|  |
| --- |
| **TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**  **VIỆN ĐIỆN TỬ - VIỄN THÔNG**  Ảnh có chứa văn bản, ký hiệu, thực phẩm  Mô tả được tạo tự động  **BÁO CÁO KIẾN TRÚC MÁY TÍNH**  **Đề tài: Thiết kế bộ xử lý RISC-V Pipeline bằng ngôn ngữ Verilog**  **Nhóm sinh viên thực hiện:**  **Tên sinh viên MSSV Mã lớp**  Phạm Ngọc Lâm 20182628 124876  Dương Văn Biên 20172426 124876  Nguyễn Văn Chiến 20172431 124876  **Giảng viên hướng dẫn:** PGS - TS. Nguyễn Đức Minh  **Hà Nội - 2021** |

MỤC LỤC

[MỤC LỤC 1](#_Toc72089230)

[DANH MỤC HÌNH ẢNH i](#_Toc72089231)

[DANH MỤC BẢNG i](#_Toc72089232)

[LỜI NÓI ĐẦU ii](#_Toc72089233)

[CHƯƠNG 1. GIỚI THIỆU (INTRODUCTION) 1](#_Toc72089234)

[1.1 Giới thiệu chung 1](#_Toc72089235)

[CHƯƠNG 2. ĐẶC TẢ THÔNG SỐ KĨ THUẬT (SPECIFICATION) 2](#_Toc72089236)

[2.1 Tổng quan hệ thống 2](#_Toc72089237)

[2.1.1 Sơ đồ khối 2](#_Toc72089238)

[2.1.2 Mô tả chức năng 3](#_Toc72089239)

[2.2 Khối Controller 3](#_Toc72089240)

[2.2.1 Sơ đồ khối 3](#_Toc72089241)

[2.2.2 Mô tả chức năng khối Controller 4](#_Toc72089242)

[2.2.3 Triển khai khối controller 6](#_Toc72089243)

[2.3 Khối Datapath 7](#_Toc72089244)

[2.3.1 Khối Instruction Fetch 7](#_Toc72089245)

[2.3.2 Khối Instruction Decode 8](#_Toc72089246)

[2.3.3 Khối Execute 9](#_Toc72089247)

[2.3.4 Khối Memory Access 11](#_Toc72089248)

[2.3.5 Mô tả chức năng của khối Datapath 12](#_Toc72089249)

[CHƯƠNG 3. KIỂM THỬ (VERTIFICATION) 13](#_Toc72089250)

[3.1 Kế hoạch kiểm thử - khối Control Logic 13](#_Toc72089251)

[3.1.1 Kịch bản – Test case 13](#_Toc72089252)

[3.1.2 Kích thích đầu vào – Stimulus 13](#_Toc72089253)

[3.1.3 Kiểm tra đầu ra 13](#_Toc72089254)

[3.1.4 Mô phỏng kiểm thử thiết kế 14](#_Toc72089255)

[3.2 Kế hoạch kiểm thử - khối Datapath 14](#_Toc72089256)

[3.2.1 Kịch bản – Test case 14](#_Toc72089257)

[3.2.2 Kích thích đầu vào – Stimulus 14](#_Toc72089258)

[3.2.3 Mô phỏng kiểm thử thiết kế 15](#_Toc72089259)

[CHƯƠNG 4. KẾT LUẬN 16](#_Toc72089260)

[TÀI LIỆU THAM KHẢO 17](#_Toc72089261)

DANH MỤC HÌNH ẢNH

[Hình 2.1 Sơ đồ khối hệ thống 2](file:///D:\EDABK\Lab%20Exercise\MegaBonus\Datapath\BÁO-CÁO-KTMT-20202%20(1).docx#_Toc72089190)

[Hình 2.2 Sơ đồ khối Controller 3](file:///D:\EDABK\Lab%20Exercise\MegaBonus\Datapath\BÁO-CÁO-KTMT-20202%20(1).docx#_Toc72089191)

[Hình 2.3 Control Logic Truth Table 5](file:///D:\EDABK\Lab%20Exercise\MegaBonus\Datapath\BÁO-CÁO-KTMT-20202%20(1).docx#_Toc72089192)

[Hình 2.4 Timing của khối control logic 5](file:///D:\EDABK\Lab%20Exercise\MegaBonus\Datapath\BÁO-CÁO-KTMT-20202%20(1).docx#_Toc72089193)

[Hình 2.5 ROM-based control 6](file:///D:\EDABK\Lab%20Exercise\MegaBonus\Datapath\BÁO-CÁO-KTMT-20202%20(1).docx#_Toc72089194)

[Hình 2.6 Kiến trúc ROM-based control 6](file:///D:\EDABK\Lab%20Exercise\MegaBonus\Datapath\BÁO-CÁO-KTMT-20202%20(1).docx#_Toc72089195)

[Hình 2.7 Sơ đồ khối Instruction Fetch 7](file:///D:\EDABK\Lab%20Exercise\MegaBonus\Datapath\BÁO-CÁO-KTMT-20202%20(1).docx#_Toc72089196)

[Hình 2.8 Sơ đồ khối Instruction Decode 8](file:///D:\EDABK\Lab%20Exercise\MegaBonus\Datapath\BÁO-CÁO-KTMT-20202%20(1).docx#_Toc72089197)

[Hình 2.9 Sơ đồ khối Execute 9](#_Toc72089198)

[Hình 2.10 Sơ đồ khối Memory Access 11](#_Toc72089199)

[Hình 2.11 Timing diagram quá trình hoạt động khối Datapath 12](file:///D:\EDABK\Lab%20Exercise\MegaBonus\Datapath\BÁO-CÁO-KTMT-20202%20(1).docx#_Toc72089200)

[Hình 3.1 Đầu ra mô phỏng trên ModelSim 14](file:///D:\EDABK\Lab%20Exercise\MegaBonus\Datapath\BÁO-CÁO-KTMT-20202%20(1).docx#_Toc72089201)

[Hình 3.2 Kết quả mô phỏng kiểm thử khối top\_datapath 15](file:///D:\EDABK\Lab%20Exercise\MegaBonus\Datapath\BÁO-CÁO-KTMT-20202%20(1).docx#_Toc72089202)

DANH MỤC BẢNG

[Bảng 2‑1 Tín hiệu vào ra hệ thống 2](#_Toc72077705)

[Bảng 2‑2 Tín hiệu vào/ra khối Controller 4](#_Toc72077706)

[Bảng 2‑3 Bảng tín hiệu vào/ra khối Instruction Fetch 7](#_Toc72077707)

[Bảng 2‑4 Bảng tín hiệu vào/ra khối Instruction Decode 9](#_Toc72077708)

[Bảng 2‑5 Tín hiệu vào/ra khối Execute 10](#_Toc72077709)

[Bảng 2‑6 Tín hiệu vào/ra khối Memory Access 11](#_Toc72077710)

LỜI NÓI ĐẦU

Mã nguồn mở đang dần trở thành một phần quan trọng của thế giới IT khi mà nó góp mặt trong khoảng 96% phần mềm thương mại. Tương tự với phần cứng, vi xử lý mã nguồn mở RISC-V đang dần được quan tâm và hứa hẹn sẽ mang tới thay đổi lớn về bối cảnh của ngành điện toán. Chương trình học môn Kiến trúc máy tính (ET4041), RISC-V Processor Design là một phần quan trọng trong RISC-V 32I nhằm hiểu rõ về quá trình hoạt động khi thực hiện các lệnh. Trong báo cáo này, chúng em triển khai kiến trúc RISC-V đơn giản sử dụng ngôn ngữ mô tả phần cứng Verilog và mô phỏng kiểm thử trên phần mêm ModelSim. Chúng em sẽ đi trình bày cụ thể những gì chúng em đã làm được thông qua 4 chương sau:

**Chương 1: Giới thiệu**

**Chương 2: Đặc tả thông số kĩ thuật**

**Chương 3: Kiểm thử**

**Chương 4: Kết luận**

Nhóm chúng em xin chân thành cảm ơn PGS.TS Nguyễn Đức Minh đã tận tâm hướng dẫn chúng em trong quá trình thực hiện đồ án cũng như hoàn hiện báo cáo này !

# GIỚI THIỆU (INTRODUCTION)

*Chương này giới thiệu khái quát về Risc-V 32I.*

## Giới thiệu chung

RISC-V là một kiến trúc tập lệnh tiêu chuẩn mở (ISA) (*Instruction Set Architecture*) dựa trên nguyên tắc *Reduced Instruction Set Computer* (RISC) đã được thiết lập. RISC – V ISA được cung cấp theo *Open Source Licenses* không yêu cầu phí sử dụng.

Các tính năng đáng chú ý của RISC – V:

* Kiến trúc *load – store*
* Các mẫu bit để đơn giản hóa bộ MUX trong CPU
* Dấu phẩy động IEEE754
* Thiết kế trung lập về mặt kiến trúc và đặt *most-significant* bits tại một vị trí cố định để tăng tốc độ *sign extension*
* Tập lệnh được thiết kế cho nhiều mục đích sử dụng. Tập lệnh cơ sở có độ dài cố định gồm các lệnh 32 bit được căn chỉnh tự nhiên.

# ĐẶC TẢ THÔNG SỐ KĨ THUẬT (SPECIFICATION)

## Khối Top Module

### Sơ đồ khối

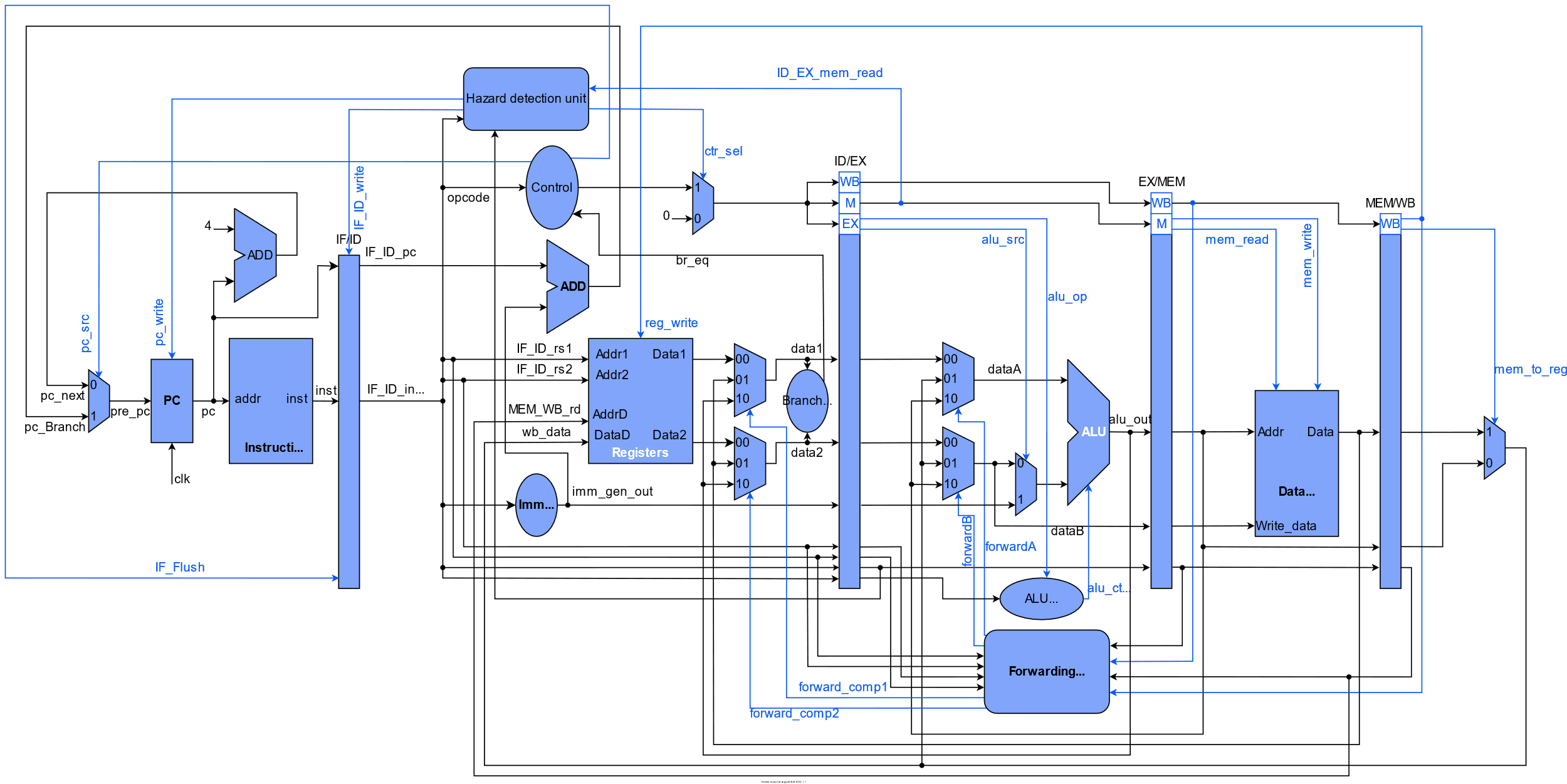


Figure . Sơ đồ khối

## Khối Intruction Fetch

Hình 2.1 Sơ đồ khối hệ thống

Dưới đây là sơ đồ khối của hệ thống bao gồm datapath và control logic trong RISC-V 32 để thực hiện một số lệnh cơ bản.

Bảng 2‑1 Tín hiệu vào ra hệ thống

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Width** | **Input/Output** | **Description** |
| **clk** | 1 | Input | Tín hiệu xung đồng hồ |
| **rst\_n** | 1 | Input | Tín hiệu reset tích cực mức thấp |
| **clk\_en** | 1 | Input | Tín hiệu enable hệ thống |

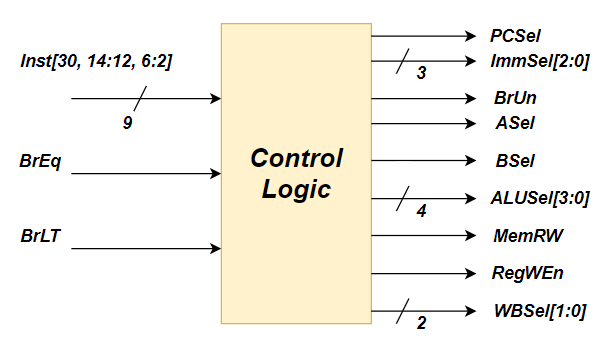
Ngoài ra, các dữ liệu trong ROM và bộ nhớ lệnh IMEM sẽ được nạp sẵn cho từng test case để thực hiện việc kiểm tra hoạt động của hệ thống.

### Mô tả chức năng

Risc-V Processor được thiết kế để thực hiện một số lệnh cơ bản như: add, sub, sw, lw, branches, JALR, JAL, … Tùy thuộc vào mỗi lệnh, mỗi kiểu dữ liệu mà việc hoạt động sẽ không giống nhau.

## Khối Controller

### Sơ đồ khối

Dưới đây là sơ đồ khối và các tín hiệu đầu vào đầu ra của khối Control Logic.

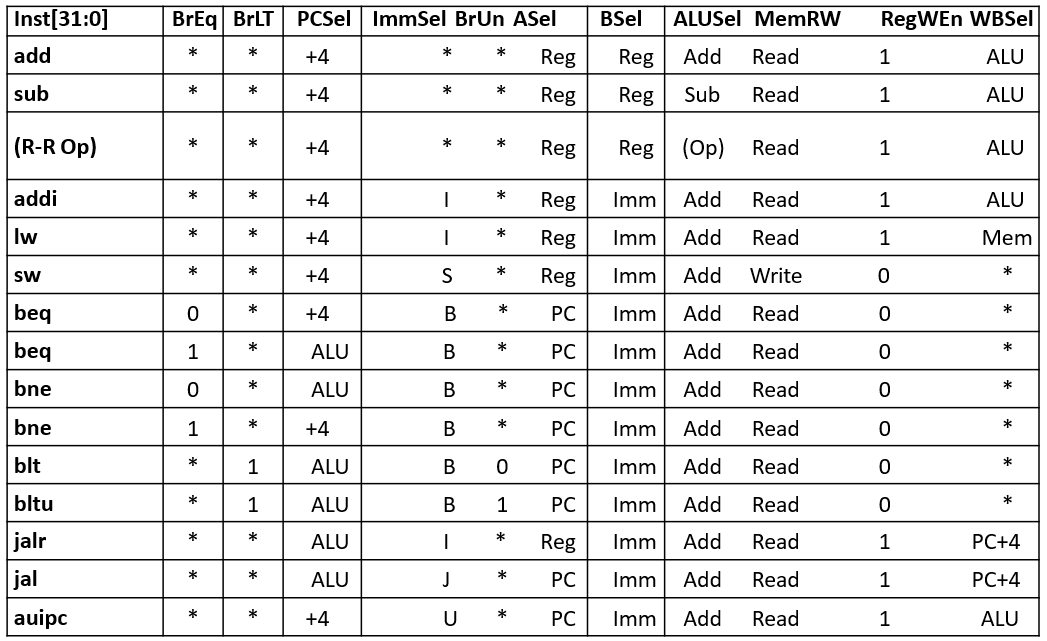
Hình 2.2 Sơ đồ khối Controller

Bảng 2‑2 Tín hiệu vào/ra khối Controller

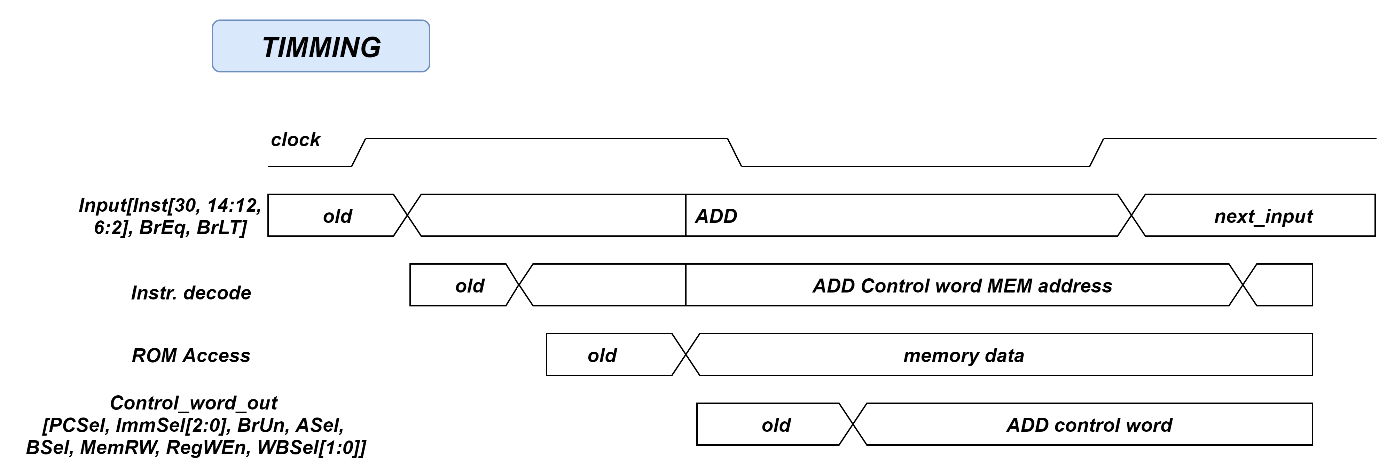
|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Width** | **Input/Output** | **Description** |
| **Inst** | 32 | Input | Instruction |
| **BrEq** | 1 | Input | Tín hiệu đầu vào xác định đầu ra khối Branch comp lệnh branch equal |
| **BrLT** | 1 | Input | Tín đầu vào xác định đầu ra khối Branch comp lệnh branch less than |
| **PCSel** | 1 | Output | Tín hiệu điều khiển (PC Select) xác định đầu vào khối PC |
| **ImmSel** | 3 | Output | Tín hiệu xác định format cho khối Immediate Generation |
| **RegWEn** | 1 | Output | Tín hiệu Enable để kích hoạt Reg[] |
| **BrUn** | 1 | Output | Tín hiệu điều khiển khối Branch comp xác định kiểu unsigned hoặc signed cho BrLT |
| **ASel** | 1 | Output | Tín hiệu điều khiển mux (lựa chọn PC+4 hoặc R[rs1]) |
| **BSel** | 1 | Output | Tín hiệu điều khiển mux (lựa chọn R[rs2] hoặc Imm) |
| **ALUSel** | 4 | Output | Tín hiệu xác định ALU select (add, sub, beq…) |
| **MemRW** | 1 | Output | Tín hiệu điều khiển chế độ đọc ghi của DMEM |
| **WBSel** | 2 | Output | Tín hiệu điều khiển Write Back Select |

### Mô tả chức năng khối Controller

Khối controller thực hiện nhiệm vụ điều khiển quá trình thực hiện lệnh trong Risc-V dựa vào các tín hiều đầu vào và đưa ra đầu ra tương ứng.

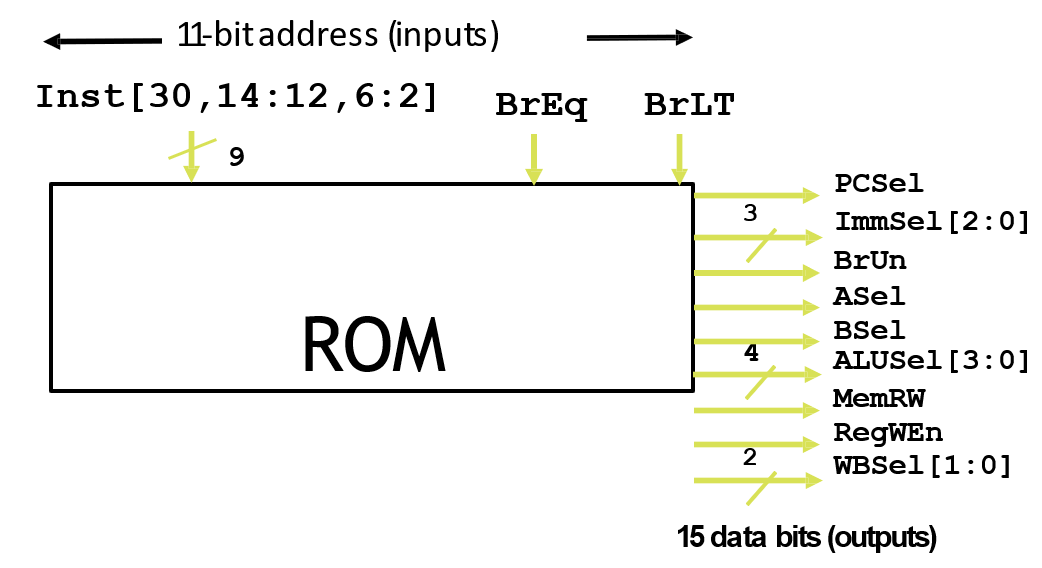
Dưới đây là bảng sự thật của khối Controller để điều khiển quá trình thực hiện lệnh.

Hình 2.3 Control Logic Truth Table

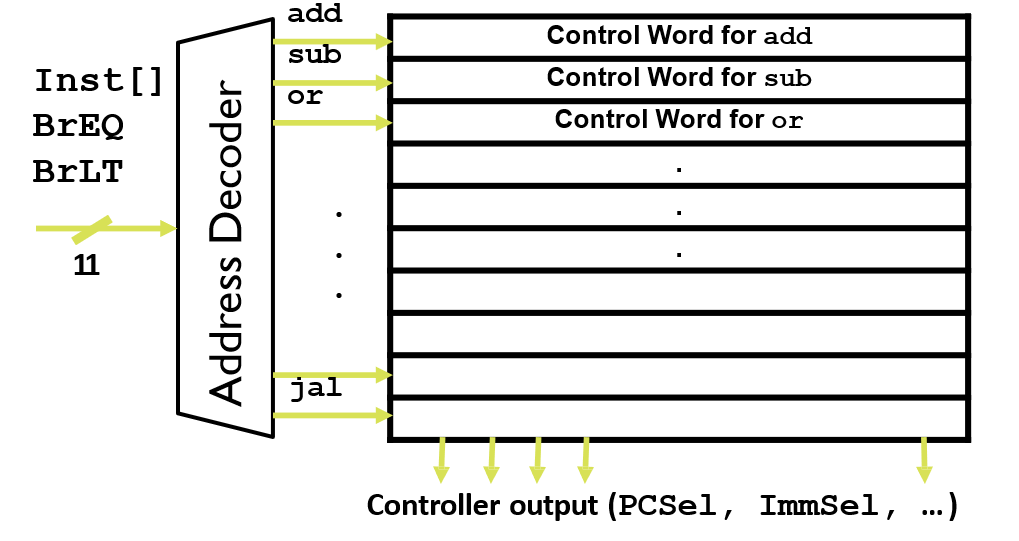
Dưới đây là hình mô tả timming tổng quát của khối Control logic, do hoạt động khá đơn giản bằng kiến trúc ROM-based nên khối hoạt động duy nhất trong một chu kì clk.

Hình 2.4 Timing của khối control logic

### Triển khai khối controller

Khối Control logic được triển khai bằng phương pháp ROM-based, các lệnh control sẽ được lưu sẵn trong 1 bộ nhớ ROM, kết hợp cùng một bộ Address Decoder nhằm đưa ra lệnh điều khiển tương ứng với từng lệnh ISA.

Hình 2.5 ROM-based control



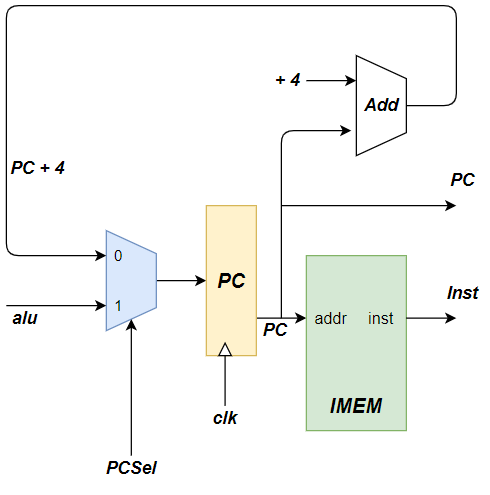
Hình 2.6 Kiến trúc ROM-based control

## Khối Datapath

Khối datapath có chức năng thực hiện các lệnh trong Risc-V, nhưng không phải tất cả các lệnh đều đi qua hết các khối trong datapath. Khối datapath bao gồm:

* Instruction Fetch (IF)
* Instruction Decode (ID)
* Execute (EX) – ALU (Arithmetic-Logic Unit)
* Memory Access
* Write Back to Register (WB)

### Khối Instruction Fetch

Dưới đây là sơ đồ khối của khối Instruction Fetch.

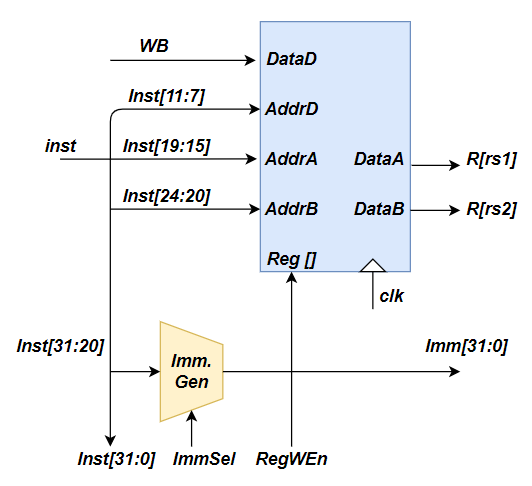
Hình 2.7 Sơ đồ khối Instruction Fetch

Dựa vào sơ đồ khối và bảng tín hiệu vào ra, khối Instruction Fetch có nhiệm vụ lựa chọn đầu vào cho khối PC dựa vào từng lệnh cụ thể, sau đó khối IMEM đưa ra mã lệnh tương ứng với địa chỉ PC.

Bảng 2‑3 Bảng tín hiệu vào/ra khối Instruction Fetch

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Width** | **Input/Output** | **Description** |
| **alu** | 32 | Input | Kết quả thực hiện lệnh trước đó của khối ALU |
| **PCSel** | 1 | Input | Tín hiệu điểu khiển để chọn đầu vào khối PC |
| **clk** | 1 | Input | Tín hiệu xung đồng hồ |
| **Inst** | 32 | Output | Instruction của lệnh cần thực hiện |
| **PC** | 32 | Output | Giá trị PC làm đầu vào cho khối IMEM |

### Khối Instruction Decode

Khối Instruction Decode là khối tiếp theo sau khối IF, nó sẽ lấy địa chỉ các thanh ghi nguồn và thanh ghi đích trong mã lệnh, sau đó đưa ra giá trị tương ứng ra **DataA** và **DataB**. Khối Immediate Generation sẽ thêm các bit sign extension và sắp xếp Immidiate từ instruction nhận được tạo ra Imm 32 bit làm đầu vào cho khối ALU.

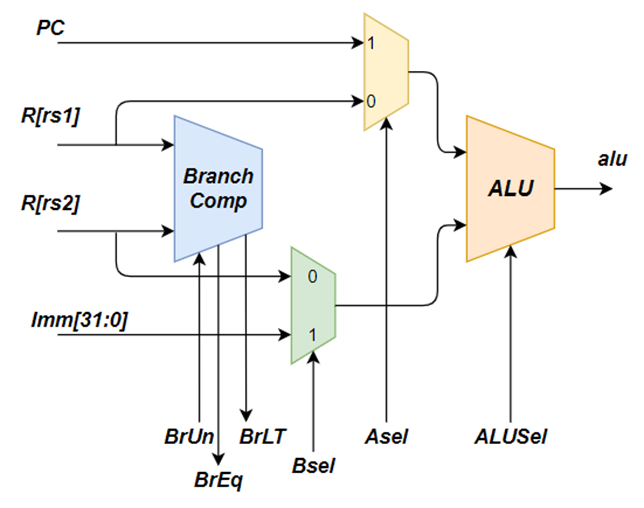
Hình 2.8 Sơ đồ khối Instruction Decode

Dựa vào sơ đồ khối bên trên, khối Instruction Fetch có hai phần chính là Register và Immediate Generation. Cụ thể ý nghĩa các đầu vào và đầu ra của khối ID được trình bày trong bảng dưới đây.

Bảng 2‑4 Bảng tín hiệu vào/ra khối Instruction Decode

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Width** | **Input/Output** | **Description** |
| **inst** | 32 | Input | Instruction 32 bit từ khối Instruction Fetch |
| **ImmSel** | 3 | Input | Immediate Select cho quá trình Immediate Generation |
| **RegWEn** | 1 | Input | Tín hiệu Write Enable cho thanh ghi đích |
| **clk** | 1 | Input | Tín hiệu đồng hồ |
| **Imm** | 31 | Output | Đầu ra 32 bit sau khi được extend |
| **DataA** | 32 | Output | Dữ liệu tương ứng với AddrA |
| **DataB** | 32 | Output | Dữ liệu tương ứng với AddrB |
| **WB** | 32 | Input | Chứa dữ liệu cần ghi vào thanh ghi đích |

### Khối Execute



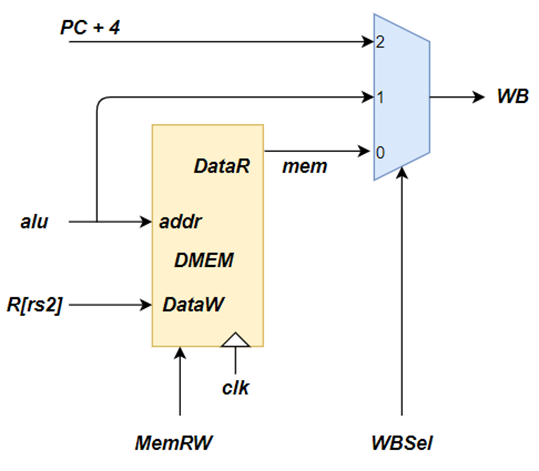
Hình 2.9 Sơ đồ khối Execute

Khối Execute bao gồm 2 module Branch Comparation và ALU. Khối Branch Comp cùng với điều kiện đầu vào **BrUn** so sánh 2 đầu vào **R[rs1]** và **R[rs2]** trả kết quả đầu ra **BrEq** và **BrLT** về khối Control Logic. A Select (**Asel**) và B Select (**Bsel**) sẽ lựa chọn đầu ra của bộ MUX dựa trên mã lệnh. ALU Select (**ALUSel**) lựa chọn hàm cho ALU thực hiện tính toán. Kết quả đầu ra có thể là dữ liệu (đối với các phép toán logic) hoặc địa chỉ (đối với Branch và Jump)

Bảng 2‑5 Tín hiệu vào/ra khối Execute

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Width** | **Input/Output** | **Description** |
| **PC** | 32 | Input | Giá trị PC hiện tại |
| **R[rs1]** | 32 | Input | Dữ liệu ứng với DataA |
| **R[rs2]** | 32 | Input | Dữ liệu ứng với DataB |
| **Imm** | 32 | Input | Giá trị Immediate 32 bits |
| **BrUn** | 1 | Input | Tín hiệu điều khiển Branch Unsigned |
| **BrEq** | 1 | Output | BrEq = 1 if R[rs1] = R[rs2] |
| **BrLT** | 1 | Output | BrLT = 1 if R[rs1] < R[rs2] |
| **ASel** | 1 | Input | Tín hiệu điều khiển bộ mux lựa chọn DataA hoặc PC + 4 |
| **BSel** | 1 | Input | Tín hiệu điều khiển bộ mux lựa chọn DataB hoặc Imm |
| **ALUSel** | 4 | Input | Tín hiệu điều khiển khối ALU |
| **alu** | 32 | Output | Đầu ra khối ALU phụ thuộc vào từng loại lệnh |

### Khối Memory Access



Hình 2.10 Sơ đồ khối Memory Access

Khối Memory Access gồm 2 module DMEM (Data Memory) và MUX 3 đầu vào. Khối DMEM lấy địa chỉ từ ALU và thực hiện lưu dữ liệu từ **R[rs2]** vào địa chỉ tương ứng khi tín hiệu **MemRW** = Write. Ngược lại đưa dữ liệu ra DataR tương ứng với địa chỉ lấy từ ALU khi **MemRW** = Read. **WBSel** điều khiển MUX đưa ra kết quả phép toán logic hay dữ liệu từ bộ nhớ hay là nhảy đến lệnh tiếp theo.

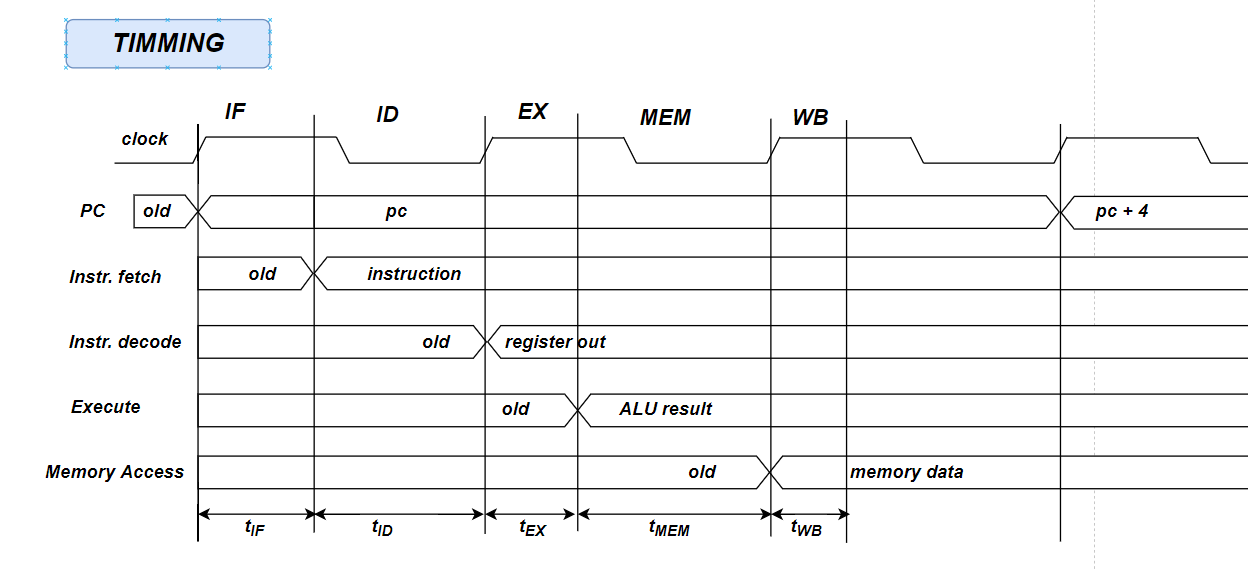
Bảng 2‑6 Tín hiệu vào/ra khối Memory Access

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Width** | **Input/Output** | **Description** |
| **PC + 4** | 32 | Input | Giá trị PC kế tiếp |
| **alu** | 32 | Input | Đầu ra của khối ALU |
| **MemRW** | 1 | Input | Tín hiệu điều khiển Read/Write Memory |
| **clk** | 1 | Input | Tín hiệu xung đồng hồ |
| **WBSel** | 2 | Output | Tín hiệu điều khiển Write Back Register Select |

### Mô tả chức năng của khối Datapath

Khối Datapath có **PC** khởi tạo mặc định bắt đầu từ 32’b0, đưa instruction vào khối Control Logic và nhận tín hiệu điều khiển hoạt động về. Hoạt động của Datapath được chia làm 5 giai đoạn:

* Instruction Fetch
* Instruction Decode
* Execute
* Memory Access
* Write Back

Timing các khối như hình dưới:

Hình 2.11 Timing diagram quá trình hoạt động khối Datapath

# KIỂM THỬ (VERIFICATION)

## Kế hoạch kiểm thử - khối Control Logic

### Kịch bản – Test case

* Kiểm tra quá trình reset, đảm bảo các đầu ra được thiết lập về đúng giá trị.
* Kiểm tra trường hợp reset bất thường khi mạch đang hoạt động
* Kiểm tra trường hợp tín hiệu reset được đặt tích cực trong nhiều chu kỳ

### Kích thích đầu vào – Stimulus

* Tạo sẵn các mã lệnh test bao gồm
  + add x8, x12, x14 🡪 0000000 01110 01100 000 01000 0110011
  + sub x10, x12, x11 🡪 0100000 01011 01100 000 01010 0110011
  + addi x15, x1, -50 🡪 111111001110 00001 000 01111 0010011
  + lw x14, 8(x2) 🡪 000000001000 00010 010 01110 0000011
  + sw x14, 8(x2) 🡪 000000001110 00010 010 01000 0100011
  + beq x19, x10, offset 🡪 0 000000 01010 10011 000 1000 0 1100011
* Sinh các đầu vào bằng cách gán giá trị trong testbench

### Kiểm tra đầu ra

Đầu ra tín hiệu điều khiển có giá trị đúng với giá trị trong Hình 2.1 Control logic Truth Table.

### Mô phỏng kiểm thử thiết kế

Hình 3.1 Đầu ra mô phỏng trên ModelSim

Đầu ra tín hiệu điều khiển của khối Control logic mô phỏng cho lệnh add đúng với giá trị trong Hình 2.1.

## Kế hoạch kiểm thử - khối Datapath

### Kịch bản – Test case

* Kiểm tra quá trình reset, đảm bảo các đầu ra được thiết lập về đúng giá trị.
* Kiểm tra trường hợp reset bất thường khi mạch đang hoạt động
* Kiểm tra trường hợp tín hiệu reset được đặt tích cực trong nhiều chu kỳ

### Kích thích đầu vào – Stimulus

* Tạo sẵn các mã lệnh test bao gồm
  + add x8, x12, x14 🡪 0000000 01110 01100 000 01000 0110011
  + sub x10, x12, x11 🡪 0100000 01011 01100 000 01010 0110011
  + addi x15, x1, -50 🡪 111111001110 00001 000 01111 0010011
  + lw x14, 8(x2) 🡪 000000001000 00010 010 01110 0000011
  + sw x14, 8(x2) 🡪 000000001110 00010 010 01000 0100011
  + beq x19, x10, offset 🡪 0 000000 01010 10011 000 1000 0 1100011
* Sinh các đầu vào bằng cách gán giá trị trong testbench

### Mô phỏng kiểm thử thiết kế

Hình 3.2 Kết quả mô phỏng kiểm thử khối top\_datapath

**Nhận xét:**

* Các tín hiệu **PC**, **inst\_tmp** lên ngay cùng với sườn dương xung **clk** (trừ **clk** đầu tiên)
* **DataA**, **DataB** nhận dữ liệu ngay sau 1 chu kì **clk**
* ALU trả kết quả và đưa ra **alu** sau khi tính toán xong
* **DataD** nhận kết quả và ghi và thanh ghi sau nửa chu kì **clk** (ở sườn âm clk tiếp theo)

# KẾT LUẬN

Báo cáo này đã trình bày kiến trúc Datapath và Control logic của một RISCV32I processor đơn giản và tiến hành triển khai mô phỏng kiểm thử trên phần mềm ModelSim. Cho ra kết quả hoạt động đúng với yêu cầu vào ra. Tuy nhiên kiến trúc được trình bày trong báo cáo mới chỉ mô phỏng và chạy được các lệnh đơn giản của kiến trúc tập lệnh RISCV gồm: add, sub, lw, sw và các phép toán logic giữa các thanh ghi với nhau. Hai khối Datapath và Control logic được triển khai độc lập, chưa có sự ghép nối hoàn thiện thành một RISCV processor. Hoạt động của kiến trúc mới chỉ dừng lạnh ở single-cycle based, tuy nhiên vẫn còn có thể tối ưu hoá hơn nữa về mặt thời gian, tăng tốc độ và hiệu quả xử lí của processor, công việc này đòi hỏi phải cần một kiến trúc phức tạp hơn mà báo cáo này chưa trình bày. Trong tương lai nhóm sẽ tiến hành hoàn thiện và triển khai kiến trúc của một RISCV32I processor hoàn chỉnh với đầy đủ các chức năng, thực hiện được đầy đủ các lệnh trong kiến trúc tập lệnh của RISCV một cách tối ưu nhất.

TÀI LIỆU THAM KHẢO

1. <https://en.wikipedia.org/wiki/RISC-V> Ngày truy cập cuối cùng: 16/5/2021
2. Slide: RISC – V 2021. PGS TS. Nguyễn Đức Minh
3. Computer Organization and Design Risc – V The Hardware Software Interface by David A. Patterson and John L. Hennessy
4. Code đính kèm: <https://github.com/lampn27022k/RISC-V---Datapath-and-Control-Logic.git>