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**

[**A.** PHÁT TRIỂN TẬP LỆNH 5](#_Toc184723865)

[1. Nhóm lệnh RRR (Register – Register – Register) 5](#_Toc184723866)

[2. Nhóm lệnh RRI (Register – Register - Immediate) 6](#_Toc184723867)

[3. Nhóm lệnh RI (Register-Immediate) 6](#_Toc184723868)

[**B.** THIẾT KẾ DATAPATH 7](#_Toc184723869)

[1. Xác định các thành phần của Datapath 7](#_Toc184723870)

[2. Chức năng cho các khối trong datapath 18](#_Toc184723871)

[**C.** THIẾT KẾ CONTROLLER 21](#_Toc184723872)

[1. Opcode + Function define 21](#_Toc184723873)

[2. Address define: 25](#_Toc184723874)

[3. Immediate define 26](#_Toc184723875)

[4. Immediate extend 26](#_Toc184723876)

[**D.** THIẾT KẾ CHƯƠNG TRÌNH 27](#_Toc184723877)

[1. Xử lí phần mềm 27](#_Toc184723878)

[a. Tập lệnh có thể biên dịch bằng phần mềm 27](#_Toc184723879)

[b. Xây dựng code asm fibonaci 27](#_Toc184723880)

[2. Lưu đồ ASM 28](#_Toc184723881)

[**E.** KIỂM THỬ 29](#_Toc184723882)

[1. Kiểm tra chức năng 29](#_Toc184723883)

[a. Nạp code 29](#_Toc184723884)

[b. Mô phỏng waveform 29](#_Toc184723885)

[c. Kiểm tra chức năng 29](#_Toc184723886)

[2. Kiểm tra tài nguyên 29](#_Toc184723887)

[3. Kiểm tra định thời 29](#_Toc184723888)

[a. Trường hợp chưa nạp code 29](#_Toc184723889)

[b. Trường hợp đã nạp code tính 12 số Fibonacci đầu tiên 30](#_Toc184723890)

[4. Kiểm tra hiệu năng 30](#_Toc184723891)

[5. Kiểm tra công suất 31](#_Toc184723892)

[a. Phân tích công suất 31](#_Toc184723893)

[b. Report công suất 31](#_Toc184723894)

[**F.** TỔNG HỢP KẾT QUẢ 32](#_Toc184723895)

[**G.** KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN CỦA ĐỀ TÀI 33](#_Toc184723896)

[1. Kết luận 33](#_Toc184723897)

[2. Hướng phát triển 33](#_Toc184723898)

**Danh sách hình**

[Hình 1. CPU 7](#_Toc184723899)

[Hình 2. Khối Datapath 7](#_Toc184723900)

[Hình 3. Khối 8 thanh ghi 16-bit với các tín hiệu điều khiển 8](#_Toc184723901)

[Hình 4. Khối 8 thanh ghi 16 bit 8](#_Toc184723902)

[Hình 5. Khối thanh ghi 16 bit 9](#_Toc184723903)

[Hình 6. Khối thanh ghi 9](#_Toc184723904)

[Hình 7. Khối ALU 9](#_Toc184723905)

[Hình 8. Khối AU 10](#_Toc184723906)

[Hình 9. Khối Full Adder 16 bit 10](#_Toc184723907)

[Hình 10. Khối Full Adder 1 bit 10](#_Toc184723908)

[Hình 11. Khối LU 10](#_Toc184723909)

[Hình 12. Khối Imem 11](#_Toc184723910)

[Hình 13. Khối Dmem 32x16 11](#_Toc184723911)

[Hình 14. Khối Dmem 16 12](#_Toc184723912)

[Hình 15. Khối Dmem 12](#_Toc184723913)

[Hình 16. Khối Data Address Decoder 13](#_Toc184723914)

[Hình 17. Khối IO controller 13](#_Toc184723915)

[Hình 18. Khối cờ zero 14](#_Toc184723916)

[Hình 19. Khối Shifter 14](#_Toc184723917)

[Hình 20. Khối PC 14](#_Toc184723918)

[Hình 21. Khối inc 1bit 14](#_Toc184723919)

[Hình 22. Khối Register file cell 16 bit 15](#_Toc184723920)

[Hình 23. Khối Register file cell 15](#_Toc184723921)

[Hình 24. Khối nhân 8 bit 16](#_Toc184723922)

[Hình 25. Khối 16 bit chia 16 bit 16](#_Toc184723923)

[Hình 26. Khối 1 bit chia 16 bit 16](#_Toc184723924)

[Hình 27. Khối 1 bit chia 1 bit 17](#_Toc184723925)

[Hình 28. Khối Controller 21](#_Toc184723926)

[Hình 29. Khối Opcode & Function 22](#_Toc184723927)

[Hình 30. Khối Address define 25](#_Toc184723928)

[Hình 31. Phần chọn Imm 26](#_Toc184723929)

[Hình 32. Phần mở rộng bit 26](#_Toc184723930)

[Hình 33. Lưu đồ ASM 28](#_Toc184723931)

[Hình 34. Mô phỏng waveform chương trình 29](#_Toc184723932)

[Hình 35. Kết quả của chạy kiểm tra tài nguyền 29](#_Toc184723933)

[Hình 36. Kiểm tra định thời chưa nạp code 30](#_Toc184723934)

[Hình 37. Fmax 30](#_Toc184723935)

[Hình 38. Kiểm tra định thời đã nạp code 30](#_Toc184723936)

[Hình 39. Chạy mô phỏng waveform với Fmax 30](#_Toc184723937)

[Hình 40. Kết quả phân tích công suất 31](#_Toc184723938)

[Hình 41. Thực hiện trên kit ALTERA DE2 32](#_Toc184723939)

# PHÁT TRIỂN TẬP LỆNH

## 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] |

## 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 |

## 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 |  |

A diagram of a computer

Description automatically generated

1. CPU

# THIẾT KẾ DATAPATH

## Xác định các thành phần của Datapath

A white background with purple lines

Description automatically generated with medium confidence

1. Khối Datapath

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

A diagram of a computer component

Description automatically generated

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

A diagram of a computer circuit

Description automatically generated

1. Khối 8 thanh ghi 16 bit

A purple line with black and white lines

Description automatically generated

1. Khối thanh ghi 16 bit

A diagram of a circuit

Description automatically generated

1. Khối thanh ghi

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

A diagram of a computer circuit

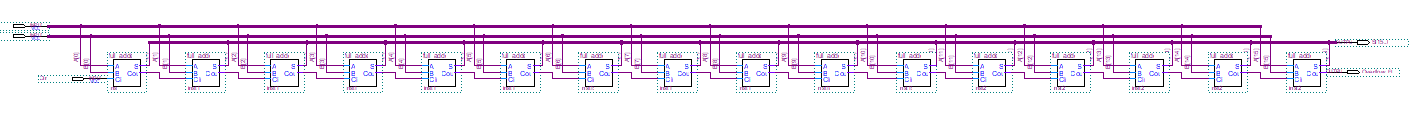
Description automatically generated

1. Khối ALU

A computer screen with a diagram

Description automatically generated with medium confidence

1. Khối AU



1. Khối Full Adder 16 bit

A diagram of a circuit

Description automatically generated

1. Khối Full Adder 1 bit

A diagram of a circuit

Description automatically generated

1. 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:**

A close-up of a data

Description automatically generated

1. Khối Imem

**32 x 16 Dmem:**

A computer screen shot of a computer scheme

Description automatically generated

1. Khối Dmem 32x16

A diagram of a computer

Description automatically generated

1. Khối Dmem 16

A diagram of a computer

Description automatically generated

1. Khối Dmem

**Data Address Decoder (DAD):**

A diagram of a circuit

Description automatically generated

1. Khối Data Address Decoder

**I/O controller:**

A diagram of a computer circuit

Description automatically generated

1. Khối IO controller

**Zero Flag:**

A diagram of a circuit

Description automatically generated

1. Khối cờ zero

**Shifter:**

A computer screen shot of a computer code

Description automatically generated with medium confidence

1. Khối Shifter

**Program counter (PC):**

A diagram of a computer

Description automatically generated

1. Khối PC

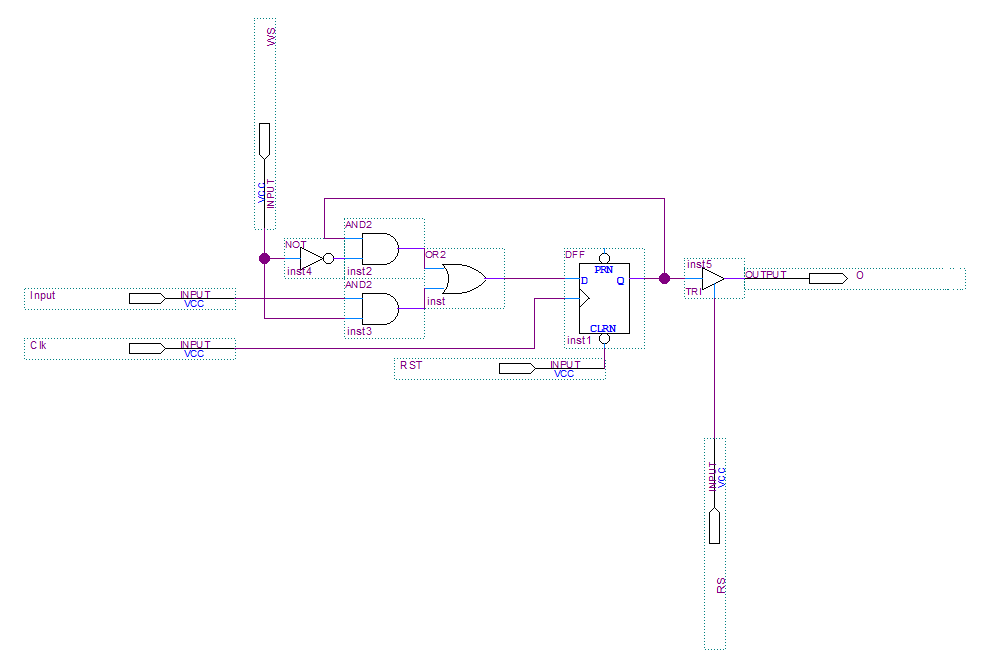


1. Khối inc 1bit

A group of blue and purple lines

Description automatically generated with medium confidence

1. Khối Register file cell 16 bit



1. Khối Register file cell

**Bộ nhân 8 bit:**

A diagram of a computer network

Description automatically generated

1. Khối nhân 8 bit

**Bộ chia 16 bit:**

A blueprint of a guitar

Description automatically generated

1. Khối 16 bit chia 16 bit

A diagram of a circuit

Description automatically generated with medium confidence

1. Khối 1 bit chia 16 bit

A diagram of a circuit

Description automatically generated

1. Khối 1 bit chia 1 bit

## 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 thục 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.

# 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

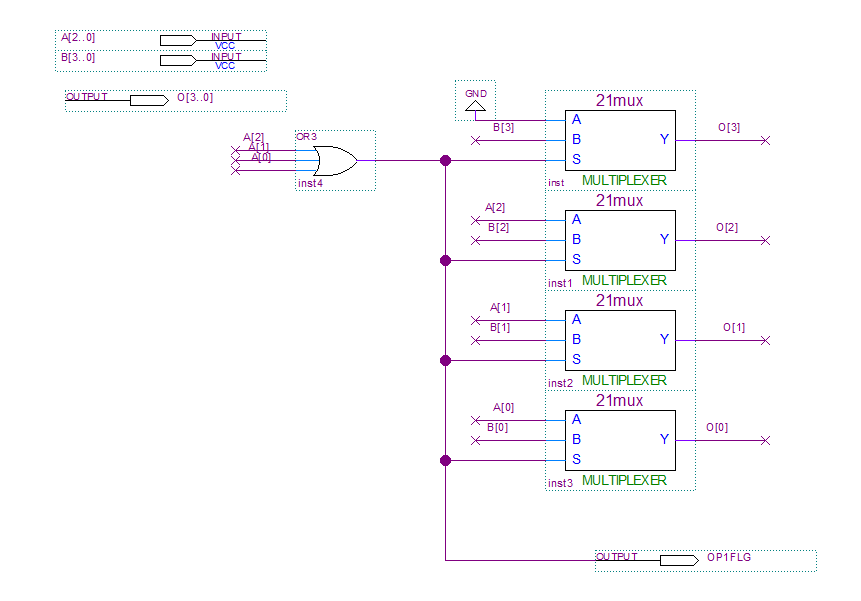
A diagram of a computer

Description automatically generated

1. Khối Controller

## 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.



A diagram of a circuit

Description automatically generated

1. 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

## Address define:

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

A diagram of a computer circuit

Description automatically generated

1. 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

## Immediate define

Chọn imm của tập lệnh RRI hay RI.

A diagram of a computer

Description automatically generated

1. Phần chọn Imm

## Immediate extend

Mở rộng imm từ 10 thành 16 bit.

A line with a line and a line with a line and a line with a line and a line with a line and a line with a line and a line with a line and a line with

Description automatically generated

1. Phần mở rộng bit

# THIẾT KẾ CHƯƠNG TRÌNH

## Xử lí phần mềm

### 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

### 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

## Lưu đồ ASM

A diagram of a algorithm

Description automatically generated

1. Lưu đồ ASM

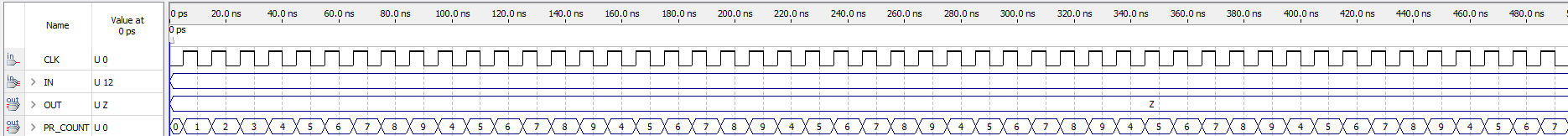
# KIỂM THỬ

## Kiểm tra chức năng

### 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.

### Mô phỏng waveform

A screenshot of a graph

Description automatically generated

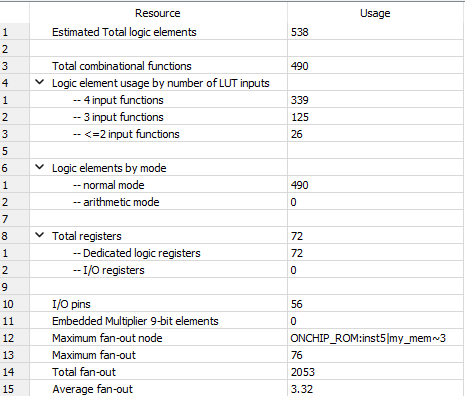
1. Mô phỏng waveform chương trình

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

### 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.

## Kiểm tra tài nguyên



1. Kết quả của chạy kiểm tra tài nguyền

## Kiểm tra định thời

### Trường hợp chưa nạp code



A screen shot of a computer

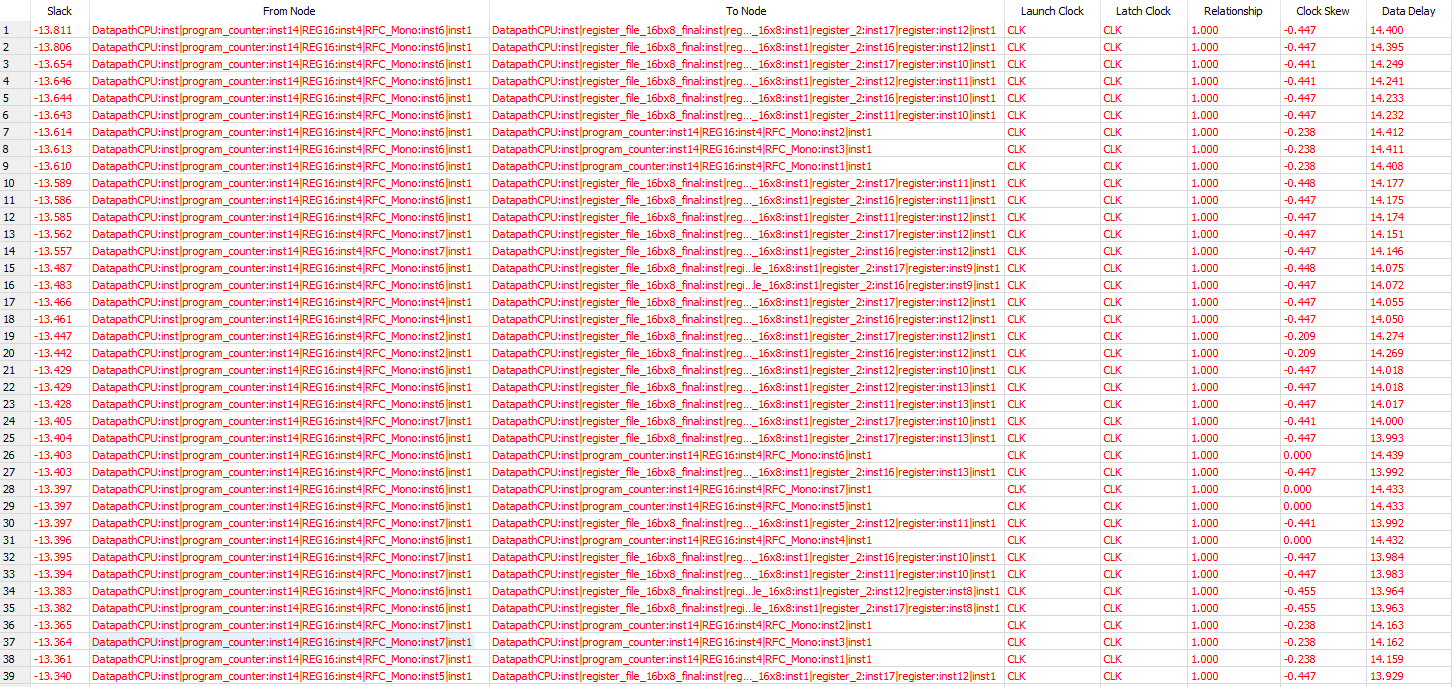
Description automatically generated

1. Kiểm tra định thời chưa nạp code

### Trường hợp đã nạp code tính 12 số Fibonacci đầu tiên



1. Fmax



1. Kiểm tra định thời đã nạp code

## Kiểm tra hiệu năng

A screenshot of a computer

Description automatically generated

1. Chạy mô phỏng waveform với Fmax

Thời gian thành chương trình là 1.058915 µs

## Kiểm tra công suất

### Phân tích công suất



1. Kết quả phân tích công suất

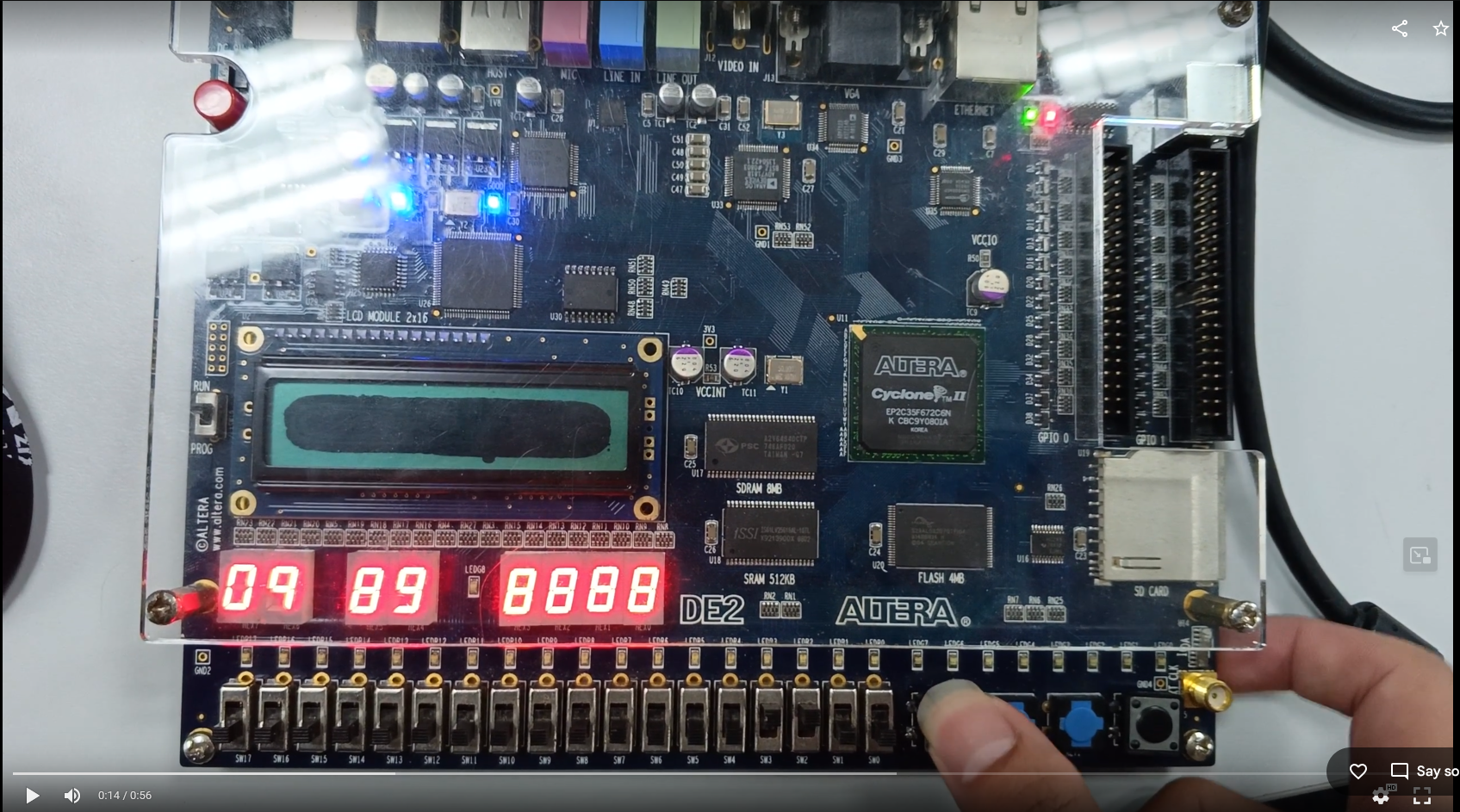
### Report công suất

Công suất tổng cộng là 115.56 mW

# 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



1. Thực hiện trên kit ALTERA DE2

[Link video demo](https://uithcm-my.sharepoint.com/:v:/g/personal/22521621_ms_uit_edu_vn/EWyp6fjDtl9Bk15G5ionnMIBlByK6Bmstxf367YTECBUeA?e=nqYfgg)

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN CỦA ĐỀ TÀI

## 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.

## 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.