Trường Đại Học Bách Khoa Hà Nội

Viện Điện Tử - Viễn Thông

====o0o====

Báo Cáo Bài Tập Lớn

Thiết kế VLSI

Đề Tài: Thiết kế MCU 8bit

Giảng viên hướng dẫn : TS. Nguyễn Vũ Thắng

Kỹ sư trợ giảng : Trần Văn Long

Mã học phần: ET4340

Nhóm Sinh viên thực hiện.

Họ và tên	Mssv
Nguyễn Văn Chuyên	20140487
Vũ Đình Nam	20179646
Lê Tiến Đạt	20179680

Hà Nội, Tháng 12 năm 2018

MỤC LỤC

Danh mục hình ảnh

Danh mục bảng biểu

Lời nói đầu

Cuộc sống ngày càng phát triển, điện tử đã trở thành một phần không thể thiếu đối với xã hội loài người. Những sản phẩm điện tử có mặt ở khắp mọi nơi với những máy móc từ đơn giản như đèn điện đến phức tạp, tinh vi như điện thoại di động, máy tính tạo nên những giá trị vật chất cho con người đến những con chip có thể điều khiển cả một hệ thống.

Đặc biệt đối với ngành thiết kế chip đang phát triển và là 1 phần không thế thiếu trong các sản phẩm điện tử ngày nay. Vì vậy, nhóm em đã quyết định chọn đề tài thiết kế MCU 8 bit để thực hiện các phép tính đơn giản và tiếp cận với công nghệ thiết kế IC.

Nhóm em xin gửi lời chân thành đến anh Trần Văn Long và thầy Nguyễn Vũ Thắng đã hướng dẫn tận tình trong suốt quá trình học tập giúp chúng em hoàn thành đề tài này.

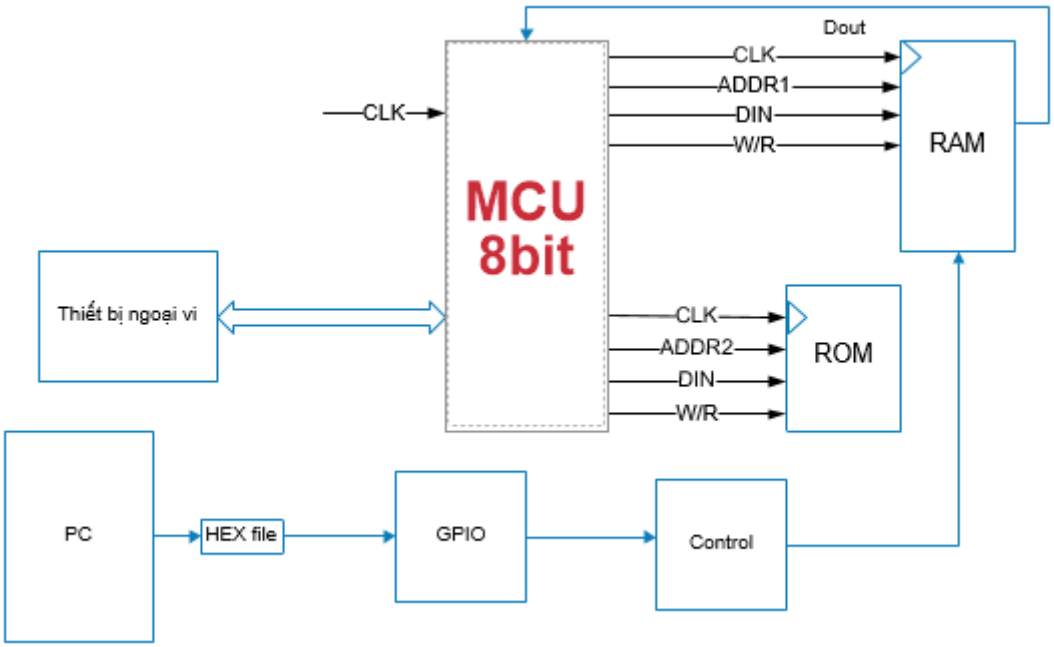
Chúng em xin chân thành cảm ơn anh Trần Văn Long và thầy Nguyễn Vũ Thắng!

Chương 1 Phân tích đề tài

1.1 Phân tích yêu cầu

❖ Tên dự án: thiết kế MCU 8bit

- Thiết kế MCU 8bit thực hiện các nhiệm vụ đơn giải như cộng, trừ, nhân, dịch…
- Có đầu vào trao đổi tính toán dữ liệu vào RAM và ROM.



Hình 1.1.1: Sơ đồ kết nối MCU

Mục tiêu thiết kế MCU 8bit: xây dựng một vi xử lý có thể thực hiện công việc trao đổi dữ liệu đọc ghi với các thiết bị Ram và Rom, điều khiển thiết bị ngoại ngoại vi và có thể lập trình trương trình từ trên máy tính PC gửi xuống để thực hiện một tác vụ như cộng, trừ, nhân số học hoặc logic mà người dùng yêu cầu.

Thành lập nhóm:

Bảng 1.1.1: Thành lập nhóm

Tên thành viên	Vai trò
Nguyễn Văn Chuyên	Trưởng nhóm: Kiểm soát, thiết kế dự án
Lê Tiến Đạt	Thành viên: Phân tích ,thiết kế
Vũ Đình Nam	Thành viên: Thiết kế, coder

1.2 Phân tích tính khả thi thực hiện

Chỉ ra các điểm khả thi về ba điểm chính một là về kỹ thuật, hai là về mặt kinh tế và ba là về mặt tổ chức.

1.2.1.1 Khả thi về mặt kỹ thuật

Thiết kế các khối của một MCU 8bit, thực hiện thực thi mạch RLT. Để làm được có thể dùng công cụ Word để mô tả dự án, dùng Visio thực hiện vẽ các khối của, dùng ngôn ngữ verilog viết trên Quartus II, dùng ModelSim để thực hiện việc kiểm tra đánh giá và dùng kit DE2 để thực hiện việc chạy dự án trên đó.

1.2.1.2 Khả thi về mặt kinh tế

Chi phí dự kiến phát triển phần mền.

- Chi phí phân tích và thiết kế: 3.300.000 VNĐ.
- Chi phí triển khai phần cứng: 8.000.000 VNĐ.
- Chi phí bản quyền phần mền: 6.000.000 VNĐ.

Chi phí hoạt động

- Hệ thống chạy trên mạch nên cần điện năng và tổn hao trong quá trình sửa dụng chi phí trung bình vào khoảng: 300.000 VNĐ.
- Chi phí nâng cấp có thuê nhân công: 6.000.000 VNĐ.

1.2.1.3 Khả thi về mặt kỹ thuật

Thiết kế các khối của một MCU 8bit, thực hiện thực thi mạch RLT. Để làm được có thể dùng công cụ Word để mô tả dự án, dùng Visio thực hiện vẽ các khối của, dùng ngôn ngữ verilog viết trên Quartus II, dùng ModelSim để thực hiện việc kiểm tra đánh giá và dùng kit DE2 để thực hiện việc chạy dự án trên đó.

1.3 Mô tả thiết kế

Đây là tài liệu cung cấp chi tiết về mô tả thiết kế của module. Sơ đồ thiết kế, các khối tín hiệu được hiển thị bên trong của từng khối tổng cũng như thiết kế tín hiệu của từng khối chi tiết.

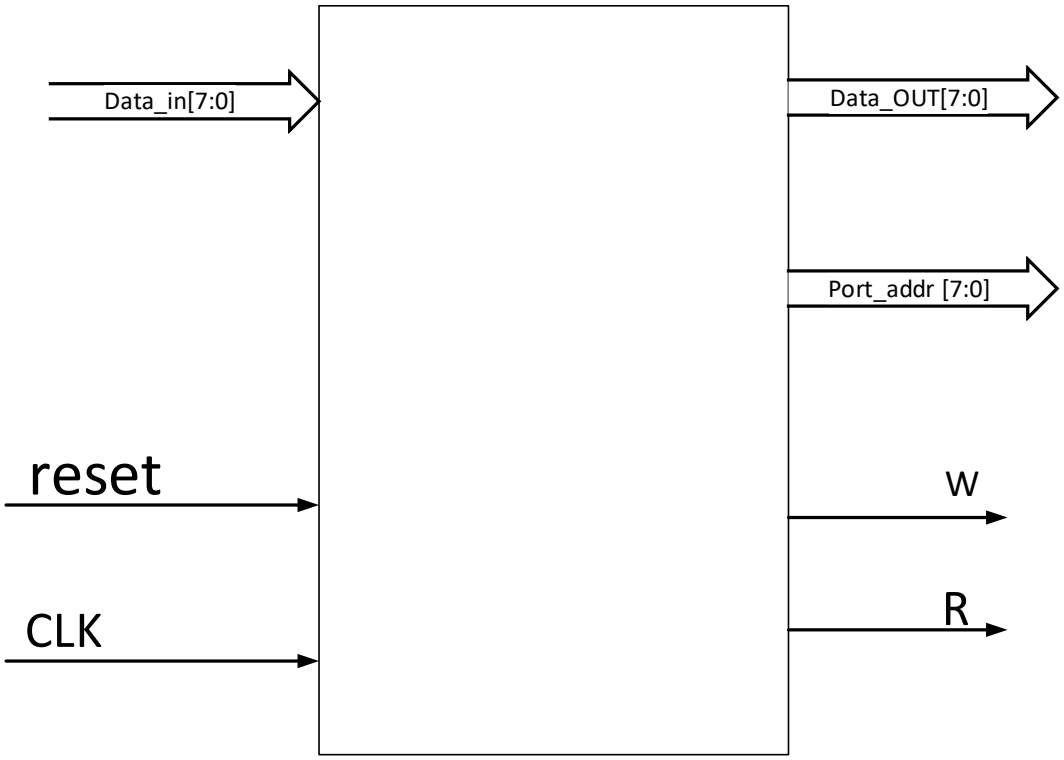
1.3.1 Mô tả MCU 8bit

MCU 8bit phù hợp, có khả năng đầy đủ của một vi điều khiển 8bit theo cấu trúc RISC được mô tả bằng ngôn ngữ 100% bằng ngôn ngữ verilog. Mã code được mô phỏng các tín hiệu vào ra thể hiện trên màn hình wave trong modelsim. MCU 8bit có thể chạy các lệnh cơ bản của một MCU 8bit được mô tả như bên dưới. Lệnh được lưu trữ trong khối lưu trữ lệnh intruction có kích thước 256 khối, mỗi khối có độ dộng 18bit, và RAM có độ dài gồm 16 thanh ghi, mỗi thanh ghi có độ dộng 8bit. Mỗi xung lệnh được chạy thông qua 9 xung CLK.

Đặng điểm MCU 8bit.

1. 8bit ALU
2. 16x8 cho kích thước thanh ghi
3. 8x8 cho kích thước RAM
4. Chín CLK/ Lệnh
5. Có các cờ nhớ, cờ tràn, cờ âm có độ dài 1bit
6. 8bit địa chỉ cổng.

1.3.2 Mô tả kết nối MCU 8bit



Hình 1.3.1: Mô tả MCU vào ra

Bảng 1.3.1: mô tả các tín hiệu của MCU 8bit

Tín hiệu	***Vào/ra***	***Mô tả***
CLK	Vào	***Clock***: tín hiệu cấp clock cho MCU 8Bit hoạt động
Reset	Vào	***Reset***: khi có tín hiệu tất cả các địa chỉ MCU 8bit trờ về không.
Data_in[7:0]	vào	***Data input***: tín hiệu cấp dữ liệu cho MCU hoạt động khi có xung CLK xườn lên vào.
Data_out[7:0]	ra	***Data output***: tín hiệu dữ liệu được tính toán và cho ra khỏi MCU khi thực hiện xong.
Data_addr[7:0]	ra	***Data address***: tín hiệu địa chỉ cổng port của vi điều khiển cho ra.

1.3.3 Mô tả chức năng các lệnh có trong MCU 8bit

Thực hiện các chức năng cơ bản của một MCU 8bit như:

1. **Memory access**: liên quan đến lệnh truy cập bộ nhớ
2. **arithmetic** : lệnh tín toán số học
3. **logic**: lệnh tín toán các phép toán liên quan đến logic
4. **Flow access:** lênh điều khiển nhảy

Bảng 1.3.2: Cấu trúc các lệnh

Memory access				
Opcodes	SM	Rd	Immediate/address	
4bits	2bits	8bit	4bit	
arithmetic				
opcodes	SM	Rs1	Rs2	Rd
4bits	2bits	4	4	4
logic				
opcodes	SM	Rs1	Rs2	Rd
4bits	2bits	4	4	4
Flow access				
opcodes	SM	Địa chỉ nhảy tới		
4bits	2bits	8		

- ***Mô tả mã lệnh***

Bảng 1.3.3: Không gian cho từng lệnh

Không gian mã nhị phân(18bit)	**Loại lệnh**	**Số lượng lệnh**
00_0000_0000_0000_0000 – 00_1111_ 1111_1111_1111	Memory access	3

01_0000_0000_0000_0000 – 01_1111_1111_1111_1111	Arithmetic	3
10_0000_0000_0000_0000 – 10_1111_1111_1111_1111	logic	10
11_0000_0000_0000 – 11_1111_1111_1111	Flow access	1 đến 5

1.3.3.1 Lệnh Memory access

Bảng 1.3.4: Lệnh Memory

SM(2bit)	Opcodes(4bit)	RD(4bit)	Dmem index(8bit)

Các lệnh **Memorry access** là lệnh truy cập bộ nhớ, giúp tải dữ liệu từ ram lưu trữ dữ liệu vào thanh ghi để phục vụ cho mục đích tính toán trong ALU vì vậy các loại lệnh trong đây sẽ bao gồm địa chỉ ô nhớ tải đến và thanh ghi đích lưu trữ dữ liệu.

Mô tả từng lệnh của khối **Memorry access.** Ở đây gồm các các trường SM 2 bit để chọn Mode là 00, trường opcode 4 bit để chọn loại lệnh thực hiện và rrrr lần lượt là địa chỉ thanh ghi cần ghi gồm 4bit , dddd_dddd là địa chỉ bộ nhớ cần đọc gồm 8bit.

Bảng 1.3.5: Các lệnh Memorry access

Lệnh	**Chức năng**	**Mã hóa(nhị phân)**	**Tác động cờ trạng thái**	**Mã lệnh (hex)** **Lệnh** **Ý nghĩa**
Lw	Tải địa chỉ ô nhớ vào thanh ghi	00_0000_rrrr_rrrr_dddd	*Không tác động*	00000 Ldi data, mem[1] addrMem[1]= data
Ldm	Tải dữ liệu từ ô nhớ vào thanh ghi	00_0001_rrrr_dddd_dddd	*Không tác động*	01000 Ldm R1,Mem[1] R1=Mem[1]

Stm	Cất dữ liệu từ thanh ghi vào trong ô nhớ	00_0010_rrrr_dddd_dddd	*Không tác động*	02000 Stm R1, Mem[1] R1=Mem[1]

1.3.3.2 Lệnh Arithmetic

Bảng 1.3.6: Lệnh Arithmetic

SM(2bit)	Opcodes(4bit)	Rs1(4bit)	Rd(4bit)	Rd(4bit)

Các lệnh toán học dùng để tính toán các phép toán số học dùng để tính toán các phép toán số học gồm có so sánh giữa hai thanh ghi để tác động lên cờ trạng thái, cộng hai số học, trừ hai số học. Các lệnh này sẽ bao gồm địa chỉ thanh ghi nguồn và thanh ghi đích, mã lựa chọn mode (**SM**), mã **Opcodes** để lựa chọn từng lệnh cụ thể trong mode.

Ở lệnh **Cmp** sẽ gồm một hai thanh ghi rs1 và rs2 được so sánh với nhau để tác động lên cờ trạng thái carry và zero. Lệnh **Add** dùng để cộng hai số học được lưu trữ trong thanh ghi rs1 và rs2 để từ đó lưu trữ vào thanh ghi đích rd sẽ tác động vào cờ trạng thái Carry, Zero và cờ tràn Over. Lệnh **Sub** dùng để trừ hai trừ hai toán hạng trong thanh ghi nguồn là rs1 và rs2 sau đó lưu trữ kết quả trong thanh ghi rd, lệnh này sẽ tác động vào cờ báo trạng thái âm S, zero, và cờ tràn O.

Bảng 1.3.7: Các lệnh Arithmetic

Lệnh	**Chức năng**	**Mã hóa**	**Tác động cờ trạng thái**	**Mã lệnh(hex)** **Lệnh** **Ý nghĩa**
Cmp	So sánh giá trị hai thanh ghi	01_0000_rrrr_rrrr_000	C, Z	10000 Cmp rs1,rs2 rs2>rs1 => C=0 rs2<rs1 => C=1
Add	Công toán học hai thanh ghi	01_0001_rrrr_rrrr_rrrr	C, Z, O	11000 Add rs1, rs2, rd rd = rs1 + rs2
Sub	Trừ toán học hai thanh ghi	01_0010_rrrr_rrrr_rrrr	S, Z, O	12000 Sub rs1, rs2, rd rd = rs2 - rs1

1.3.3.3 Lệnh Logic

Bảng 1.3.8: Lệnh Logic

SM(2bit)	Opcodes(4bit)	Rs1(4bit)	Rd(4bit)	Rd(4bit)

Ở trong đây sẽ gồm các lệnh ***And, Or, Xor, Sl0, Sl1, Sr0, Sr1, Rrl, Rrr,*** gồm có bit hai toán tử nguồn rs1 và rs2 sau khi qua phép tính toán sẽ cho ra kết quả như trong bảng dưới đây.

Bảng 1.3.9: Các lệnh logic

Tên lệnh	**Chức năng**	**Mã hóa**	**Tác động cờ trạng thái**	**Mã lệnh(hex)** **Lệnh** **Ý nghĩa**

And	Phép và số logic	10_0000_rrrr_rrrr_rrrr	Z	10000 And rs1, rs2, rd Rd= rs1 and rs2
Or	Phép hoặc logic	10_0001_rrrr_rrrr_rrrr	Z	21000 Or rs1, rs2, rd Rd = rs1 or rs2
Xor	Phép xor logic	10_0010_rrrr_rrrr_rrrr	Z	22000 Xor rs1, rs2, rd Rd = rs1 xor rs2
Not	Lệnh đảo logic	10_0011_rrrr_0000_rrrr	Z	23000 Not rs, rd Rd = not(rs)
Sl0	Dịch trái cho thêm 0 vào sau	10_0100_rrrr_rrrr_rrrr	Z, C	24000 Sl0 rd Rd = {rs[6:0], 0}
Sl1	Dịch trái cho thêm 1 vào sau	10_0101_rrrr_rrrr_rrrr	Z, C	25000 Sl1 rd Rd = {rs[6:0], 1}
Sr0	Dịch phải cho thêm 0 vào trước	10_0110_rrrr_rrrr_rrrr	Z, C	26000 Sr0 rd Rd = {0, rs[6:0]}
Sr1	Dịch phải cho thêm 1 vào trước	10_0111_rrrr_rrrr_rrrr	Z, C	27000 Sr1 rd Rd = {1, rs[6:0]}

Rrl	Quay vòng trái	10_1000_rrrr_rrrr_rrrr	Z, C	28000 Rr1 rd Rd = {rs[6:0], rs[7]}
Rrr	Quay vòng phải	10_1001_rrrr_rrrr_rrrr	Z, C	29000 Rrr rd Rd = {rs[0], rs[7:1]}

1.3.3.4 Lệnh flow control

Bảng 1.3.10: Mã Flow control

SM(2bit)	Opcodes(4bit)	Rs1(4bit)	Rd(4bit)	Rd(4bit)

Ở trong bảng dưới sẽ gồm lệnh nhảy Jmp, Jpz, Jnz, Jpc theo như mô tả trong bản dưới đây!

Bảng 1.3.11: Các lệnh Flow control

Tên lệnh	Chức năng	**Mã hóa**	**Tác động cờ trạng thái**	**Mã lệnh(hex)** **Lệnh** **Ý nghĩa**
Jmp	Nhảy khí gặp	11_0000_rrrr_rrrr_rrrr	không	30000 Jmp addr Pc = addr
Jpz	Nhảy nếu cờ zero bật lên là 1	11_0001_rrrr_rrrr_rrrr	không	31000 Jmp addr Pc = addr (zero =1)
Jnz	Nhảy nếu cờ zero bật lên là 0	11_0000_rrrr_rrrr_rrrr	không	32000 Jnz addr

				Pc = addr (zero=0)
Jpc	Nhảy nếu cờ carry bật lên là 1	11_0010_rrrr_rrrr_rrrr	không	33000 Jnc addr Pc = addr (carry=1)
Jnc	Nhảy nếu cờ carry bật lên là 0	11_0011_rrrr_rrrr_rrrr	không	34000 Jnc addr Pc = addr (carr=0)

MCU 8bit có thể thực hiện các lệnh liên quan đến việc xử lý có điều kiện cũng như những lệnh nhảy không điều kiện được mô tả trên bảng trên.[1]

1.4 Quản trị dự án

1.4.1 Yêu cầu dự án

Bảng 1.4.1: Bảng nguồn lực dự án

Loại tài nguyên	Kiểu	Từ viết tắt	Đơn vị
Word	Vật liệu	W	
PowerPoint	Vật liệu	P	
Quartus II	Vật liệu	Q	
ModelSim	Vật liệu	M	
Visio	Vật liệu	V	
Visual studio	Vật liệu	VS	
Chuyen	**Công việc**	**C**	**100%**
Dat	**Công việc**	**D**	**100%**
Nam	**Công việc**	**N**	**100%**
AnhLONG	Công việc	A	100%

Nguồn nhân lực bên trong bao gồm ba thành viên trong nhóm: Chuyên, Đạt, Nam

Nguồn nhân lực bên ngoài: AnhLong

Phần mềm thực hiện : Word, PowerPoint, Quartus II, ModelSim, Visio, Visual studio.

1.4.2 Kế hoạch dự án

Kế hoạch dự án được tạo trong file báo cáo trên phần mền Microsoft Project nhằm mục đích thực hiện phân công công việc dự án đến từng thành viên trong nhóm, xác định thời gian bắt đầu cũng như thời gian kết thúc của một dự án, các công việc đi trước có liên quan đến công việc sau bằng sơ đồ Grantt Chart.

ID	Kiểu	Tên CV	Thời gian	Thời gian bắt đầu	Thời gian kết thúc	Công việc đi trước	Tài Nguyên
1		**Xác định yêu cầu MCU**	3 d	10/4/18	10/6/18		Visio[1],Chuyen,Dat,Nam
2		**Mô tả thiết kế**	14 d	10/7/18	10/20/18	1	Chuyen,Dat,Nam,Word[1],Visio[1]
3		**Thiết kế trương trình dịch**	14 d	10/21/18	11/3/18	2	Dat
4		**thiết kế sơ đồ khối Top**	7 d	10/21/18	10/27/18	2	Nam,Visio[1]
5		**thiết kế sơ đồ khối chi tiết**	40 d	10/21/18	11/29/18	1,2,4	
6		thiết kế khối PC	9 d	10/31/18	11/8/18	2	Chuyen,Nam,Visio[1]
7		Thiết kế khối giải mã và bộ nhớ	15 d	11/1/18	11/15/18	2	Dat,Chuyen,Visio[1]
8		Thiết kế khối thanh ghi	5 d	10/22/18	10/26/18	2	Chuyen,Dat,Nam,Visio[1]
9		thiết kế khối ALU	7 d	10/21/18	10/27/18	2	Chuyen,Visio[1]
10		Thiết kế khối Control	25 d	10/26/18	11/19/18	9FS-10 d	Nam,Dat,Visio[1]
11		**thiết kế FSM**	14 d	11/1/18	11/14/18	4FS+4 d	Dat,Nam,Visio[1]
12		**Thực thi(code)**	45 d	10/31/18	12/14/18		
13		**Thực thi trương trình dịch**	13 d	11/4/18	11/16/18	3	Visual[1],Dat
14		**khối ALU**	12 d	10/31/18	11/11/18	9	
15		Lệnh access memory	7 d	10/31/18	11/6/18	9	Dat,Nam,Quartus II[1],Chuyen
16		Lệnh số học	7 d	10/31/18	11/6/18	9	Chuyen,Dat,Nam,Quartus II[1]
17		Lệnh logic	7 d	10/31/18	11/6/18	9	Chuyen,Dat,Nam,Quartus II[1]
18		Lệnh flow control	12 d	10/31/18	11/11/18	6FS-15 d	Nam,Dat
19		Thực thi khối PC	12 d	10/31/18	11/11/18	6FS-15 d	Nam,Chuyen,Quartus II[1]
20		Khối giải mã	14 d	11/7/18	11/20/18	7FS-9 d	Dat,Nam,Quartus II[1]
21		**Khối bộ nhớ**	**15 d**	**10/31/18**	**11/14/18**	**8**	
22		khối bộ nhớ(RAM)	9 d	11/4/18	11/12/18	7FS-12 d	Dat,Quartus II[1]
23		Khối thanh ghi	7 d	10/31/18	11/6/18	8	Chuyen,Dat,Quartus II[1]
24		Khối Control	18 d	11/3/18	11/20/18	10SS+8 d	Nam,Quartus II[1]
25		ghép code	24 d	11/21/18	12/14/18	14,19,20,2	
26		**Kiểm thử**	**32 d**	**11/9/18**	**12/10/18**		
27		**test trên modelsim**	**21 d**	**11/9/18**	**11/29/18**	**12**	
28		Kiểm thử khối PC	9 d	11/9/18	11/17/18	19SS+5 d	Nam,ModelSim[1],Chuyen
29		Kiểm thử khối giá mã	9 d	11/21/18	11/29/18	20	Dat,Nam,ModelSim[1]
30		kiểm thử khối thanh ghi	10 d	11/15/18	11/24/18	21	Chuyen,Dat,ModelSim[1]
31		Kiểm thử khối ALU	9 d	11/12/18	11/20/18	14	Chuyen,Dat,Nam,ModelSim[1]
32		**kiểm thử trên KIT**	11 d	11/30/18	12/10/18	27	AnhLONG,Chuyen,Dat,Nam
33		**Làm báo cáo**	7 d	12/11/18	12/19/18	26	Chuyen,Dat,Nam,PowerPoint[1],Visio[1],Word[1.25]

Ke Hoach nhóm 2

Hình 1.4.1: Kế hoạch dự án

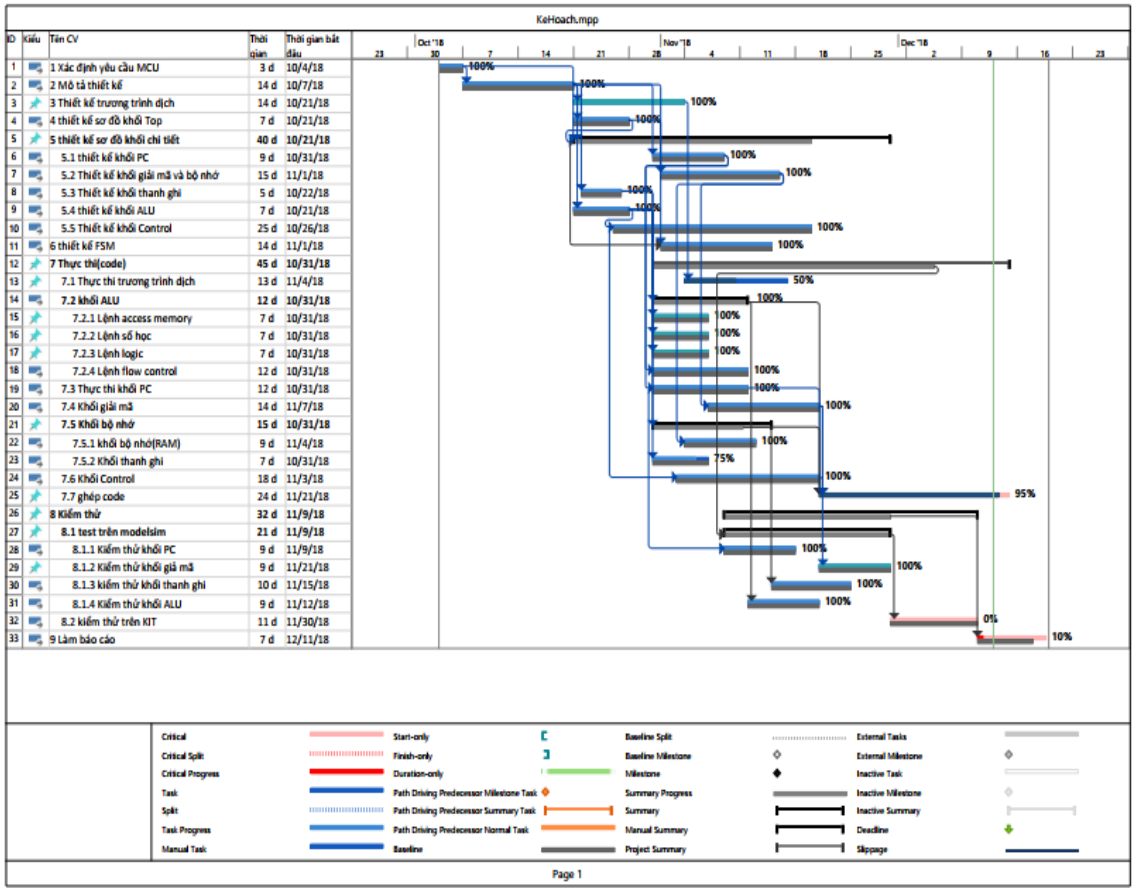
Các công việc chính của dự án bao gồm:

- Xác định yêu cầu MCU

- Mô tả thiết kế
- Thiết kế trương trình dịch
- Thiết kế khối Top
- Thiết kế chi tiết
- Thực thi và kiểm thử code
- Làm báo cáo

1.4.2.1 Theo dõi tiến độ dự án

Theo dõi tiến độ cụ thể tiến độ của dự án, công việc được theo dõi cập nhật từng tuần liên liên tiếp để đảm bảo tiến độ của dự án được theo đúng kế hoạch ban đầu nhóm đã đề ra để. Ở trong sơ đồ các công việc được đánh giá bằng mức độ phần trăm và nếu như có công việc nào của từng tuần đang chậm tiến độ trong dự án thì người đứng đầu dự án sẽ ngay lập tức có thể theo dõi, tìm ra nguyên nhân và cho ra các giải pháp khắc phục cần thiết như thêm người vào nhiệm vụ đó nếu có nhiệm vụ đã hoàn thành vượt kế hoạch để đảm bảo dự án luôn trong kiểm soát đúng tiến độ.



Hình 1.4.2: Tiến độ dự án

Nhận xét:

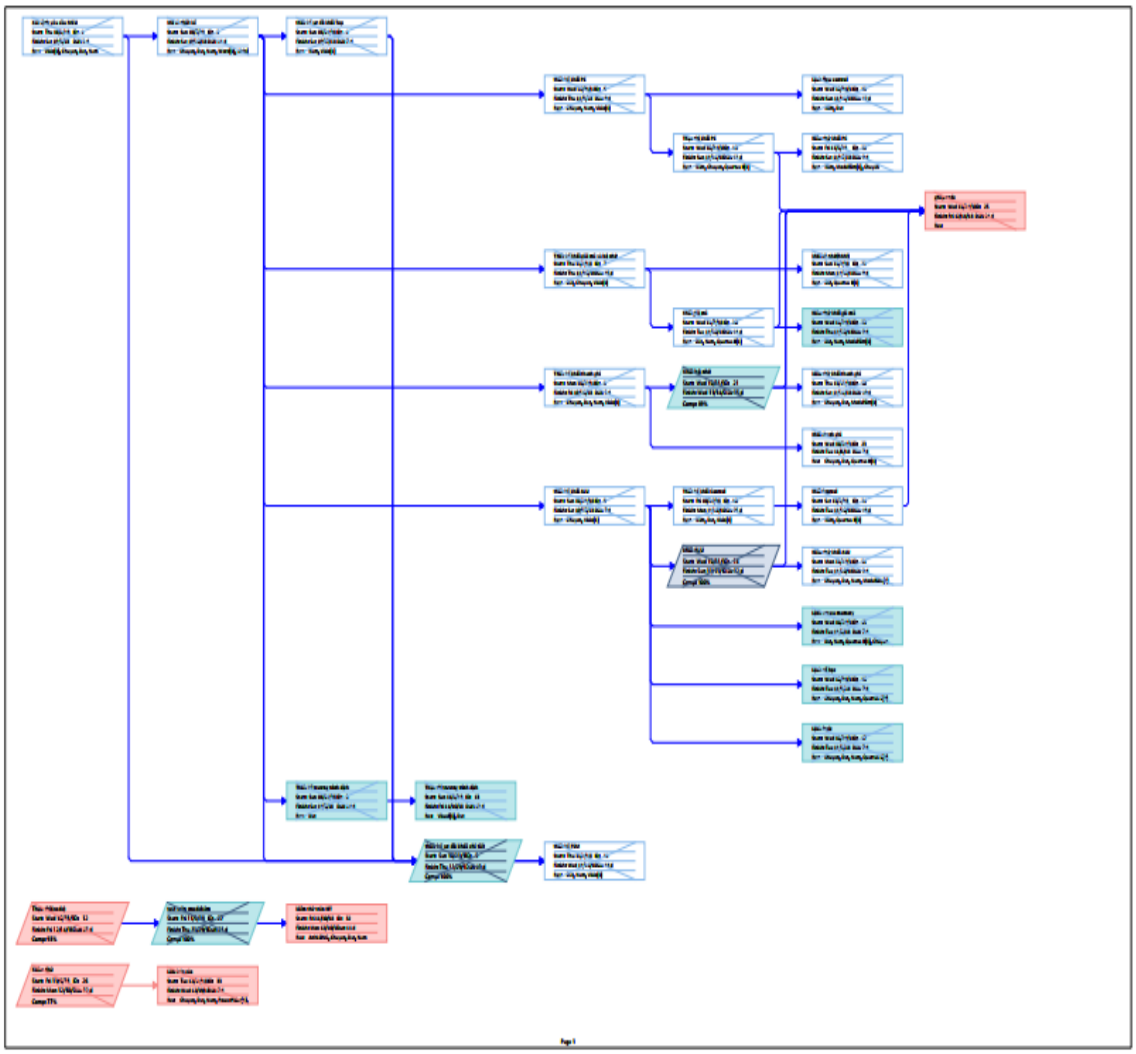
Công việc số 5.Thiết kế sơ đồ khối chi tiết có dự kiến hoàn thành trong 30 ngày nhưng thực tế do yêu cầu sửa đổi trong quá trình thực thi code cần phải có thêm thời gian để chỉnh sửa lại thiết kế khi thiết kế.

Công việc số 7.thực thi code ước lượng làm trong 35 ngày nhưng do các thành viên trong nhóm chưa có nhiều kinh nghiệm về code thêm vào đó là thiết kế có nhiều chỉnh sửa do đó công việc diễn ra không theo dự định sai số mất 11 ngày.

Công việc số 3.thiết kế trương trình dịch và số 7.1 thực thi truong trình dịch là công việc phát sinh trong quá trình thực hiện gây ra tốn thời gian và nhân lực cần cho công việc đó do đó tiến độ dự án chậm so với dự kiến mất 14 ngày.

1.4.2.2 Sơ đồ liên kết thực thể

Sơ network diagram thể hiện liên kết các công việc liên quan đến nhau trong dự án, nhìn vào sơ đồ ta thấy biểu thị được gạch chéo ký hiệu cho sự hoàn thành trong dự án. Mỗi ô trong dự án được hiển thị bao gồm tên công việc, mã ID, ngày bắt đầu, ngày kết thúc, các công cụ cần cho nhiệm vụ cũng như người thực hiện nhiệm vụ đó.

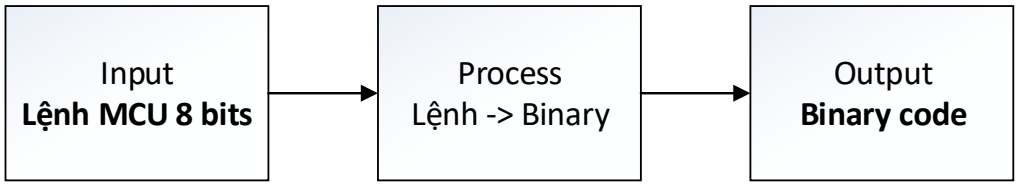


Hình 1.4.3: Sơ đồ thực thể liên kết

Chương 2 Thiết kế MCU 8bit

2.1 Thiết kế chương trình dịch mã

Chương trình dịch chuyển mã lệnh 8 bits sang mã máy dưới dạng nhị phân theo mô tả mã lệnh [1.2] được viết bằng C#. Các thanh ghi được sử dụng R0 – R15 biểu diễn bởi 4 bit [0000 - 1111].



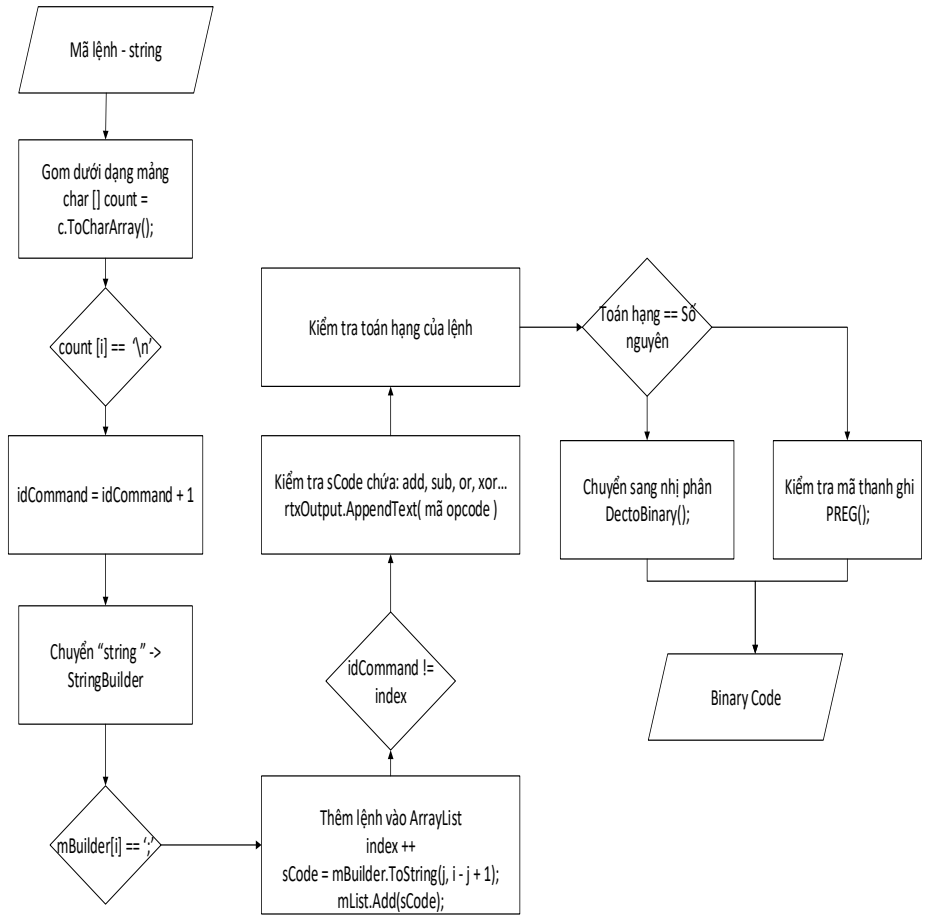
Hình 2.1.1: Sơ đồ khối chương trình dịch mã

Input: Lệnh theo cú pháp được kết thúc ";" cách nhau bởi dấu cách

Ví dụ: chương trình cộng hai số c = a + b. c, a, b lần lượt được lưu trữ r3, r1, r2

```
ldi r1 6;
ldi r2 8;
add r1 r2 r3;
stm r3 10;
```

Process: Kiểm tra từng lệnh xem có đúng cú pháp sau đó chuyển sang mã nhị phân theo quy tắc đã định nghĩa.

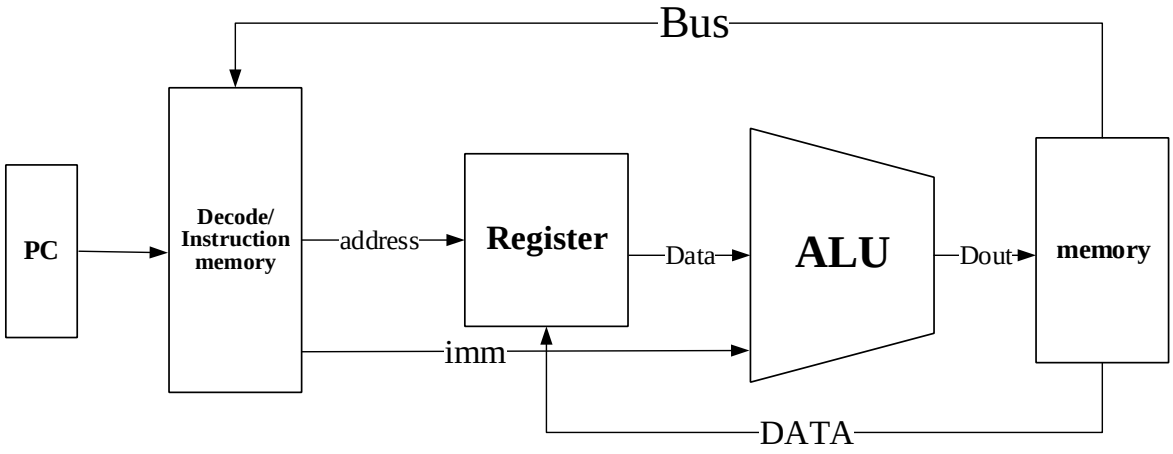


Hình 2.1.2: Sơ đồ chương trình dịch mã

Output: Bảng mã nhị phân.

2.2 Thiết kế sơ đồ khối

2.2.1 Thiết kế khối TOP BLOCK



Hình 2.2.1: TOP BLOCK

Trong thiết kế top-level nhóm thiết kế MCU 8bit gồm 5 trạng thái chính tương ứng với 5 khối chính trong. Bao gồm các khối như mô tả bảng sau.

Bảng 2.2.1: Chức năng các khối

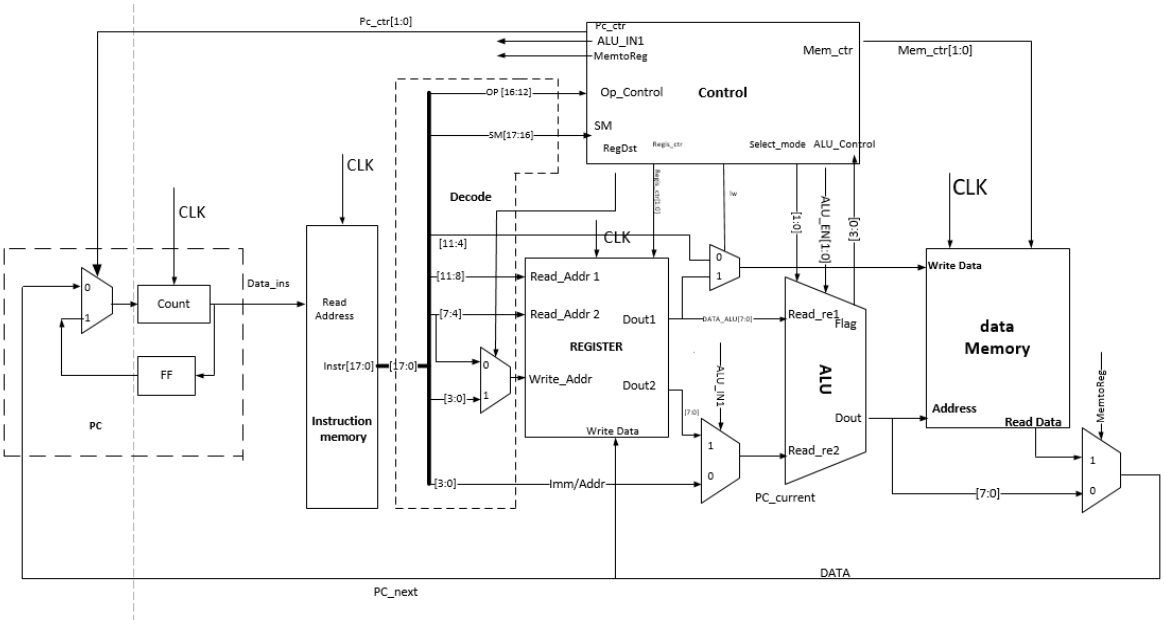
Khối	Chức năng thực hiện
PC	Thực hiện chức năng tính toán ra địa chỉ tiếp theo sẽ thực hiện trong lệnh.
Decode/ Intruction memory	Thực hiện chức năng vừa là khối lưu trữ các lệnh thực hiện của một trương trình tính toán của người lập trình cần thực thi vừa làm chức năng giải mã lệnh để đưa ra các bít cho vào các khối tiếp theo trong MUC 8bit.
Register	Thực hiện nhiệm vụ lưu trữ dữ liệu tính toán để đưa và MCU và bộ nhớ trung gian trao đổi dữ liệu giữ Ram và MCU.
ALU	Thực hiện chứ năng tính toán số liệu được đưa từ thanh ghi, Ram như cộng, trừ, nhân, dịch… để đưa số liệu ra ngoài lưu trữ ở thanh ghi, Ram, giá trị nhảy ở PC.
Ram	Thực hiện chức năng lưu trữ dữ liệu từ của trương trình tính toán.

2.2.2 Thiết kế khối TOP DETAIL

Các block detail thiết kế bao gồm mô tả các chi tiết các trạng thái vào ra cũng như kết nối của từng khối theo số bit với nhau được thiết kế bao gồm:

- ***Khối PC***: có đầu vào 8bit, đầu ra là địa chỉ 8bit có nhiệm vụ trỏ đến câu lệnh tiếp theo được lưu trữ, ban đầu tín hiệu PC sẽ có mặc định là ô nhớ đầu tiên trong khối lưu trữ lệnh.
- ***Intruction memory***: được dùng để làm lưu trữ các lệnh tiếp theo sẽ được thực hiện trong trương trình, khôi có kích thức 256 ô nhớ, mỗi ô có độ dài 18bit.
- ***Register***: là khối lưu trữ các giá trị tính toán đưa vào ALU, khối có kích thước 16 thanh ghi, mỗi thanh ghi có độ dài 8bit.

- ***ALU***: Thực hiện chức năng tính toán.
- ***Data memory***: thực hiện việc lưu trữ giá trị tính toán, khối có kích thước 256 ô nhớ, mỗi ô có độ dài 8bit.
- ***Control***: làm việc đưa ra các tín hiệu đến các khối khác trong MCU làm nhiệm vụ điều khiển hoạt động của toàn bộ các khối khi có nhận được mã lệnh từ memory intruction.[2]



Hình 2.2.2: BLOCK DETAIL

2.3 Thiết kế các trạng thái

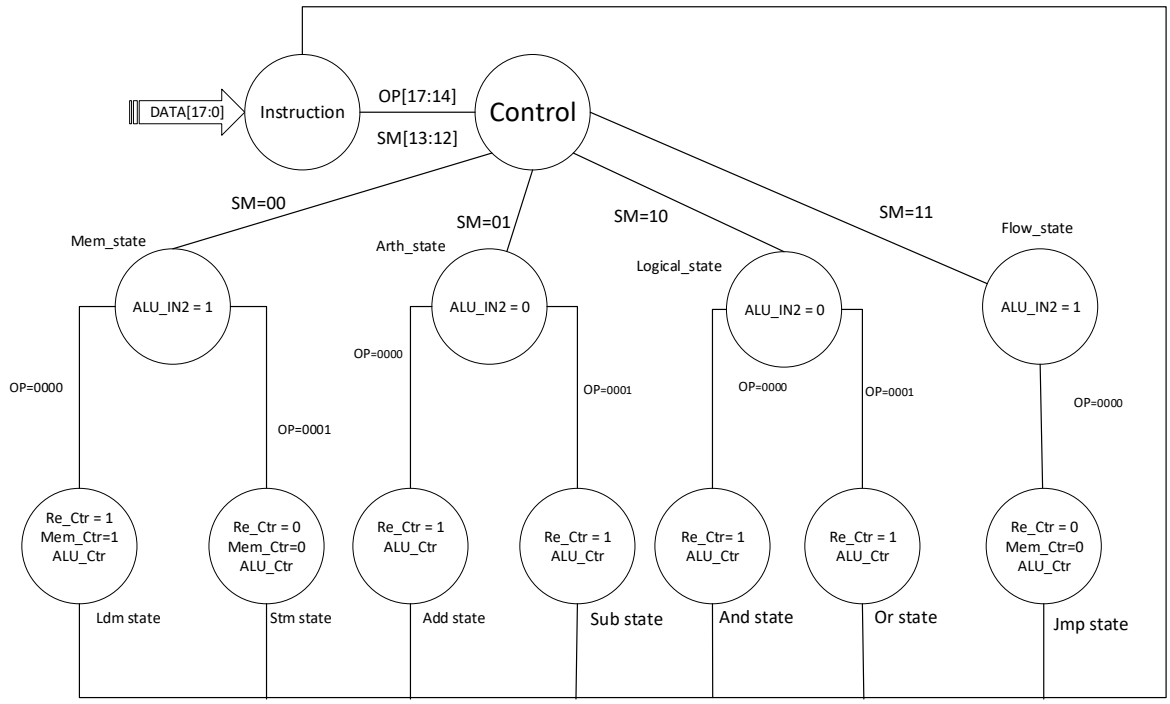
Khi nhận được các tín hiệu được giải mã trong khối intruction memory MCU thực hiện bao gồm 4 trạng thái chính:

Intruction: trạng thái đọc dữ liệu từ trong khối lưu trữ dữ liệu ra khi có thanh ghi PC chỉ đến địa chỉ cần đọc.

Control: trạng thái đưa ra các tín hiệu điều khiển khi nhận được mã opcode và select mode từ khối intruction đưa tới.

ALU_IN: Trạng thái tính toán trong ALU bao gồm 4 trạng thái trong đó khi nhận được tín hiệu điều khiển từ trạng thái control gửi đến trong mã select mode(SM).

***Memory và một số lệnh khác*:** Khi nhận được mã opcode ALU sẽ thực hiện các công việc để nhảy vào các trạng thái tiếp theo, nếu ALU nhận được lệnh tính toán đến số học, logic, truy cập bộ nhớ sẽ có trạng thái lưu trữ được thực hiện, nếu ALU nhận được các lệnh liên quan nhảy sẽ trực tiếp quay trở về intruction để thực hiện lệnh tiếp theo dựa vào khối PC.



Hình 2.3.1: Sơ đồ trạng thái

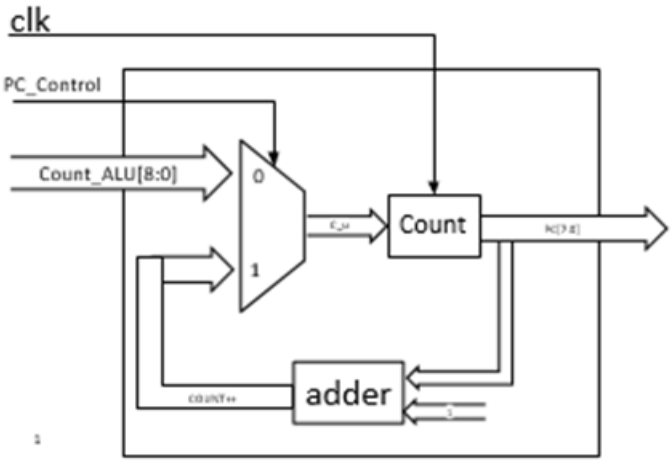
2.4 Thiết kế chi tiết từng khối

Đưa ra thiết kế tổng thể cũng như các thiết kế chi tiết được chỉ ra trong từng khối cụ thể.

2.4.1 Thiết kế khối PC

Trình bày thiết kế chi tiết về khối **PC** bao gồm thiết kế của khối, timing, mô tả vào ra, và lưu đồ thuật toán.

Thiết chi tiết khối PC



Hình 2.4.1: Khối PC

Trong thiết kế gồm có 1 bộ mux 1bit điều khiển thực hiện chứ năng nhận lệnh từ khối điều khiển control để chọn địa chỉ lệnh cần nhảy tới một là địa chỉ nhảy như bình thường(tăng 1 đơn vị) hai là nhảy tới vị trí đã chỉ định trong những lệnh liên qua đến điều khiển luồng. Một bộ đếm count_ALU[8:0] thực hiện công việc đếm chỉ định đến lệnh tiếp theo cần nhảy. Một bộ cộng adder 8bit thực hiện phép tính cộng tăng một đơn vị để nhảy đến lệnh kế tiếp cần chỉ tới khi thực hiện tính toán thông thường mà không có tín hiệu điều khiển luồng từ khối control.

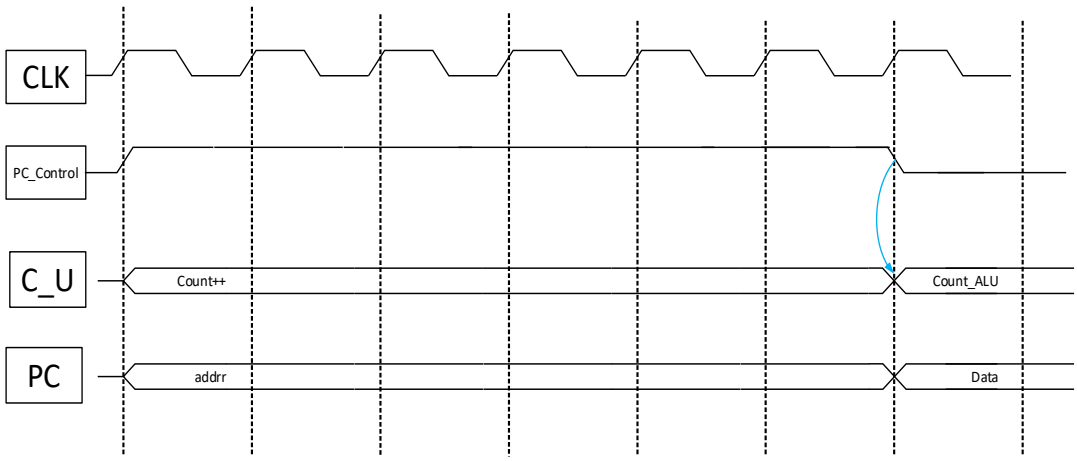
- Mô tả vào ra khối PC

Tín hiệu vào ra của khối PC được mô tả thiết kế như trong bảng dưới đây:

tín hiếu	kết nối	I/0	Mô tả
PC_Control	PC_Contrl	I	tín hiệu chọn loại lệnh nhảy có điều kiện hay không có
count	Data(PC_next)	I	tính toán ra lệnh tiếp theo cần nhảy của lệnh có điều kiện
Intruction	Intruction	O	địa chỉ cần nhảy tới trong thanh ghi lệnh

Hình 2.4.2: Mô tả vào ra khối PC

- Mô tả timming khối PC

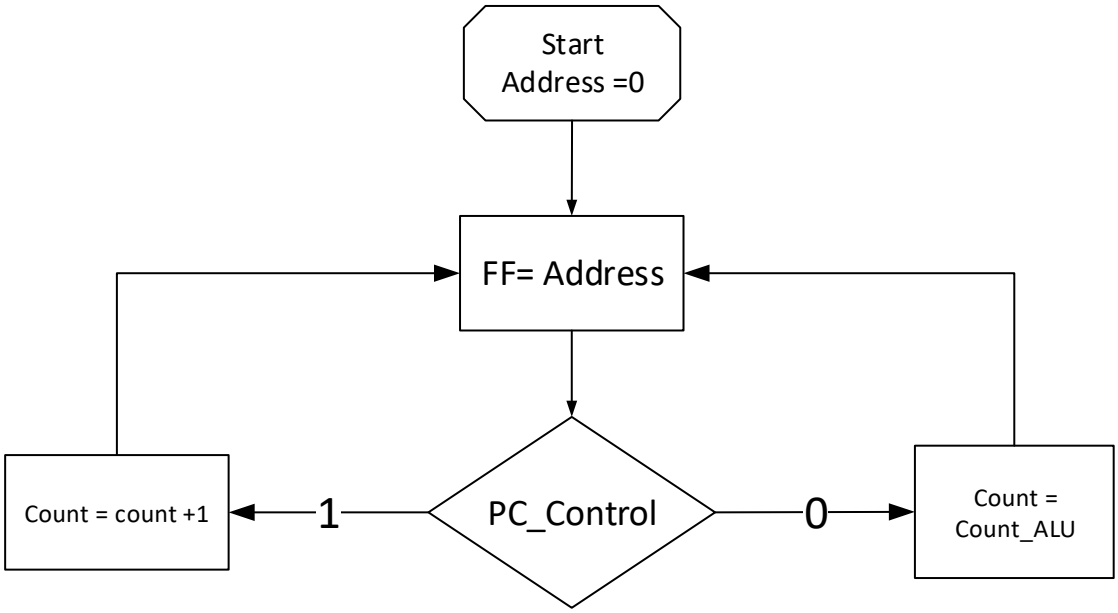


Hình 2.4.3: Timming PC

Khối PC: sườn lên số 1: khởi tạo giá trị và tính toán, sườn lên số 2: output.

Tín hiệu timming được thực hiện sẽ nằm ở xung **CLK** đầu tiền của đồng hồ, khi xuất hiện trạng thái xườn lên của xung đầu tiên tín hiệu **PC_Control** luôn luôn chỉ là 1 khi thực hiện lệnh đầu tiên => dữ liệu từ **PC[7:0]** sẽ có mặc định địa chỉ nhảy tới là 1 đơn vị khi tín hiệu **PC_Control** chưa đảo trạng thái về 0, khi tín hiệu **PC_Control** đảo trạng thái về 0 => tín hiệu đầu ra **PC[7:0] .**

- Lưu đồ thuật toán khối PC



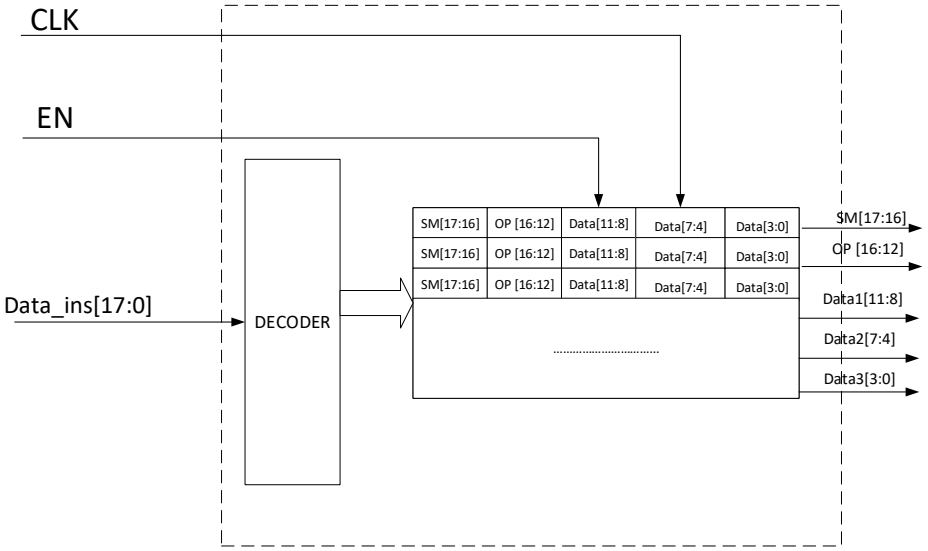
Hình 2.4.4: Lưu đồ thuật toán trong khối PC

Ban đầu khi khởi động **MCU 8bit** lên địa chỉ lệnh nằm trong khối **PC** trỏ đến luôn nằm ở ô nhớ đầu tiên của các lệnh nằm trong Intruction và khối **count** lưu trữ giá trị

đầu tiên của dữ liệu. Tín hiệu đầu vào **PC_Control** kiểm tra xem nếu tín hiệu là 0 thì chọn đầu vào của khối **count** là **Count_ALU**, ngược lại nếu tín hiệu là 1 thì chọn đầu vào khối **count** tăng lên 1 đơn vị ô nhớ.

2.4.2 Thiết kế khối Instruction

- Thiết chi tiết khối giải mã lệnh



Hình 2.4.5: Khối giải mã lệnh

Khối Instruction gồm 2 khối con:

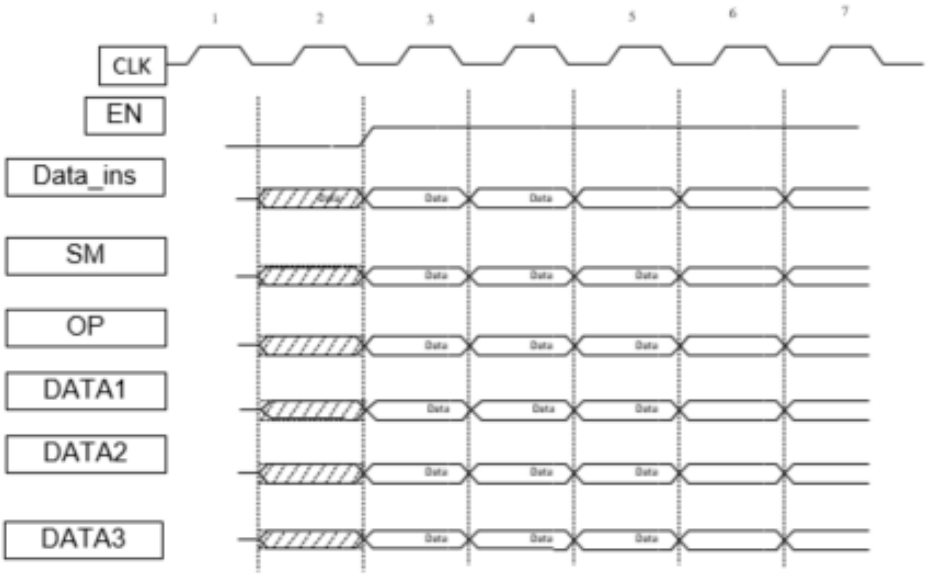
- Khối Decode: phân tích dữ liệu output từ khối PC thành các trường SM, OP, Data 1, Data 2, Data 3 lấy ở trong khối lưu trữ mã op.
- Khối lưu trữ mã op: lưu trữ các mã op.

- Mô tả vào ra khối giải mã lệnh

Tín hiệu	Kết nối	I/0	Mô tả
EN	EN_INT	I	Tín hiệu cho phép khối hoạt động
Data_ins[17:0]	Data_ins	I	Mã lệnh
SM	SM	O	Tín hiệu select mode
OP	OP	O	Tín hiệu mã opcode
Data1[11:8]	Read_Addr1	O	Dữ liệu để ghi vào thanh ghi
Data2[7:4]	Read_Addr2	O	Dữ liệu đọc từ thanh ghi
Data3[3:0]	Write_Addr	O	Dữ liệu ghi vào thanh ghi

Hình 2.4.6: Mô tả và ra khối INS

- Mô tả timming khối giải mã lệnh



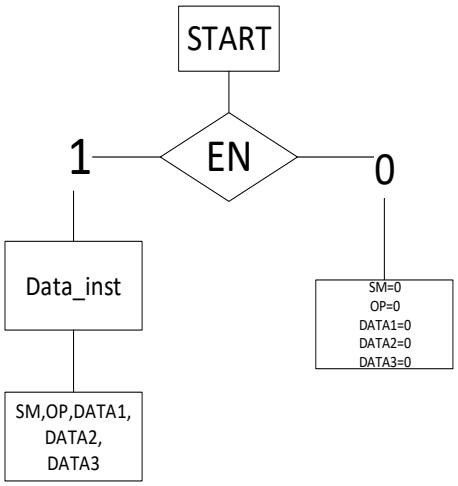
Hình 2.4.7: Timming INS

Khối Instruction: sườn lên số 2 khởi tạo giá trị và tính toán. Sườn 3: output.

Khối Instruction sẽ hoạt động ở chu kỳ thứ 2 sau khối PC. Khi đến chu kỳ thứ 2, tín hiệu EN được kích hoạt sẽ làm khối INSTRUCTION hoạt động, tín hiệu Data_ins sẽ được đọc và các tín hiệu ra tương ứng sẽ được tính toán ngay trong chu kỳ đó.

Tín hiệu đâu vào Data_INS là địa chỉ mã lệnh gồm 18 bit và đầu ra mã hex sẽ được phân tích và đưa đầu ra tương ứng với với tín hiệu SM, OP, DATA1, DATA 2, DATA 3.

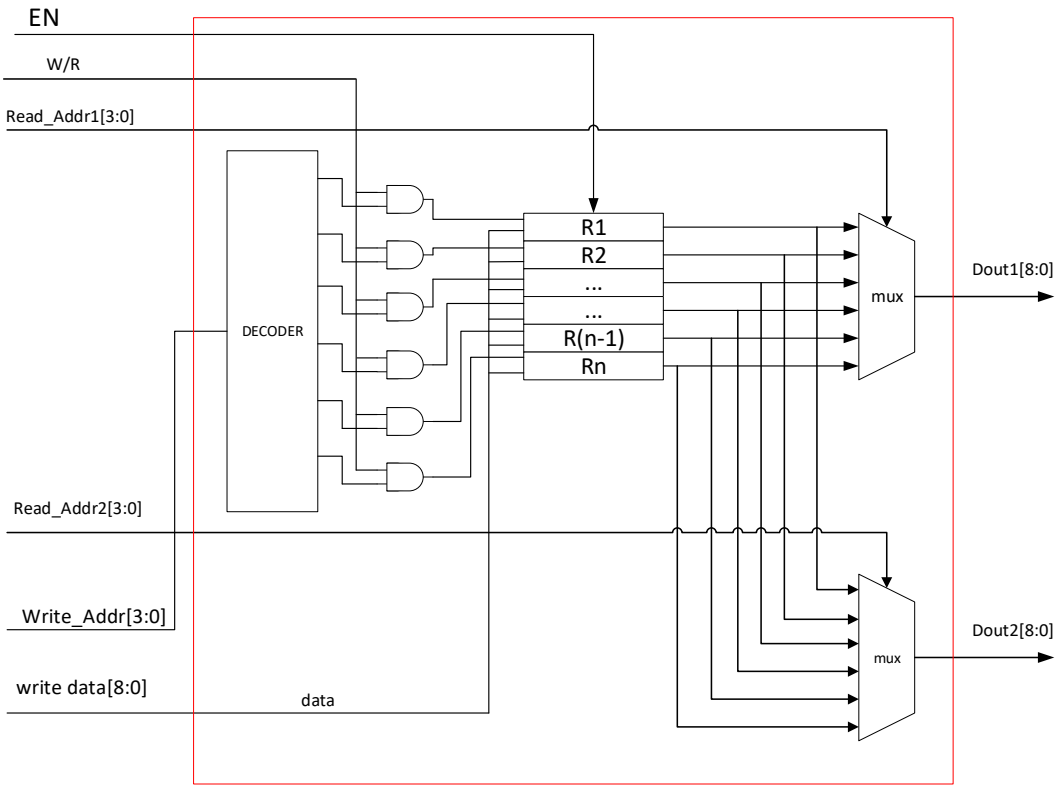
- Lưu đồ thuật toán khối giải mã lệnh



Hình 2.4.8: Lưu đồ thuật toán INS

2.4.3 Thiết kế khối Register

- Thiết chi tiết khối thanh ghi



Hình 2.4.9: BLOCK khối PC

Khối Register gồm 3 khối con:

- Khối Decoder: giải mã tín hiệu Write_Addr, kết hợp với bộ AND để điều khiển Read_Write của hệ thống.
- Khối lưu trữ dữ liệu: lưu trữ data của bộ register.

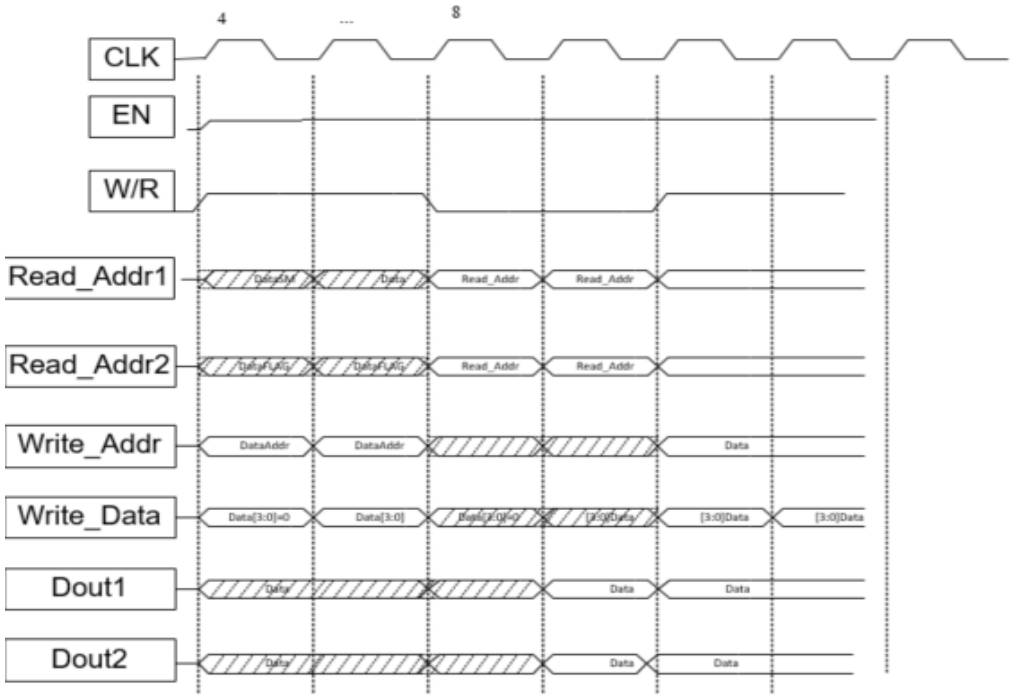
- Khối mux: lựa chọn tín hiệu đầu ra dựa trên các thanh ghi.

➢ Mô tả vào ra khối thanh ghi

Tín hiệu	Kết nối	I/0	Mô tả
W/R	Re_Ctr	I	Tín hiệu điều khiển Read=0/Write=1
Read_Addr1	Read_Addr1	I	Địa chỉ đọc thanh ghi
Read_Addr2	Read_Addr2	I	Địa chỉ đọc thanh ghi
Write_Addr	Write_Addr	I	Địa chỉ ghi thanh ghi
Write_Data	Write_Data	I	Dữ liệu để ghi vào thanh ghi
Dout1	Read_Re1	O	Dữ liệu đọc từ thanh ghi
Dout2	Read_Re2	O	Dữ liệu đọc từ thanh ghi

Hình 2.4.10: Mô tả vào ra khối register

➢ Mô tả timming khối thanh ghi

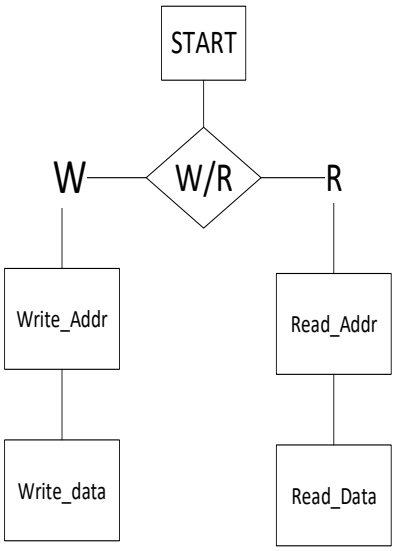


Hình 2.4.11: Sơ đồ Timming register

Khối Register: sườn lên số 4: khởi tạo giá trị và read data. Sườn lên số 5 output. Sườn số 7 khởi tạo giá trị và readata. Sườn lên số 8 write data.

Khối REGISTER sẽ hoạt động ở chu kỳ thứ 4 với chức năng đọc dữ liệu và ở chu kỳ thứ 8 sẽ có chức năng là ghi dữ liệu. Tương ứng với các chu kỳ thì tín hiệu đầu ra sẽ được tính toán tương ứng.

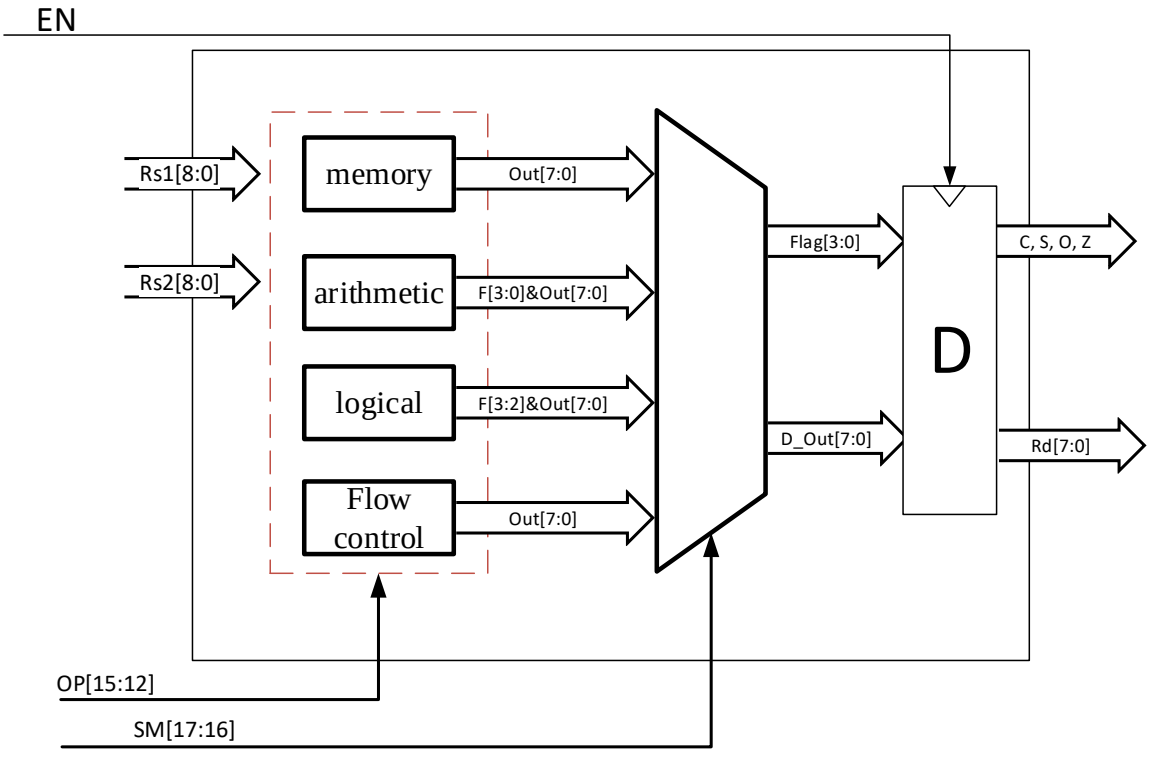
➢ Lưu đồ thuật toán khối thanh ghi



Hình 2.4.12: Sơ đồ thuật toán khối Register

2.4.4 Thiết kế khối ALU

- Thiết chi tiết khối ALU



Hình 2.4.13: BLOCK ALU

Thiết kế khi nhìn từ trái qua phải bao gồm 3 khối chính: khối tính toán, khối mux, và khối D flip-flop.

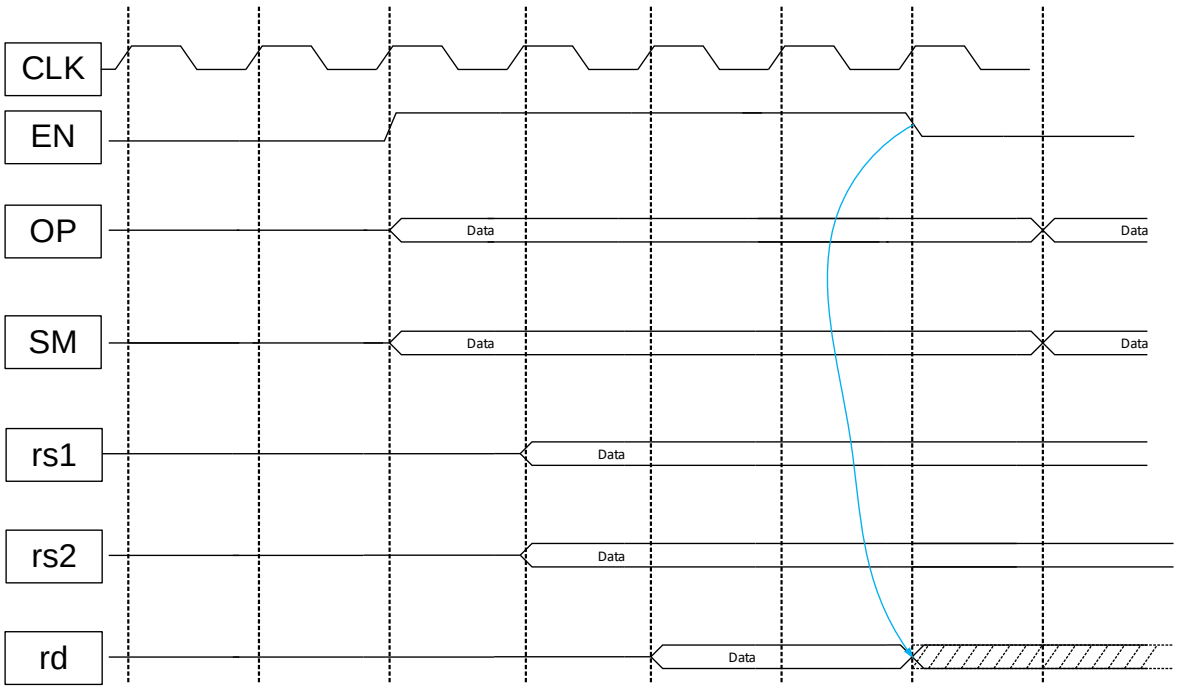
- tính toán chứa trong đó là các khối thực hiện chức năng như tính toán số học, logic, nhảy, và truy cập bộ nhớ.
- Khối mux thực hiện chức năng chọn đầu ra tín hiệu dựa trên lệnh điều khiển SM được đưa ra khi giải mã lệnh.
- Khối D flip-flop thực hiện chứ năng dữ kết quả tính toán cho đến khi có xung CLK được đưa từ bên ngoài vào mới cho kết quả đầu ra.

➢ Mô tả vào ra khối ALU

tín hiệu	kết nối	I/O	Mô tả
EN	CLK	I	tín hiệu báo kết nối ALU
Rs1	Data_rs1	I	Dữ liệu đi từ thanh ghi 1 vào
Rs2	Data_rs2	I	Dữ liệu đi từ thanh ghi 2 vào
OP	OP_control	I	Mã opcode để chọn từng lệnh cụ thể trong mode
SM	SM_control	I	select mode để chọn loại lệnh
flag	C, Z, O, S	O	cờ nhớ(C), cờ báo không(Z), cờ báo tràn(O), cờ báo âm(S)
out	out	O	Kết quả tính toán được từ khối ALU

Hình 2.4.14: Mô tả vào ra khối ALU

➢ Mô tả timming khối ALU

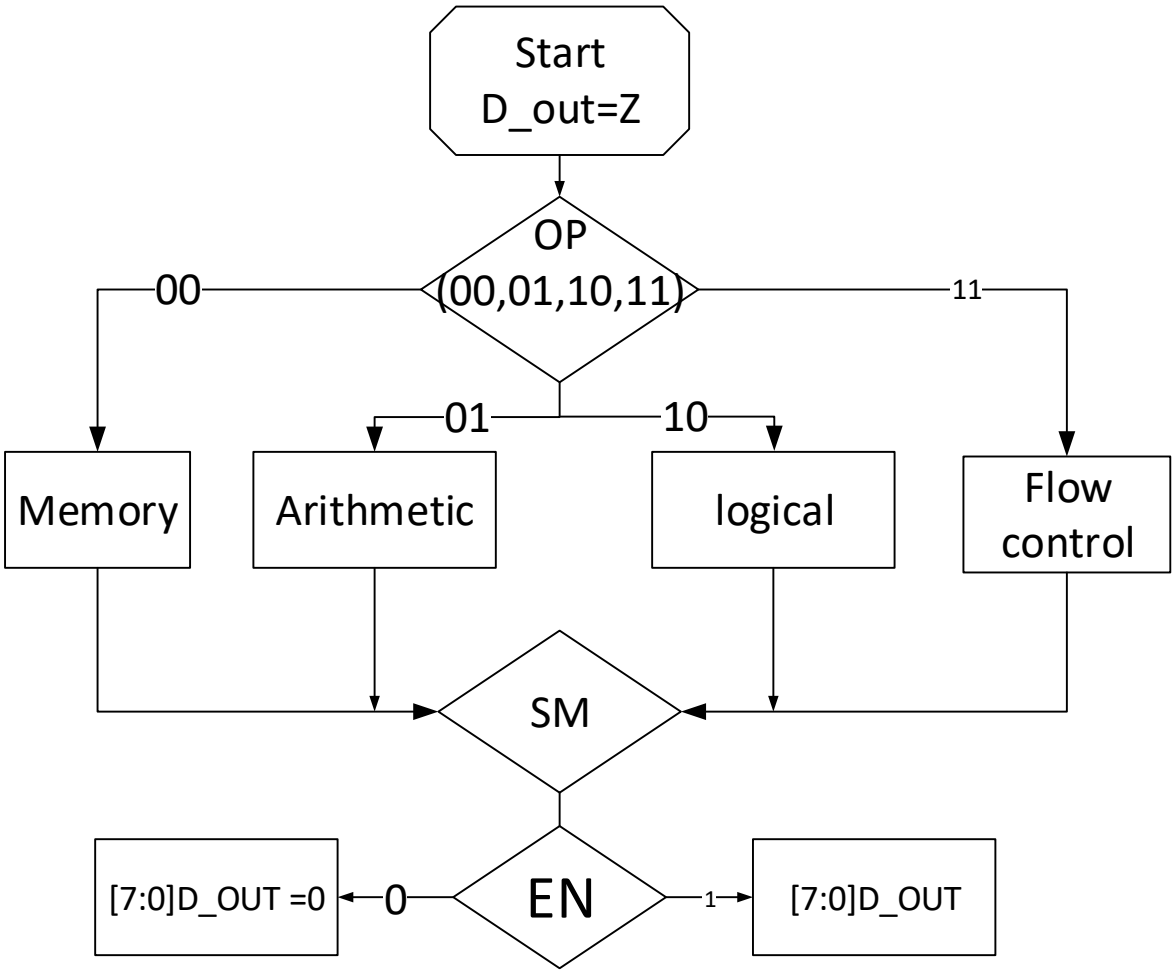


Hình 2.4.15: Timming khối ALU

Bộ tính toán ALU: sườn lên thứ 5 nhận giá trị đầu vào. Sườn lên thứ 6 output đầu ra.

Bộ tính toán **ALU** sẽ nằm ở khối xung lệnh 5 của xung **CLK**, khi xung **CLK** thực hiện được đến chu kỳ thứ 3 tín hiệu **EN, OP, SM** báo hiệu **ALU** có thể bắt đầu được cho phép thực hiện công việc tính toán, xung thứ 4 của **CLK** dữ liệu được đọc ra từ thanh ghi làm đầu vào cho khối **ALU**, xung thứ 5 dữ liệu được xuất ra từ **ALU** để hoàn thành một task vụ tính toán kết thúc một câu lệnh trong khối **ALU**.

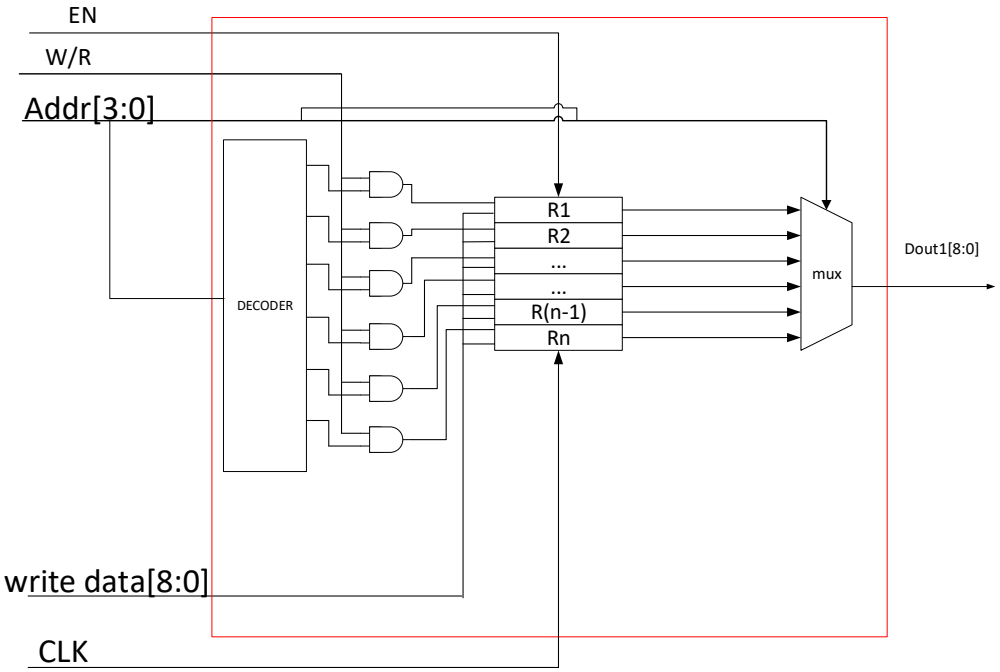
- Lưu đồ thuật toán khối ALU



Hình 2.4.16: Lưu đồ thuật toán ALU

2.4.5 Thiết kế khối memory

- Thiết chi tiết khối lưu trữ dữ liệu



Hình 2.4.17: BLOCK MEMORY

Khối Memory gồm 3 khối con:

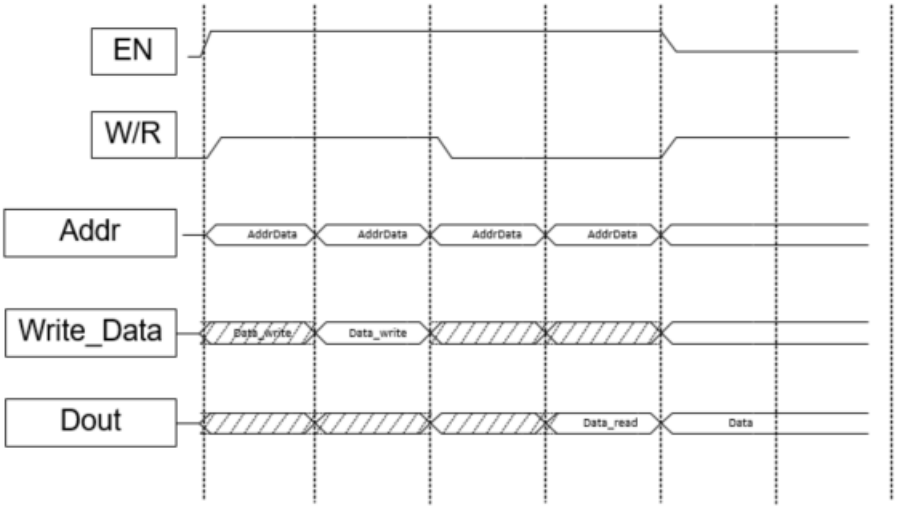
- Khối Decoder: giải mã lệnh Addr.
- Khối lưu trữ dữ liệu: lưu trữ data được cấp phát.
- Khối mux: lựa chọn dữ liệu trong khối lưu trữ.

➢ Mô tả vào ra khối lưu trữ dữ liệu

Tín hiệu	Kết nối	I/0	Mô tả
EN	EN_INT	I	Tín hiệu cho phép khối hoạt động
W/R	Write_Addr	I	Tín hiệu write hoặc read
Addr	Addr	I	Địa chỉ read hoặc write
Write_data	Data	I	Dữ liệu ghi vào ram
Dout	Dout	O	Dữ liệu đọc từ ram

Hình 2.4.18: Mô tảvào ra khối Mem

➢ Mô tả timming khối lưu trữ dữ liệu



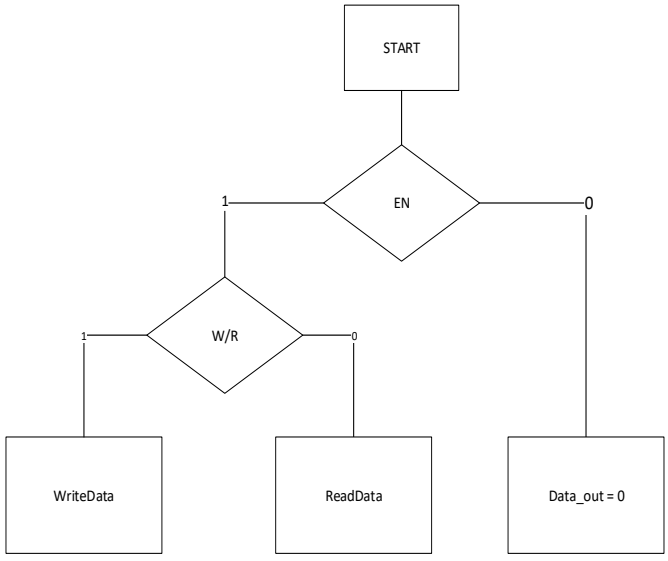
Hình 2.4.19: Timming mem

Khối Memory sườn lên thứ 7 read data, sườn lên thứ 9 write data.

Khối Memory sẽ được hoạt động ở chu kỳ thứ 6 của xung clk. Khối memory sẽ có 2 chức năng là read hoặc write tùy thuộc vào chức năng từng câu lệnh.

Khi có tín hiệu Write, khối memory sẽ thực hiện ghi dữ liệu Write_Data vào bộ nhớ có địa chỉ là đầu vào Addr. Nếu có tín hiệu Read, khối sẽ có chức năng đọc dữ liệu từ bộ nhớ có địa chỉ là Addr.

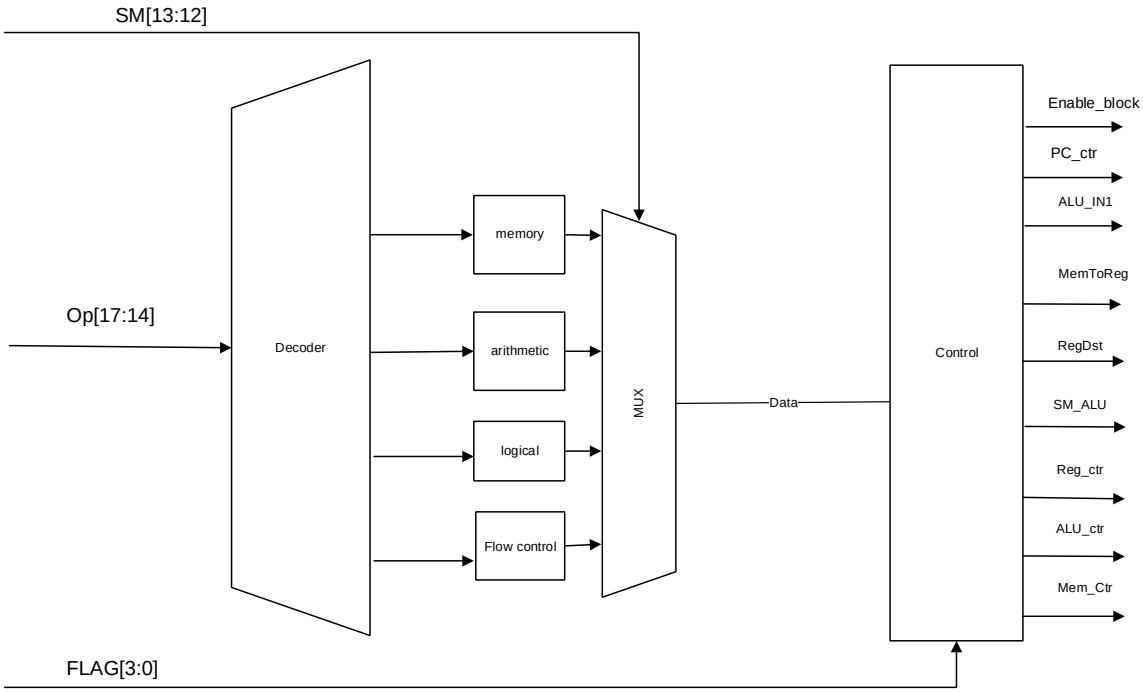
➢ Lưu đồ thuật toán khối lưu trữ dữ liệu



Hình 2.4.20: Lưu đồ thuật toán MEM

2.4.6 Thiết kế khối control

- Thiết chi tiết khối control



Hình 2.4.21: BLOCK CONTROL

Khối control được chia làm 3 khối con

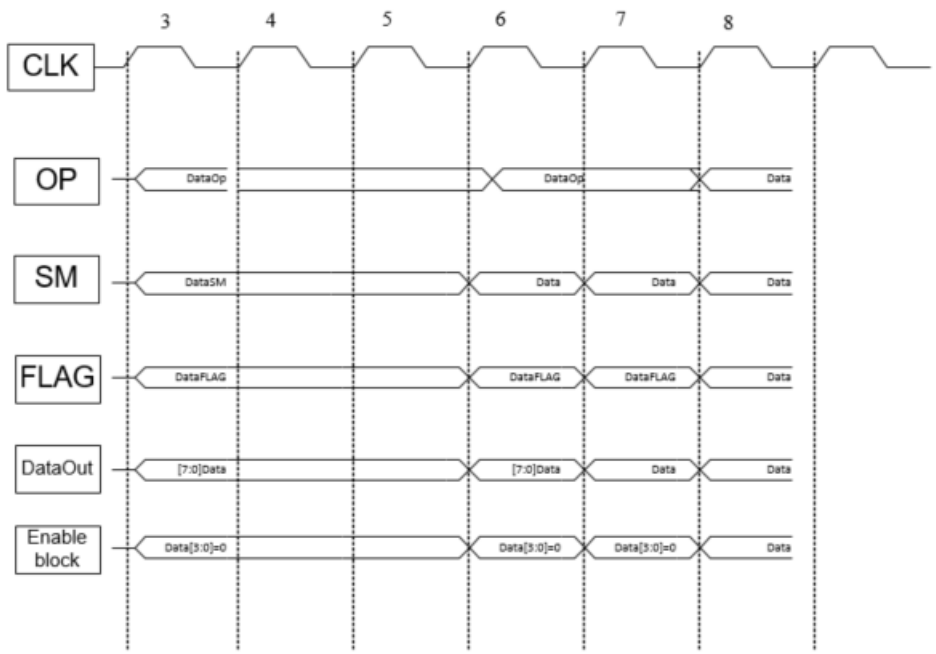
- Khối Decoder: giải mã lệnh OP.
- Khối Mux: Lựa chọn các tính hiệu OP.
- Khối Control: Dựa vào đầu ra bộ mux để setup các tín hiệu đầu ra điều khiển các khối.

- Mô tả vào ra khối control

Tín hiệu	Kết nối	I/0	Mô tả
SM	SM_control	I	Chọn mode tín hiệu
Op	OP_control	I	MÃ opcode của lệnh
FLAG	FLAG	I	cờ nhớ(C), cờ báo không(Z), cờ báo tràn(O), cờ báo âm(S)
PC_ctr	PC_Ctr	O	Tín hiệu điều khiển khối PC
ALU_IN1	ALU_IN1	O	Tín hiệu điều khiển chọn đầu vào ALU
MemToReg	MemToReg	O	Tín hiệu điều khiển Write_Data
RegDst	RegDst	O	Tín hiệu điều khiển Write_Addr
SM_ALU	SM_ALU	O	Chọn mode trong ALU
REG_ctr	REG_ctr	O	Tín hiệu điều khiển Register write hoặc read data
ALU_ctr	ALU_ctr	O	Tín hiệu enable ALU
Mem_Ctr	Mem_Ctr	O	Tín hiệu điều khiển Memory write hoặc read data
Enable_block	Enable	O	Tín hiệu kích hoạt các khối PC, REG, ALU, MEM

Hình 2.4.22: Mô tả vào ra khối control

- Mô tả timming khối control

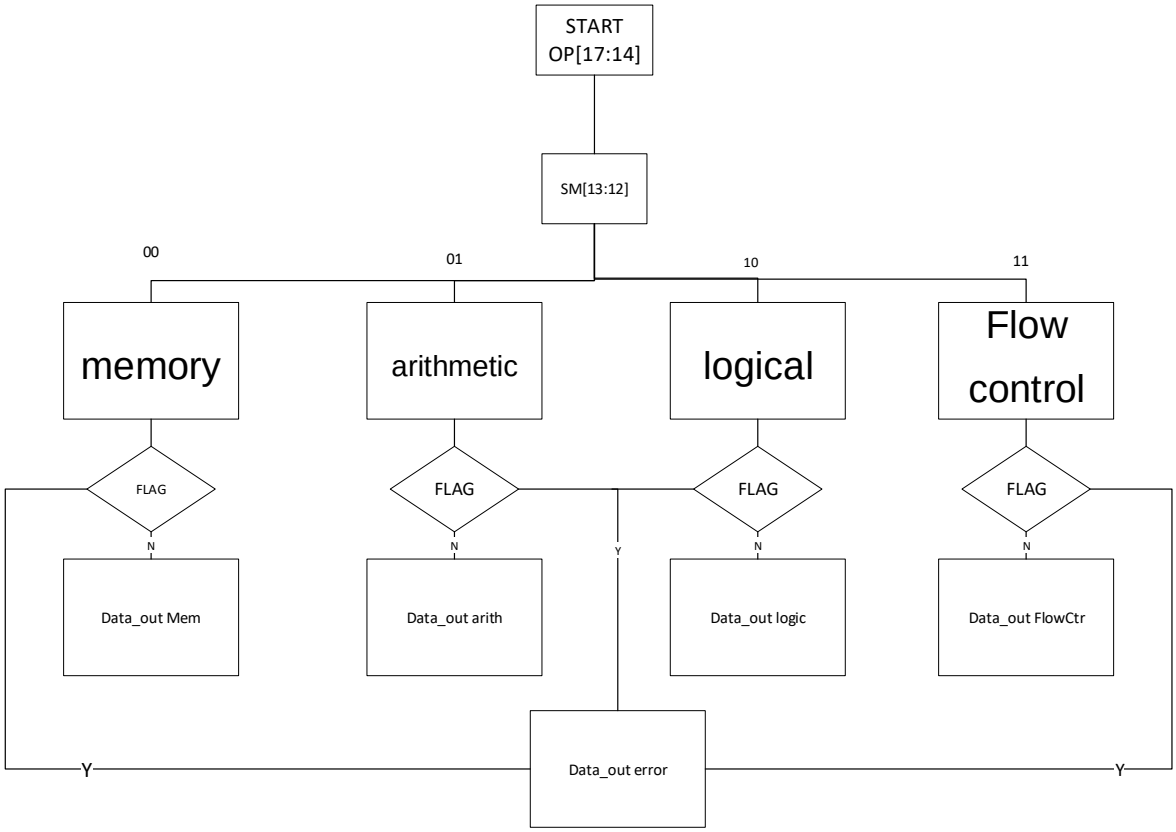


Hình 2.4.23: Timming control

Khối Control là khối điều khiển các tín hiệu Control của các khối register, memory, ALU. Khối sẽ hoạt được hoạt động ở chu kỳ thứ 3 sau khối PC và khối INSTRUCTION.

Tại chu kỳ thứ 3, đầu vào OP sẽ được phân tích và khối Control sẽ phân loại kiểu dữ liệu và cho đầu ra tương ứng để điều khiển các khối còn lại ở những chu kỳ tiếp theo.

- Lưu đồ thuật toán khối control

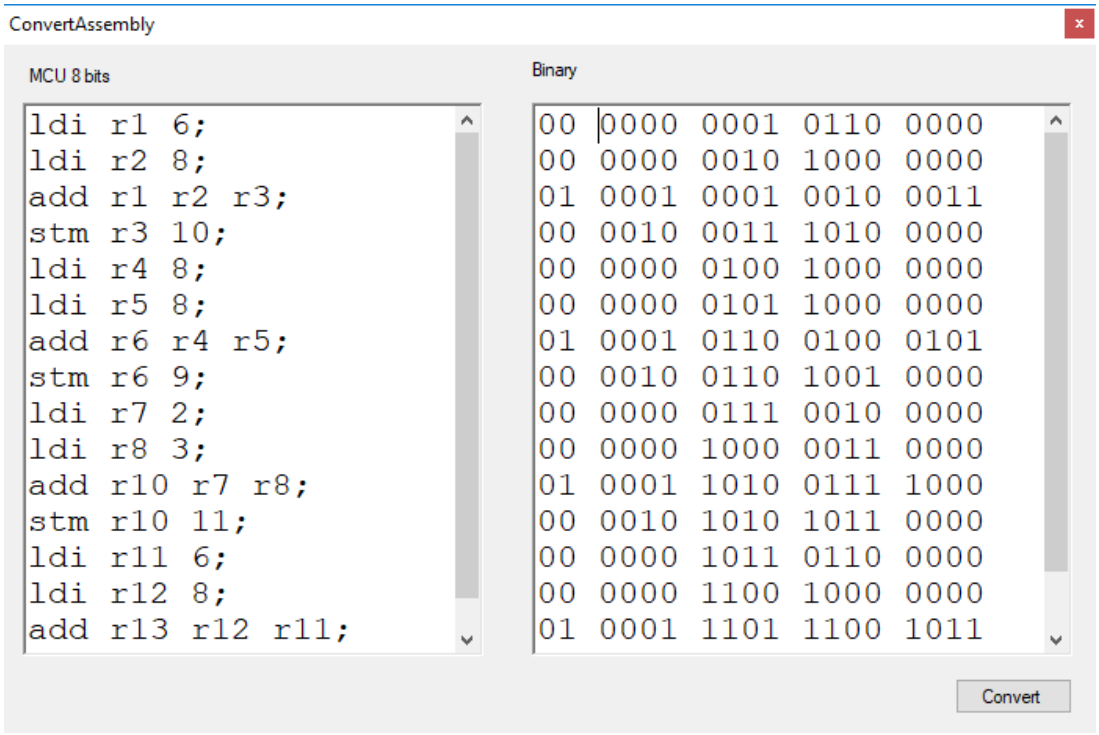


Hình 2.4.24: Lưu đồ thuật toán control

Chương 3 Thực thi và kiểm thử

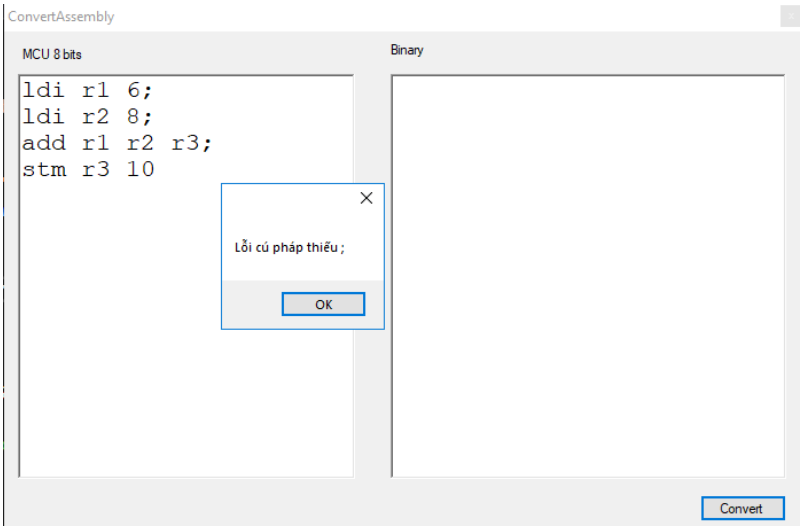
3.1 Thực thi trương trình dịch

Quá trình thực thi, kiểm thử phần mềm trình biên dịch lệnh sang mã nhị phân. So sánh kết quả đầu ra so với giá trị tính toán.



Hình 3.1.1: Chương trình thực hiện

Sau khi nhập xong mã lệnh nhấn nút “Convert”, chương trình hiển thị bảng mã nhị phân và lưu dưới dạng file .txt để Quartus đọc các mã lệnh.



Hình 3.1.2: lỗi cú pháp

Chương trình có chức năng báo lỗi sai cú pháp, người dùng sửa lại lệnh.

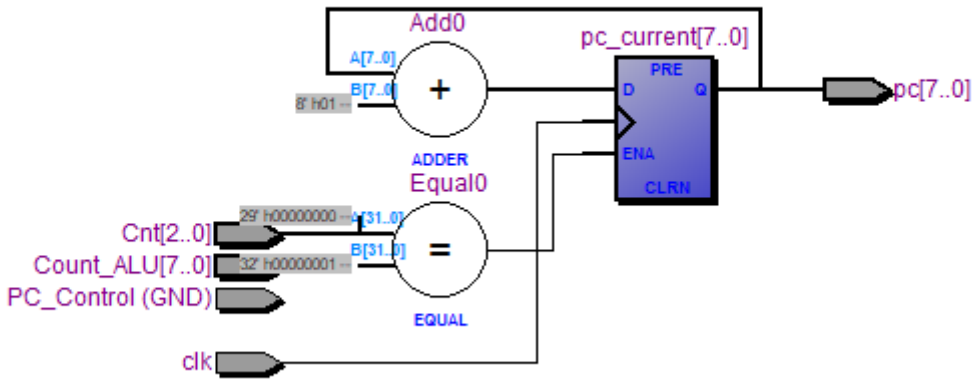
3.2 Thực thi và kiểm thử từng khối

Điễn đạt quá trình thực thi,kiểm thử trong thiết kế các khối trong ALU 8bit và cuối cùng là đưa ra các nhận xét đánh giá về mạch logic để từ đó rút ra kinh nghiệm cho các phiên bản kế tiếp.

3.2.1 Thực thi khối PC

Mô tả tổng qua về khối PC khi thực bằng code verilog, đưa ra các đánh giá thiết kế, ưu, nhược điểm có trong thiết kế để có thể tương thích với những khối khác trong thiết kế góp phần nhận biết các lỗi khi gộp các khối lại.

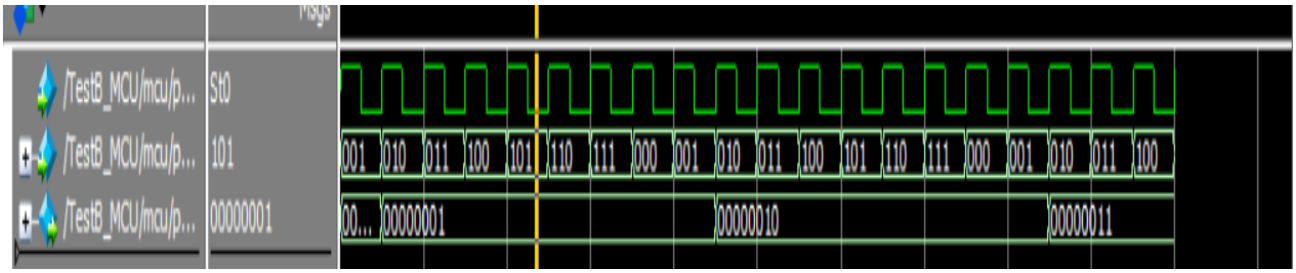
❖ Thực thi RTL khối PC



Hình 3.2.1: RTC PC

Sau khi code chạy kiểm thử nhóm thu được khối PC như trên làm nhiệm chỉ đến lệnh tiếp theo cần nhảy trong MCU 8bit.

❖ Kiểm thử khối PC trên test bench

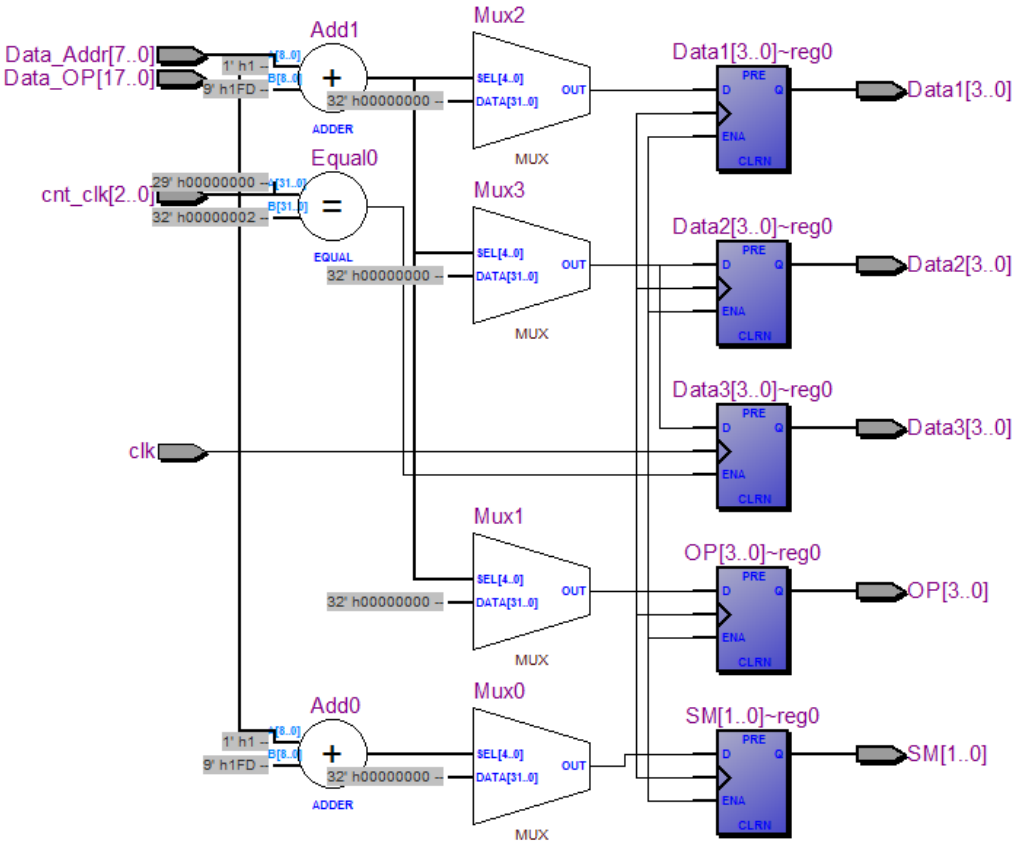


Hình 3.2.2: Test Bench PC

Theo như hình trên, tín hiệu CNT_CLK sẽ được cộng thêm 1 ở mỗi chu kỳ thứ 2. Tín hiệu này sẽ điều khiển số thứ tự lệnh sẽ được thực thi trong chương trình.

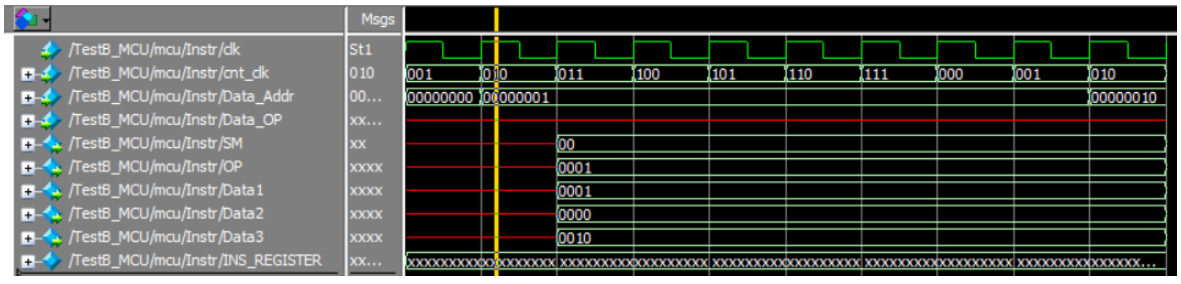
3.2.2 Thực thi khối Instruction

❖ Thực thi RTL khối Instruction



Hình 3.2.3: RTL INS

❖ Kiểm thử khối giải mã lệnh

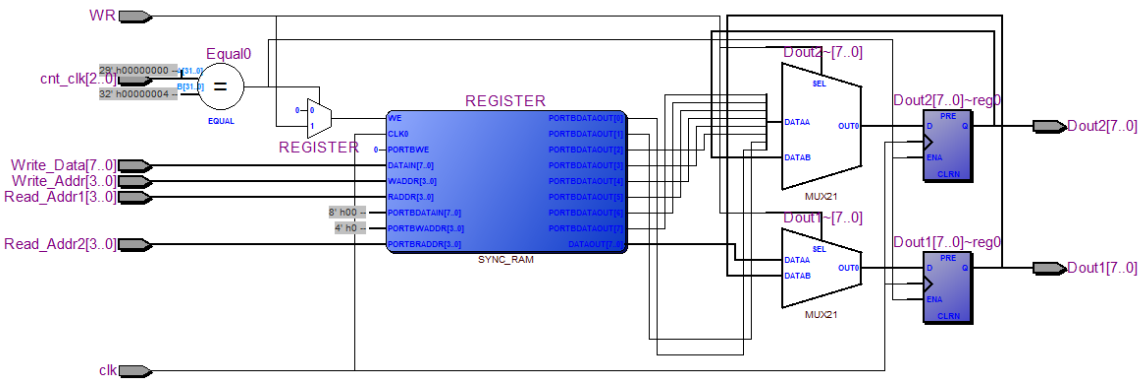


Hình 3.2.4: Wave INS

Theo như hình trên, mã hexa sẽ được thực thi trong bộ nhớ của khối là thanh ghi thứ 01. Thanh ghi này có chứa dữ liệu là 00_0001_0001_0000_0010 và sau khi qua khối sẽ được phân tích và tách ra các đầu ra tương ứng: SM = 00, OP = 0001, Data1 = 0001, Data2 = 0000, Data3 = 0010 sẽ đưa vào khối ALU để thực hiện tính toán.

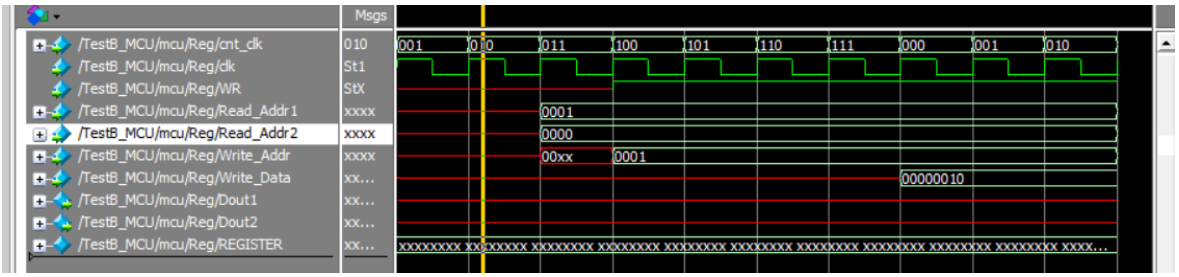
3.2.3 Thực thi khối Register

❖ Thực thi RTL khối thanh ghi



Hình 3.2.5: RTL REG

3.2.3.2 Kiểm thử khối Register

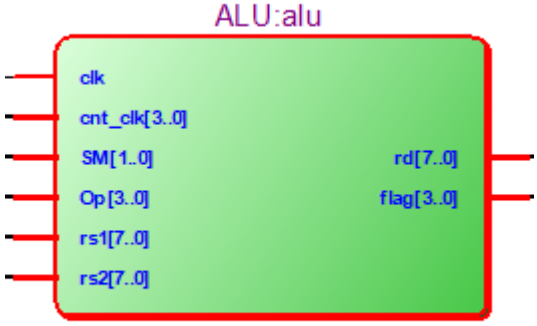


Hình 3.2.6: Wave REG

Khối Register sẽ được thực hiện ở chu kỳ thứ 3, các tín hiệu đầu vào Read_Add1 và Read_Add2 sẽ được đọc. Ở chu kỳ thứ 8 thì tín sẽ Write data vào thanh ghi register khi thực hiện mã Ldm.

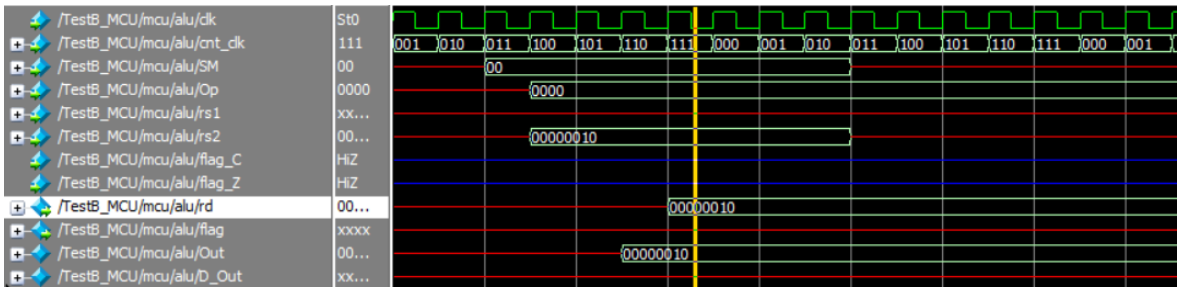
3.2.4 Thực thi khối ALU

❖ Thực thi RTL khối ALU



Hình 3.2.7 RTL ALU

Kiểm thử khối ALU

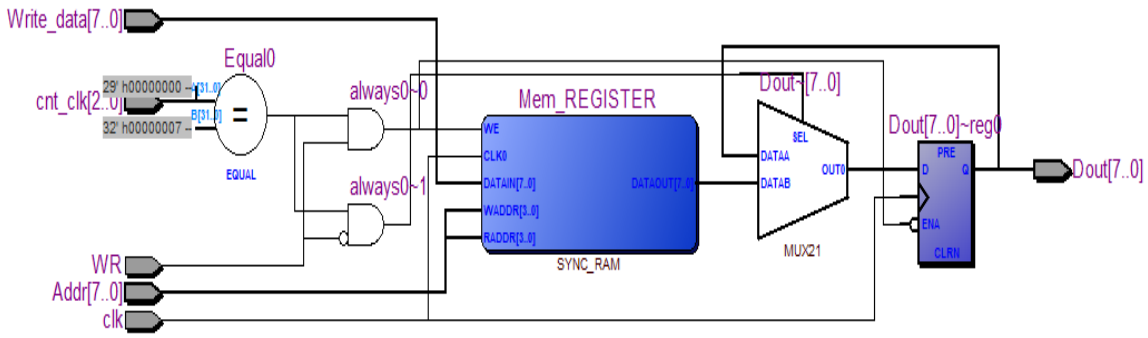


Hình 3.2.8: Wave ALU

Khối ALU sẽ được thực hiện và lấy đầu ra ở chu kỳ thứ 5. Ở đây, thực hiện lệnh ldm, dữ liệu đầu vào rs2 = 00000010 sẽ được tính toán và đưa ra tín hiệu đầu ra tương ứng đưa vào khối memory để tải tải dữ liệu.

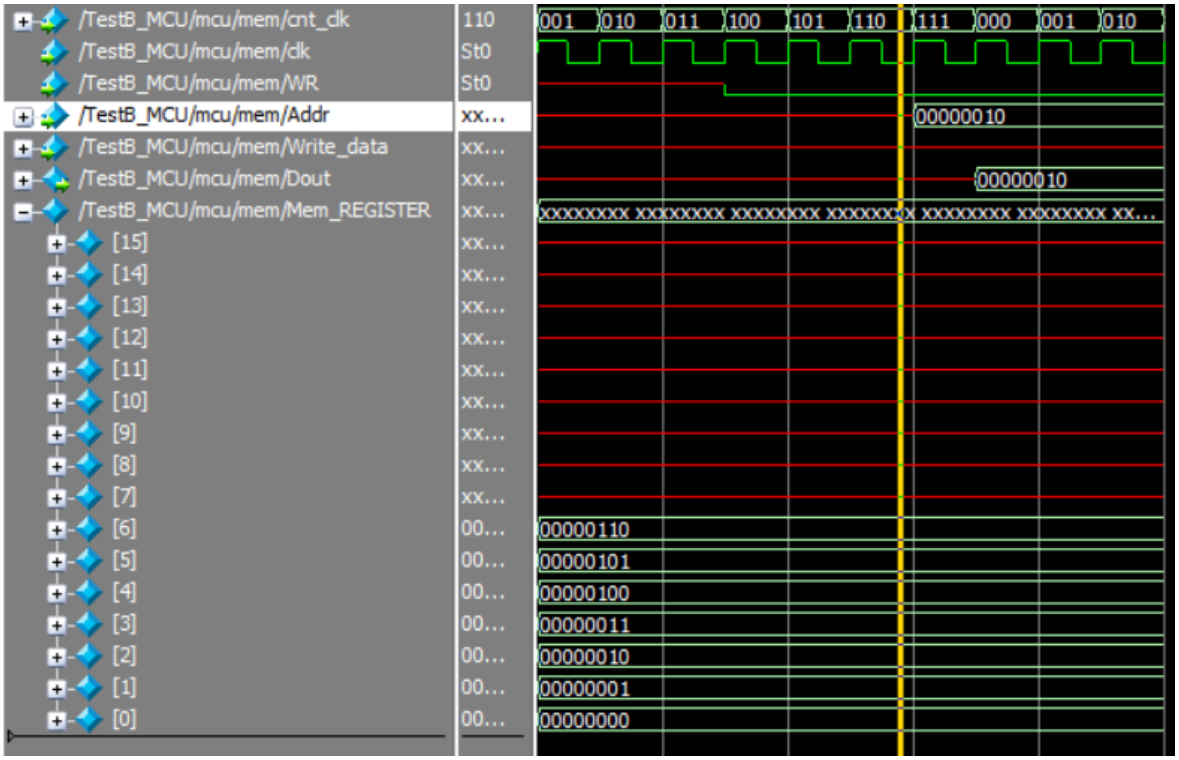
3.2.5 Thực thi khối Memmory

Thực thi RTL khối Memory



Hình 3.2.9: RTL mem

Kiểm thử khối lưu trữ dữ liệu

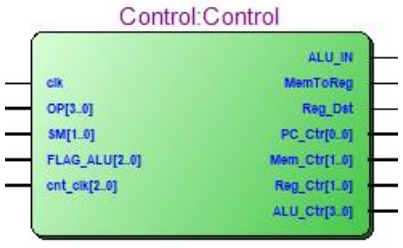


Hình 3.2.10: Wave MEM

Khối memory là khối lưu trữ dữ liệu có chức năng Write/ Read tưng ứng. Ở đây, dữ liệu sẽ được tải ở chu kỳ đầu tiên và đếm chu kỳ thứ 7 với tín hiệu Read thì dữ liệu ở thành ghi 2 sẽ được load ra và cho đầu ra Dout = 00000010 tương ứng với thanh ghi thứ 2 ở Mem_REGISTER.

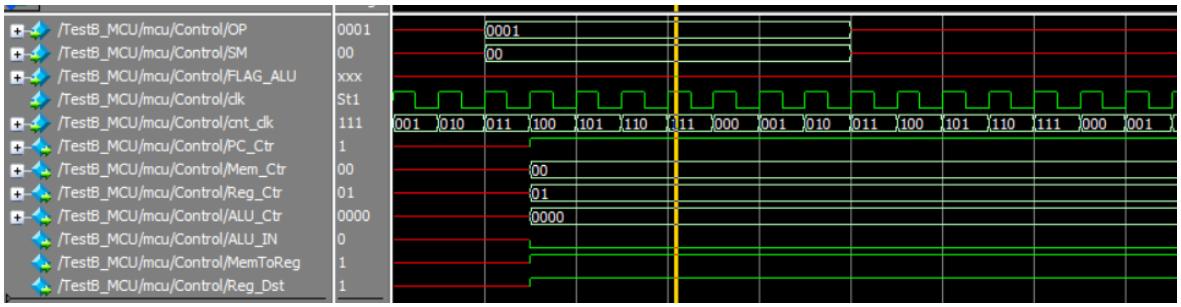
3.2.6 Thực thi khối control

Thực thi RTL khối control



Hình 3.2.11: RTL Control

Kiểm thử khối control



Hình 3.2.12: Wave control

Khối Control sẽ được thực hiện ở chu kỳ thứ 4. Sau khi tín hiệu input OP và SM được đưa vào thì các tín hiệu ra tương ứng là PC_Ctr = 1, Mem_Ctr = 00, Reg_Ctr = 01, ALU_Ctr = 0000, ALU_IN = 0, MemtoReg = 1, Reg_Dst = 1.

3.3 Kiểm thử ghép nối khối

Để kiểm tra hoạt động của tất cả các khối, ta sẽ thực thi đoạn chương trình sau:

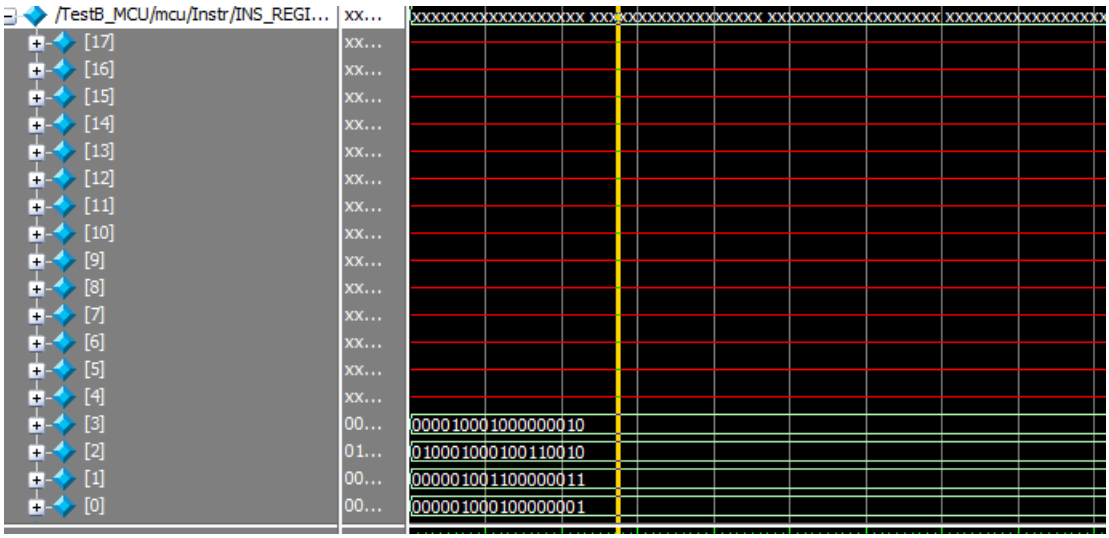
Ldm r1 m1

Ldm r3 m3

Add r1 r3 r2

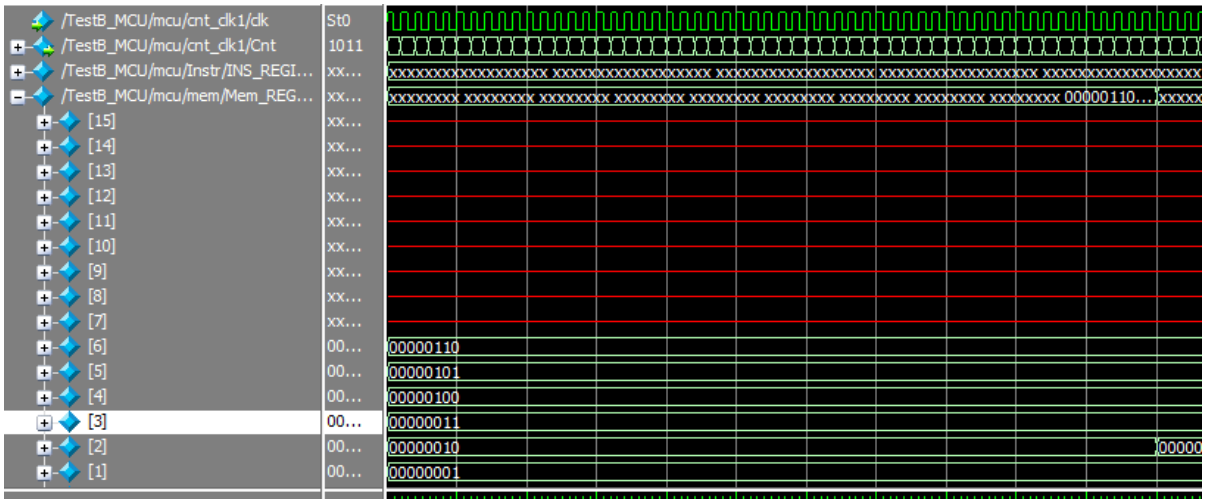
Stm r2 m2

- ❖ Kiểm thử trên test bench (Kiểm tra các thanh ghi Reg, Memory, Ins)



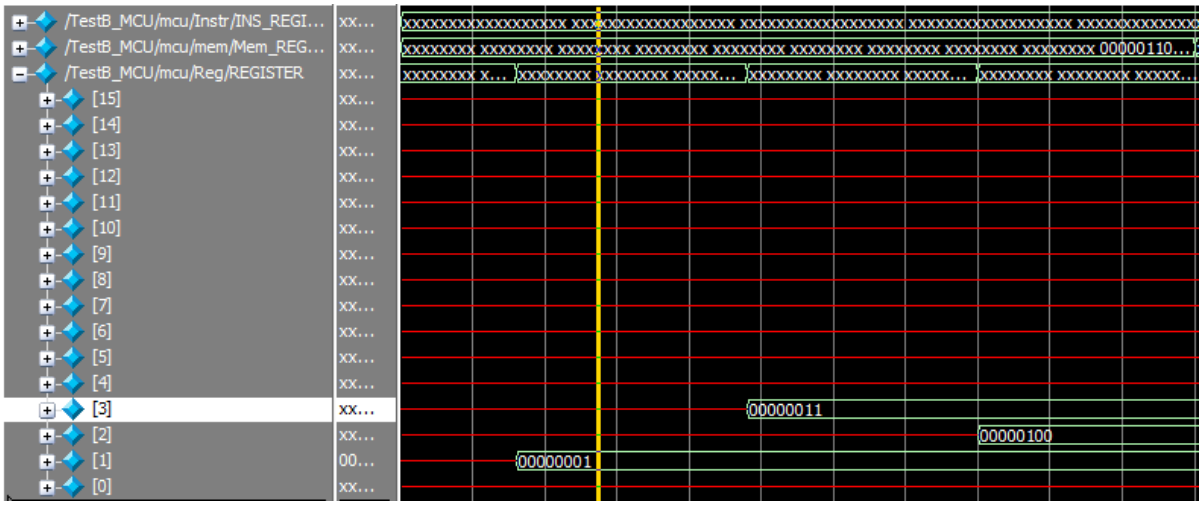
Hình 3.3.1: Wave INS

Trên thanh ghi INS_REGISTER sẽ chưa các mã hex được lưu lần lượt ở các thanh ghi từ 0 đến 3.



Hình 3.3.2: Wave MEM

Khối memory sẽ lưu trữ giá trị khởi tạo từ đầu do ta nhập vào tương ứng với các thanh ghi thứ 0 đến 6 là các data có giá trị từ 0 đến 6.



Hình 3.3.3: Wave register

Khối Register: sẽ lần lượt load giá trị data = 000001 ở m1 vào r1, data = 000011 ở m3 và r3 và tính toán ra data = 000100

KẾT LUẬN

Trong quá trình làm đê tài, chúng em đã thiết kế thành công MCU 8 bit thực hiện các phép tính đơn giản như ADD,SUB, OR, XOR, NOT,… Bằng việc lên kế hoạch thiết kế chi tiết cho từng phần như thiết kế SPEC, thiết kế sơ đồ khối, thiết kế chi tiết cho từng khối, và hiện thực hóa trên code verilog.

Qua quá trình thiết kế, chúng em đã hiểu rõ hơn về các bước thiết kế 1 MCU 8 bit và tự thiết kế , lên kế hoạch thực hiệnđó nên không tránh khỏi những sai sót, nhưng đó là những kinh nghiệm quý báu cho chúng em học tập sau này.

Để hoàn thiện đề tài này, nhóm chúng em xin cảm ơn anh Long và thầy Thắng đã giúp đỡ nhóm trong quá trình làm bài tập lớn này.

DANH MỤC TÀI LIỆU THAM KHẢO

[1] "Overview :: Natalius 8 bit RISC :: OpenCores." [Online]. Available: https://opencores.org/projects/natalius_8bit_risc. [Accessed: 21-Dec-2018].

[2] N. H. E. Weste and Da. M. Harris, *CMOS VLSI Design : A Circuit and Systems Perspective*, vol. 53, no. 9. 2011.