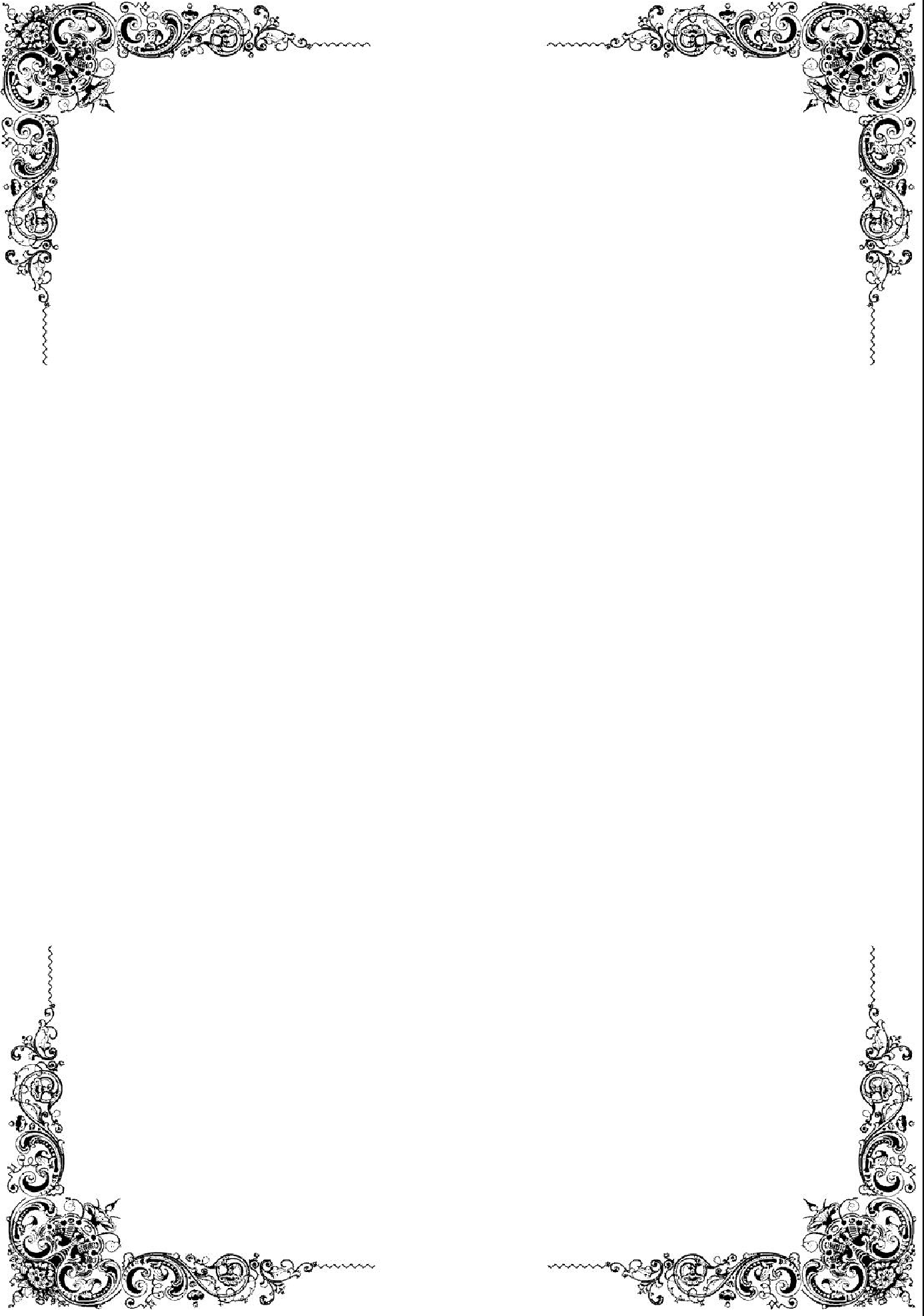
**BỘ GIÁO DỤC VÀ ĐÀO TẠO**



**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HỒ CHÍ MINH**

**KHOA ĐIỆN - ĐIỆN TỬ**

**\*\*\*\*\*\***

**ĐỒ ÁN MÔN HỌC 2**

**DESIGN AND SIMULATION I2C CONTROLLER BY USING VERILOG**

**SINH VIÊN THỰC HIỆN:**

ĐINH HOÀNG LONG -18119093

NGUYỄN MINH TÙNG -18119131

**NGÀNH:** CÔNG NGHỆ KỸ THUẬT MÁY TÍNH

**KHÓA:** 2018

**Giáo viên hướng dẫn:** THS. HUỲNH HOÀNG HÀ

Tp. Hồ Chí Minh, tháng năm 2021

|  |  |
| --- | --- |
| LOGO CLC.JPG | CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM  Độc lập – Tự do – Hạnh phúc  ----\*\*\*---- |

Tp. Hồ Chí Minh, ngày 8 tháng 7 năm 2021

**NHIỆM VỤ ĐỒ ÁN MÔN HỌC**

|  |  |
| --- | --- |
| Họ và tên sinh viên: Đinh Hoàng Long  Nguyễn Minh Tùng | MSSV: 18119093  MSSV: 18119131 |
| Ngành: Công nghệ kỹ thuật máy tính | Lớp: 18119CL3B |
| Giảng viên hướng dẫn: Huỳnh Hoàng Hà | ĐT: -- |
| Ngày nhận đề tài: 30/09/2021 | Ngày nộp đề tài: -- |
| 1. Tên đề tài: Design and simulation I2C Controller by using verilog |  |
| 2. Các số liệu, tài liệu ban đầu: Các kiến thức liên quan đến môn: kỹ thuật số, thiết kế hệ thống và vi mạch tích hợp, truyền số liệu, kiến trúc và tổ chức máy tính. |  |
| 1. Nội dung thực hiện đề tài:  * Tóm tắt lý thuyết có liên quan và đặc tính của I2C controller * Hoàn thiện sơ đồ mạch nguyên lý của hệ thống * Hoàn thiện bản thiết kế và mô tả hành vi từng khối * Hoàn thiện RTL code * Hoàn thiện testcase và test từng khối * Test toàn bộ hệ thống * Viết báo cáo và trình bày kết quả |  |
| 1. Sản phẩm: I2C CONTROLLER |  |

|  |  |
| --- | --- |
| Logo CLC | CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM  Độc lập – Tự do – Hạnh Phúc  \*\*\*\*\*\*\* |

# **PHIẾU NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN**

Họ và tên Sinh viên: Đinh Hoàng Long MSSV: 18119093

Họ và tên Sinh viên: Nguyễn Minh Tùng MSSV: 18119131

Ngành: Công nghệ kỹ thuật Máy Tính

Tên đề tài: Design and simulation I2C Controller by using verilog

Họ và tên Giáo viên hướng dẫn: Huỳnh Hoàng Hà

**NHẬN XÉT**

1. Về nội dung đề tài & khối lượng thực hiện:

1. Ưu điểm:

1. Khuyết điểm:

1. Đề nghị cho bảo vệ hay không?

1. Đánh giá loại:

1. Điểm:……………….(Bằng chữ: )

Tp*. Hồ Chí Minh, ngày tháng năm 20…*

Giáo viên hướng dẫn

*(Ký & ghi rõ họ tên)*

|  |  |
| --- | --- |
| Logo CLC | CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM  Độc lập – Tự do – Hạnh phúc  ----\*\*\*---- |

**PHIẾU NHẬN XÉT CỦA GIÁO VIÊN PHẢN BIỆN**

Họ và tên Sinh viên: Đinh Hoàng Long MSSV: 18119093

Họ và tên Sinh viên: Nguyễn Minh Tùng MSSV: 18119131

Ngành: Công nghệ kỹ thuật Máy tính

Tên đề tài: Design and simulation I2C Controller by using verilog

Họ và tên Giáo viên phản biện:

**NHẬN XÉT**

1. Về nội dung đề tài & khối lượng thực hiện:

1. Ưu điểm:

1. Khuyết điểm:

1. Đề nghị cho bảo vệ hay không?

1. Đánh giá loại:

1. Điểm:……………….(Bằng chữ: )

Tp*. Hồ Chí Minh, ngày tháng năm 2021*

Giáo viên phản biện

*(Ký & ghi rõ họ tên)*

**LỜI CẢM ƠN**

Sau một thời gian nghiên cứu thực hiện đề tài, nhóm em đã có hoàn thành được đề tài “I2C Controller”, để có thể hoàn thành đề tài trên ngoài sự cố gắng của tình thành viên trong nhóm thì nhóm em xin chân thành cảm ơn đến thầy: Huỳnh Hoành Hà là người trực tiếp hướng dẫn và góp ý trong suốt quá trình chúng em thực hiện đề tài. Cảm ơn thầy vì đã giành thời gian quý báu của mình để hướng dẫn nhóm, và đưa ra những giải pháp, những ý kiến mỗi khi nhóm gặp khó khăn. Bên cạnh đó chúng em xin cảm mơn đến tất cả các thầy cô tại trường Đại học sư phạm kỹ thuật vì những kiến thức mà thầy cô đã truyền đạt trong những năm qua để từ đó chúng em có cơ sở để nghiên cứu và xây dựng đề tài.

Nhóm em cũng chân thành gửi lời cảm ơn chân thành đến những bạn sinh viên ngành công nghệ kỹ thuật máy tính đã cùng chúng em sát cánh trong những năm học qua, họ cũng là người tạo động lực để chúng em có thể hoàn thành tốt đề tài. Xin chân trọng cảm mơn!

**TÓM TẮT**

Ngày nay con người đang bước chân trên nền công nghiệp hiện đại hóa 4.0 cùng với nhu cầu sử dụng các thiết bị tiện lợi ngày càng nhiều, kéo theo sự tiến bộ không ngừng của khoa học và công nghệ, các thiết bị ngoại vi, đặc biệt là các chuẩn giao tiếp đang không ngừng cải tiến và phát triển đa dạng. Vậy làm thế nào để chúng ta có thể tích hợp các chuẩn giao tiếp trên chính các thiết bị ngoại vi của mình? Trên cơ sở đó chúng em sẽ chọn cách thiết kế I2C Controller để hiểu rõ hơn hoạt động của chuẩn giao tiếp I2C, một chuẩn giao tiếp được ứng dụng rộng rãi hiện nay, đồng thời đó cũng là chìa khóa quan trọng cho chúng em biết cách thức giao tiếp của các thiết bị ngoại vi.

Trong nội dung đồ án này, lõi I2C Controller được chúng em thiết kế theo quy trình thiết kế vi mạch số của các công ty vi mạch hiện nay. RTL code được chúng em sữa lỗi dựa trên trình mô phỏng của phần mềm ISE Design suite.

**CÁC TỪ VIẾT TẮT**

|  |  |  |
| --- | --- | --- |
| **STT** | **Các từ viết tắt** | **Nghĩa đầy đủ** |
| 1 | I2C | Intel – Integrated Circuit |
| 2 | IC | Integrated Circuit |
| 3 | RTL | Register Transfer Level |
| 4 | UART | Universal Asynchronous Receiver / Transmitter |
| 5 | SPI | Serial Peripheral Interface |
| 6 | IP | intellectual property |
| 7 | SCL | Serial clock line |
| 8 | SDA | Serial data line |
| 9 | ACK | Acknowledge |
| 10 | NACK | No-acknowlegde |
| 11 | R/ | Read/write |
| 12 | FSM | Finite state machine |

**MỤC LỤC**

[CHƯƠNG 1: TỔNG QUAN 9](#_Toc89337938)

[1.1 Tình hình nghiên cứu trong và ngoài nước 9](#_Toc89337939)

[1.1.1 Đặt vấn đề 9](#_Toc89337940)

[1.1.2 Giới thiệu 10](#_Toc89337941)

[1.2 Mục tiêu đề tài. 10](#_Toc89337942)

[1.3 Giới hạn đề tài. 10](#_Toc89337943)

[1.4 Nội dung nghiên cứu. 10](#_Toc89337944)

[1.5 Bố cục. 11](#_Toc89337945)

[CHƯƠNG 2: CƠ SỞ LÝ THUYẾT 11](#_Toc89337946)

[2.1 Tổng quan về qui trình thiết kế vi mạch số [1],[4],[5] 11](#_Toc89337947)

[2.1.1 Quy trình thiết kế vi mạch số 11](#_Toc89337948)

[2.1.1.1 Specification 12](#_Toc89337949)

[2.2.1.2 Verification 13](#_Toc89337950)

[2.2.1.3 Physical design: 14](#_Toc89337951)

[2.1.2 Công cụ sử dụng trong thiết kế vi mạch số 16](#_Toc89337952)

[2.1.2.1 Specification 16](#_Toc89337953)

[2.1.2.2 Verification 16](#_Toc89337954)

[2.1.2.3 Physical Design: 16](#_Toc89337955)

[2.2 Giao thức I2C [2],[6],[8] 16](#_Toc89337956)

[2.2.1 Đặc điểm của giao thức I2C 16](#_Toc89337957)

[2.2.2 Chức năng của I2C controller 17](#_Toc89337958)

[2.2.3 Sơ đồ khối: 17](#_Toc89337959)

[2.2.4 Hoạt động của I2C 18](#_Toc89337960)

[2.2.4.1 Định dạng khung dữ liệu 19](#_Toc89337961)

[2.2.4.2 Điều kiện START và STOP 19](#_Toc89337962)

[2.2.4.3 Địa chỉ thiết bị 20](#_Toc89337963)

[2.2.4.4 Chế độ truyền, nhận Master-Slave 20](#_Toc89337964)

[2.3 Máy trạng thái (FSM) [3] 21](#_Toc89337965)

[2.3.1 Mô hình cơ bản của máy trạng thái 21](#_Toc89337966)

[2.3.2 Phân loại 22](#_Toc89337967)

[CHƯƠNG 3: THIẾT KẾ, TÍNH TOÁN VÀ XÂY DỰNG HỆ THỐNG HỆ THỐNG 24](#_Toc89337968)

[3.1 Yêu cầu hệ thống 24](#_Toc89337969)

[3.1.1 Hệ thống có các chức năng: 24](#_Toc89337970)

[3.1.2 Sơ đồ khối và chức năng mỗi khối 24](#_Toc89337971)

[3.1.2.1 Chức năng từng khối 24](#_Toc89337972)

[3.1.2.2 Sơ đồ khối của hệ thống 24](#_Toc89337973)

[3.1.2.3 Hoạt động của hệ thống 25](#_Toc89337974)

[3.2 Thiết kế hệ thống các khối 25](#_Toc89337975)

[3.2.1 Khối chia xung 25](#_Toc89337976)

[3.2.2 Khối thanh ghi dịch 26](#_Toc89337977)

[3.2.3 Khối FSM 27](#_Toc89337978)

[3.2.4 Sơ đồ nguyên lý toàn mạch 27](#_Toc89337979)

[3.3 Thiết kế hệ thống phần mềm 28](#_Toc89337980)

[3.2.2 Lưu đồ chương trình tổng quát 28](#_Toc89337981)

[3.3.2 Lưu đồ chương trình chính 29](#_Toc89337982)

[3.4 Thiết kế kiểm tra lỗi [7], [9] 30](#_Toc89337983)

[3.4.1 Test Plan 30](#_Toc89337984)

[3.4.2 Sơ đồ kiểm tra lỗi 31](#_Toc89337985)

[3.4.3 Testcase 32](#_Toc89337986)

[3.5 Thực thi hệ thống 35](#_Toc89337987)

[CHƯƠNG 4: KẾT QUẢ VÀ THỰC NGHIỆM 36](#_Toc89337988)

[4.1 Kết quả 36](#_Toc89337989)

[4.2 Thực nghiệm 37](#_Toc89337990)

[CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 40](#_Toc89337991)

[5.1 Kết luận: 40](#_Toc89337992)

[5.2 Hướng phát triển: 40](#_Toc89337993)

[TÀI LIỆU THAM KHẢO 41](#_Toc89337994)

[PHỤ LỤC VỀ CHƯƠNG TRÌNH 42](#_Toc89337996)

CHƯƠNG 1: TỔNG QUAN

1.1 Tình hình nghiên cứu trong và ngoài nước

1.1.1 Đặt vấn đề

Trong cuộc cách mạng công nghiệp 4.0 hiện nay với sự phát triển không ngừng của khoa học và công nghệ đã làm cho cuộc sống của con người được tốt hơn và hoàn thiện hơn. Các thiết bị điều khiển thông minh và tự động cũng được phát triển nhằm phục vụ con người. Trong đó chuẩn giao tiếp đóng vai trò quan trọng trong việc giao tiếp giữa các thiết bị như các thiết bị trong nhà hoạt động một cách tự động hoặc bán tự động, các thiết bị giao tiếp với nhau trở nên ổn định, thuận lợi và an toàn hơn nhờ ứng dụng của các chuẩn giao tiếp. Với sự xuất hiện và phát triển mạnh mẽ của các chuẩn giao tiếp các vi xử lý, vi điều khiển, công nghệ IoT đã được ra đời, các thiết bị được kết nối với nhau thông qua các chuẩn giao tiếp có thể giao tiếp với nhau từ đó hình thành từ đó ta có thể kết nối chúng để phục vụ nhu cầu của con người.

1.1.2 Giới thiệu

Khi kết nối nhiều thiết bị với một bộ vi điều khiển, địa chỉ và đường dữ liệu của mỗi thiết bị được kết nối riêng lẻ. Điều này sẽ chiếm các chân quý giá trên vi điều khiển, dẫn đến rất nhiều dấu không đủ chân trên PCB và yêu cầu nhiều thành phần hơn để kết nối mọi thứ với nhau. Điều này làm cho các hệ thống này đắt tiền để sản xuất. Để giải quyết vấn đề này, Philips đã phát triển Inter-IC bus (mạch tích hợp liên), hay I2C, vào những năm 1980.

I2C, viết tắt của cụm từ “Intel – Integrated Circuit”, là một chuẩn bus nối tiếp hai chiều với hai dây tín hiệu được phát triển bởi hãng sản xuất linh kiện điện tử Philips. Ban đầu, chuẩn này chỉ được sử dụng trong các linh kiện điện tử của Philips. Sau đó, nhờ tính ưu việt và đơn giản của nó, I2C đã được chuẩn hóa và sử dụng rộng rãi trong các module truyền thông nối tiếp của vi mạch tích hợp ngày nay. Truyền thông với bus I2C là quá trình [truyền thông đồng bộ nối tiếp](https://vi.wikipedia.org/wiki/Truy%E1%BB%81n_th%C3%B4ng_n%E1%BB%91i_ti%E1%BA%BFp), hỗ trợ nhiều master và slave trên đường truyền. I2C phù hợp với các thiết bị ngoại vi ưu tiên về kết nối đơn giản và chi phí sản xuất thấp, quan trọng hơn là yêu cầu về tốc độ truyền.

I2C trở thành một chuẩn công nghiệp cho các giao tiếp điều khiển, có thể kể ra đây một vài tên tuổi kể đế như: Texas Instrument (TI), Maxim- Dallas, Analog Device, National Semiconductor, …Bus I2C được sử dụng làm bus giao tiếp ngoại vi cho rất nhiều loại IC khác nhau như các loại vi điều khiển 8051, PIC, AVR, ARM, chíp nhớ như Ram tĩnh (Static Ram), EEPROM, bộ chuyển tương tự số (ADC), số tương tự (DAC), IC điều khiển LCD, LED, ….

1.2 Mục tiêu đề tài.

Thiết kế, thi công và mô phỏng giao thức I2C controller bằng ngôn ngữ phần cứng Verilog. Master gửi địa chỉ cho Salve và khi Salve xác định đúng địa chỉ thì quá trình truyền nhận dữ liệu được diễn ra, đồng thời phát hiện được lỗi của chuẩn giao thức I2C trong quá trình giao tiếp.

1.3 Giới hạn đề tài.

Mô hình chỉ mang tính chất nghiên cứu. Không thể tạo ra sản phẩm thực tế vì khi muốn sản xuất chip thực tế phải đi qua rất nhiều công đoạn và chi phí sản xuất rất cao.

1.4 Nội dung nghiên cứu.

Mục tiêu xây dựng hệ thống như trên thì em sẽ thực hiện những nội dung sau:

* So sánh giữa các chuẩn giao tiếp được ứng dụng phổ biến hiện nay như I2C, UART, SPI.
* Tìm hiểu đề tài và các tài liệu có liên quan về đề tài cần thiết kế.
* Hoàn thiện bản thiết kế và mô tả hành vi từng khối.
* Hoàn thiện RTL code và thực hiện test plan.
* Hoàn thiện testcase và test từng khối trong hệ thống.
* Test toàn bộ hệ thống.
* Viết báo cáo và trình bày kết quả.

1.5 Bố cục.

Bố cục bài báo cáo bao gồm 5 phần chính:

* ***Chương 1: Tổng quan***

Chương này trình bày các vấn đề dẫn nhập lý do chọn đề tài, giới thiệu, mục tiêu, nội dung nghiên cứu, các giới hạn và bố cục của tiểu luận

* ***Chương 2: Cơ sở lý thuyết***

Chương này trình bày về quy trình thiết kế mạch số và giải thích cơ chế hoạt động của chuẩn giao tiếp I2C đồng thời so sách chuẩn giao tiếp I2C với các chuẩn giao tiếp hiện nay

* ***Chương 3: Thiết kế hệ thống***

Chương này trình bày về việc thực hiện viết RTL code hệ thống, giải thích các khối trong hệ thống, trình bày test case và thiết kế kiểm tra trong hệ thống

* ***Chương 4: Kết quả và thực nghiệm***

Chương này trình về kết quả testbench dạng sóng trên waveform

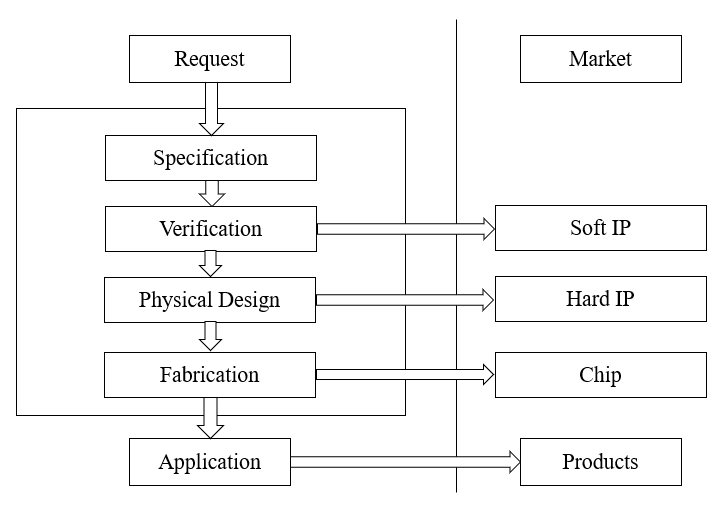
* ***Chương 5: Kết Luận và hướng phát triển***

Chương này đưa ra kết luận và hướng phát triển cho đề tài

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1 Tổng quan về qui trình thiết kế vi mạch số [1],[4],[5]

2.1.1 Quy trình thiết kế vi mạch số



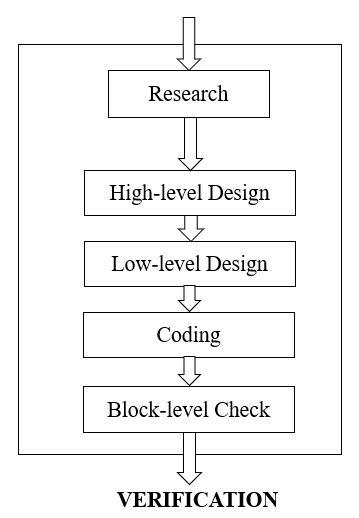
Hình 2.1 Tổng quan về quy trình thiết kế vi mạch số

IP core (lõi IP) có ba loại bao gồm: lõi IP mềm (soft core), lõi IP cứng (hard core) và firm core.

* Lõi IP mềm (Soft IP) đơn giản chính là RTL code (Verilog hoặc VHDL) thực hiện một chức năng nào đó.
* Lõi IP cứng (Hard IP) được sinh ra từ lõi IP mềm, lõi IP sau khi được tổng hợp và layout sẽ tạo ra lõi IP cứng, lõi IP cứng chính là bản layout theo một công nghệ sản xuất chip nào đó và có thể đem đi chế tạo thành chip.

2.1.1.1 Specification

Specification là quá trình nghiên cứu các đặc tính kĩ thuật của lõi IP, từ đó phân tích thiết kế lõi IP đến mức cổng logic, bao gồm 5 bước chính:



Hình 2.2 Chi tiết quy trình Specification

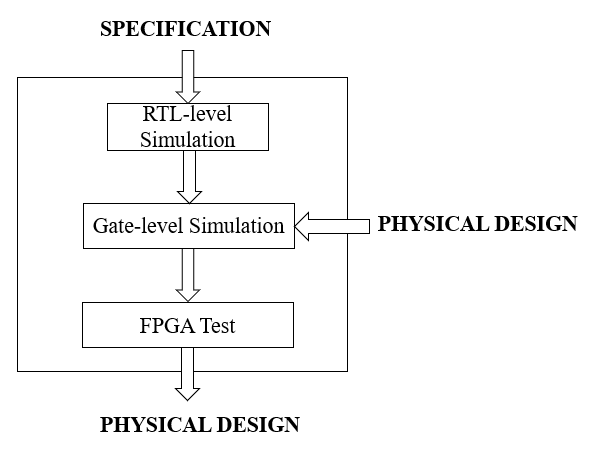
* Research: Tìm kiếm, đọc hiểu tài liệu và phân tích các giao thức, chức năng, các thông số mong muốn, … để tạo ra bảng chức năng chi tiết.
* High-level Design: Thiết kế mức kiến trúc: bao gồm vẽ sơ đồ khối và thiết kế được chia thành các khối riêng biệt, mỗi khối sẽ đảm nhiệm một chức năng riêng theo ý muốn của kĩ sư thiết kế và vẽ sơ đồ tín hiệu là xác định số lượng, chức năng và định thời các tín hiệu. Nếu thiết kế có giao diện được mô tả trong các giao thức chuẩn, các tín hiệu đó sẽ được liệt kê trong tài liệu về các giao thức đó.
* Low-level Design: Phân tích cấu trúc của thiết kế ở mức cổng (gate). Tất cả các khối được phân tích bằng việc dùng các bộ MUX, các cổng AND, OR, XOR, NOT và Flip-flop, … Bước này tiêu tốn rất nhiều thời gian do đó đóng vai trò đặc biệt và quan trọng trong luồng thiết kế. Kết quả thu được sau quá trình phân tích là sơ đồ cấu trúc chi tiết của thiết kế.
* Coding: Việc viết RTL được hoàn thành dựa vào sơ đồ cấu trúc chi tiết của thiết kế đã được phân tích ở bước thiết kế mức chức năng.
* Block-level Check: Kiểm tra và mô phỏng tất cả các khối riêng biệt bằng cách sử dụng công cụ kiểm tra cũng như các phần mềm mô phỏng.

2.2.1.2 Verification

Verification: quá trình được sử dụng để kiểm tra tính đúng đắn về chức năng của thiết kế, nhằm đảm bảo chắc chắn thiết kế logic phù hợp với Specification. Có 2 loại Verification thông dụng hiện nay là Functional Verification và Timing Verification.

Mục đích phát hiện lỗi có trong thiết kế. Bởi vì lỗi tìm thấy ở block-level ít gây ra thiệt hại. Lỗi tìm thấy ở system-level ảnh hưởng đến thời gian sản xuất IC tung ra thị trường. Lỗi tìm thấy sau khi chế tạo gây thiệt hại khá lớn vì chi phí chế tạo lại. Lỗi tìm thấy bởi khách hàng, người dùng có thể gây thiệt hại hàng triệu đô la và mất uy tín.

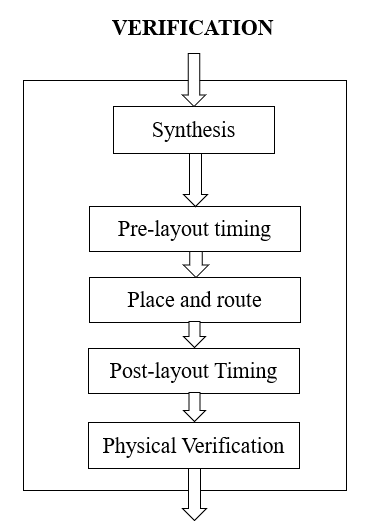
Verification gồm 3 bước chính:



Hình 2.3 Chi tiết quy trình Verification

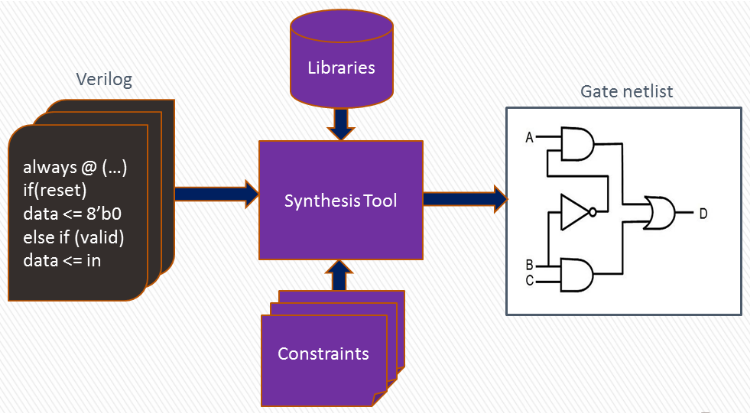
* RTL-level Simulation: Mô phỏng, phát hiện và sữa lỗi trong thiết kế ở mức RTL code.
* Gate-level Simulation: Mô phỏng và kiểm tra tính tương đương trong thiết kế ở mức cổng logic.
* FPGA Test: Kiểm tra hoạt động cơ bản của thiết kế trên KIT FPGA.

2.2.1.3 Physical design:



Hình 2.4 Chi tiết quy trình Physical design

* Synthesis: là quá trình chuyển đổi thiết kế dưới dạng mô tả bằng ngôn ngữ phần cứng (RTL -level) sang mô tả bằng các cổng logic (Gate-level). Quá trình tổng hợp logic được thực hiện bằng cách sử dụng tổ hợp RTL code và thư viện công nghệ, thư viện công nghệ là thư viện chứa các cell chuẩn, thông số của chúng và bao gồm các cổng logic cơ bản (AND OR XOR NOT) và các Macro cell (Adder, Mux, Memory, Flip-flop).



Hình 2.5 Quá trình tổng hợp logic

* Pre-layout Timing: Phân tích các vi phạm về thời gian (timing) sau khi tổng hợp
* Place and Route: Xác định vị trí của các cổng logic đặt trên chip vật lý đồng thời đảm bảo tối thiểu hóa độ trễ trên các Critical Path, độ nghẽn khi định tuyến (routing) và độ trễ đường dây bằng cách thay đổi kích thước các cổng logic, chèn thêm các bộ đệm hoặc thực hiện tổng hợp lại từng phần.
* Post-layout Timing: Phân tích, kiểm tra timing cho các đường timing sau quá trình Place and Route.
* Physical Verification: Kiểm tra layout của IC có thỏa mãn các điều kiện, tiêu chí kĩ thuật theo các luật lệ thiết kế (Design Rule check), tính đồng nhất giữa Schematic và Layout.

2.1.2 Công cụ sử dụng trong thiết kế vi mạch số

2.1.2.1 Specification

Online Diagram Software: Vẽ sơ đồ khối, sơ đồ tín hiệu, sơ đồ chi tiết mức cổng logic và sơ đồ máy trạng thái hữu hạn.

LEDA: Kiểm tra lỗi cú pháp trong RTL Code.

Công cụ FPGA: Quartus (Altera), Leonardo Spectrum (Mentor Graphics), ISE Design Suite (Xilinx) được dùng để biên dịch code RTL trong FPGA.

2.1.2.2 Verification

VCS: Trình mô phỏng hoạt động trên hệ điều hành Linux, được sử dụng trong công nghiệp.

ModelSim (Mentor Graphic): Trình mô phỏng sử dụng để kiểm tra chức năng logic của thiết kế.

2.1.2.3 Physical Design:

Design Compiler: Tổng hợp code RTL thành các công logic.

IC Compiler: Thực hiện trình biên dịch layout thiết kế.

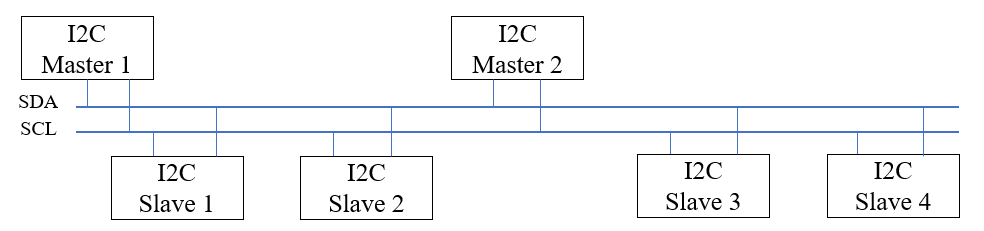
Prime Timer: Kiểm tra thời gian (timing) của thiết kế

Hercules: Kiểm tra thiết kế vật lý (Physical design), Design Rule Check (DRC) và layout Versus Schematic (LVS)

2.2 Giao thức I2C [2],[6],[8]

2.2.1 Đặc điểm của giao thức I2C

* Truyền bán song công.
* Có xung đồng bộ nên ít xảy ra hiện tượng mất mát dữ liệu.
* Hoạt động theo mô hình Mesh. Thiết bị Master điều khiển nhiều thiết bị slaver. Thiết bị chủ nắm vai trò tạo xung clock cho toàn hệ thống, khi thiết bị chủ/tớ giao tiếp thì thiết bị chủ có nhiệm vụ tạo xung clock và quản lý địa chỉ của thiết bị tớ trong suốt quá trình giao tiếp.
* Dữ liệu truyền theo dạng nối tiếp.
* Tốc độ truyền nhanh 1Mbs.
* Mỗi thiết bị luôn được đánh một giá trị địa chỉ duy nhất (do nhà sản xuất quy định).

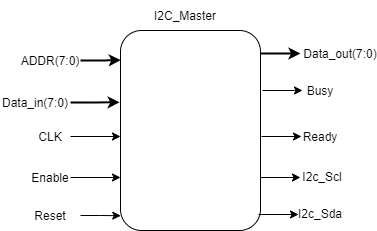


Hình 2.6 Mô hình hệ thống giao thức I2C

2.2.2 Chức năng của I2C controller

* Đọc/Ghi dữ liệu theo byte (8 bits).
* Có các cờ báo hiệu.
* Có khả năng kiểm trả phát hiện lỗi truyền dữ liệu.
* Kết nối trao đổi dữ liệu với nhiều thiết bị khác nhau.

2.2.3 Sơ đồ khối:



Hình 2.7 Sơ đồ khối của I2C Controller

Chức năng:

|  |  |  |
| --- | --- | --- |
| STT | Chức năng | Chân |
| 1 | Kiểm soát địa chỉ (7bit) | ADDR (7:0) |
| 2 | Cấp xung | CLK |
| 3 | Trạng thái sẵn sàng (I2C controller không hoạt động | Ready |
| 4 | Trạng thái bận (đường truyền đang có dữ liệu) | Busy |
| 5 | Đường truyền xung | SCL |
| 6 | Đường truyền dữ liệu | SDA |
| 7 | Cho phép I2C hoạt động | Enable |
| 8 | Chân cấp dữ liệu (8-bit) | Data\_in (7:0) |
| 9 | Chân dữ liệu ra truyền cho vi điều khiển (8-bit) | Data\_out (7:0) |
| 10 | Reset quá trình truyền về trạng thái ban đầu | Reset |
| 11 | Kiểm soát quá trình đọc ghi dữ liệu | R/ |

2.2.4 Hoạt động của I2C

Hệ thống I2C sử dụng đường truyền dữ liệu nối tiếp (SDA) và đường clock nối tiếp (SCL) để truyền dữ liệu. SDA là đường truyền/nhận dữ liệu, SCL là đường truyền xung clock. Mỗi dây SDA và SCL đều được kết nối với dương nguồn qua điện trở kéo lên 1KΩ – 4.7 KΩ tùy thuộc vào thiết bị.

Thông thường, một giao tiếp tiêu chuẩn bao gồm bốn phần:

1) START signal generation: Thiết bị chủ xác định địa chỉ của thiết bị tớ cần giao tiếp và gửi 1 tín hiệu START, liền sau đó gửi 1-byte dữ liệu cho thiết bị tớ cùng với yêu cầu truyền hay nhận dữ liệu tương ứng.  
2) Slave address transfer: Thiết bị tớ nhận thấy đúng với địa chỉ của mình thì xác nhận bằng 1 xung ACK.  
3) Data transfer: Quá trình truyền dữ liệu bắt đầu, bên nhận dữ liệu sẽ kết thúc mỗi byte nhận bằng việc trả ACK.  
4) STOP signal generation: Thiết bị chủ gửi một 1 tín hiệu STOP, kết thúc quá trình truyền nhận.

2.2.4.1 Định dạng khung dữ liệu



Hình 2.8 Định dạng khung dữ liệu theo chuẩn I2C

Dữ liệu trên Bus I2C được truyền theo từng bit một, tại mỗi sườn dương của clock trên dây ACK. Dữ liệu thay đổi trên SDA theo sườn dốc xuống của clock.

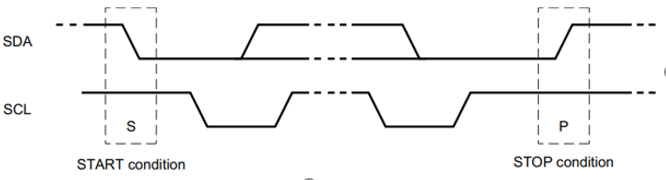
Slave có địa chỉ khớp với địa chỉ được truyền bởi Master mới phản hồi bằng cách trả lại một bit xác nhận (xung ACK) bằng cách kéo SDA thấp ở chu kỳ clock SCL thứ 9. Khi đã đạt được địa chỉ Slave thành công, quá trình truyền dữ liệu có thể tiến hành trên cơ sở từng byte theo hướng được chỉ định bởi bit R/ được gửi bởi Master

Dữ liệu được chuyển đồng bộ giữa Master và Slave tới SCL trên dòng SDA trên cơ sở từng byte. Mỗi byte dữ liệu dài 8 bit. Có một xung clock SCL cho mỗi bit dữ liệu với MSB được truyền đầu tiên. Một bit xác nhận theo sau mỗi byte được chuyển. Mỗi bit được lấy mẫu trong chu kỳ cao của SCL; do đó, đường SDA chỉ được thay đổi trong chu kỳ SCL thấp và phải được giữ ổn định trong chu kỳ SCL cao.

Trường hợp thiết bị nhận không đủ 8 bits dữ liệu, thiết bị nhận sẽ gửi 1 xung NOT-ACK (SDA cao, SCL thứ 9), từ đó thiết bị chủ sẽ gửi tín hiệu STOP hay RESTART.

2.2.4.2 Điều kiện START và STOP

Khi muốn thiết lập hay kết thúc một quá trình truyền nhận thì thiết bị chủ đều phải gửi tín hiệu START hay STOP để báo hiệu cho thiết bị tớ biết. Ban đầu đường truyền đang ở chế độ rỗi (idle), có nghĩa là không có thiết bị chủ (Master) đang tham gia vào bus (Cả 2 dòng SCL và SDA đều ở mức cao), Một Master có thể bắt đầu chuyển giao bằng cách gửi tín hiệu START.



Hình 2.9 Tín hiệu START và STOP

Tín hiệu START, thường được gọi là S-bit, được định nghĩa là sự chuyển đổi từ cao đến thấp của SDA trong khi SCL ở mức logic '1', tín hiệu START biểu thị sự bắt đầu của quá trình truyền dữ liệu mới.

Tín hiệu STOP, thường được gọi là P-bit, được định nghĩa là sự chuyển đổi từ thấp đến cao của SDA trong khi SCL ở mức logic '1'.

Cả 2 tín hiệu START và STOP đều được tạo bởi thiết bị chủ. Sau tín hiệu START, bus I2C coi như đang trong trạng thái hoạt động (busy). Bus I2C sẽ rãnh(idle), sẵn sàng cho một giao tiếp mới sau tín hiệu STOP từ phía Master.

2.2.4.3 Địa chỉ thiết bị

Mỗi thiết bị tham gia vào bus I2C đều có một địa chỉ duy nhất do nhà sản xuất quy định để tránh gây nhầm lẫn đường truyền. Độ dài byte địa chỉ là 7 bits, do đó sẽ có tối đa 128 thiết bị trên bus I2C. Khi thiết bị chủ (Master) muốn giao tiếp với thiết bị ngoại vi nào trên bus I2C, nó sẽ gửi 7 bits địa chỉ của thiết bị đó ra bus ngay sau xung START. Byte đầu tiên được gửi sẽ bao gồm 7 bits địa chỉ và một bit thứ 8 điều khiển hướng truyền dữ liệu. Nếu bit thứ 8 bằng “0” có nghĩa là byte dữ liệu tiếp theo sau sẽ được truyền từ Master đến Slave, còn ngược lại nếu bằng “1” thì các byte theo sau byte đầu tiên sẽ là dữ liệu từ thiết bị tớ gửi đến thiết bị chủ hoặc ngược lại.

2.2.4.4 Chế độ truyền, nhận Master-Slave

Việc truyền dữ liệu trên bus I2C thực hiện theo 2 hướng: từ Master đến Slave và ngược lại, hướng truyền được quy định bởi bit thứ 8 trong byte đầu tiên.

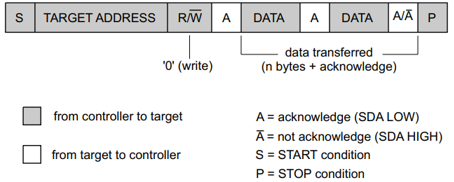
Master truyền Slave nhận:

Master tạo tín hiệu START, bắt đầu quá trình truyền dữ liệu

Master gửi byte đầu tiên gồm 7 bits địa chỉ của Slave cần giao tiếp, và bit thứ 8 R/ “0” và đợi xung phản hồi ACK từ Slave.

Nhận được ACK báo đúng địa chỉ từ Slave, Master bắt đầu truyền dữ liệu theo từng bit một. Kết thúc mỗi byte 8 bits thì chờ xung ACK phản hồi. Số lượng byte được truyền không giới hạn.

Muốn kết thúc quá trình truyền thì Master gửi 1 tín hiệu STOP



Hình 2.10 Master truyền Slave nhận

Master nhận Slave truyền:

Master tạo tín hiệu START, bắt đầu quá trình truyền dữ liệu

Master gửi byte đầu tiên gồm 7 bits địa chỉ của Slave cần giao tiếp, và bit thứ 8 R/ “1” và đợi xung phản hồi ACK từ Slave.

Nhận được ACK báo đúng địa chỉ từ Slave, Slave bắt đầu truyền dữ liệu theo từng bit một. Kết thúc mỗi byte 8 bits thì chờ xung ACK phản hồi. Số lượng byte được truyền không giới hạn.

Muốn kết thúc quá trình truyền thì Master gửi 1 tín hiệu STOP

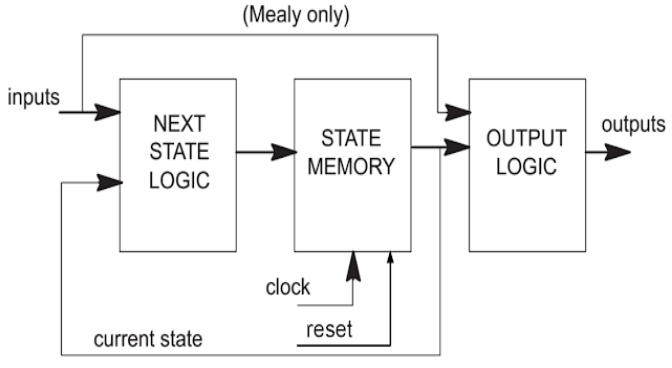


Hình 2.11 Master nhận Slave truyền

2.3 Máy trạng thái (FSM) [3]

Máy trạng thái hữu hạn (finite-state machine FSM) là một [mô hình toán học](https://vi.wikipedia.org/wiki/M%C3%B4_h%C3%ACnh_t%C3%ADnh_to%C3%A1n). Nó là một thành phần quan trọng không thể thiếu được sử dụng phổ biến trong thiết kế vi mạch số với ưu điểm là dễ kiểm soát quá trình hoạt động của thiết kế và dễ debug khi gặp lỗi ở thiết kế.

2.3.1 Mô hình cơ bản của máy trạng thái



Hình 2.12 Mô hình của một máy trạng thái

Mô hình máy trạng thái gồm 3 thành phần cơ bản như sau:

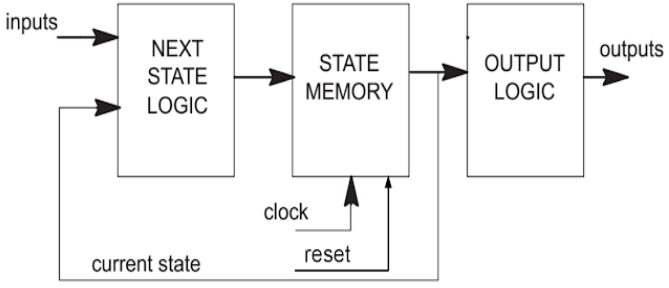
Mạch tạo trạng thái kế tiếp (Next state logic) là một mạch tổ hợp phụ thuộc vào ngõ vào của FSM và giá trị trạng thái hiện tại lấy từ bộ nhớ trạng thái (state memory)

Bộ nhớ trạng thái (state memory) là phần tử nhớ để lưu trạng thái hiện tại của FSM nó có thể là Flip-Flop, Latch, ... lấy ngõ vào từ mạch tạo trạng thái kế tiếp (next state logic). Bộ nhớ trạng thái thường được sử dụng trong các thiết kế số đồng bộ là Flip Flop hoạt động theo xung clock. Một tín hiệu reset có thể được sử dụng để khởi động FSM trở về giá trị ban đầu.

Mạch tạo ngõ ra (output logic) là mạch tổ hợp tạo ra giá trị ngõ ra tương ứng với trạng thái hiện tại của máy trạng thái. Mạch này lấy ngõ vào là giá trị trạng thái hiện tại và có thể tổ hợp thêm ngõ vào của FSM tùy vào phân loại của máy trạng thái.

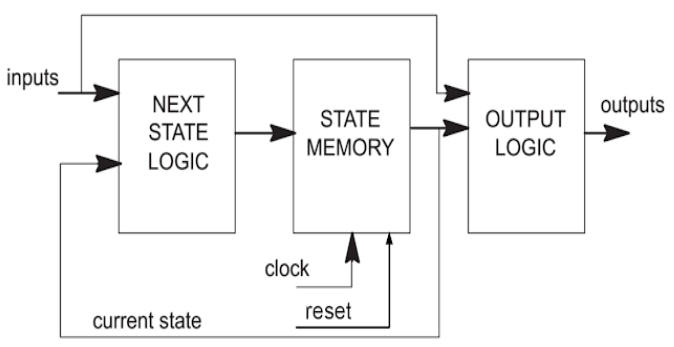
2.3.2 Phân loại

Máy trạng thái được chia làm 2 loại:



Hình 2.13: Mô hình máy trạng thái Moore

Máy trạng thái Moore là máy trạng thái có ngõ ra không phụ thuộc trực tiếp vào ngõ vào FSM



Hình 2.14: Mô hình máy trạng thái mealy

Máy trạng thái Mealy là máy trạng thái có ngõ ra phụ thuộc trực tiếp vào ngõ vào FSM

CHƯƠNG 3: THIẾT KẾ, TÍNH TOÁN VÀ XÂY DỰNG HỆ THỐNG HỆ THỐNG

3.1 Yêu cầu hệ thống

3.1.1 Hệ thống có các chức năng:

* Truyền, nhận dữ liệu giữa Master và Slave
* Phản hồi xung ACK
* Xác nhận địa chỉ
* Phát hiện lỗi truyền nhận sai các bit địa chỉ và các bit dữ liệu
* Reset

3.1.2 Sơ đồ khối và chức năng mỗi khối

3.1.2.1 Chức năng từng khối

* **Khối CLOCK GENERATOR:** Dùng để chia xung CLK đầu vào từ thạch anh ra các xung với tần số khác hau phù hợp với tốc độ truyền của module. Khi muốn truyền dữ liệu cho slave ta chỉ cần đưa lệnh cho CPU xuất tín hiệu 0/1 vào 8 input của thanh ghi...
* **Khối REGISTER:** Khối này thực hiện chức năng lưu trữ các giá trị truyền đi hay nhận vào. Ngoài ra còn được sử dụng làm bộ đệm trong quá trình xử lí dữ liệu.
* **Khối FSM:** là module máy trạng thái, tại dậy module sẽ thực hiện các chức năng và trạng thái để truyền nhận dữ liệu giữa master và slaver. Cụ thể là các trạng thái reset, enable, start, idle, read, write…

3.1.2.2 Sơ đồ khối của hệ thống

CLK

Register

Clock generator

FSM

Data

Data\_in

SDA

SCL

Hình 3.1: Sơ đồ khối của hệ thống

3.1.2.3 Hoạt động của hệ thống

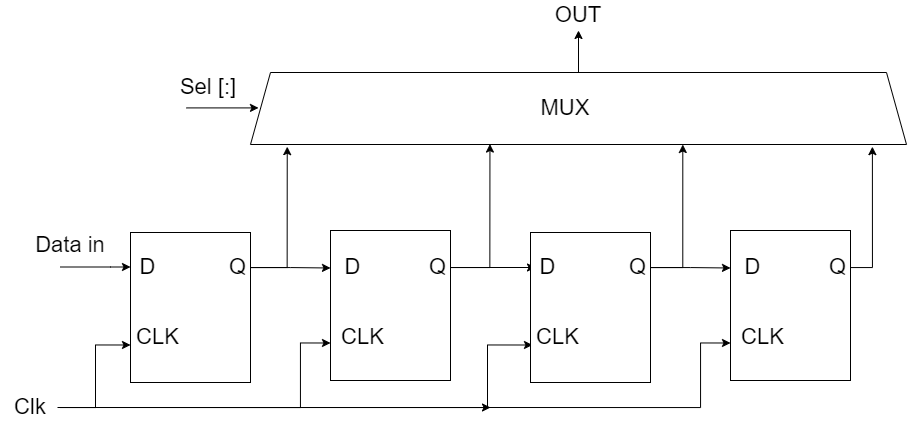
Khi hệ thống được cấp xung hệ thống sẽ hoạt động theo trình tự như sau:

**Bước 1:** Từ xung clock thạch anh, sẽ đi qua CLK\_DIV để chia tần số thành tần số phù hợp  
**Bước 2:** Data và địa chỉ sẽ được lưu vào thanh ghi 8 bit

**Bước 3:** Tương ứng với các trạng thái trong máy trạng thái, các bit sẽ được truyền ra ngoài chân SDA đồng bộ với xung SCL

3.2 Thiết kế hệ thống các khối

3.2.1 Khối chia xung

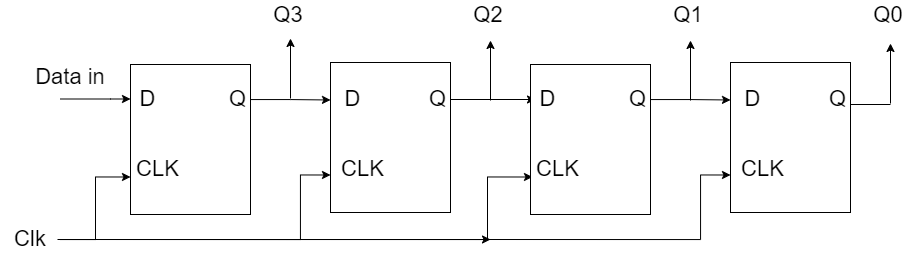


Hình 3.2: Sơ đồ khối chia xung

**Công thức tính tần số ngõ ra: Fout = Fin / 2(tổng số flip flop)**

Khối chia xung được sử dụng với mục đích hạ xung clock hệ thống xuống tần số nhất định phù hợp cho quá trình truyền nhận tốt nhất. Người dùng có thể tự điểu chỉnh và tính toàn mức tần số móng muốn bằng cách thay đổi giá trị thanh ghi tương ứng với số flip flop.

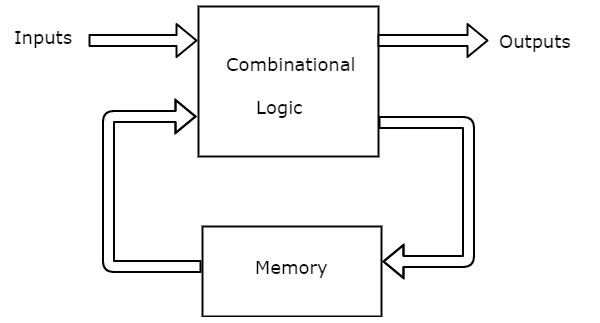
3.2.2 Khối thanh ghi dịch



Hình 3.3: Sơ đồ khối thanh ghi dịch

* Thanh ghi dịch được tạo ra bằng cách xếp tầng các Flip-Flop (Register) thành một chuỗi. Tất cả thanh ghi phải sử dụng cùng một **clock** và đầu ra của một thanh ghi phải được kết nối với đầu vào của thanh ghi tiếp theo trong chuỗi đó
* Khi dữ liệu được truyền vào từ chân SDA thì thanh ghi dich có chức năng chuyển đổi dữ liệu từ dạng nối tiếp sang dạng song song.

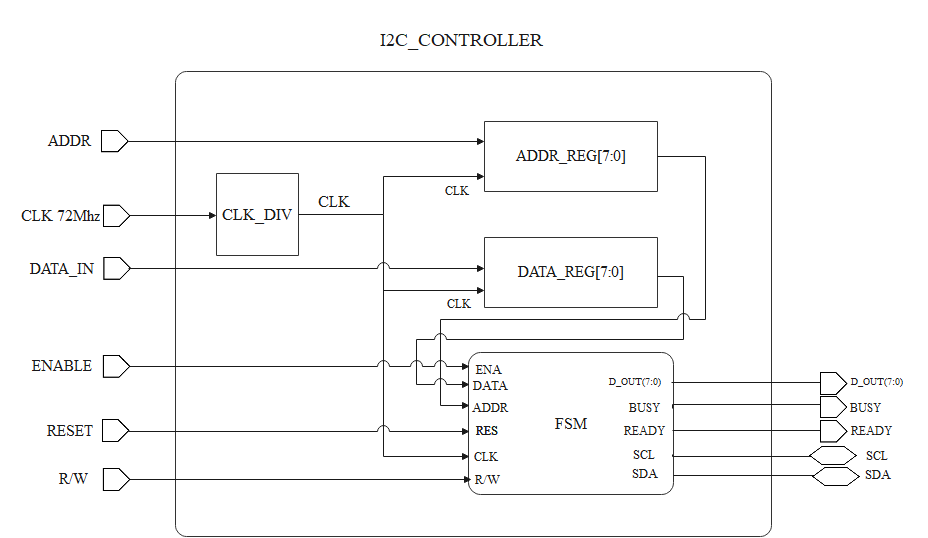
3.2.3 Khối FSM



Hỉnh 3.4: Sơ đồ khối FSM

Máy trạng thái bao gồm hai khói chính đó là khối logic tổng hợp khối này chứa các là nơi thay đổi trạng thái của hệ thống. Còn khối memory là dùng để lưu trữ trạng thái hiện tại của hệ thống đồng thời hồi qui lại cho khối logic tổng hợp để thay đổi trạng thái.

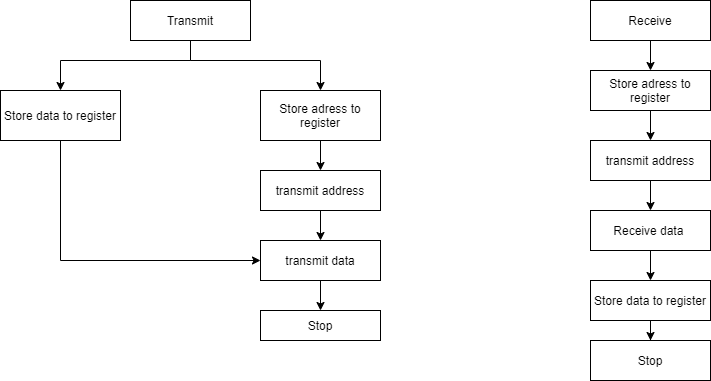
3.2.4 Sơ đồ nguyên lý toàn mạch



Hình 3.5: Sơ đồ nguyên lý toàn mạch

3.3 Thiết kế hệ thống phần mềm

3.2.2 Lưu đồ chương trình tổng quát



Hình 3.2 Lưu đồ chương trình tổng quát toàn bộ hoạt động

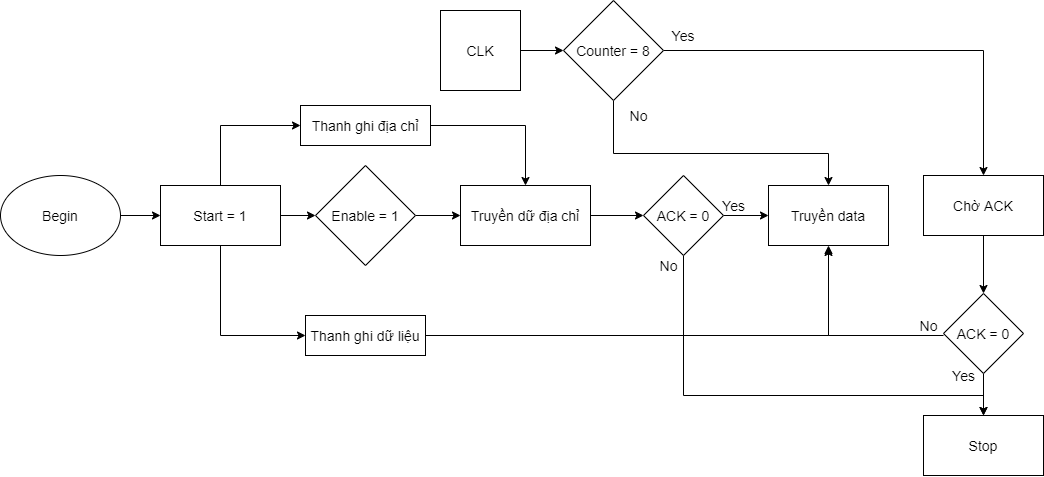
Khi mạch bắt đầu truyền:

1. Dữ liệu cần truyền và địa chỉ slaver sẽ được lưu vô thanh ghi để chuẩn bị cho quá trình truyền.
2. Sau khi lưu xong. CLK sẽ truyền sang một bộ chia xung để xuống tần số phù hợp
3. Tại môi cạnh lên của xung SCL thì data sẽ được truyền lần lượt ra chân SDA. Sau đó các data sẽ được lưu vào trong thanh ghi.
4. Khi dữ liệu được lưu đầy bộ đệm thì master sẽ chờ ACK xác nhận từ slaver và sau đó kết thúc quá trình truyền.

Khi mạch bắt đầu đọc

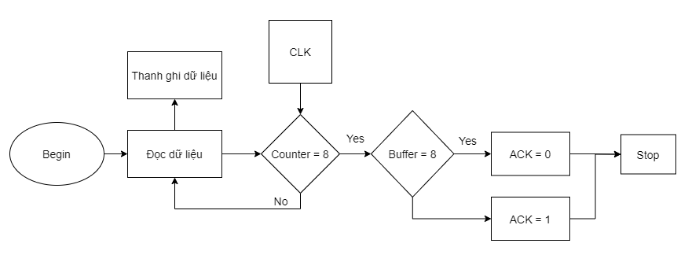
1. Ban đầu master sẽ lưu địa chỉ slaver vô thanh ghi
2. Tiếp theo tại xung cạnh lên SCL thì địa chỉ được truyền lần lượt ra chân SDA tới slaver.
3. Master chờ xác nhận ACK từ slaver.
4. Slaver truyền lần lượt data qua chân SDA tới Master.
5. Sau khi truyền đủ 8 bits thì kết thúc quá trình truyền.

3.3.2 Lưu đồ chương trình chính



Hình 3.3 Lưu đồ chương trình chính của hệ thống

Khi bắt đầu hệ thông bit start sẽ được set lên mức 1 đồng thời enable cho phép thì data và địa chỉ được load vô thanh ghi. Sau đó tại chu kì tiếp theo sẽ chuyển sang trạng thái truyền địa chỉ. Mỗi bit được truyền thi sẽ được đếm thông qua một bộ counter để đếm số bit và chu kì. Nếu counter = 8 quá trình truyền địa chỉ và R/W đã đủ. Lúc này máy trạng thái sẽ chuyển sang trạng thái chờ nhận bit ACK từ slaver. Nếu nhận được bit 0 thì master bắt đầu truyền các bit data từ thanh ghi ra chân sda theo thứ tự. Tại lúc này, counter reset giá trị về 0 và tiếp tục đếm cho đến khi bằng 8. Sau khi truyền xong thì master chờ ACK từ slaver và sau đó kết thúc quá trình truyền.

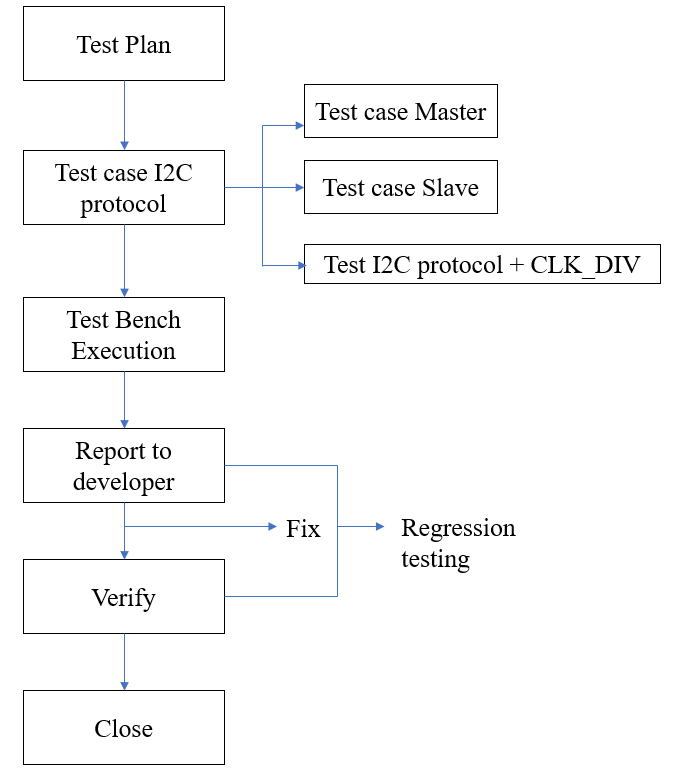


Hình 3.4 Lưu đồ quá trình đọc dữ liệu

Khi bắt đầu nhận dữ liệu từ slaver thì master sẽ clear hết giá trị trong thanh ghi dữ liệu. Sau đó lần lượt các bit từ slaver được truyền tới master và lưu vào trong thanh ghi. Ngay khi bộ đếm tràn thì quá trình nhận sẽ kết thúc. Sau đó Master sẽ truyền dữ liệu từ thanh ghi ra ngoài chân data\_out đồng thời gửi ACK xác nhận về cho slaver. Cuối cùng là kết thúc quá trình nhận dữ liệu.

3.4 Thiết kế kiểm tra lỗi [7], [9]

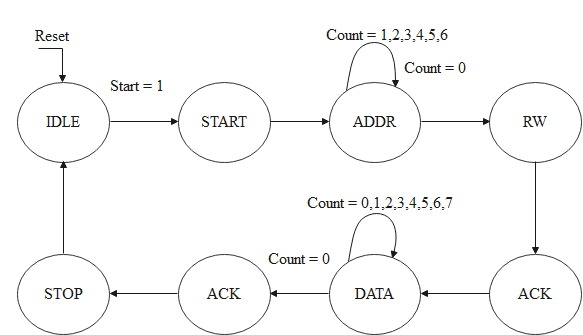
3.4.1 Test Plan



Hình 3.4 Sơ đồ test plan hệ thống

* CLK\_DIV: Chia tần số phù hợp với mỗi khối trong hệ thống.
* Test case Master: Test khối Master đã tạo với khối Slave chuẩn
* Test case Slave: Test khối Slave chuẩn với khối Master đã tạo
* Test I2C protocol + CLK\_DIV: test quá trình giao tiếp giữa Master và Slave, sử dụng bộ chia xung
* Test Bench Execution: Test Bench quá trình giao tiếp giữa Master với Slave và ngược lại

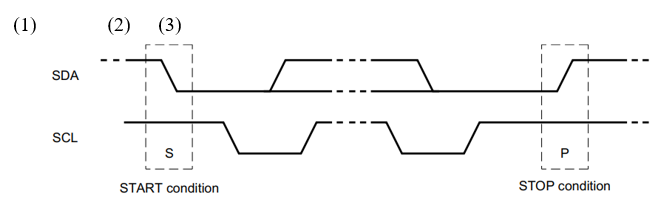
3.4.2 Sơ đồ kiểm tra lỗi



Hình 3.5 Sơ đồ kiểm tra lỗi

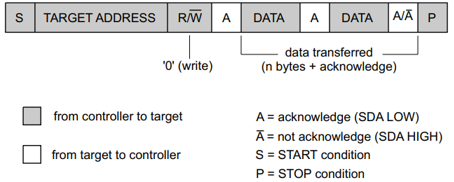
* IDLE: Trạng thái rỗi, mạch không hoạt động
* START: Tín hiệu Start, mạch bắt đầu hoạt động
* ADDR: Địa chỉ Slave
* RW: Hướng truyền dữ liệu
* ACK: xung xác nhận
* DATA: Dữ liệu được truyền đi
* STOP: Tín hiệu STOP, mạch dừng hoạt động

3.4.3 Testcase



Hình 3.6: Trạng thái Start và Stop

* (1) Trạng thái reset: reset= 1, enable= X thì SDA và SCL cao, ready = 0, busy = 0
* (2) Trạng thái ready: reset = 0, enable= 1, SDA và SCL cao, ready = 1, busy = 0
* (3) Trạng thái Start: reset= 0, enable = 1, SDA mức cao sang SDA mức thấp, SCL ở mức cao, ready = 0, busy = 1
* Hướng truyền dữ liệu:



Hình 3.7 Quá trình truyền dữ liệu

RW = 0: Mater truyền dữ liệu



Hình 3.8 Quá trình nhận dữ liệu

RW = 1: Master nhận dữ liệu

* Trạng thái truyền nhận dữ liệu



Hình 3.9 Frame truyền nhận dữ liệu chuẩn

Địa chỉ Slave: 7’b0001111

Trạng thái xác nhận:

addr(6:0) = 7’b0001111, RW = 0, SDA = 0 (ở SCL thứ 9)

=> data\_in(7:0) = 8’b101010101

addr(6:0) = 7’b0001111, RW = 1, SDA = 0 (ở SCL thứ 9)

=> data\_out(7:0) = 8’b101010101

Trạng thái từ chối:

Th1: Truyền, nhận không đúng số bit địa chỉ

Addr(6:0) = 7’b1111111 và RW= 0, SDA = 1( ở SCL thứ 9)

=>data\_in(7:0) = 8’bxxxxxxxx

Addr(6:0) = 7’1111111 và RW= 1, SDA = 1( ở SCL thứ 9)

=>data\_out(7:0) = 8’bxxxxxxxx

Th2: Truyền không đủ số bit dữ liệu

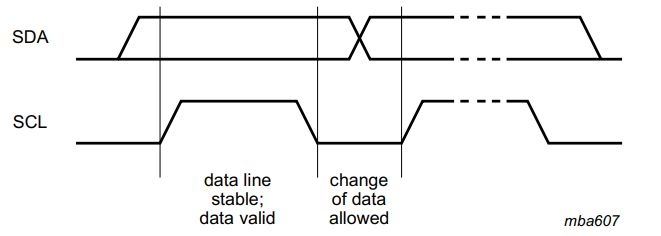
Addr(6:0) = 7’b0001111 và RW= 0, SDA = 0( ở SCL thứ 9), data\_in(5:0) = 6’b000111

Th3: Nhận không đủ số bit dữ liệu

Addr(6:0) = 7’b0001111 và RW= 1, SDA = 0( ở SCL thứ 9),

=>data\_out(5:0) = 8’bxxxxxxxx

* Quá trình truyền 1-bit

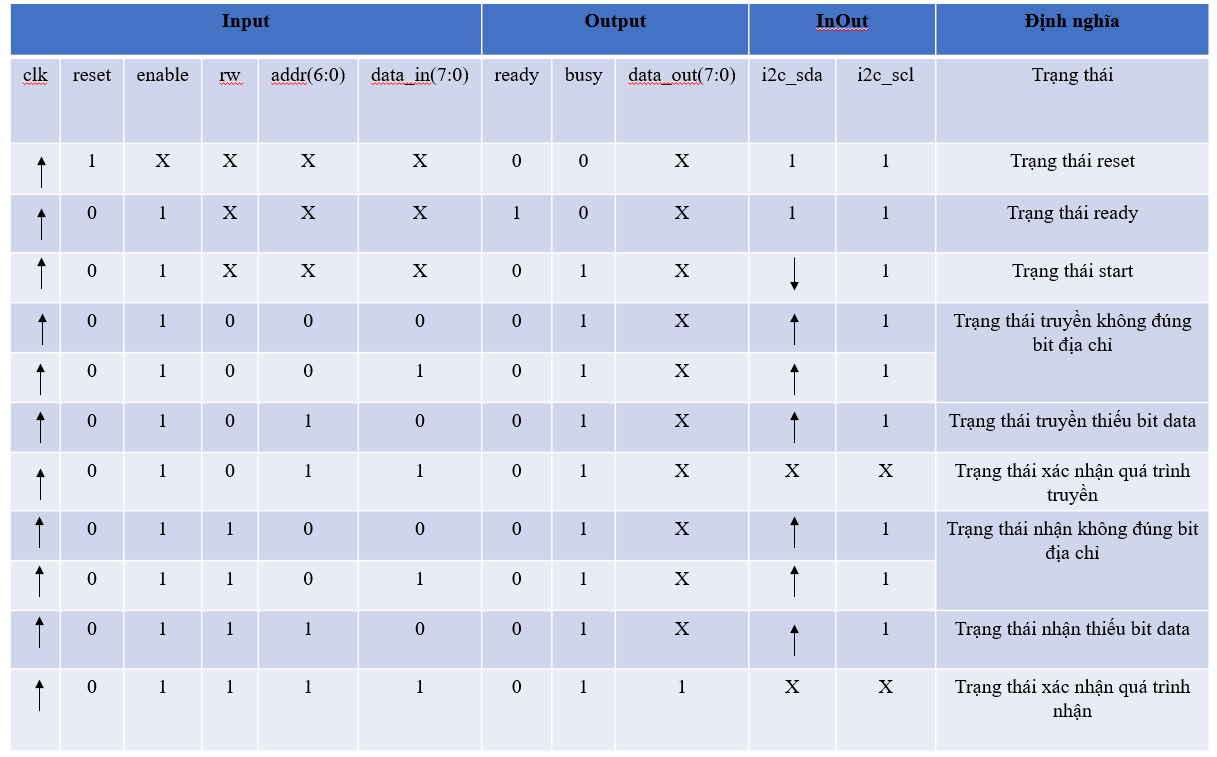


Hình 3.10: Quá trình truyền 1-bit dữ liệu

Dữ liệu trong SDA phải ổn định trong thời gian SCL mức cao, trạng thái cao hoặc thấp của đường dữ liệu chỉ có thể thay đổi khi SCL ở mức thấp

|  |  |
| --- | --- |
| STEP TESTCASE | DEFINITION |
| Trạng thái reset | Tín hiệu reset |
| Trạng thái start không sẵn sàng | Trạng thái ready |
| Trạng thái start sẵn sàng | Trạng thái start |
| Hướng truyền dữ liệu | RW = 0 (Master -> Salve)  RW = 1 (Slave -> Mater) |
| Trạng thái truyền nhận dữ liệu | Truyền không đúng số bit địa chỉ  Truyền không đủ số bit dữ liệu  Nhận không đủ số bit dữ liệu |
| Quá trình truyền 1-bit | Sự ổn định trong SDA  Sự thay đổi trong SDA |

Bảng trạng thái



3.5 Thực thi hệ thống

Các bước hoạt động của mô hình:

Bước 1: Thiết bị chủ xác định địa chỉ của thiết bị tớ cần giao tiếp và gửi 1 tín hiệu START, liền sau đó gửi 1-byte dữ liệu cho thiết bị tớ cùng với yêu cầu truyền hay nhận dữ liệu tương ứng.

Bước 2: Thiết bị tớ nhận thấy đúng với địa chỉ của mình thì xác nhận bằng 1 xung ACK.

Bước 3: Quá trình truyền dữ liệu bắt đầu, bên nhận dữ liệu sẽ kết thúc mỗi byte nhận bằng việc trả ACK.

Bước 4: Thiết bị chủ gửi một 1 tín hiệu STOP, kết thúc quá trình truyền nhận.

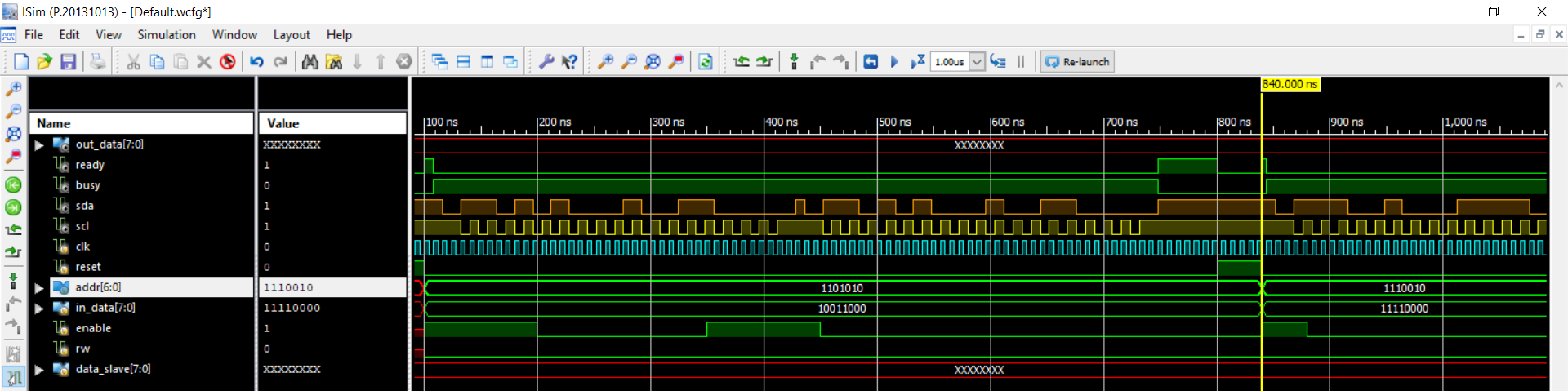
CHƯƠNG 4: KẾT QUẢ VÀ THỰC NGHIỆM

4.1 Kết quả

Trong quá trình thực hiện đề tài em đã hoàn thành được các khối trong hệ thống như:

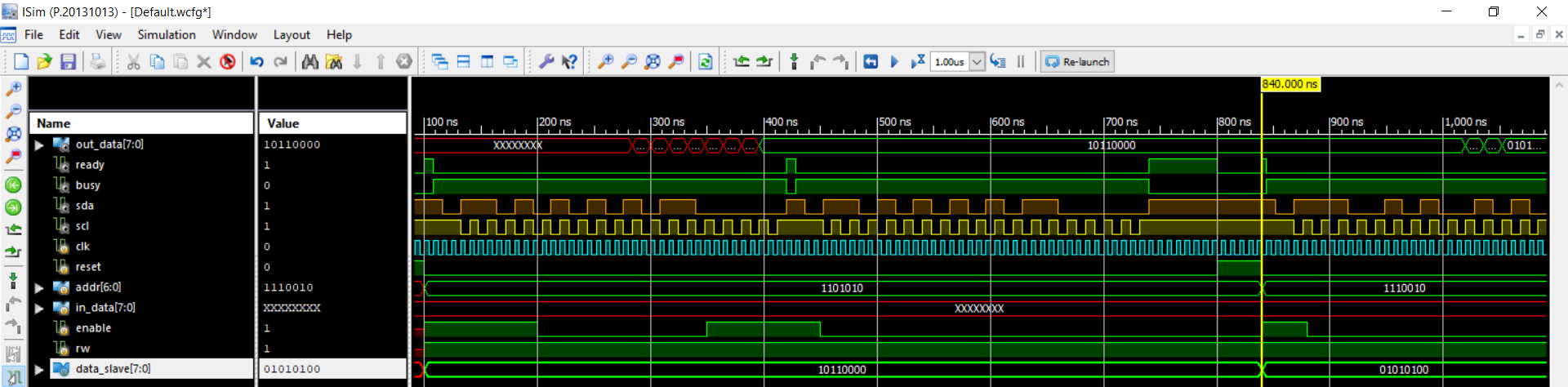
* FSM
* CLOCK GENERATOR
* REGISTER

Thành công trong việc thiết kế và mô phỏng **“I2C Controller “**



Hình 4.1 Testbench quá trình Master truyền dữ liệu

Dạng sóng quá trình Master truyền lần lượt từng byte dữ liệu đến Slave. Ban đầu ở trạng thái ready tức lúc này Ready mức ‘1’, Busy mức ‘0’ và (I2C hoạt động ở trạng thái rỗi). Tiếp theo là tín hiệu start tức S-bit. SDA chuyển trạng thái từ mức ‘1’ xuống mức 0 trong khi SCL mức ‘1’, sau tín hiệu start master sẽ gửi 7 bits địa chỉ và 1 bits RW để xác định hướng truyền dữ liệu, trong trường hợp này RW=’0’ và khi Slave xác định đó là địa chỉ của mình thì nó sẽ trả về 1-bit ACK xác nhận và quá trình truyền dữ liệu diễn ra trong giai đoạn tiếp theo, sau khi master truyền đủ 8 bits dữ liệu thì Slave sẽ xác nhận điều đó bằng cách gửi 1-bit ACK xác nhận, trường hợp truyền không đủ dữ liệu thì Slave sẽ gửi 1-bit NACK, và master sẽ gửi tín hiệu stop để kết thúc quá trình truyền.



Hình 4.2 Testbench quá trình Master nhận dữ liệu

Dạng sóng quá trình Slave truyền lần lượt từng byte dữ liệu đến Master. Ban đầu ở trạng thái ready tức lúc này Ready mức ‘1’, Busy mức ‘0’ và (I2C hoạt động ở trạng thái rỗi). Tiếp theo là tín hiệu start tức S-bit. SDA chuyển trạng thái từ mức ‘1’ xuống mức 0 trong khi SCL mức ‘1’, sau tín hiệu start master sẽ gửi 7 bits địa chỉ và 1 bits RW để xác định hướng truyền dữ liệu, trong trường hợp này RW=’1’ và khi Slave xác định đó là địa chỉ của mình thì nó sẽ trả về 1-bit ACK xác nhận và quá trình truyền dữ liệu diễn ra trong giai đoạn tiếp theo, sau khi slave truyền đủ 8 bits dữ liệu thì Master sẽ xác nhận điều đó bằng cách gửi 1-bit ACK xác nhận, trường hợp sai truyền không đủ dữ liệu thì Master sẽ gửi 1-bit NACK, và kết thúc quá trình truyền.

Link video kết quả: <https://www.youtube.com/watch?v=tM1QKFVfYCo>

4.2 Thực nghiệm

Master với Slave chuẩn

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Checkbox | Testcase | result |
| 1 |  | Trạng thái ready và reset |  |
| 2 |  | Trạng thái start |  |
| 3 |  | Trạng thái truyền sai bit địa chỉ |  |
| 4 |  | Trạng thái truyền thiếu bit dữ liệu |  |
| 5 |  | Master xác nhận truyền Slave 1 và Salve 2 |  |
| 6 |  | Trạng thái nhận sai bit địa chỉ |  |
| 7 |  | Trạng thái nhận thiếu bit dữ liệu |  |
| 8 |  | Master nhận đúng  Slave 1 và Slave 2 |  |

1) Trạng thái ready và reset: ready = 1 và busy = 0 và ngược lại, khi có tín hiệu reset ready và busy ở mức ‘0’

2) Trạng thái Start: SDA chuyển trạng thái từ mức ‘1’ xuống mức 0 trong khi SCL mức ‘1’

3) Trạng thái truyền sai bits địa chỉ: Kết thúc quá trình truyền

4) Trạng thái truyền thiếu bits dữ liệu: Slave xác nhận đúng bits địa chỉ quá trình truyền diễn ra nhưng Master gửi thiếu bits dữ liệu, Slave sẽ gửi tín hiệu NOT ACK và Master sẽ gửi tín hiệu stop và kết thúc quá trình truyền

5) Master xác nhận truyền dữ liệu cho Slave 1 và Slave 2: Master truyền lần lượt dữ liệu cho 2 con Slave có địa chỉ khác nhau, Ban đầu ở trạng thái ready tức lúc này Ready mức ‘1’, Busy mức ‘0’ và (I2C hoạt động ở trạng thái rỗi). Tiếp theo là tín hiệu start tức S-bit. SDA chuyển trạng thái từ mức ‘1’ xuống mức 0 trong khi SCL mức ‘1’, sau tín hiệu start master sẽ gửi 7 bits địa chỉ và 1 bits RW để xác định hướng truyền dữ liệu, trong trường hợp này RW=’0’ và khi Slave xác định đó là địa chỉ của mình thì nó sẽ trả về 1-bit ACK xác nhận và quá trình truyền dữ liệu diễn ra trong giai đoạn tiếp theo, sau khi master truyền đủ 8 bits dữ liệu thì Slave sẽ xác nhận điều đó bằng cách gửi 1-bit ACK xác nhận, tiếp theo sẽ cho 1 tín hiệu reset và Master sẽ truyền dữ liệu đến Slave thứ 2

6) Trạng thái nhận sai bits địa chỉ: Kết thúc quá trình truyền

7) Trạng thái nhận thiếu bits dữ liệu: Slave xác nhận đúng bits địa chỉ quá trình truyền diễn ra nhưng Slave gửi thiếu bits dữ liệu, Master sẽ gửi tín hiệu NOT ACK và Master và kết thúc quá trình truyền

8) Master nhận đúng Slave 1 và Slave 2: Master nhận lần lượt dữ liệu của 2 con Slave có địa chỉ khác nhau, Ban đầu ở trạng thái ready tức lúc này Ready mức ‘1’, Busy mức ‘0’ và (I2C hoạt động ở trạng thái rỗi). Tiếp theo là tín hiệu start tức S-bit. SDA chuyển trạng thái từ mức ‘1’ xuống mức 0 trong khi SCL mức ‘1’, sau tín hiệu start master sẽ gửi 7 bits địa chỉ và 1 bits RW để xác định hướng truyền dữ liệu, trong trường hợp này RW=’1’ và khi Slave xác định đó là địa chỉ của mình thì nó sẽ trả về 1-bit ACK xác nhận và quá trình truyền dữ liệu diễn ra trong giai đoạn tiếp theo, sau khi master nhận đủ 8 bits dữ liệu thì Master sẽ xác nhận điều đó bằng cách gửi 1-bit ACK xác nhận, tiếp theo sẽ cho 1 tín hiệu reset và Slave sẽ nhận dữ liệu từ Slave thứ 2

**Ưu điểm:**

* Các module giao tiếp với nhau với độ ổn định cao.
* Dễ dàng giao tiếp với các module khác qua hai dây.
* Khả năng lỗi bit trên đường truyền thấp.
* **Hạn chế:**
* Chỉ truyền được 1byte data /frame
* Không thể phát hiện khi truyền/nhận sai data.
* Chỉ được ứng dụng trong các ứng dụng có khoảng cách kết nối gần.
* Tốc độ truyền còn hạn chế, chưa được tối ưu tối đa tốc độ.

CHƯƠNG 5: KẾT LUẬN, ỨNG DỤNG VÀ HƯỚNG PHÁT TRIỂN

5.1 Kết luận:

* So với các mục tiêu đề ta thì hệ thông đạt được kết quả như mong muốn. Đáp ứng được tất cả các yêu cầu về mặt thiết kế, phát hiện lỗi truyền…Tuy nhiên so với một hệ thống I2C đạt chuẩn trên các thiết bị hiện nay thì hệ thống vẫn còn nhiều điểm yếu và hạn chế cần khắc phục.
* Với các yêu cầu cao về kết quá, nhóm cũng đã thực hiện kiểm tra trên các thiết bị slaver mẫu trên mạng. Kết quả đạt được như kì vọng.
  1. Hướng phát triển:
* Trong tương lai gần, sẽ thiết kế một số chức năng cho module để trăng cường độ tin cậy trong quá trình truyền
* Phát triển thêm chức năng ngắt khi phát hiện quá trình truyền bị lỗi hay mất bit. Sau đó sẽ yêu cầu master/slaver truyền lại data nhằm tăng cường tốc độ truyền.
* Phát triển và cải thiện tính năng truyền/nhận nhiều byte.

|  |
| --- |
| TÀI LIỆU THAM KHẢO |
| Tiếng việt |
| 1. Nguyễn Quân (2017),” Quy trình cơ bản tạo ra mạch số”.   (https://vimach.net/threads/quy-trinh-co-ban-tao-ra-vi-mach-so.970/)   1. N.V.Hùng - N.N.Lâm - N.V.Phúc - Đ.P.H.Trang (2013), “Giáo Trình Kỹ Thuật Truyền Số Liệu, Đại học Sư Phạm Kỹ Thuật TP.HCM: Nhà Xuất Bản Đại Học Quốc Gia TP.HCM. 2. Nguyễn Quân (2020),” Máy trạng thái hữu hạn FSM”   (https://nguyenquanicd.blogspot.com/2017/08/verilogsystem-verilog-may-trang-thai.html) |
| **Tiếng anh** |
| [4] Integrated circuit design, Wikipedia  (<https://en.wikipedia.org/wiki/Integrated_circuit_design>).  [5] HEMANTH SURKA (July 20,2020),” Integrated Chip (IC) Design Flow”  (https://learnai1.home.blog/2020/07/20/integrated-chip-ic-design-flow/).  [6] YIDA (2018), “UART vs I2C vs SPI - Communication Protocols and Uses”  (<https://www.seeedstudio.com/blog/2019/09/25/uart-vs-i2c-vs-spi-communication-protocols-and-uses/>).  [7] SHIVANI MEHROTRA, NISHA CHARAYA (2015),” Design of I2C Single Master Using Verilog”.  (<https://www.researchgate.net/publication/275771333_Design_of_I2C_Single_Master_Using_Verilog>)  [8] RICHARD HERVEILLE (July 3,2003), “I2C -Master Core Specification”. (<https://opencores.org/websvn/filedetails?repname=i2c&path=%2Fi2c%2Ftrunk%2Fdoc%2Fi2c_specs.pdf>)  [9] NXP SEMICONDUCTORS (October 1,2021), “UM10204 -I2C bus specification and user manual”.  (https://www.pololu.com/file/0J435/UM10204.pdf) |

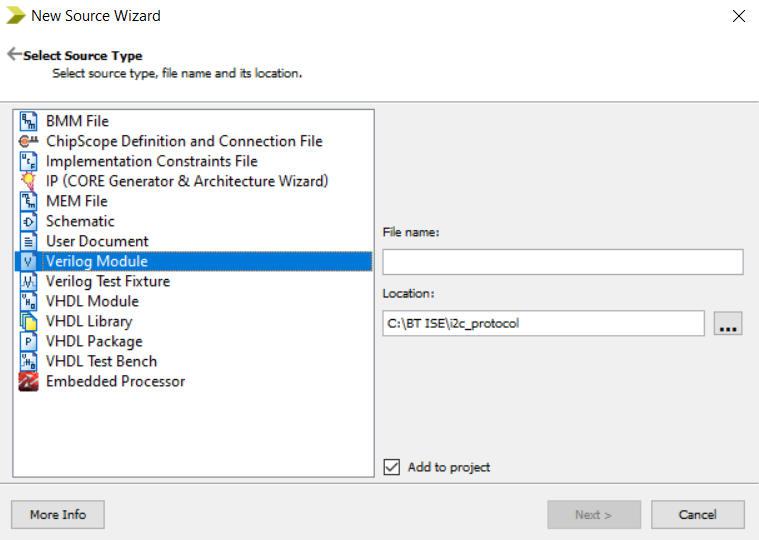
PHỤ LỤC VỀ CHƯƠNG TRÌNH

**Hướng dẫn thiết lập và sử dụng**

Bước 1: Download phần mềm ISE Design Suite 14.7



Bước 2: Vào phần mềm ISE Design Suite 14.7 và tạo các file module cần thiết



Bước 3: Lập trình cho chương trình