins	222
11	Embedded Multiplier 9-bit elements	0
12	Maximum fan-out node	TEST_DP:inst|IMEM:inst3|rom~5
13	Maximum fan-out	51
14	Total fan-out	450
15	Average fan-out	1.35

Kiểm tra định thời

Slow Model Fmax Summary

	Fmax	Restricted Fmax	Clock Name	Note
1	61.28 MHz	61.28 MHz	CLK	

Tần số tối đa

Slow Model Fmax Summary

	Fmax	Restricted Fmax	Clock Name	Note
1	699.3 MHz	420.17 MHz	CLK	limit due to minimum period restriction (max I/O toggle rate)

Kiểm tra công suất

PowerPlay Power Analyzer Summary

PowerPlay Power Analyzer Status	Successful - Sun Dec 29 20:11:53 2024
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition
Revision Name	RISC
Top-level Entity Name	RISCV1
Family	Cyclone II
Device	EP2C35F672C6
Power Models	Final
Total Thermal Power Dissipation	134.16 mW
Core Dynamic Thermal Power Dissipation	0.00 mW
Core Static Thermal Power Dissipation	80.00 mW
I/O Thermal Power Dissipation	54.16 mW
Power Estimation Confidence	Low: user provided insufficient toggle rate data

Bảng tổng hợp tài nguyên

Tài nguyên	475 thành phần logic
Tần số tối đa	420.17MHz
Định thời	61.28MHz
Công suất	134,16mW