Thiết kế RISC CPU đơn giản

GIỚI THIỆU

1. MỤC TIÊU

- Rèn luyện, nâng cao khả năng sử dụng ngôn ngữ đặc tả phần cứng Verilog HDL để thiết kế mạch logic số.
- Rèn luyện kỹ năng phân tích, thiết kế mạch số theo hướng tiếp cận mô hình phân cấp: phân chia chức năng, module,... Nâng cao tính sáng tạo trong thiết kế các sản phẩm ứng dụng hệ thống số.
- Rèn luyện kỹ năng nghiên cứu, tự học, có thêm hiểu biết về các lĩnh vực liên quan: các kỹ thuật trong các mạch điện tử số thông dụng, các lĩnh vực ứng dụng hệ thống số,...

2. YÊU CẦU

- Thực hiện theo nhóm (tối đa 5 thành viên).
- Báo cáo: yêu cầu sử dụng Latex để viết báo cáo. Nộp trực tiếp khi demo với GVHD. Yêu cầu báo cáo cần có các nội dung sau đây:
  - **Phần 1**: Giới thiệu. Bao gồm các nội dung về giới thiệu đề tài, công cụ sử dụng, thiết bị sử dụng, các chức năng của sản phẩm.
  - **Phần 2**: Lý thuyết sơ lược về sản phẩm, những tính năng được ứng dụng cho sản phẩm trong thực tế.
  - **Phần 3**: Thiết kế. Bao gồm sơ đồ khối, các khối chức năng, trình bày chức năng của các khối.
  - **Phần 4**: Hiện thực. Bao gồm cách thức hiện thực thiết kế (Sơ đồ khối, cấu trúc mạch, mô hình thiết kế, v.v.). Có thể vẽ các flow-chart thể hiện luồng xử lý của hệ thống.
  - **Phần 5**: Kiểm thử và đánh giá. Bao gồm kết quả mô phỏng các trường hợp kiểm thử. Đánh giá thông số hiệu năng về tài nguyên, thời gian, năng lượng, ...
  - **Phần 6**: Kết luận. Bao gồm kết luận của nhóm, hướng phát triển trong tương lai, những khó khăn gặp phải và bảng phân chia công việc.
- Đối với các yêu cầu cụ thể trong các đề tài, sinh viên có thể đề xuất thay đổi để nâng cao tính dễ sử dụng, dễ mở rộng tính năng, phù hợp với thiết kế của nhóm, ... Sinh viên trao đổi với GVHD nếu có thay đổi.

3. TIÊU CHÍ ĐÁNH GIÁ

Về sản phẩm:

- Chức năng cơ bản theo yêu cầu.
- Giao diện người dùng đơn giản, tính dễ sử dụng,... (UX – User Experience).
- Chức năng nâng cao, đề xuất thêm của nhóm.

Về kỹ thuật:

- Phân chia khối chức năng hợp lý
- Mã nguồn tốt (không vi phạm đặc tính phần cứng, coding style, ...)

4. TÀI LIỆU THAM KHẢO

- Slide bài giảng.
- Bài tập thực hành, các tài liệu hướng dẫn.
- Internet

ĐỀ TÀI

Danh sách đề tài

1. Thiết kế RISC CPU đơn giản

1.1. Giới thiệu

RISC (Reduced Instructions Set Computer) là một phương pháp thiết kế bộ xử lý hiện nay. Trong đề tài này, chúng ta sẽ thiết kế RISC CPU đơn giản với 3-bit opcode và và 5-bit toán hạng. Tức là có 8 loại câu lệnh và 32 không gian địa chỉ.
Bộ xử lý hoạt động dựa trên tín hiệu clock và reset. Chương trình sẽ dừng lại khi có tín hiệu HALT.

1.2. Yêu cầu

Hãy sử dụng các kiến thức về Thiết kế vi mạch để thiết kế RISC CPU đơn giản sử dụng các công cụ của Cadence.

Về thiết kế hệ thống cần có những khối chức năng như sau:

- Program Counter: chức năng lưu trữ thanh ghi địa chỉ của chương trình (program address)
- Address Mux: nhằm lựa chọn giữa địa chỉ chương trình hoặc địa chỉ của câu lệnh (instruction)
- Memory: lưu trữ và cung cấp dữ liệu cho chương trình
- Instruction Register: xử lý dữ liệu instruction
- Accumulator Register: xử lý dữ liệu từ ALU
- ALU: xử lý dữ liệu từ Memory, Accumulator và opcode của Instruction,
- Mỗi khối chức năng đều cần có testbench tương ứng.

Về chức năng hệ thống:

- Nạp lệnh từ Memory
- Giải mã lệnh
- Lấy dữ liệu toán hạng từ Memory nếu cần
- Thực thi câu lệnh, xử lý các phép toán nếu cần
- Lưu trữ kết quả trở lại Memory hoặc Accumulator. Quá trình này sẽ lặp lại cho tới khi có câu lệnh kết thúc chương trình

1.3. Nội dung chi tiết

Program Counter

- Counter là bộ đếm quan trọng dùng để đếm câu lệnh của chương trình. Ngoài ra còn có thể dùng đếm các trạng thái của chương trình.
- Counter phải hoạt động khi có xung lên của clk.
- Reset kích hoạt mức cao, bộ đếm trở về 0.
- Counter với độ rộng số đếm là 5.
- Counter có chức năng load một số bất kì vào bộ đếm. Nếu không, bộ đếm sẽ hoạt động bình thường.

Address Mux

- Khối Address Mux với chức năng của Mux sẽ chọn giữa địa chỉ lệnh trong giai đoạn nạp lệnh và địa chỉa toán hạng trong giai đoạn thực thi lệnh.
- Mux sẽ có độ rộng mặc định là 5.
- Độ rộng cần sử dụng parameter để vẫn thay đổi được nếu cần

ALU

- ALU thực thi những phép toán số học. Phép tính được thực thi sẽ phụ thuộc vào toán tử của câu lệnh.
- ALU thực thi 8 phép toán trên số hạng 8-bit (inA và inB). Kết quả sẽ cho ra 8-bit output và 1-bit is_zero
- is_zero bất đồng bộ nhằm cho biết input inA có bằng 0 hay không.
- Đầu vào opcode 3-bit sẽ quyết định phép toán nào được sử dụng như mô tả trong bảng sau:

Opcode	**Mã**	**Hoạt động**	Output
HLT	000	Dừng hoạt động chương trình	inA
SKZ	001	Trước tiên sẽ kiểm tra kết quả của ALU có bằng 0 hay không, nếu bằng 0 thì sẽ bỏ qua câu lệnh tiếp theo, ngược lại sẽ tiếp tục thực thi như bình thường	inA
ADD	010	Cộng giá trị trong Accumulator vào giá trị bộ nhớ địa chỉ trong câu lệnh và kết quả được trả về Accumulator.	inA+inB
AND	011	Thực hiện AND giá trị trong Accumulator và giá trị bộ nhớ địa chỉ trong câu lệnh và kết quả được trả về Accumulator.	inA and inB
XOR	100	Thực hiện XOR giá trị trong Accumulator và giá trị bộ nhớ địa chỉ trong câu lệnh và kết quả được trả về Accumulator.	inA or inB
LDA	101	Thực hiện đọc giá trị từ địa chỉ trong câu lệnh và đưa vào Accumulator.	inB
STO	110	Thực hiện ghi dữ liệu của Accumulator vào địa chỉ trong câu lệnh.	inA
JMP	111	Lệnh nhảy không điều kiện, nhảy đến địa chỉ đích trong câu lệnh và tiếp tục thực hiện chương trình	inA

Controller

- Controller quản lý những tín hiệu điều khiển của CPU. Bao gồm nạp và thực thi lệnh.
- Controller phải hoạt động khi có xung lên của clk.
- Tín hiệu rst đồng bộ và kích hoạt mức cao.
- Tín hiệu đầu vào opcode 3-bit tương ứng với ALU
- Controller có 7 output như bảng sau:

Output	**Function**
sel	select
rd	memory read
ld_ir	load instruction register
halt	halt
inc_pc	increment program counter
ld_ac	load accumulator
ld_pc	load program counter
wr	memory write
data_e	data enable

- Controller có 8 trạng thái hoạt động liên tục trong 8 chu kỳ clk theo thứ tự: $INST\_ADDR, INST\_FETCH, INST\_LOAD, IDLE, OP\_ADDR, OP\_FETCH, ALU\_OP, STORE$. Trạng thái reset là $INST\_ADDR$.
- Output của Controller dựa theo trạng thái và opcode như bảng sau:

Outputs	**Phase**								**Notes**
	INST_ ADDR	INST_ FETCH	INST_ LOAD	IDLE	OP_ ADDR	OP_ FETCH	ALU_ OP	STORE	
sel	1	1	1	1	0	0	0	0	ALU OP = 1 if opcode is ADD, AND, XOR or LDA
rd	0	1	1	1	0	ALUOP	ALUOP	ALUOP	
ld_ir	0	0	1	1	0	0	0	0	
halt	0	0	0	0	HALT	0	0	0	
inc_pc	0	0	0	0	1	0	SKZ & & zero	0	
ld_ac	0	0	0	0	0	0	0	ALUOP	
ld_pc	0	0	0	0	0	0	JMP	JMP	
wr	0	0	0	0	0	0	0	STO	
data_e	0	0	0	0	0	0	STO	STO	

Register

- Tín hiệu đầu vào có độ rộng 8 bits.
- Tín hiệu rst đồng bộ và kích hoạt mức cao.
- Register phải hoạt động khi có xung lên của clk.
- Khi có tín hiệu load, giá trị đầu vào sẽ chuyển đến đầu ra.
- Ngược lại giá trị đầu ra sẽ không đổi.

Memory

- Memory sẽ lưu trữ instruction và data.
- Memory cần được thiết kế tách riêng chức năng đọc/ghi bằng cách sử dụng Single bidirectional data port. Không được đọc và ghi cùng lúc.
- 5-bit địa chỉ và 8-bit data.
- 1-bit tín hiệu cho phép đọc/ghi
- Memory phải hoạt động khi có xung lên của clk.

1.4. Nội dung làm thêm gợi ý

- Vận dụng kiến thức môn học Kiến trúc máy tính để hoàn thiện thiết kế, xử lý Hazard
- Hoàn thiện chức năng cho khối ALU (fix/floating point, multiplier, divider, ...)
- Sinh viên có thể đề xuất các nội dung khác.

Tham khảo: