HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
CƠ SỞ TẠI THÀNH PHỐ HỒ CHÍ MINH
KHOA KỸ THUẬT ĐIỆN TỬ 2

ĐỒ ÁN
TỐT NGHIỆP ĐẠI HỌC

CHUYÊN NGÀNH: THIẾT KẾ VI MẠCH
HỆ CHÍNH QUY
NIÊN KHÓA: 2021-2026

Đề tài:

THIẾT KẾ VÀ MÔ PHỎNG BỘ ĐIỀU KHIỂN PWM

Sinh viên thực hiện:	**NGUYỄN VIỆT THÀNH**
MSSV	**N21DCDT083**
Sinh viên thực hiện:	**LƯƠNG PHÚC MINH DUY**
MSSV	**N21DCDT018**
Lớp:	**D21CQDTVM01-N**
Giáo viên hướng dẫn:	**NGUYỄN THANH BÌNH**

08/2025
TP.HCM – 2025

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
CƠ SỞ TẠI THÀNH PHỐ HỒ CHÍ MINH
KHOA KỸ THUẬT ĐIỆN TỬ 2

ĐỒ ÁN
TỐT NGHIỆP ĐẠI HỌC

CHUYÊN NGÀNH: THIẾT KẾ VI MẠCH
HỆ CHÍNH QUY
NIÊN KHÓA: 2021-2026

Đề tài:

THIẾT KẾ VÀ MÔ PHỎNG BỘ ĐIỀU KHIỂN PWM

NỘI DUNG:

- CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI
- CHƯƠNG 2: CƠ SỞ LÝ THUYẾT
- CHƯƠNG 3: THIẾT KẾ HỆ THỐNG
- CHƯƠNG 4: MÔ PHỎNG VÀ KIỂM THỬ
- CHƯƠNG 5: TRIỂN KHAI VÀ ĐO ĐẠC TRÊN FPGA
- CHƯƠNG 6: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Sinh viên thực hiện:	**NGUYỄN VIỆT THÀNH**
MSSV	**N21DCDT083**
Sinh viên thực hiện:	**LƯƠNG PHÚC MINH DUY**
MSSV	**N21DCDT018**
Lớp:	**D21CQDTVM01-N**
Giáo viên hướng dẫn:	**NGUYỄN THANH BÌNH**

TÊN ĐỀ TÀI **M ĐT: 14 - xyz**

PHIẾU GIAO NHIỆM VỤ

(Phiếu này các bạn sẽ nhận ở trên văn phòng Khoa, khi nào có cô sẽ thông báo tới nhận)

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

TP. HCM, ngày ……tháng… .năm 2025

Giáo viên

(Ký và ghi rõ họ tên)

NHẬN XÉT CỦA GIÁO VIÊN PHẢN BIỆN

TP. HCM, ngày ……tháng… .năm 2025

Giáo viên

(Ký và ghi rõ họ tên)

LỜI CẢM ƠN

Xin gửi lời cảm ơn chân thành đến thầy Nguyễn Thanh Bình và thầy Phạm Thế Duy, người đã tận tình hướng dẫn và đồng hành cùng nhóm trong suốt quá trình thực hiện dự án I²C–PWM trên FPGA. Sự chỉ bảo chi tiết, những góp ý chuyên môn sâu sắc và sự theo dõi thường xuyên của hai thầy đã giúp nhóm hiểu rõ hơn về mục tiêu đề tài, phương pháp thiết kế, cũng như cách triển khai và tích hợp giao tiếp I²C Slave với hệ thống PWM đa kênh một cách hiệu quả.

Trong quá trình thực hiện dự án, nhóm đã gặp không ít khó khăn về mặt kỹ thuật, quản lý thời gian và phối hợp nhóm, đặc biệt trong các giai đoạn thiết kế logic, kiểm thử module và triển khai thực tế trên kit FPGA. Nhờ sự hướng dẫn kiên nhẫn và hỗ trợ kịp thời của hai thầy, nhóm đã từng bước điều chỉnh phương pháp làm việc, nâng cao kỹ năng thiết kế số, kiểm chứng chức năng và trình bày kết quả một cách khoa học, rõ ràng.

Những kiến thức, kinh nghiệm và tinh thần nghiên cứu nghiêm túc mà thầy truyền đạt là hành trang quý giá đối với nhóm trong quá trình học tập và nghiên cứu sau này.

Lời cảm ơn này thay cho sự tri ân sâu sắc của nhóm đối với sự quan tâm, hỗ trợ và động viên của hai thầy, góp phần quan trọng giúp báo cáo I²C–PWM được hoàn thành một cách tương đối đầy đủ, chỉnh chu và có ý nghĩa thực tiễn.

TP. HCM, ngày ……tháng… .năm 2025
Sinh viên thực hiện
(Ký và ghi rõ họ tên)

MỤC LỤC

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG BIỂU

LỜI MỞ ĐẦU

Trong bối cảnh công nghệ số phát triển mạnh mẽ, việc nắm vững các kiến thức và kỹ năng về thiết kế hệ thống số, giao tiếp ngoại vi và xử lý tín hiệu trên FPGA trở thành yêu cầu quan trọng đối với sinh viên ngành kỹ thuật. Xuất phát từ nhu cầu đó, nhóm đã thực hiện đồ án thiết kế mô-đun I2C–PWM Controller tích hợp giao tiếp I2C, khối giải mã địa chỉ, tầng thanh ghi và khối sinh xung PWM đa kênh.

Cốt lõi của đồ án tập trung vào việc hiện thực hóa các lý thuyết đã được nghiên cứu về điều chế độ rộng xung (PWM). Thay vì chỉ dừng lại ở mức mô phỏng và phân tích dạng sóng trong môi trường lý thuyết, đồ án hướng đến quá trình thiết kế, xây dựng và triển khai hệ thống một cách hoàn chỉnh. Trong đó, mục tiêu mô hình hóa kiến trúc phần cứng, hiện thực thuật toán PWM ở mức RTL, cũng như biểu diễn và kiểm chứng kết quả thông qua các tín hiệu thực tế và hành vi khả kiến được của hệ thống đóng vai trò trọng tâm. Cách tiếp cận này giúp thu hẹp khoảng cách giữa lý thuyết và ứng dụng, đồng thời đánh giá được tính đúng đắn, khả thi và khả năng mở rộng của thiết kế trong các hệ thống số và nhúng thực tế.

Báo cáo này được xây dựng nhằm tổng hợp toàn bộ quá trình thực hiện đồ án, bao gồm các kết quả đạt được, những khó khăn gặp phải, bài học kinh nghiệm và các hướng phát triển tiếp theo. Mục tiêu của báo cáo là cung cấp một tài liệu tham khảo có giá trị cho sinh viên trong quá trình nghiên cứu và triển khai các dự án liên quan đến thiết kế phần cứng số, giao tiếp ngoại vi và hệ thống điều chế PWM trên nền tảng FPGA.

Thông qua việc trình bày xuyên suốt các giai đoạn từ thiết kế RTL, xây dựng và xác minh testbench, đến tổng hợp và triển khai trên phần cứng thực tế, báo cáo góp phần làm rõ mối liên hệ giữa lý thuyết và ứng dụng. Đồng thời, các kinh nghiệm rút ra từ quá trình thực hiện được kỳ vọng sẽ hỗ trợ người đọc nâng cao hiệu quả thiết kế và mở rộng hệ thống trong các nghiên cứu và ứng dụng tiếp theo.

Đồ án gồm có 6 chương:

Chương 1: Tổng quan đề tài

Chương 2: Cơ sở lý thuyết

Chương 3: Thiết kế hệ thống

Chương 4: Mô phỏng và kiểm thử

Chương 5: Triển khai và đo đạc trên fpga

Chương 6: Kết luận và hướng phát triển

CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI

1.1. Giới thiệu

Dựa trên cơ sở lý thuyết về kỹ thuật điều chế độ rộng xung (PWM – Pulse Width Modulation), bộ PWM hoàn chỉnh cần đảm bảo khả năng tạo, điều chỉnh và kiểm soát các tín hiệu PWM với các tham số đặc trưng như tần số hoạt động, độ rộng xung (duty cycle) và thời gian chết (deadtime). Bên cạnh đó, khối PWM cần có khả năng giao tiếp với hệ thống điều khiển bên bên ngoài thông qua giao thức I²C (Inter-Integrated Circuit), một chuẩn truyền thông nối tiếp phổ biến, cho phép truyền dữ liệu hai chiều giữa bộ điều khiển (master) và khối PWM (slave) để cho phép người dùng cấu hình các thông số hoạt động theo thời gian thực.

Lý do lựa chọn đề tài xuất phát từ mong muốn làm quen với quy trình thiết kế một khối IP hoàn chỉnh, từ giai đoạn hình thành ý tưởng, phân tích chức năng, thiết kế kiến trúc khối, lập trình mô tả bằng ngôn ngữ Verilog HDL, cho đến mô phỏng và kiểm chứng hoạt động trên công cụ FPGA. Đề tài không chỉ giúp hiện thực hóa các kiến thức lý thuyết đã học về mạch số, vi mạch và thiết kế phần cứng, mà còn rèn luyện tư duy hệ thống, khả năng xử lý vấn đề và quy trình triển khai một dự án kỹ thuật thực tế.

1.2. Mục tiêu đề tài

Mục tiêu chính của đề tài tập trung vào việc thiết kế bộ điều khiển PWM bao gồm 3 nhiện vụ chính sau:

- Thiết kế khối điều khiển PWM với chức năng hoàn chỉnh.
- Thiết kế khối ngoại vi I²C Slave với chức năng đọc ghi.
- Kiểm tra và đánh giá chức năng của 2 khối trên thông qua mô phỏng và phần cứng sử dụng kit FPGA.

1.3. Phạm vi và giới hạn của đề tài

Phạm vi nghiên cứu giới hạn trong việc thiết kế bộ PWM ở mức RTL (Register Transfer Level) sử dụng ngôn ngữ Verilog HDL.

Giới hạn: Đề tài không đi sâu vào các giai đoạn chế tạo vật lý vi mạch (như layout, mask, fabrication) mà tập trung vào thiết kế logic, mô phỏng chức năng và tổng hợp trên FPGA.

1.4. Phương pháp thực hiện

Quá trình thiết kế được thực hiện theo Quy trình Thiết kế Vi Mạch Số (TKVM), bao gồm các hoạt động chính sau:

- Quy trình thiết kế tổng thể:
    + Phân tích và xác định yêu cầu chức năng của khối PWM và khối I²C.
    + Thiết kế kiến trúc Mô-đun (Prescaler, Counter, Comparator, Deadtime, Register Interface) và lập trình mô tả bằng Verilog HDL.
    + Mô phỏng và kiểm thử chức năng cho từng mô-đun độc lập và toàn hệ thống.

+ Tổng hợp (Synthesis) và triển khai trên FPGA để kiểm chứng khả năng thực thi thực tế.
+ Đánh giá kết quả đầu ra thông qua phân tích dạng sóng (waveform) và các tham số hoạt động (tần số, duty cycle, deadtime).

- Các Bước Triển Khai Cụ Thể
  + Xây dựng ý tưởng và kiến trúc tổng thể.
  + Thiết kế và lập trình Verilog HDL cho từng mô-đun riêng biệt.
  + Kiểm thử độc lập (Unit Test): Kiểm tra chức năng của từng mô-đun nhỏ (ví dụ: bộ đếm, bộ so sánh) để phát hiện lỗi logic cục bộ.
  + Kiểm thử hệ thống (System Test): Kết hợp toàn bộ mô-đun (integration) và kiểm tra tổng thể để xác nhận sự tương thích và hoạt động chính xác của toàn khối PWM.
  + Sử dụng mô phỏng hành vi (behavioral simulation) và mô phỏng thời gian (timing simulation) để kiểm tra tính chính xác về mặt thời gian của tín hiệu.

CHƯƠNG 2: CƠ SỞ LÍ THUYẾT

2.1. Tổng quan về điều chế độ rộng xung

2.1.1. Tần số và chu kỳ của tín hiệu số

Khái niệm tín hiệu số (Digital Signal) trong hệ thống điều khiển thường được thể hiện dưới dạng tín hiệu vuông (xung), chỉ có hai mức (cao/ON và thấp/OFF). Hai tham số nền tảng của tín hiệu số tuần hoàn là Tần số và Chu kỳ.

Chu kỳ (Period, T): Là khoảng thời gian cần thiết để tín hiệu lặp lại một mẫu hoàn chỉnh. Đơn vị thường là giây (s) hoặc mili giây (ms).

Tần số (Frequency, f): Là số lần tín hiệu lặp lại mẫu của nó trong một đơn vị thời gian (thường là 1 giây). Đơn vị là Hertz (Hz).

Mối quan hệ: Tần số và chu kỳ là hai đại lượng nghịch đảo của nhau:

$$f\ [Hz] = \frac{1}{T}\ hoặc\ T\ [s] = \frac{1}{f} \quad (2.1)$$

Hình 2.1: Công thức mối liên hệ giữa tần số (f) và chu kỳ (T)

Trong các hệ thống điều khiển bằng vi điều khiển, tần số PWM là cố định, được thiết lập dựa trên tần số xung nhịp hệ thống (System Clock) và bộ chia (Prescaler).

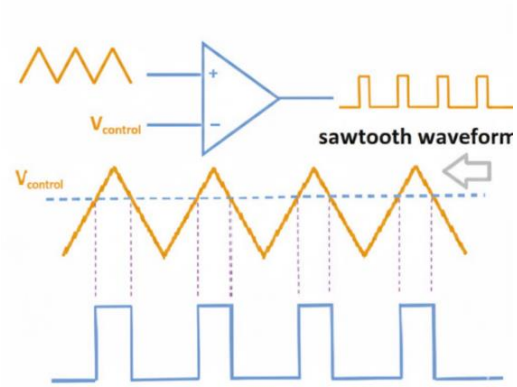
2.1.2. Nguyên lý hoạt động của điều chế độ rộng xung

Điều chế độ rộng xung (Pulse Width Modulation - PWM) là một kỹ thuật được sử dụng để điều khiển công suất cung cấp cho tải bằng cách thay đổi thời gian bật (ON) của tín hiệu trong một chu kỳ (Period) cố định.

Nguyên lý cơ bản nhất của PWM là phương pháp so sánh trực tiếp (Direct Comparison):

- Sóng mang (Carrier Wave): Một tín hiệu tuần hoàn có biên độ và tần số cố định, thường là sóng tam giác (Triangle) hoặc sóng răng cưa (Sawtooth). Đây chính là đại diện cho chu kỳ PWM (T_{PWM}).
- Tín hiệu điều khiển (Control Signal): Là điện áp một chiều (DC) hoặc tín hiệu tham chiếu có điện thay đổi. Đại lượng này đại diện cho giá trị mong muốn của đầu ra.
- Bộ so sánh (Comparator): Có chức năng so sánh tín hiệu điều khiển với sóng mang.
    + Khi tín hiệu điều khiển > sóng mang, đầu ra của bộ so sánh ở mức cao (on).
    + Khi tín hiệu điều khiển < sóng mang, đầu ra của bộ so sánh ở mức thấp (off).

Kết quả là một chuỗi xung vuông có tần số cố định, nhưng độ rộng của xung ở mức on thay đổi tỷ lệ thuận với biên độ của tín hiệu điều khiển.

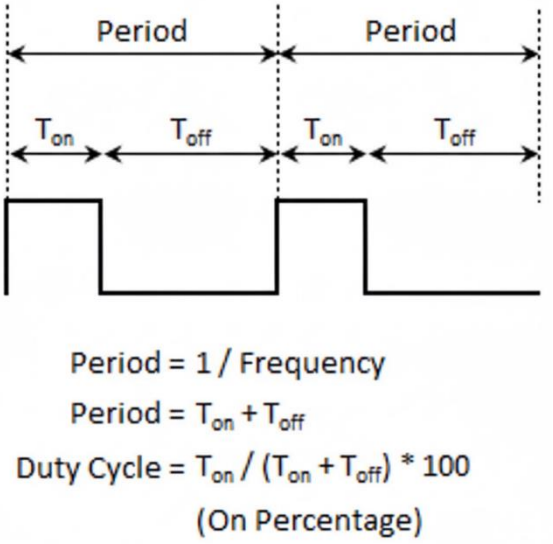


Hình 2.2 Mô tả nguyên lý cơ bản của điều chế độ rộng xung

2.1.3. Tham số độ rộng xung trong điều chế độ rộng xung

Độ rộng xung (Duty Cycle) là tỷ lệ phần trăm giữa thời gian tín hiệu ở mức cao (T_{ON}) so với tổng chu kỳ (T) của tín hiệu đó. Độ rộng xung là tham số chính để định lượng mức độ điều khiển của tín hiệu PWM.

$$DutyCycle[\%] = \frac{T_{ON}}{T_{ON}+T_{OFF}} \times 100 = \frac{T_{ON}}{Period(T)} \times 100[\%] \quad (2.2)$$



Hình 2.3 Sơ đồ biểu diễn tín hiệu PWM và các thông số liên quan

Trong đó:

T_{ON} : Thời gian xung ở mức cao (ON-time).

T : Chu kỳ xung.

Tín hiệu PWM có bản chất là tín hiệu bật/tắt, nhưng khi được lọc qua tải có tính quán tính (như cuộn dây động cơ, tụ điện lọc...), nó sẽ cho ra một giá trị trung bình (Average Value). Giá trị trung bình này tỷ lệ thuận với Duty Cycle. Ví dụ, nếu điện áp cấp là V_{CC}:

$$V_{TB} = D * V_{CC} \quad (2.3)$$

Điều chỉnh Duty Cycle từ 0% đến 100% cho phép điều chỉnh giá trị trung bình từ 0 đến V_{CC} một cách hiệu quả.

2.1.4. Độ phân giải trong điều chế độ rộng xung

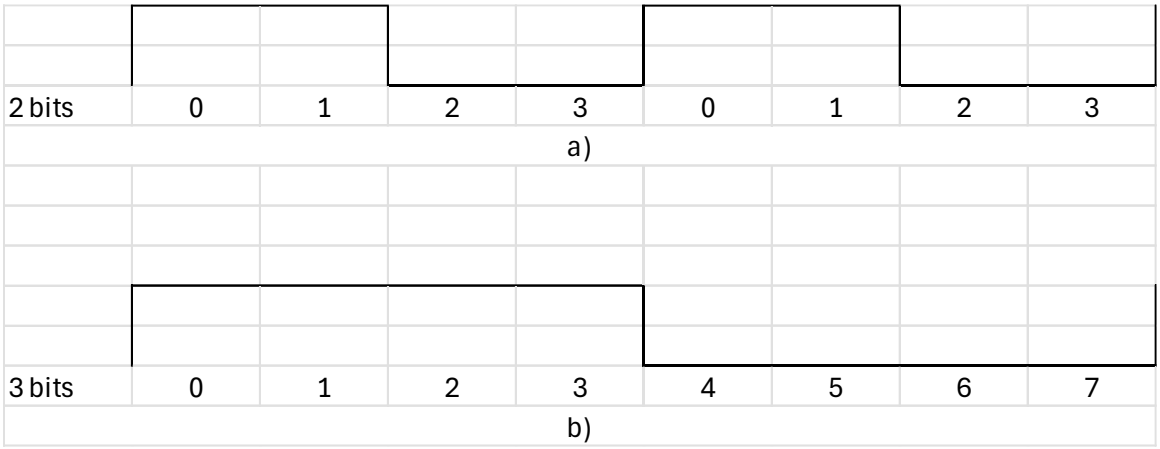
Độ phân giải (Resolution) của PWM là khả năng điều chỉnh Duty Cycle chi tiết nhất mà hệ thống có thể thực hiện được. Nó quyết định số lượng bước mà ta có thể thay đổi điện áp/công suất đầu ra.

Trong các hệ thống vi điều khiển, PWM được tạo ra bằng cách đếm đến một giá trị đỉnh và so sánh với giá trị điều khiển.

Xác định theo số bit: Nếu bộ đếm có n bit, thì tổng số bước đếm có thể đạt được là 2^n bước.

Ví dụ:

- PWM 8-bit: $2^8 = 256$ bước. Độ phân giải là $\frac{1}{256} * 100 \approx 0{,}39\%/bước$.
- PWM 10-bit:$2^{10} = 1024$ bước. Độ phân giải là $\frac{1}{1024} * 100 \approx 0{,}098\%/bước$.



Hình 2.4 So sánh độ phân giải của 2 tín hiệu PWM có cùng độ rộng xung

(a) Tín hiệu PWM 2-bit với 4 mức điều khiển (0-3).

(b) Tín hiệu PWM 3-bit với 8 mức điều khiển (0-7).

Độ phân giải càng cao (số bit càng lớn) thì khả năng điều chỉnh Duty Cycle càng mịn và chính xác hơn, nhưng thường đi kèm với giảm tần số PWM (nếu giữ nguyên tần số xung nhịp).

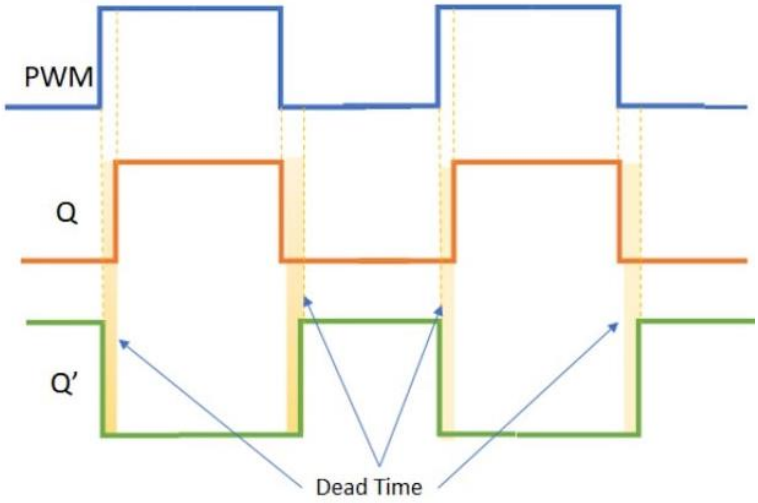
2.1.5. Thời gian chết trong quá trình chuyển mạch công suất

Deadtime (thời gian chết) là khoảng thời gian trễ nhỏ được chèn vào giữa hai xung điều khiển của cặp transistor công suất (thường là MOSFET hoặc IGBT) trong cùng một nhánh cầu H. Trong khoảng thời gian này, cả hai transistor đều ở trạng thái tắt, không dẫn dòng.

Trong thực tế, các linh kiện bán dẫn không thể tắt/mở ngay lập tức mà có thời gian trễ chuyển mạch (switching delay).

Nếu hai transistor trên cùng một nhánh dẫn đồng thời (do trễ tắt chưa kịp), sẽ xảy ra hiện tượng ngắn mạch xuyên thẳng (shoot-through), gây hỏng mạch công suất.

Để tránh điều này, bộ điều khiển PWM sẽ tự động chèn một khoảng trễ (deadtime) giữa xung tắt của transistor này và xung bật của transistor kia.



Hình 2.5 Mô tả tín hiệu PWM gốc với 2 tín hiệu đầu ra là Q và Q'

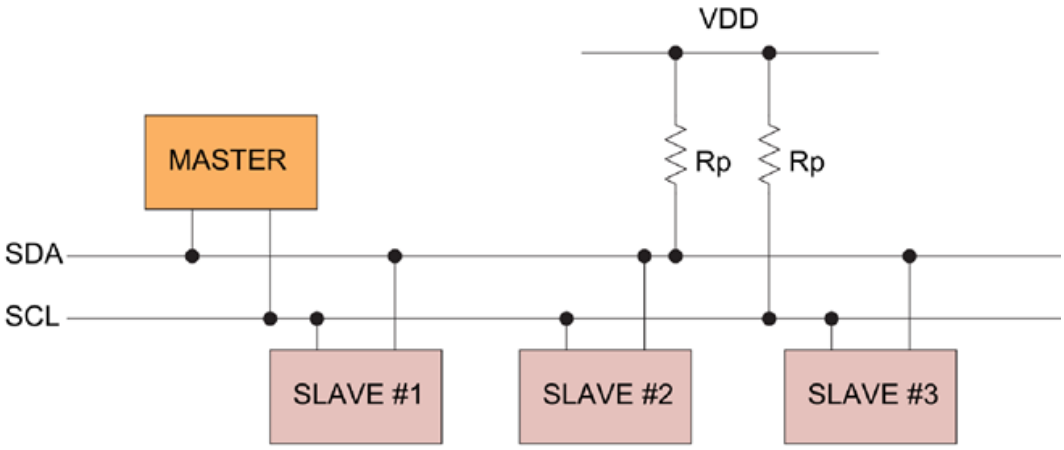
2.2. Tổng quan về giao thức I²C

2.2.1. Giới thiệu và đặc điểm kỹ thuật

Giao thức I²C (Inter-Integrated Circuit) là một giao thức truyền thông nối tiếp, đồng bộ được phát triển bởi Philips (nay là NXP Semiconductors) vào những năm 1980. Mục đích chính là tạo ra một bus truyền thông hiệu quả, đơn giản để giao tiếp giữa các thiết bị trên cùng một bảng mạch in.

Các đặc điểm kỹ thuật nổi bật của I²C bao gồm:

- Bus hai dây: Chỉ sử dụng hai đường tín hiệu là SCL và SDA. Điều này giúp giảm thiểu số lượng chân (pin) cần thiết trên vi điều khiển và các thiết bị ngoại vi.
- Truyền thông nối tiếp, đồng bộ: Dữ liệu được truyền từng bit một và được đồng bộ hóa bằng xung nhịp SCL.
- Cấu trúc đa chủ - đa tớ (Multi-Master, Multi-Slave): Cho phép nhiều thiết bị đóng vai trò là chủ (Master) khởi tạo truyền thông và nhiều thiết bị đóng vai trò là tớ (Slave) phản hồi.
- Tốc độ truyền: I²C hỗ trợ nhiều chế độ tốc độ, phổ biến nhất là Standard-mode (100 kbps) và Fast-mode (400 kbps), cùng với các chế độ tốc độ cao hơn như Fast-mode Plus (1 Mbps) và High-speed mode (3.4 Mbps).



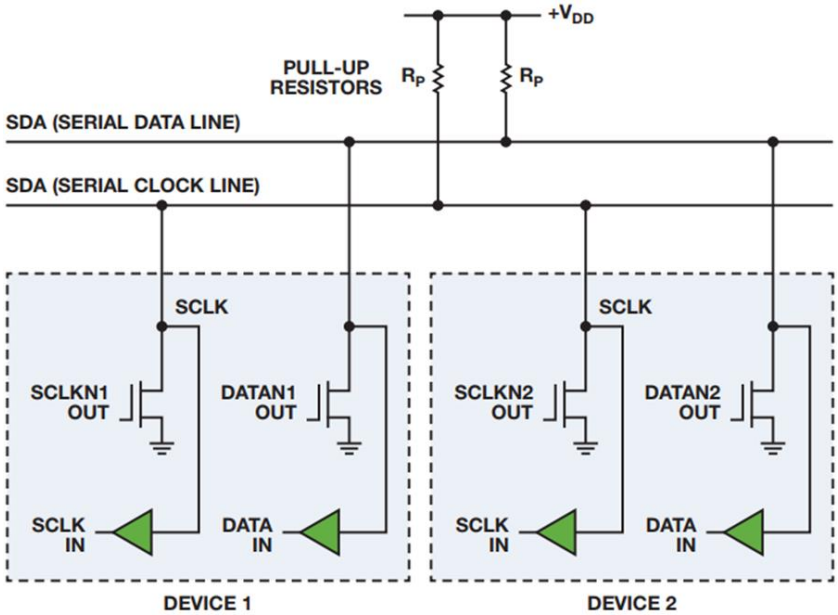
Hình 2.6 Mạng I²C cơ bản với 1 Master và nhiều Slave

2.2.2. Cấu trúc phần cứng và tín hiệu bus

Giao thức I²C hoạt động dựa trên hai đường tín hiệu mở (Open-Drain) cần sử dụng điện trở kéo lên:

- SCL (Serial Clock - xung nhịp nối tiếp): Đường truyền xung nhịp do thiết bị Master phát ra để đồng bộ hóa quá trình truyền dữ liệu.
- SDA (Serial Data - dữ liệu nối tiếp): Đường truyền dữ liệu hai chiều giữa Master và Slave.

Điện trở kéo lên (Pull-up Resistor): Đây là thành phần thiết yếu. Do cả SCL và SDA đều là các đường dây mở (Open-Drain), chúng chỉ có thể kéo xuống mức thấp (GND). Các điện trở này sẽ kéo mức điện áp của bus lên mức cao V_{CC} khi không có thiết bị nào kéo xuống.

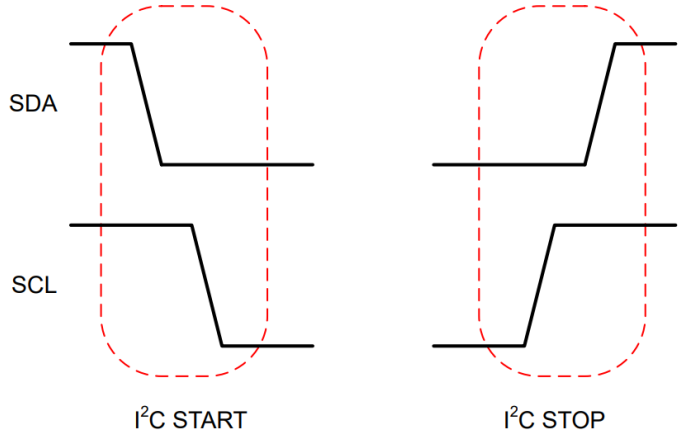


Hình 2.7 Cấu trúc phần cứng của thiết bị I²C kết nối với bus I²C

2.2.3. Nguyên lý hoạt động và khung giao tiếp dữ liệu

Quá trình giao tiếp trên bus I²C được kiểm soát bởi Master thông qua các trạng thái tín hiệu đặc biệt:

- Trạng thái bắt đầu (Start Condition): Master khởi tạo giao tiếp bằng cách chuyển tín hiệu SDA từ mức cao xuống mức thấp trong khi SCL vẫn ở mức cao.
- Trạng thái kết thúc (Stop Condition): Master kết thúc giao tiếp bằng cách chuyển tín hiệu SDA từ mức thấp lên mức cao trong khi SCL vẫn ở mức cao.

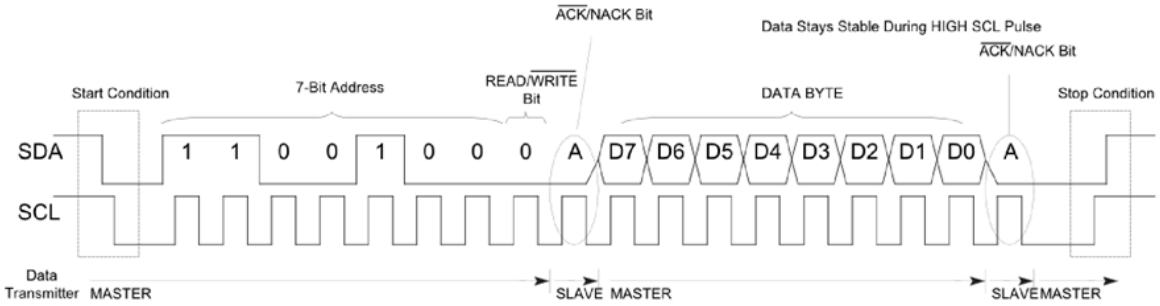


Hình 2.8 Điều kiện START và STOP của bus I²C

- Cơ chế ACK/NACK (Acknowledge/Not Acknowledge) diễn ra sau mỗi byte dữ liệu được truyền. Sau khi thiết bị nhận (Slave hoặc Master) được 8 bit dữ liệu sẽ phản hồi bằng cách kéo đường SDA xuống mức thấp trong một chu kỳ xung nhịp (ACK - xác nhận) hoặc giữ SDA ở mức cao (NACK - không xác nhận).Có năm điều kiện chính khiến tín hiệu NACK được tạo ra:
  + Không có thiết bị nhận: Trên bus không có Slave nào phản hồi với địa chỉ được truyền đi.
  + Thiết bị nhận chưa sẵn sàng: Thiết bị nhận đang thực hiện một chức năng thời gian thực khác và chưa sẵn sàng để bắt đầu hoặc tiếp tục giao tiếp.
  + Lỗi dữ liệu/lệnh: Trong quá trình truyền, bên nhận nhận được dữ liệu hoặc lệnh mà nó không hiểu.
  + Bộ đệm đầy: Trong quá trình truyền, bên nhận không thể nhận thêm byte dữ liệu nào nữa (ví dụ: bộ đệm đã đầy).
  + Kết thúc truyền dữ liệu (Controller-Receiver): Khi Controller đóng vai trò là bên nhận, nó phải chủ động gửi NACK để báo hiệu cho bên phát (Target Transmitter) rằng Controller muốn kết thúc phiên truyền dữ liệu.

2.2.4. Quá trình ghi dữ liệu của giao thức I²C

Giao thức I²C được chia thành các khung (frames). Giao tiếp bắt đầu bằng việc thiết bị điều khiển (Master) gửi một điều kiện START sau đó là địa chỉ của Slave (slave address) và Bit đọc/ghi. Khung địa chỉ này được theo sau bởi một hoặc nhiều khung dữ liệu (data frame), mỗi khung chứa một byte. Mỗi khung cũng có một bit xác nhận (acknowledge bit) để báo cho thiết bị điều khiển biết rằng thiết bị đích hoặc thiết bị điều khiển đã nhận được thông tin.



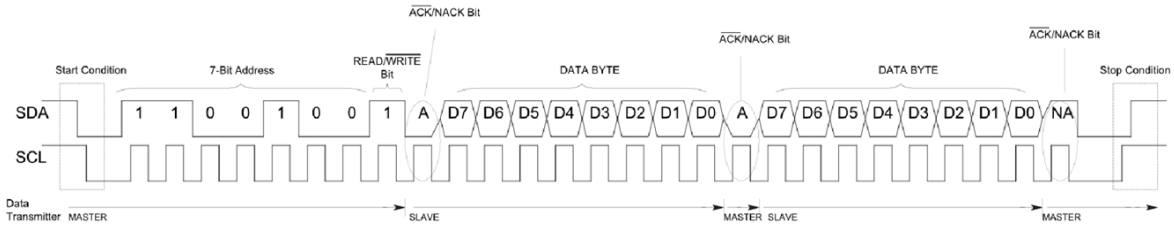
Hình 2.9 Quá trình MASTER ghi 1 Byte dữ liệu vào SLAVE

Một thao tác ghi dữ liệu từ Master đến Slave trên bus I²C sẽ diễn ra theo trình tự các bước sau:

- Trạng thái bắt đầu (Start Condition): Master khởi tạo giao tiếp bằng cách phát điều kiện START (SDA chuyển từ mức cao xuống mức thấp khi SCL ở mức cao).
- Khung địa chỉ và lệnh ghi: Master gửi Byte địa chỉ (Address Frame) gồm:
  + 7 bit/10 bit địa chỉ của thiết bị Slave muốn giao tiếp.
  + 1 bit R/W được đặt là 0 (Chỉ thị Master muốn ghi dữ liệu).
- Phản hồi địa chỉ (ACK/NACK):
  + Slave được định địa chỉ sẽ phản hồi ACK (kéo SDA xuống thấp) nếu đúng địa chỉ và sẵn sàng giao tiếp.
  + Nếu Slave không phản hồi (NACK), Master sẽ kết thúc giao tiếp bằng điều kiện STOP.
- Truyền dữ liệu: Master tiến hành truyền một byte dữ liệu.
- Phản hồi dữ liệu (ACK/NACK):
  + Sau mỗi byte dữ liệu, Slave được định địa chỉ sẽ phản hồi ACK nếu byte được nhận thành công.
  + Nếu Slave phản hồi NACK (ví dụ: bộ đệm đầy), Master phải kết thúc giao tiếp.
- Tiếp tục hoặc kết thúc:
  + Tiếp tục: Nếu Slave phản hồi ACK và Master muốn truyền thêm dữ liệu, Master quay lại bước 4 (Truyền Dữ liệu).
  + Kết thúc: Khi Master không muốn truyền dữ liệu nữa hoặc Slave phản hồi NACK thì Master sẽ phát điều kiện STOP.

- Trạng thái kết thúc (Stop Condition): Master kết thúc giao tiếp bằng cách phát điều kiện STOP (SDA chuyển từ mức thấp lên mức cao khi SCL ở mức cao), giải phóng bus.

2.2.5. Quá trình đọc dữ liệu của giao thức I²C



Hình 2.10 Quá trình MASTER đọc 1 Byte dữ liệu từ SLAVE

Tương tự như thao tác ghi, giao tiếp bắt đầu bằng khung địa chỉ. Điểm khác biệt nằm ở bit R/W và vai trò của ACK/NACK trong khung dữ liệu.

- Trạng thái bắt đầu (Start Condition): Master khởi tạo giao tiếp bằng cách phát điều kiện START (SDA chuyển từ mức cao xuống mức thấp khi SCL ở mức cao).
- Khung địa chỉ và lệnh đọc: Master gửi Byte Địa chỉ (Address Frame) gồm:
  + 7 bit/10 bit địa chỉ của thiết bị Slave muốn giao tiếp.
  + 1 bit R/W được đặt là 1 (chỉ thị Master muốn đọc dữ liệu).
- Phản hồi địa chỉ (ACK/NACK)
  + Slave được định địa chỉ sẽ phản hồi ACK (kéo SDA xuống thấp) nếu đúng địa chỉ và sẵn sàng phát dữ liệu.
  + Nếu Slave không phản hồi (NACK), Master sẽ kết thúc giao tiếp bằng điều kiện STOP.
- Truyền Dữ liệu
  + Slave chuyển vai trò thành bên phát (Transmitter) và tiến hành truyền một byte dữ liệu.
- Phản hồi dữ liệu (ACK/NACK) sau mỗi byte dữ liệu, Master (lúc này là bên nhận) sẽ phản hồi để điều khiển quá trình đọc:
  + ACK (tiếp tục đọc): Nếu Master muốn đọc thêm byte tiếp theo sẽ phản hồi ACK (kéo SDA xuống thấp).
  + NACK (ngừng đọc): Nếu Master đã nhận đủ dữ liệu và muốn kết thúc phiên đọc sẽ phản hồi NACK (giữ SDA ở mức cao).
- Tiếp tục hoặc kết thúc
  + Tiếp tục: Nếu Master phản hồi ACK, Slave sẽ phát byte dữ liệu tiếp theo (quay lại bước 4).
  + Kết thúc: Nếu Master phản hồi NACK, Master sẽ phát điều kiện STOP để giải phóng bus.
- Trạng thái kết thúc (Stop Condition) Master kết thúc giao tiếp bằng cách phát điều kiện STOP (SDA chuyển từ mức thấp lên mức cao khi SCL ở mức cao).

2.3. Môi trường và công cụ

2.3.1. Môi trường phát triển

Ngôn ngữ sử dụng: Sử dụng Verilog HDL (Hardware Description Language) cho toàn bộ dự án, bao gồm cả mã thiết kế RTL (Register-Transfer Level) và các Testbench mô phỏng.

IDE soạn thảo:

- Visual Studio Code (VS Code): Là trình soạn thảo chính, được cài đặt với các extension chuyên biệt để hỗ trợ cú pháp Verilog (syntax highlighting), tự động hoàn thành (auto-complete).
- EDA Playground (Online Simulation): Được sử dụng như một nền tảng đồng bộ và chia sẻ code trực tuyến, giúp các thành viên trong nhóm dễ dàng chia sẻ, trao đổi, đồng bộ và chạy thử nghiệm nhanh.

2.3.2. Công cụ biên dịch, mô phỏng và tổng hợp

Để đảm bảo chất lượng và khả năng triển khai của thiết kế, nhóm sử dụng một bộ công cụ chuyên dụng cho từng giai đoạn:

Giai đoạn	Công cụ	Mục đích sử dụng
Mới bắt đầu	EDA Playground	Kiểm tra nhanh chức năng của từng module nhỏ hoặc testbench đơn giản.
Kiểm tra chất lượng	Verilator	Kiểm tra Lint và Rule-Check (kiểm tra quy tắc và lỗi cú pháp nâng cao) sau khi tích hợp module. Phát hiện các lỗi nghiêm trọng như: unsynthesizable (không tổng hợp được), cảnh báo width mismatch, tín hiệu không được gán, latch (chốt), và kiểm tra coding style.
Tổng hợp module	Icarus Verilog	Sử dụng cho giai đoạn kiểm thử module-level do tốc độ nhanh. Thực hiện kiểm thử chức năng ở cấp độ hệ thống (system-level test). Hỗ trợ tạo file sóng .vcd để phân tích waveform.
Mô phỏng và coverage	QuestaSim	Dùng cho các test case phức tạp và để đánh giá Functional Coverage và Code Coverage, đảm bảo mọi kịch bản và dòng code đều được kiểm tra.
Tổng hợp và triển khai trên FPGA	Gowin IDE	Thực hiện các bước Tổng hợp (Synthesis), Place and Route, và tạo file bitstream để nạp và triển khai thiết kế lên chip FPGA Tang Nano 20K.

Bảng 2.1 Bảng công cụ sử dụng theo giai đoạn và mục đích

2.3.3. Công cụ phân tích dạng sóng

GTKWave: Công cụ phân tích waveform từ các file .vcd, cho phép quan sát tín hiệu, đo thời điểm, đánh dấu các sự kiện quan trọng và xuất ảnh waveform phục vụ báo cáo kỹ thuật.

PulseView: Dùng để quan sát tín hiệu đo được từ logic analyzer trong quá trình chạy thử trên FPGA, hỗ trợ kiểm tra hoạt động thực tế của hệ thống, chẳng hạn như xem dạng sóng PWM hoặc kiểm tra tín hiệu giao tiếp với MCU.

2.3.4. Công cụ và nền tảng phần cứng triển khai

Đề tài sử dụng Kit Phát triển Tang Nano 20K (của Sipeed) tích hợp FPGA GW2AR-LV18QN88C6/I5 (dòng Arora V của Gowin Semiconductor) làm nền tảng kiểm chứng phần cứng.

Đặc điểm chính của Tang Nano 20K:

- Cấu trúc Logic: Gồm 20.736 Logic units (LUT4), 15.552 Flip-Flop(FF), cho phép triển khai các thiết kế logic có độ phức tạp cao hơn.
- Bộ nhớ: 468 Kb Block SRAM, và tích hợp DRAM 64 Mbit ngoài (8MB).
- Khối xử lý: 16 Khối nhân DSP (DSP Blocks) và 2 PLL nội, hỗ trợ các phép tính số học tốc độ cao và tạo xung nhịp linh hoạt.
- Tích hợp thạch anh 27MHz được sử dụng làm nguồn xung nhịp chính cho hệ thống.
- Giao tiếp ngoại vi: Tích hợp USB-C, Cổng MIPI CSI (Camera), Cổng MIPI DSI (Màn hình), GPIO mở rộng, hỗ trợ UART, SPI, I^2C.
- Ưu điểm: Hiệu năng mạnh mẽ hơn so với các kit nhỏ, chi phí hợp lý, tích hợp sẵn mạch nạp cấu hình qua USB, hỗ trợ quan sát trực quan tín hiệu thông qua đèn LED/nút nhấn.

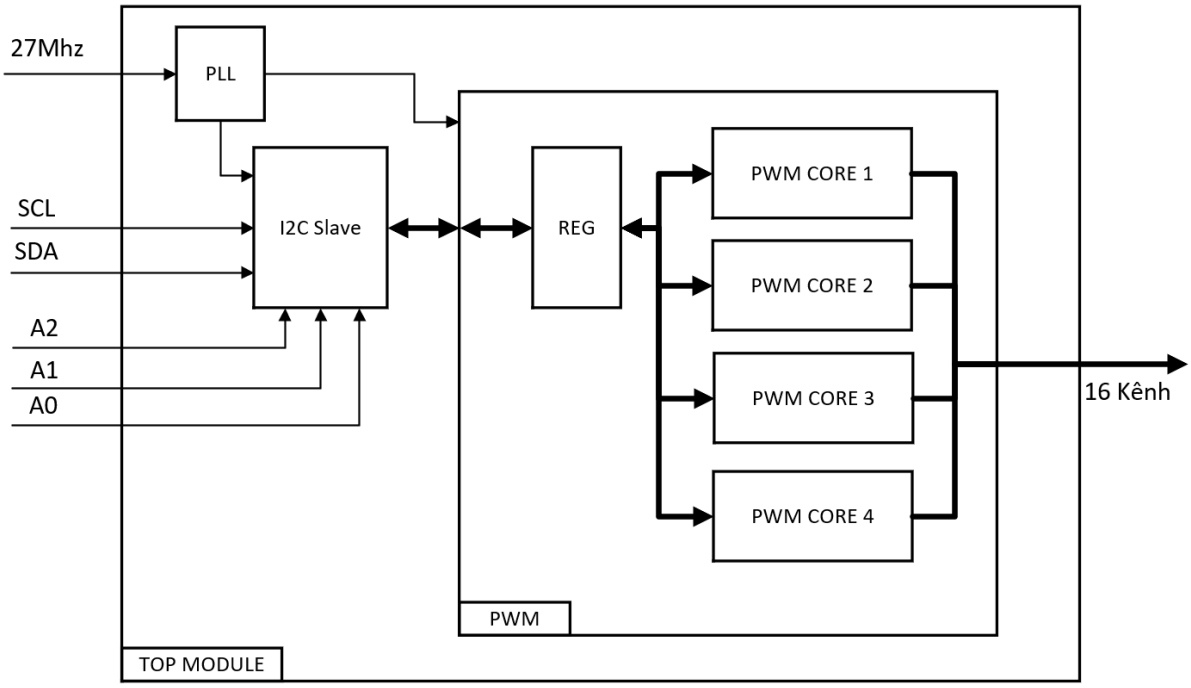
Kit Tang Nano 20K cung cấp hiệu năng tốt, đủ mạnh để triển khai các khối IP phức tạp như bộ PWM tích hợp giao tiếp I^2C. Kết hợp với các công cụ biên dịch, mô phỏng, tổng hợp và phân tích dạng sóng đã trình bày ở trên, đây là lựa chọn tối ưu để hiện thực hóa và kiểm chứng thiết kế IP PWM ở mức RTL, đồng thời giúp sinh viên tiếp cận quy trình thiết kế và triển khai FPGA trong thực tế.

CHƯƠNG 3: THIẾT KẾ HỆ THỐNG

3.1. Yêu cầu và sơ đồ khối tổng quát của hệ thống

Thiết kế hệ thống điều khiển PWM bao gồm các yêu cầu sau:

- Tín hiệu đầu ra PWM
  - + Hỗ trợ 16 đầu ra PWM
  - + Hỗ trợ đầu ra: Hỗ trợ tổng cộng 16 kênh đầu ra PWM, được điều khiển bởi 4 Lõi PWM.
  - + Cho phép cấu hình tần số: Tần số được xác định bởi hai thanh ghi 16-bit: PSC_PRELOAD (Bộ chia tần số) và ARR_PRELOAD (Chu kỳ/Độ phân giải).
  - + Cho phép cấu hình độ rộng xung: Độ rộng xung (Duty Cycle) được kiểm soát bởi hai thanh ghi 16-bit: CMP_START và CMP_END. Điều này cho phép tạo ra các chế độ xung nâng cao (ví dụ: đếm trung tâm).
  - + Cho phép cấu hình Dead Time: Hỗ trợ chèn thời gian chết (*Dead Time*) thông qua giá trị DTG_PRELOAD 8-bit cho mỗi cặp kênh, tạo ra xung chính và xung bù.
  - + Cho phép cấu hình loại xung đầu ra: Chế độ hoạt động (Mode) và phân cực (Polarity) của đầu ra được cấu hình thông qua thanh ghi CFG_REG 8-bit.
  - + Cho phép bật/tắt: Cho phép bật/tắt bộ đếm của từng Core thông qua các bit riêng biệt trong thanh ghi CEN_REG (tại địa chỉ 0x00).
- Giao tiếp I^2C:
  - + Phân bổ địa chỉ: Địa chỉ Slave 7-bit được cấu hình động bằng cách sử dụng 3 chân vật lý đầu vào (A0, A1, A2).
  - + Giao thức đọc/ghi: Hệ thống hỗ trợ đọc/ghi dữ liệu qua các thanh ghi nội bộ theo cấu trúc: 1 Byte địa chỉ thanh ghi và 2 Byte dữ liệu.
  - + Hỗ trợ tốc độ: Được thiết kế để hỗ trợ tốc độ Fast Mode Plus (Fm+) (lên đến 1 MHz)
  - + Có bộ lọc tín hiệu đầu vào: các tín hiệu I^2C được xử lý qua bộ lọc đầu vào (Input Filter) để đảm bảo độ tin cậy.
  - + Kết nối Logic: Khối Frame Bridge chịu trách nhiệm ánh xạ luồng dữ liệu I^2C liên tục sang các thao tác truy cập thanh ghi rời rạc.



Hình 3.1 Sơ đồ khối tổng thể của toàn bộ hệ thống điều khiển PWM

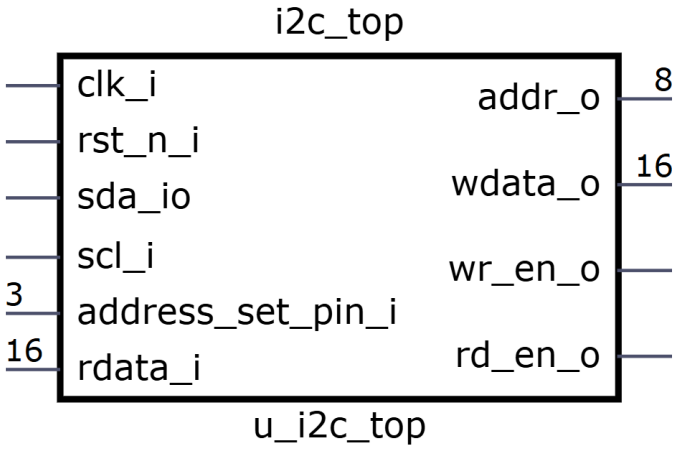
Chức năng của từng khối trong thiết kế là:

- Khối PLL (Phase-Locked Loop): Khối này dùng để tạo và ổn định tần số clock để cung cấp cho khối PWM và I²C Slave.
- Khối I²C Slave: Khối này dùng để giao tiếp và nhận lệnh/dữ liệu từ một thiết bị chủ (Master) bên ngoài thông qua giao thức I²C.
- Khối PWM module: Đây là khối chính để tạo ra tín hiệu điều chế độ rộng xung. Khối này bao gồm các khối nhỏ sau:
    + REG (Register): Khối này dùng để lưu trữ các tham số cấu hình cho các PWM CORE.
    + PWM CORE: Khối này dùng để tạo ra tín hiệu PWM dựa trên các tham số nhận được từ khối REG và clock từ khối PLL. Mỗi core có thể điều khiển 2 cặp kênh đầu ra.

3.2. Thiết kế khối I²C Slave

3.2.1. Sơ đồ khối tổng quát của khối I²C Slave

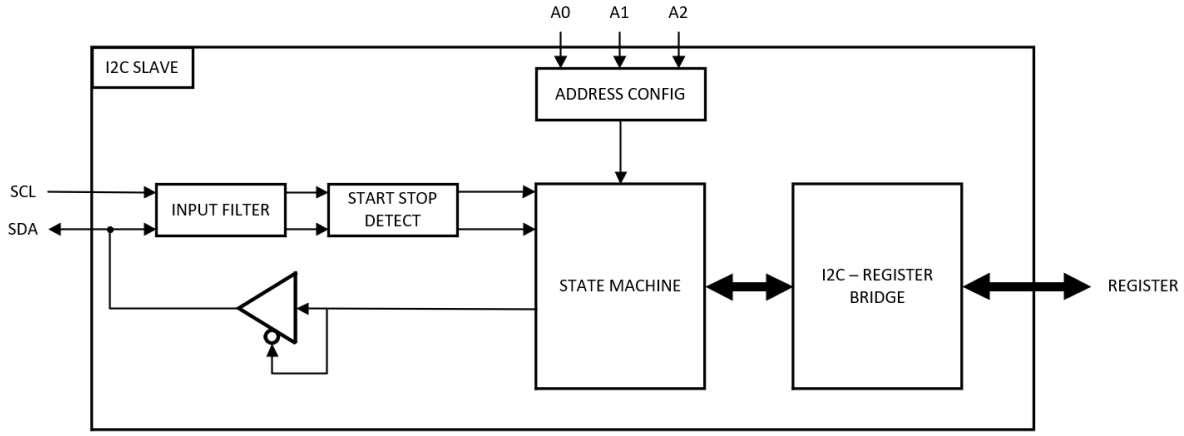
Sơ đồ khối tổng quát của thiết kế khối I²C Slave được thể hiện ở hình sau:



Hình 3.2 Sơ đồ khối giao diện I²C_top

STT	Tín hiệu	Số bit	Phân loại	Mô tả
1	clk_i	1	Ngõ vào	Xung clock hoạt động cho module I²C.
2	rst_n_i	1	Ngõ vào	Tín hiệu reset tích cực thấp
3	sda_io	1	Ngõ vào/ra	Chân dữ liệu hai chiều của giao thức I²C.
4	scl_i	1	Ngõ vào	Chân xung đồng hồ của giao thức I²C.
5	address_set_pin_i	3	Ngõ vào	Các chân thiết lập địa chỉ I²C (ví dụ: A0, A1, A2).
6	rdata_i	16	Ngõ vào	Dữ liệu đầu vào để đọc từ các thanh ghi nội bộ (register).
7	addr_o	8	Ngõ ra	Địa chỉ thanh ghi (8-bit) nội bộ .
8	wdata_o	16	Ngõ ra	Dữ liệu đầu ra (16-bit) để ghi vào các thanh ghi.
9	wr_en_o	1	Ngõ ra	Tín hiệu cho phép ghi, được kích hoạt khi Master I²C thực hiện thao tác ghi.
10	rd_en_o	1	Ngõ ra	Tín hiệu cho phép, được kích hoạt khi Master I²C thực hiện thao tác đọc.

Bảng 3.1 Bảng mô tả các tín hiệu của khối I²C_top



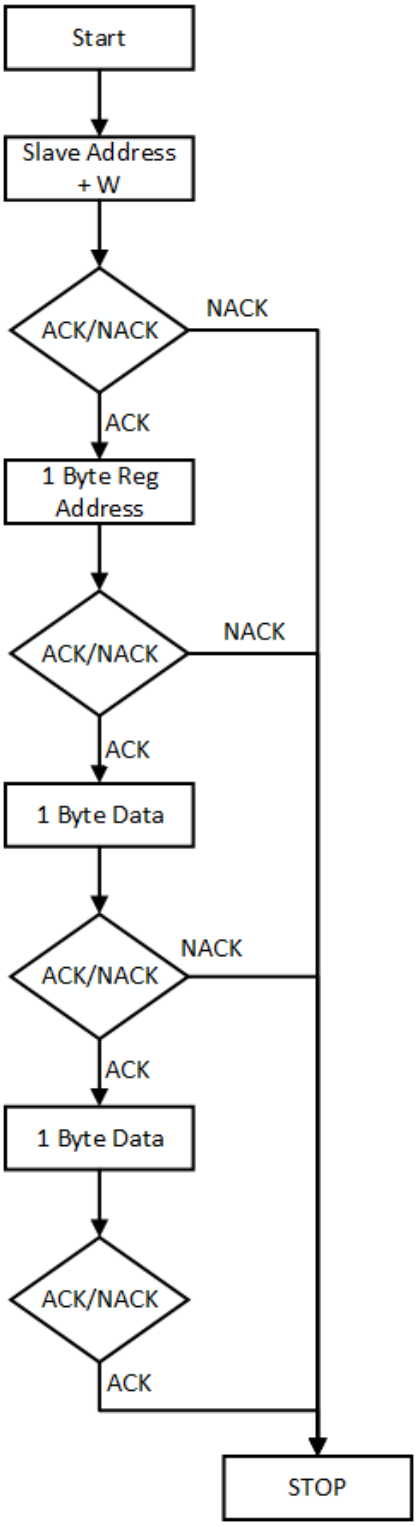
Hình 3.3 Sơ đồ khối tổng thể hệ thống I²C Slave

Chức năng của từng khối trong thiết kế là:

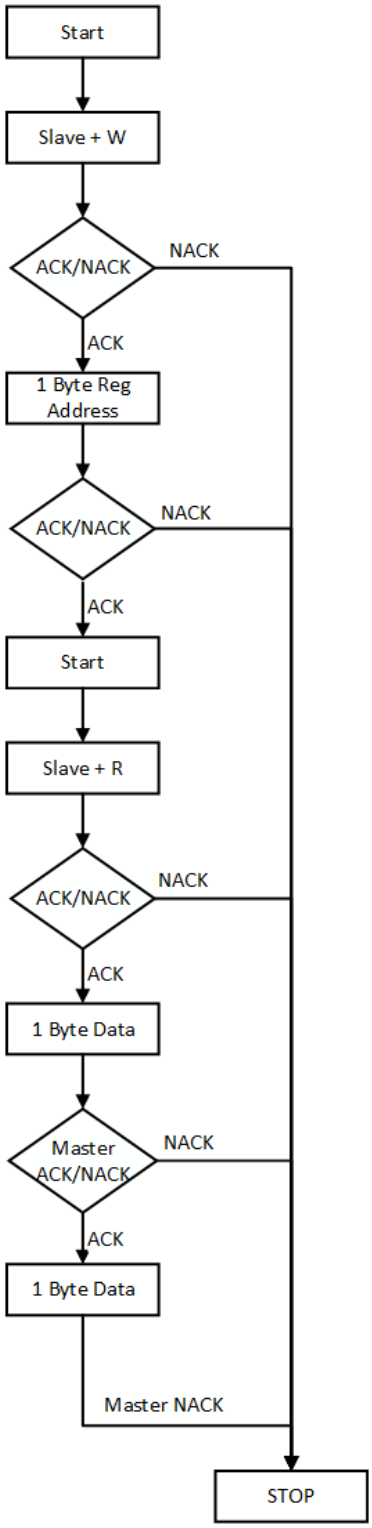
- Khối Input Filter: Chức năng của khối này là tiếp nhận các tín hiệu I²C thô từ bên ngoài (SCL, SDA) và thực hiện lọc nhiễu, ổn định tín hiệu.
- Khối Start Stop Detect: Chức năng của khối này là phát hiện các điều kiện giao dịch quan trọng như START và STOP để đảm bảo quá trình truyền nhận được kích hoạt đúng thời điểm.
- Khối Address Config: Có nhiệm vụ nhận tín hiệu đầu vào (A0, A1, A2), sau đó chuyển thành 7 bit địa chỉ vật lý cho I²C.
- Khối State Machine: Đây là khối chịu trách nhiệm giải mã toàn bộ giao dịch I²C. Nó kiểm tra địa chỉ slave nhận được từ Master và so sánh với địa chỉ từ khối Address Config để xác định thiết bị có phải là đích của giao dịch hay không. Khối này cũng xác định loại thao tác đang diễn ra (đọc hoặc ghi) dựa trên bit điều khiển trong gói tin.
- Khối I²C Frame Bridge: Chức năng của khối này là chuyển đổi các dữ liệu I²C dạng chuỗi byte thành lệnh truy cập thanh ghi nội bộ. Byte dữ liệu đầu tiên sau khi địa chỉ slave được xác nhận sẽ được xem như địa chỉ thanh ghi cần truy cập, các byte tiếp theo sẽ được dùng làm dữ liệu ghi hoặc dữ liệu phản hồi cho giao dịch đọc tùy theo lệnh từ Master.

3.2.2. Lưu đồ trạng thái

Dựa vào yêu cầu hệ thống, hình thành nên được 2 lưu đồ trạng thái đại diện cho 2 trường hợp đọc và ghi như sau:



Hình 3.4a



Hình 3.4b

Hình 3.4 Lưu đồ trạng thái mô tả quá trình truyền 2 byte dữ liệu trên Slave

Đối với sơ đồ 3.4a mô tả cách Master (thiết bị chủ) ghi 2 byte dữ liệu vào một thanh ghi cụ thể trên Slave (thiết bị phụ). Gồm các bước cơ bản sau:

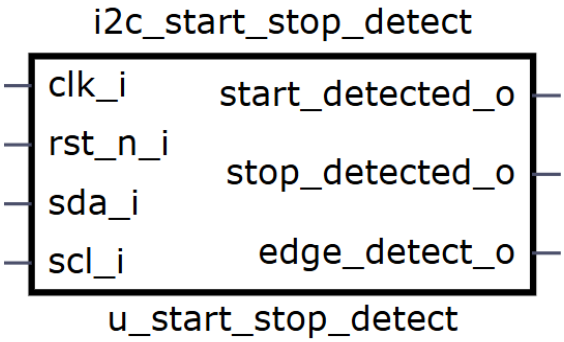
- Bắt đầu: Master tạo tín hiệu Start.
- Địa chỉ Slave + W: Master gọi Slave bằng địa chỉ vật lý và cho biết muốn thực hiện quá trình ghi (Write). Nếu Slave không phản hồi (NACK), giao dịch dừng lại (Stop).
- 1 Byte Reg Address (địa chỉ thanh ghi): Master gửi địa chỉ vị trí thanh ghi muốn ghi dữ liệu vào. Nếu Slave không ACK (NACK), giao dịch dừng lại.
- 1 Byte Data (Byte Dữ liệu 1): Master gửi byte dữ liệu thứ nhất. Nếu Slave không ACK (NACK), giao dịch dừng lại.
- 1 Byte Data (Byte Dữ liệu 2): Master gửi byte dữ liệu thứ hai. Nếu Slave không ACK (NACK), giao dịch dừng lại.
- STOP: Master tạo tín hiệu Stop để kết thúc giao dịch.

Còn sơ đồ 3.4b miêu tả quá trình đọc. Quá trình này phức tạp hơn ghi vì phải yêu cầu Master trước hết phải ghi địa chỉ thanh ghi mà nó muốn đọc, sau đó mới đọc dữ liệu thực tế. Quy trình được chia thành hai pha chính, được ngăn cách bởi một tín hiệu Start lặp lại (Repeated Start).

- Pha ghi (Write Phase): Mục đích chính là báo cho Slave biết Master muốn đọc từ thanh ghi có địa chỉ nào. Gồm các bước sau:
    + Bắt đầu: Master tạo tín hiệu Start.
    + Địa chỉ Slave + W: Master gọi Slave bằng địa chỉ vật lý và cho biết muốn thực hiện quá trình ghi (Write). Nếu Slave không phản hồi (NACK), giao dịch dừng lại (Stop).
    + 1 Byte Reg Address (địa chỉ thanh ghi): Master gửi địa chỉ vị trí thanh ghi muốn ghi dữ liệu vào. Nếu Slave không ACK (NACK), giao dịch dừng lại.
- Pha đọc (Read Phase): Sau khi truyền cho Slave địa chỉ thanh ghi mà Master muốn nhận dữ liệu. Đến pha này là lúc để Slave gửi dữ liệu từ thanh ghi đó về cho Master. Gồm các bước sau:
    + Start (Lặp lại): Master tạo tín hiệu Repeated Start (không có Stop ở giữa) để chuyển hướng giao dịch.
    + Địa chỉ Slave + R: Master gọi Slave bằng địa chỉ vật lý và cho biết muốn thực hiện quá trình đọc (Read). Nếu Slave không phản hồi (NACK), giao dịch dừng lại (Stop).
    + 1 Byte Data (Byte Dữ liệu 1): Slave gửi byte dữ liệu thứ nhất. Nếu Master không ACK (NACK), giao dịch dừng lại.
    + 1 Byte Data (Byte Dữ liệu 2): Slave gửi byte dữ liệu thứ hai. Lúc này Master sẽ gửi không xác nhận (NACK), để báo hiệu cho Slave biết Master đã nhận đủ dữ liệu.
    + STOP: Master tạo tín hiệu Stop để kết thúc giao dịch.

3.2.3. Sơ đồ và nguyên lý hoạt động theo từng khối nội bộ

3.2.3.1. Khối I²C_start_stop_detect



Hình 3.5 Sơ đồ giao diện I²C_start_stop_detect

STT	Tín hiệu	Số bit	Phân loại	Mô tả
1	clk_i	1	Ngõ vào	Xung clock hoạt động của module.
2	rst_n_i	1	Ngõ vào	Tín hiệu reset tích cực thấp cho module.
3	sda_i	1	Ngõ vào	Tín hiệu dữ liệu (Serial Data) của bus I²C (đầu vào từ bus).
4	scl_i	1	Ngõ vào	Tín hiệu xung đồng hồ (Serial Clock) của bus I²C.
5	start_detected_o	1	Ngõ ra	Tín hiệu được kích hoạt khi module phát hiện điều kiện START của giao thức I²C.
6	stop_detected_o	1	Ngõ ra	Tín hiệu được kích hoạt khi module phát hiện điều kiện STOP của giao thức I²C.
7	edge_detect_o	1	Ngõ ra	Tín hiệu được kích hoạt khi phát hiện có sự thay đổi cạnh lên trên đường xung SCL

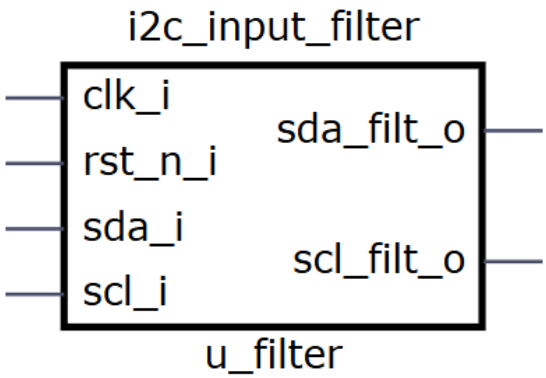
Bảng 3.2 Bảng mô tả các tín hiệu của module I²C_start_stop_detect

Module I²C_start_stop_detect có chức năng chính là giám sát các đường tín hiệu I²C (SDA và SCL) để phát hiện các sự kiện kiểm soát quan trọng:

- Điều kiện START: Được định nghĩa là một sườn xuống trên SDA khi SCL ở mức cao.
- Điều kiện STOP: Được định nghĩa là một sườn lên trên SDA khi SCL ở mức cao.

Đồng thời, module còn có tín hiệu edge_detect_o dùng để phát hiện cạnh lên trên đường tín hiệu SCL đã được lọc. Tín hiệu này giúp đồng bộ hóa các module giải mã trạng thái tiếp theo.

3.2.3.2. Khối I²C_input_filter



Hình 3.6 Sơ đồ khối giao diện I²C_input_filter

STT	Tín hiệu	Số bit	Phân loại	Mô tả
1	clk_i	1	Ngõ vào	Xung clock hoạt động của module.
2	rst_n_i	1	Ngõ vào	Tín hiệu reset tích cực thấp cho module.
3	sda_i	1	Ngõ vào	Tín hiệu dữ liệu thô từ bus I²C.
4	scl_i	1	Ngõ vào	Tín hiệu xung đồng hồ thô từ bus I²C.
5	sda_filt_o	1	Ngõ ra	Tín hiệu dữ liệu đã được lọc và chống dội.
6	scl_filt_o	1	Ngõ ra	Tín hiệu xung đồng hồ đã được lọc và chống dội.

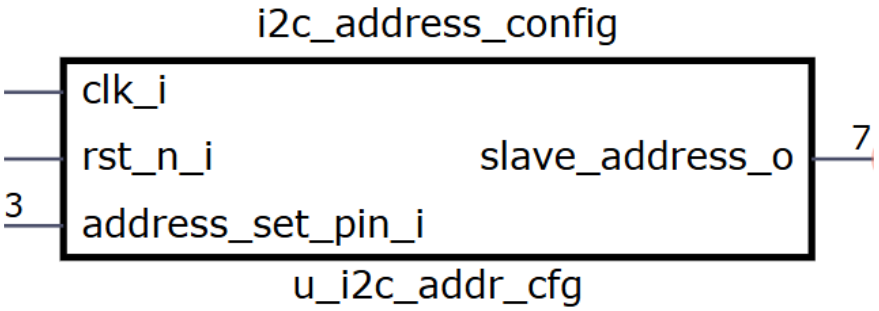
Bảng 3.3 Bảng mô tả các tín hiệu của module I²C_input_filter

Module I²C_input_filter có vai trò thiết yếu trong việc đảm bảo ổn định và độ tin cậy của giao tiếp I²C:

- Lọc nhiễu (Noise Filtering): Bus I²C là bus tốc độ thấp và thường chịu ảnh hưởng của nhiễu (noise) và các đột biến điện áp (glitches) trên đường dây, đặc biệt là trong môi trường công nghiệp hoặc dây cáp dài. Bộ lọc này loại bỏ các xung nhiễu có thời gian ngắn. Module sử dụng shift-register để loại bỏ các xung nhiễu, các đột biến điện áp có thời gian ngắn trên bus.
- Chống dội (Debouncing/Spike Suppression): Chuyển đổi trạng thái của các tín hiệu I²C không hoàn hảo. Bộ lọc này đảm bảo rằng các trạng thái logic (HIGH hoặc LOW) được xác định rõ ràng bằng cơ chế giữ tín hiệu ở một trạng thái trong một khoảng thời gian nhất định (thường là vài chu kỳ clock) trước khi xác nhận sự thay đổi trạng thái.

Tín hiệu đầu ra sau đó được sử dụng bởi các khối logic I²C tiếp theo để đảm bảo hoạt động trên dữ liệu ổn định.

3.2.3.3. Khối I²C_address_config



Hình 3.7 Sơ đồ khối giao diện I²C_address_config

STT	Tín hiệu	Số bit	Phân loại	Mô tả
1	clk_i	1	Ngõ vào	Xung clock hoạt động của module.
2	rst_n_i	1	Ngõ vào	Tín hiệu reset tích cực thấp cho module.
3	address_set_pin_i	3	Ngõ vào	Các chân cấu hình địa chỉ vật lý (hardware pin) của Slave (ví dụ: A0, A1, A2).
4	slave_address_o	7	Ngõ ra	Địa chỉ Slave I²C cuối cùng (7-bit) được tạo ra và sử dụng bởi bộ giải mã địa chỉ I²C.

Bảng 3.4 Bảng mô tả các tín hiệu của module I²C_address_config

Module I²C_address_config có chức năng chính là tạo ra địa chỉ I²C Slave (7-bit).

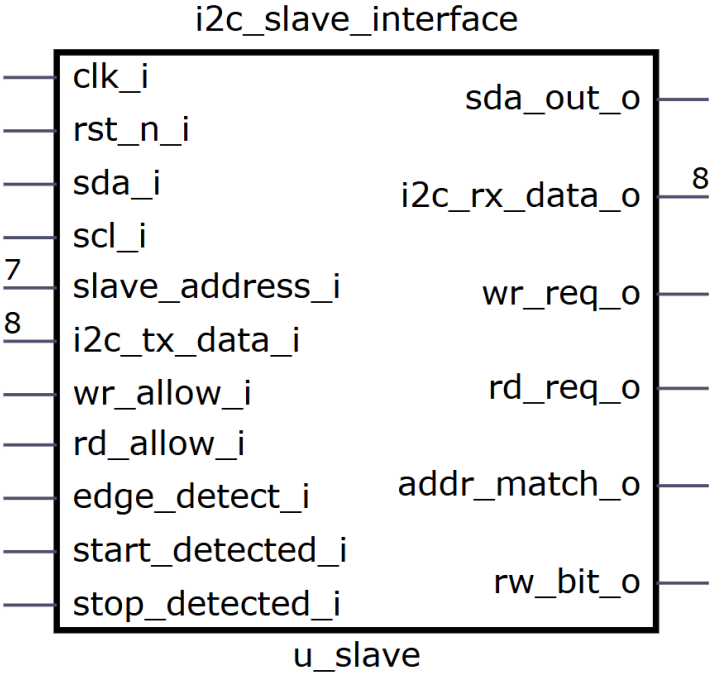
Cấu hình địa chỉ: Giao thức I²C thường yêu cầu 7-bit địa chỉ vật lý. Ở đây, do mong muốn đơn giản hóa, nhóm đã chọn 4-bit địa chỉ cố định (Fixed Address) và 3 bit địa chỉ có thể lập trình/cấu hình qua chân vật lý (Configurable Address).

Hoạt động: Module này sẽ nhận 3 bit đầu vào từ các chân cấu hình (address_set_pin_i) và kết hợp chúng với các bit địa chỉ cố định được mã hóa cứng bên trong module để tạo ra slave_address_o (7-bit).

Ý nghĩa: Điều này cho phép nhiều thiết bị Slave giống hệt nhau được kết nối trên cùng một bus I²C, mỗi thiết bị có một địa chỉ duy nhất được xác định bởi cách người dùng cấu hình các chân vật lý bên ngoài (A0, A1, A2).

3.2.3.4. Khối I²C_slave_interface

Module này đóng vai trò là khối logic lõi để xử lý giao tiếp I²C ở phía Slave.



Hình 3.8 Sơ đồ khối giao diện I²C_slave_interface

STT	Tín hiệu	Số bit	Phân loại	Mô tả
1	clk_i	1	Ngõ vào	Xung clock hoạt động của module.
2	rst_n_i	1	Ngõ vào	Tín hiệu reset tích cực thấp.
3	sda_i	1	Ngõ vào	Tín hiệu dữ liệu (Serial Data) đã được lọc từ bus I²C (nhận từ I²C_input_filter).
4	scl_i	1	Ngõ vào	Tín hiệu xung đồng hồ (Serial Clock) đã được lọc từ bus I²C.
5	slave_address_i	7	Ngõ vào	Địa chỉ Slave 7-bit của module (nhận từ I²C_address_config).
6	I²C_tx_data_i	8	Ngõ vào	Dữ liệu 8-bit được đưa vào để gửi đi trong giao dịch Đọc (Read).
7	wr_allow_i	1	Ngõ vào	Tín hiệu cho phép ghi (Write Allow) từ khối điều khiển thanh ghi nội bộ.

8	rd_allow_i	1	Ngõ vào	Tín hiệu cho phép đọc (Read Allow) từ khối điều khiển thanh ghi nội bộ.
9	edge_detect_i	1	Ngõ vào	Tín hiệu phát hiện cạnh lên của SCL (nhận từ I²C_start_stop_detect).
10	start_detected_i	1	Ngõ vào	Tín hiệu phát hiện điều kiện START (nhận từ I²C_start_stop_detect).
11	stop_detected_i	1	Ngõ vào	Tín hiệu phát hiện điều kiện STOP (nhận từ I²C_start_stop_detect).
12	sda_out_o	1	Ngõ ra	Tín hiệu dữ liệu gửi ra bus I²C (kết nối với logic điều khiển bus open-drain).
13	I²C_rx_data_o	8	Ngõ ra	Dữ liệu 8-bit nhận được từ Master I²C.
14	wr_req_o	1	Ngõ ra	Yêu cầu ghi (Write Request) gửi đến khối điều khiển thanh ghi (đã nhận được 8-bit dữ liệu).
15	rd_req_o	1	Ngõ ra	Yêu cầu đọc (Read Request) gửi đến khối điều khiển thanh ghi.
16	addr_match_o	1	Ngõ ra	Tín hiệu xác nhận địa chỉ Slave khớp với địa chỉ Master đã gửi.
17	rw_bit_o	1	Ngõ ra	Bit R/W của giao dịch hiện tại (chỉ ra giao dịch là Đọc hay Ghi).

Bảng 3.5 Bảng mô tả các tín hiệu của module I²C_slave_interface

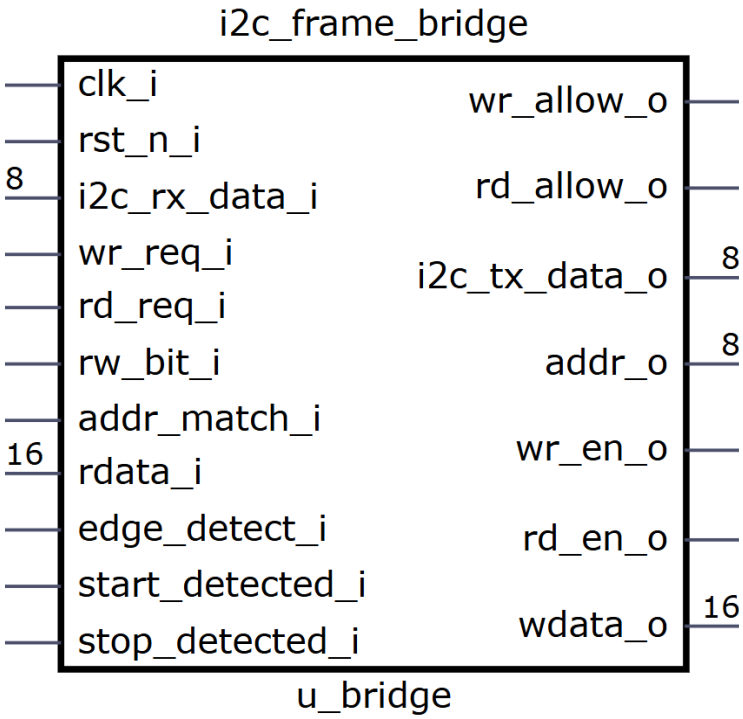
Module I²C_slave_interface là bộ xử lý trạng thái (State Machine) cốt lõi của Slave, chịu trách nhiệm:

1. Phân tích các tín hiệu I²C đến (SCL, SDA) và các điều kiện kiểm soát (start, stop, edge-detect).
2. So sánh địa chỉ Master gửi đến với slave_address_i để kích hoạt addr_match_o.
3. Giải mã bit đọc/ghi (R/W) và xuất ra rw_bit_o.
4. Nhận dữ liệu 8-bit và xuất ra I²C_rx_data_o cùng với yêu cầu wr_req_o.
5. Gửi dữ liệu 8-bit (I²C_tx_data_i) ra bus qua sda_out_o khi có yêu cầu Đọc.

Đây là khối trung tâm kết nối logic I²C ngoài với logic điều khiển nội bộ (thanh ghi).

3.2.3.5. Khối I²C_slave_bridge

Module này hoạt động như một bộ điều khiển trung gian (Bridge Controller) giữa giao diện I²C Slave (từ I²C_slave_interface) và bus thanh ghi nội bộ (Register Bus).



Hình 3.9 Sơ đồ khối giao diện I²C_frame_bridge

STT	Tín hiệu	Số bit	Phân loại	Mô tả
1	clk_i	1	Ngõ vào	Xung clock hoạt động của module.
2	rst_n_i	1	Ngõ vào	Tín hiệu reset tích cực thấp.
3	I²C_rx_data_i	8	Ngõ vào	Dữ liệu 8-bit nhận được từ Master I²C (nhận từ I²C_slave_interface).
4	wr_req_i	1	Ngõ vào	Yêu cầu ghi (Write Request) từ giao diện Slave.
5	rd_req_i	1	Ngõ vào	Yêu cầu đọc (Read Request) từ giao diện Slave.
6	rw_bit_i	1	Ngõ vào	Bit Đọc/Ghi (Read/Write bit) từ giao diện Slave.

7	addr_match_i	1	Ngõ vào	Tín hiệu xác nhận địa chỉ Slave khớp.
8	rdata_i	16	Ngõ vào	Dữ liệu 16-bit được đọc từ Thanh ghi nội bộ.
9	edge_detect_i	1	Ngõ vào	Tín hiệu phát hiện cạnh (từ khối phát hiện).
10	start_detected_i	1	Ngõ vào	Tín hiệu phát hiện điều kiện START.
11	stop_detected_i	1	Ngõ vào	Tín hiệu phát hiện điều kiện STOP.
12	wr_allow_o	1	Ngõ ra	Cho phép ghi (Write Allow) được gửi đến giao diện Slave.
13	rd_allow_o	1	Ngõ ra	Cho phép đọc (Read Allow) được gửi đến giao diện Slave.
14	I²C_tx_data_o	8	Ngõ ra	Dữ liệu 8-bit được gửi đến giao diện Slave để truyền đi (dữ liệu được lấy từ Thanh ghi).
15	addr_o	8	Ngõ ra	Địa chỉ thanh ghi 8-bit nội bộ mà I²C Master đang truy cập.
16	wr_en_o	1	Ngõ ra	Tín hiệu Cho phép Ghi Thanh ghi (Register Write Enable).
17	rd_en_o	1	Ngõ ra	Tín hiệu Cho phép Đọc Thanh ghi (Register Read Enable).
18	wdata_o	16	Ngõ ra	Dữ liệu 16-bit để ghi vào Thanh ghi nội bộ.

Bảng 3.6 Bảng mô tả các tín hiệu của khối I²C_frame_bridge

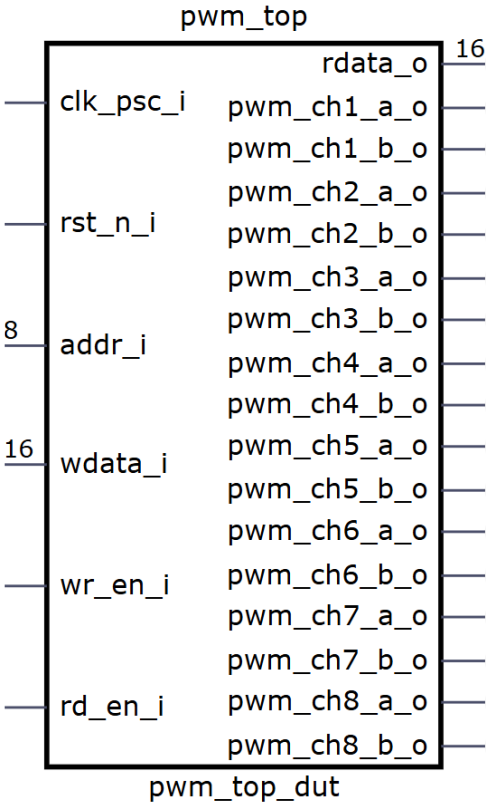
Module I²C_frame_bridge chịu trách nhiệm chuyển đổi giao thức I²C (truyền từng byte) thành các thao tác truy cập thanh ghi (Register Access) cụ thể:

- Giải mã khung (Frame Decoding): Nó theo dõi chuỗi dữ liệu I²C. Sau khi Master gửi địa chỉ Slave và rw_bit, byte dữ liệu đầu tiên thường là địa chỉ thanh ghi (Register Address). Module này phải lưu trữ byte đó và kích hoạt addr_o.
- Xác định thao tác: Khi nhận được wr_req_i hoặc rd_req_i, Bridge sẽ:
- Ghi (Write): Nếu rw_bit_i là bit 0 (Ghi), byte tiếp theo (I²C_rx_data_i) sẽ được dùng làm dữ liệu ghi, kích hoạt wdata_o và wr_en_o. Sau đó gửi wr_allow_o về I²C_slave_interface.
- Đọc (Read): Nếu rw_bit_i là bit 1 (Đọc), Bridge kích hoạt rd_en_o và đợi rdata_i (16-bit) từ bus thanh ghi, sau đó chia nó thành các byte 8-bit và gửi tuần tự đến I²C_tx_data_o để Slave truyền đi. Nó cũng gửi rd_allow_o về I²C_slave_interface.

Nói tóm lại, Bridge này là bộ quản lý luồng dữ liệu I²C liên tục thành các truy cập đọc/ghi rời rạc vào bộ nhớ hoặc thanh ghi nội bộ.

3.3. Thiết kế khối PWM

3.3.1. Sơ đồ khối tổng quát của khối PWM

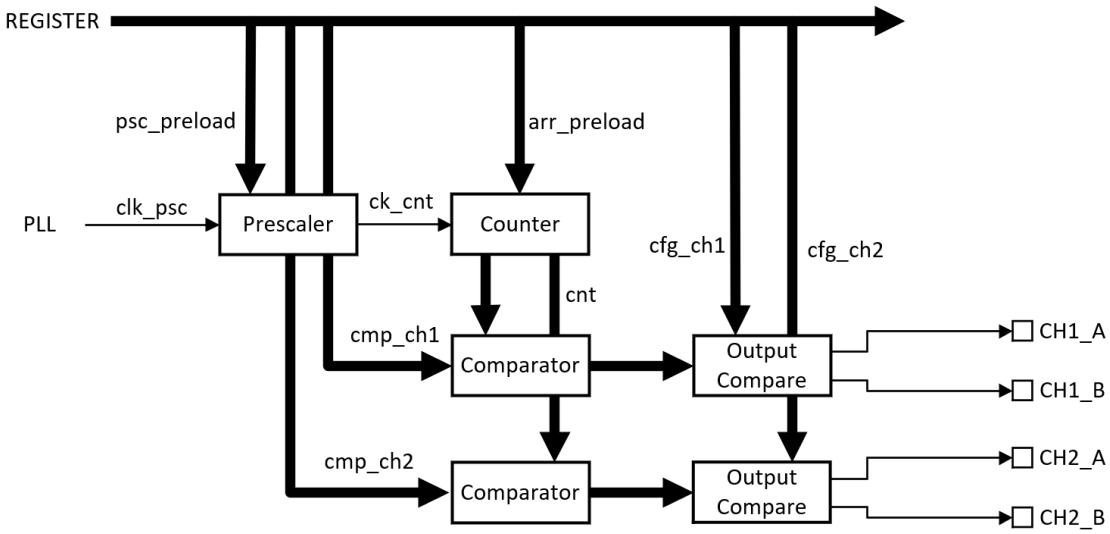


Hình 3.10 Sơ đồ khối giao diện của khối pwm_top

STT	Tín hiệu	Số bit	Phân loại	Mô tả
1	clk_psc_i	1	Ngõ vào	Xung clock hoạt động
2	rst_n_i	1	Ngõ vào	Tín hiệu reset tích cực thấp.
3	addr_i	8	Ngõ vào	Địa chỉ thanh ghi (8-bit)
4	wr_en_i	1	Ngõ vào	Tín hiệu ghi từ I²C Frame Bridge.
5	rd_en_i	1	Ngõ vào	Tín hiệu đọc từ I²C Frame Bridge.
6	wdata_i	16	Ngõ vào	Dữ liệu 16-bit để ghi vào thanh ghi.
7	rdata_o	16	Ngõ ra	Dữ liệu 16-bit được đọc từ thanh ghi.
8	pwm_ch1_a_o	1	Ngõ ra	Xung PWM chính cho Kênh 1.
9	pwm_ch1_b_o	1	Ngõ ra	Xung PWM bù cho Kênh 1.
10	pwm_ch2_a_o	1	Ngõ ra	Xung PWM chính cho Kênh 2.
11	pwm_ch2_b_o	1	Ngõ ra	Xung PWM bù cho Kênh 2.
12	pwm_ch3_a_o	1	Ngõ ra	Xung PWM chính cho Kênh 3.
13	pwm_ch3_b_o	1	Ngõ ra	Xung PWM bù cho Kênh 3.
14	pwm_ch4_a_o	1	Ngõ ra	Xung PWM chính cho Kênh 4.

15	pwm_ch4_b_o	1	Ngõ ra	Xung PWM bù cho Kênh 4.
16	pwm_ch5_a_o	1	Ngõ ra	Xung PWM chính cho Kênh 5.
17	pwm_ch5_b_o	1	Ngõ ra	Xung PWM bù cho Kênh 5.
18	pwm_ch6_a_o	1	Ngõ ra	Xung PWM chính cho Kênh 6.
19	pwm_ch6_b_o	1	Ngõ ra	Xung PWM bù cho Kênh 6.
20	pwm_ch7_a_o	1	Ngõ ra	Xung PWM chính cho Kênh 7.
21	pwm_ch7_b_o	1	Ngõ ra	Xung PWM bù cho Kênh 7.
22	pwm_ch8_a_o	1	Ngõ ra	Xung PWM chính cho Kênh 8.
23	pwm_ch8_b_o	1	Ngõ ra	Xung PWM bù cho Kênh 8.

Bảng 3.7 Bảng mô tả các tín hiệu của khối pwm_top



Hình 3.11 Sơ đồ khối tổng quát của hệ thống PWM

Chức năng của từng khối trong thiết kế là:

- Prescaler (bộ chia tần số): Khối này nhận xung nhịp clk_psc và chia tần số của nó theo một hệ số được cấu hình bởi psc_preload. Điều này tạo ra một xung nhịp chậm hơn (clk_cnt) để cấp cho Counter, cho phép thay đổi tần số hoạt động linh hoạt.
- Counter (bộ đếm): Khối này đếm theo xung nhịp clk_cnt. Nó đếm từ 0 cho đến giá trị tối đa được cấu hình bởi arr_preload, sau đó chuyển về 0 và bắt đầu đếm lại. Giá trị đếm hiện tại (cnt) được cấp cho các Comparator.
- Comparator (bộ so sánh): Khối này so sánh giá trị đếm hiện tại (cnt) với giá trị so sánh đã được cấu hình (cmp_ch1 hoặc cmp_ch2). Từ đó tạo ra 4 tín hiệu (cnt bằng với cmp_ch1_start , cnt lớn hơn cmp_ch1_start, cnt bằng với cmp_ch1_end, cnt lớn hơn cmp_ch1_end).
- Output Compare (bộ so sánh đầu ra): Khối này nhận 4 tín hiệu từ Comparator và sử dụng các bit cấu hình (cfg_ch) để xác định cách tạo ra dạng sóng đầu ra. Tùy vào chế độ cài đặt (chạy đơn kênh hoặc chạy) mà khối sẽ tạo ra một

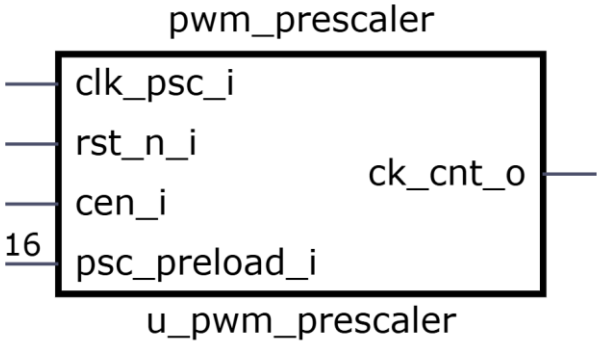
tín hiệu. Đây là điểm mấu chốt để xác định độ rộng xung (Duty Cycle) trong ứng dụng PWM.

Có 2 chế độ hoạt động chính:

- Chế độ đơn kênh (Independent Channel Mode)
  + Nguyên tắc: Mỗi kênh đầu ra (ví dụ: CH1) sử dụng một cặp giá trị so sánh riêng (cmp_ch1_start, cmp_ch1_end) để tạo ra tín hiệu đầu ra độc lập.
  + Hoạt động: Nếu có một cặp giá trị so sánh, hệ thống có thể tạo ra hai kênh đầu ra độc lập (ví dụ: CH1_A và CH1_B), mỗi kênh có cùng tần số và độ rộng xung (Duty Cycle) riêng biệt.
  + Ứng dụng: Thích hợp để tạo tín hiệu PWM đơn giản, điều khiển động cơ DC hoặc servo.
- Chế độ thuận-nghịch (Complementary Mode)
  + Nguyên tắc: Một cặp giá trị so sánh duy nhất (cmp_chX) được sử dụng để tạo ra hai tín hiệu đầu ra ngược pha nhau (ví dụ: CH1_A và CH1_B).
  + Hoạt động: Tạo ra một cặp tín hiệu bổ sung (Complementary Signals), trong đó khi CH1_A ở mức HIGH thì CH1_B ở mức LOW và ngược lại.Tùy theo cấu hình (cfg_chX), các tín hiệu này có thể được đưa qua bộ tạo Dead-time hoặc đưa ra chân IO.
  + Ứng dụng: Thường dùng trong điều khiển động cơ cầu H, bộ biến tần, hoặc các bộ chuyển đổi nguồn cần tín hiệu điều khiển bổ sung kèm bảo vệ Dead-time.

3.3.2. Sơ đồ và nguyên lý hoạt động của từng khối nội bộ

3.3.2.1. Khối pwm_prescaler



Hình 3.12 Sơ đồ khối giao diện pwm_prescaler

STT	Tín hiệu	Số bit	Phân loại	Mô tả
1	clk_psc_i	1	Ngõ vào	Xung clock tần số cao đầu vào cho bộ chia tần số (Prescaler Clock).
2	rst_n_i	1	Ngõ vào	Tín hiệu reset tích cực thấp.
3	cen_i	1	Ngõ vào	Tín hiệu cho phép đếm (*Counter Enable*).
4	psc_preload_i	16	Ngõ vào	Giá trị nạp trước 16-bit, xác định tỷ lệ chia tần số (Prescaler Value).
5	ck_cnt_o	1	Ngõ ra	Tín hiệu xung đồng hồ đầu ra đã được chia tần số (Clock Counter Output), được dùng làm clock cho bộ đếm PWM chính.

Bảng 3.8 Bảng mô tả các tín hiệu của khối pwm_prescaler

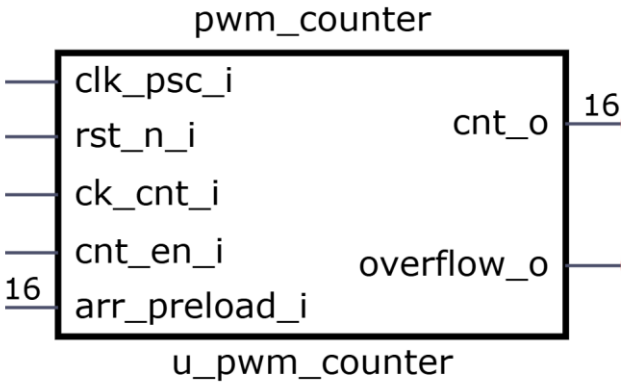
Module pwm_prescaler đóng vai trò quan trọng trong việc tạo ra tần số PWM:

- Chia tần số Clock: Chức năng chính là lấy xung clock đầu vào tần số cao (clk_psc_i) và tạo ra một xung clock đầu ra (ck_cnt_o) có tần số thấp hơn.
- Xác định tần số: Tần số đầu ra được xác định bởi giá trị psc_preload_i (giá trị bộ chia N). Xung clock đầu ra thường được kích hoạt mỗi khi bộ đếm đạt đến giá trị N này, theo công thức:

$$Tần\ số\ đầu\ ra = \frac{tần\ số\ đầu\ vào}{hệ\ số\ chia} = \frac{tần\ số\ đầu\ vào}{psc\_preload + 1} \quad (3.1)$$

- Phạm vi rộng: Bằng cách sử dụng psc_preload_i 16-bit, module có thể chia tần số trong một phạm vi rất rộng, cho phép bộ điều khiển PWM tạo ra các tần số xung rất khác nhau (từ rất cao đến rất thấp) từ một xung clock hệ thống cố định.

3.3.2.2. Khối pwm_counter



Hình 3.13 Sơ đồ giao diện của khối pwm_counter

STT	Tín hiệu	Số bit	Phân loại	Mô tả
1	clk_psc_i	1	Ngõ vào	Xung clock tần số cao (thường là clock hệ thống) dùng cho logic nội bộ của bộ đếm.
2	rst_n_i	1	Ngõ vào	Tín hiệu reset tích cực thấp.
3	ck_cnt_i	1	Ngõ vào	Xung clock đã được chia tần số (Prescaled Clock) dùng để đếm (nhận từ pwm_prescaler).
4	cnt_en_i	1	Ngõ vào	Tín hiệu cho phép bộ đếm hoạt động (Counter Enable).
5	arr_preload_i	16	Ngõ vào	Giá trị thanh ghi tự động nạp lại (Auto-Reload Register - ARR), xác định giá trị tối đa của bộ đếm (chu kỳ PWM).
6	cnt_o	16	Ngõ ra	Giá trị đếm hiện tại 16-bit của bộ đếm PWM.
7	overflow_o	1	Ngõ ra	Tín hiệu được kích hoạt khi bộ đếm đạt đến giá trị tối đa (arr_preload_i) và tràn (overflow) hoặc nạp lại (reset) về 0.

Bảng 3.9 Bảng mô tả các tín hiệu của khối pwm_counter

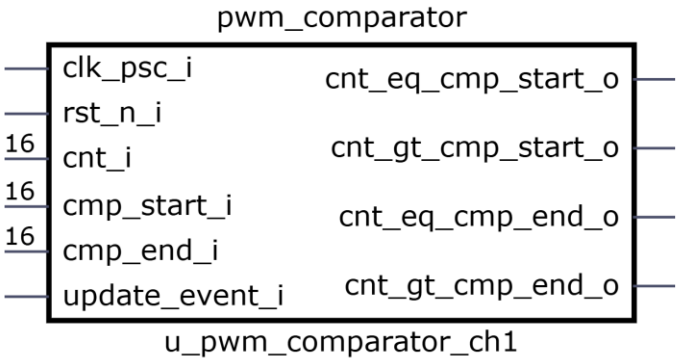
Module pwm_counter là khối tạo ra một bộ đếm hình răng cưa (sawtooth wave) có chu kỳ cố định, thực hiện các chức năng cơ bản của một bộ định thời.

Chu kỳ (Period): Giá trị arr_preload_i (ARR) xác định số lượng xung ck_cnt_i cần thiết để bộ đếm hoàn thành một chu kỳ đầy đủ. Đây là yếu tố chính xác định tần số cơ bản của tín hiệu PWM:

$$Tần\ số\ PWM = \frac{tần\ số\ ck\_cnt}{arr\_preload + 1} \quad (3.2)$$

Tạo tín hiệu cờ (Flag): Tín hiệu overflow_o là cờ quan trọng nhất. Nó báo hiệu sự kết thúc của một chu kỳ PWM, được sử dụng để tạo tín hiệu cập nhật giá trị thanh ghi nội cho các các khối PWM Core tiếp theo.

3.3.2.3. Khối comparator



Hình 3.14 Sơ đồ giao diện của khối comparator

STT	Tín hiệu	Số bit	Phân loại	Mô tả
1	clk_psc_i	1	Ngõ vào	Xung clock hoạt động cho logic nội bộ của bộ so sánh.
2	rst_n_i	1	Ngõ vào	Tín hiệu reset tích cực thấp.
3	cnt_i	16	Ngõ vào	Giá trị đếm hiện tại 16-bit của Bộ đếm PWM (nhận từ pwm_counter).
4	cmp_start_i	16	Ngõ vào	Giá trị so sánh Bắt đầu 16-bit, xác định thời điểm bật xung PWM.
5	cmp_end_i	16	Ngõ vào	Giá trị so sánh Kết thúc 16-bit, xác định thời điểm tắt xung PWM.
6	update_event_i	1	Ngõ vào	Tín hiệu sự kiện cập nhật (thường là cờ tràn/overflow từ bộ đếm) để đồng bộ hóa việc nạp lại giá trị mới.
7	cnt_eq_cmp_start_o	1	Ngõ ra	Kích hoạt khi cnt_i bằng cmp_start_i.
8	cnt_gt_cmp_start_o	1	Ngõ ra	Kích hoạt khi cnt_i lớn hơn cmp_start_i.
9	cnt_eq_cmp_end_o	1	Ngõ ra	Kích hoạt khi cnt_i bằng cmp_end_i.
10	cnt_gt_cmp_end_o	1	Ngõ ra	Kích hoạt khi cnt_i lớn hơn cmp_end_i.

Bảng 3.10 Bảng mô tả các tín hiệu của khối pwm_comparator

Module pwm_comparator là trái tim của việc tạo xung PWM. Dùng để so sánh giá trị bộ đếm đang chạy (cnt_i) với hai ngưỡng cài đặt (cmp_start_i và cmp_end_i).

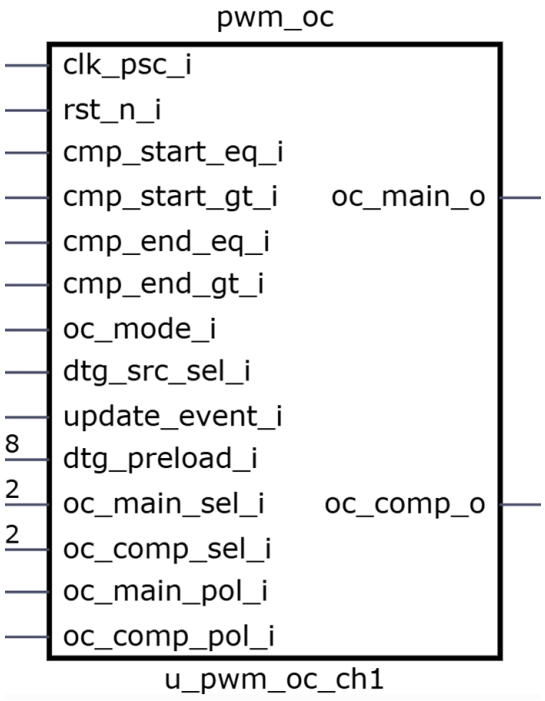
Nguyên lý hoạt động của bộ này như sau:

- Tín hiệu cnt_eq_cmp_start_o: được bật khi giá trị cnt_i = cmp_start_i.
- Tín hiệu cnt_gt_cmp_start_o: được bật khi giá trị cnt_i > cmp_start_i.
- Tín hiệu cnt_eq_ end _start_o: được bật khi giá trị cnt_i = cmp_ end _i.
- Tín hiệu cnt_gt_ end _start_o: được bật khi giá trị cnt_i > cmp_ end _i.

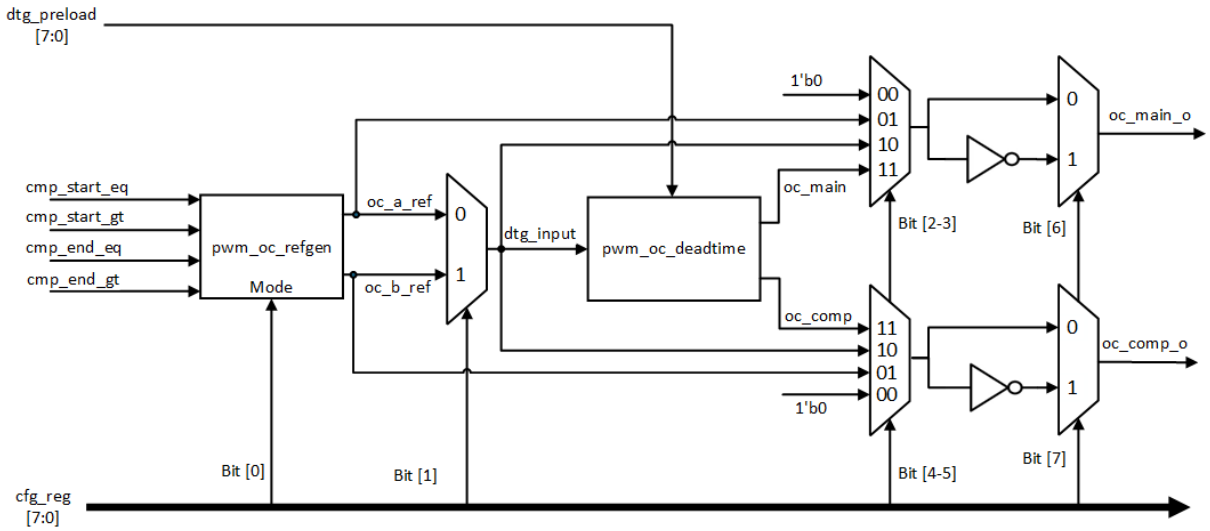
Các tín hiệu đầu ra (cnt_eq_ và cnt_gt_) là các cờ báo hiệu thời điểm xảy ra sự kiện so sánh. Các cờ này sau đó được sử dụng bởi khối logic điều khiển đầu ra (Output Control) để chuyển đổi trạng thái (HIGH/LOW) của chân PWM thực tế.

Với hai giá trị so sánh (cmp_start_i và cmp_end_i), module này hỗ trợ các chế độ PWM nâng cao hơn, bao gồm khả năng tạo xung bị dịch pha (phase-shifted) hoặc chế độ cơ bản (edge-aligned PWM).

3.3.2.4. Khối pwm_oc:



Hình 3.15 Sơ đồ giao diện của khối pwm_oc



Hình 3.16 Sơ đồ khối bên trong của khối pwm_oc

STT	Tín hiệu	Số bit	Phân loại	Mô tả
1	clk_psc_i	1	Ngõ vào	Xung clock hoạt động.
2	rst_n_i	1	Ngõ vào	Tín hiệu reset tích cực thấp.
3	cmp_start_eq_i	1	Ngõ vào	Cờ so sánh: Đếm bằng giá trị Start Compare.
4	cmp_start_gt_i	1	Ngõ vào	Cờ so sánh: Đếm lớn hơn giá trị Start Compare.
5	cmp_end_eq_i	1	Ngõ vào	Cờ so sánh: Đếm bằng giá trị End Compare.
6	cmp_end_gt_i	1	Ngõ vào	Cờ so sánh: Đếm lớn hơn giá trị End Compare.
7	oc_mode_i	1	Ngõ vào	Chế độ hoạt động của đầu ra so sánh
8	dtg_src_sel_i	1	Ngõ vào	Lựa chọn nguồn cho bộ tạo Dead Time.
9	update_event_i	1	Ngõ vào	Sự kiện cập nhật (tràn bộ đếm) để đồng bộ hóa.
10	dtg_preload_i	8	Ngõ vào	Giá trị cài đặt cho Thời gian Chết (Dead Time) 8-bit.
11	oc_main_sel_i	2	Ngõ vào	Lựa chọn đầu ra chính (oc_main_o).
12	oc_comp_sel_i	2	Ngõ vào	Lựa chọn đầu ra bù (oc_comp_o).
13	oc_main_pol_i	1	Ngõ vào	Cài đặt Phân cực (Polarity) cho đầu ra chính (oc_main_o - tích cực cao/thấp).
14	oc_comp_pol_i	1	Ngõ vào	Cài đặt Phân cực (Polarity) cho đầu ra bù (oc_comp_o - tích cực cao/thấp).
15	oc_main_o	1	Ngõ ra	Tín hiệu xung PWM chính đã được xử lý
16	oc_comp_o	1	Ngõ ra	Tín hiệu xung PWM bù đã được xử lý

Bảng 3.11 Bảng mô tả các tín hiệu của khối pwm_oc

Module pwm_oc là bộ phận cuối cùng trong chuỗi tạo xung, chuyển đổi các cờ so sánh logic thành tín hiệu vật lý:

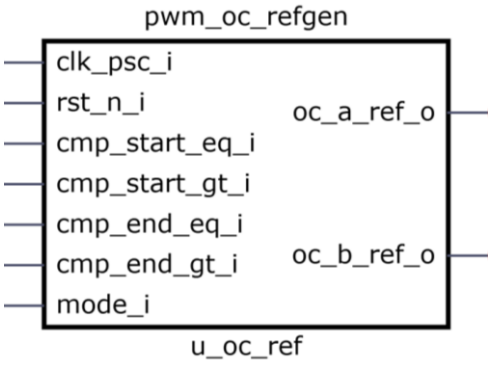
Tạo xung: Sử dụng các cờ so sánh (cmp_*) và chế độ hoạt động (oc_mode_i) để tạo ra xung PWM cơ bản (oc_main_o).

Tạo xung bù (Complementary): Tạo ra tín hiệu oc_comp_o, là tín hiệu đảo pha của oc_main_o, cần thiết cho các ứng dụng điều khiển động cơ hoặc bộ chuyển đổi công suất.

Thời gian chết (Dead Time Generation - DTG): Sử dụng dtg_preload_i và logic nội bộ để chèn một khoảng thời gian trễ ngắn giữa thời điểm một xung tắt và xung bù của nó bật, hoặc ngược lại.

Phân cực (Polarity): Cho phép người dùng đảo ngược logic đầu ra (oc_main_o, oc_comp_o) để phù hợp với yêu cầu của tải.

3.3.2.5. Khối pwm_oc_refgen



Hình 3.17 Sơ đồ giao diện của khối pwm_oc_refgen

STT	Tín hiệu	Số bit	Phân loại	Mô tả
1	clk_psc_i	1	Ngõ vào	Xung clock hoạt động của module.
2	rst_n_i	1	Ngõ vào	Tín hiệu reset tích cực thấp.
3	cmp_start_eq_i	1	Ngõ vào	Cờ so sánh: Đếm bằng giá trị Start Compare.
4	cmp_start_gt_i	1	Ngõ vào	Cờ so sánh:Đếm lớn hơn giá trị Start Compare.
5	cmp_end_eq_i	1	Ngõ vào	Cờ so sánh: Đếm bằng giá trị End Compare.
6	cmp_end_gt_i	1	Ngõ vào	Cờ so sánh: Đếm lớn hơn giá trị End Compare.
7	mode_i	1	Ngõ vào	Chế độ hoạt động được cấu hình để tạo tín hiệu tham chiếu.
8	oc_a_ref_o	1	Ngõ ra	Tín hiệu tham chiếu Chính (Reference A) cho đầu ra PWM.

9	oc_b_ref_o	1	Ngõ ra	Tín hiệu tham chiếu Bù (Reference B) cho đầu ra PWM.

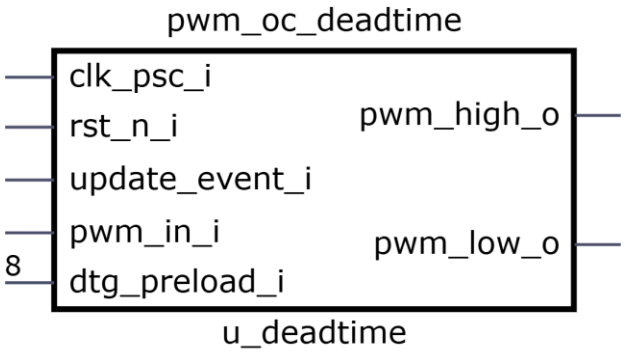
Bảng 3.12 Bảng mô tả các tín hiệu của khối pwm_oc_refgen

Module này thực hiện logic chuyển đổi các tín hiệu so sánh từ bộ đếm thành các tín hiệu oc_a_ref_o và oc_b_ref_o cấp cho đầu ra, theo các chế độ PWM khác nhau dựa vào mode_i (chế độ so sánh đơn (1 cạnh) hoặc chế độ hai cạnh đối xứng).

Ví dụ:

- Trong Mode 0 (chế độ so sánh đơn (1 cạnh)): oc_a_ref_o và oc_b_ref_o có thể được đặt HIGH khi bộ đếm tràn (overflow) và đặt LOW khi đếm bằng giá trị so sánh.
- Trong Mode 1 (chế độ hai cạnh đối xứng): Sử dụng đồng thời cả hai cờ *Start* và *End* để định cạnh lên và xuống của xung oc_a_ref_o, còn oc_b_ref_o là nghịch đảo của oc_a_ref_o.

3.3.2.6. Khối pwm_oc_deadtime



Hình 3.18 Sơ đồ giao diện của khối pwm_oc_deadtime

STT	Tín hiệu	Số bit	Phân loại	Mô tả
1	clk_psc_i	1	Ngõ vào	Xung clock hoạt động của module (Prescaler Clock).
2	rst_n_i	1	Ngõ vào	Tín hiệu reset tích cực thấp.
3	update_event_i	1	Ngõ vào	Tín hiệu sự kiện cập nhật (thường là tràn bộ đếm) để đồng bộ hóa.
4	pwm_in_i	1	Ngõ vào	Tín hiệu PWM đơn đầu vào chưa có thời gian chết (nhận từ pwm_oc_refgen).
5	dtg_preload_i	8	Ngõ vào	Giá trị cài đặt 8-bit cho độ dài Thời gian Chết (Dead Time Value).
6	pwm_high_o	1	Ngõ ra	Tín hiệu đầu ra chính (HIGH side drive) đã được xử lý thời gian chết.
7	pwm_low_o	1	Ngõ ra	Tín hiệu đầu ra bù (LOW side drive) đã được xử lý thời gian chết.

Bảng 3.13 Bảng mô tả các tín hiệu của khối pwm_oc_deadtime

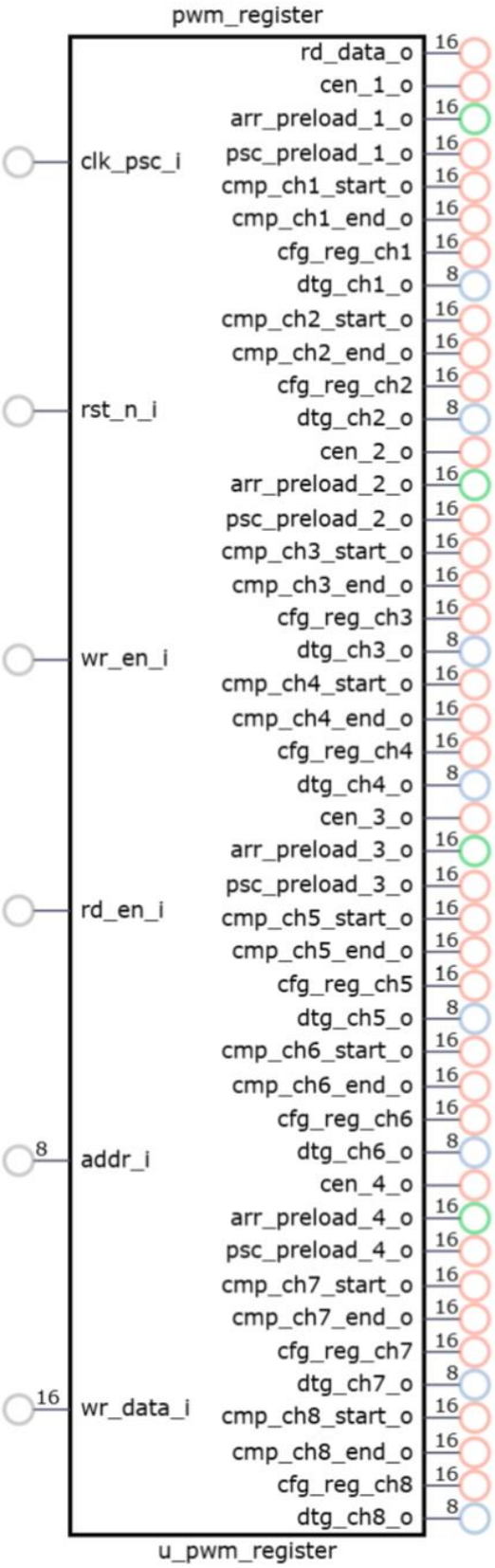
Module pwm_oc_deadtime chuyển đổi một tín hiệu PWM đơn thành hai tín hiệu bù nhau được cách ly bằng một khoảng thời gian trễ ngắn (Dead Time):

Tạo xung bù và trễ:

- Nhận tín hiệu pwm_in_i làm cơ sở. Sau đó tạo ra pwm_high_o và pwm_low_o là hai tín hiệu đảo pha, với pwm_low_o là bù của pwm_high_o.
- Chèn thời gian Chết: Khoảng thời gian dtg_preload_i được chèn vào hai thời điểm quan trọng:

3.4. Khối pwm_register

3.4.1. Sơ đồ khối tổng quát của khối pwm_register



Hình 3.19 Sơ đồ giao diện của khối pwm_register

STT	Tín hiệu	Số bit	Phân loại	Mô tả
1	clk_psc_i	1	Ngõ vào	Xung clock cho bộ điều khiển Prescaler.
2	rst_n_i	1	Ngõ vào	Tín hiệu Reset tích cực mức thấp (Active-low Reset).
3	wr_en_i	1	Ngõ vào	Tín hiệu cho phép ghi (Write Enable) vào các thanh ghi.
4	rd_en_i	1	Ngõ vào	Tín hiệu cho phép đọc (Read Enable) từ các thanh ghi.
5	addr_i	8	Ngõ vào	Địa chỉ (Address) 8-bit để chọn thanh ghi cần truy cập (đọc/ghi).
6	wr_data_i	16	Ngõ vào	Dữ liệu đầu vào 16-bit để ghi vào thanh ghi được chọn.
7	rd_data_o	16	Ngõ ra	Dữ liệu đầu ra 16-bit đọc từ thanh ghi được chọn.
8	cen_1_o	1	Ngõ ra	Tín hiệu cho phép bộ đếm (Counter Enable) cho kênh 1.
9	arr_preload_1_o	16	Ngõ ra	Giá trị thanh ghi Auto-Reload (ARR) (preload) cho kênh 1.
10	psc_preload_1_o	16	Ngõ ra	Giá trị thanh ghi Prescaler (PSC) (preload) cho kênh 1.
11	cmp_ch1_start_o	16	Ngõ ra	Giá trị thanh ghi Compare (CMP) khi bắt đầu chu kỳ cho kênh 1.
12	cmp_ch1_end_o	16	Ngõ ra	Giá trị thanh ghi Compare (CMP) khi kết thúc chu kỳ cho kênh 1.
13	cfg_reg_ch1	8	Ngõ ra	Thanh ghi cấu hình (Configuration Register) 8-bit cho kênh 1.
14	dtg_ch1_o	8	Ngõ ra	Giá trị Dead Time Generator (DTG) 8-bit cho kênh 1.
15	cen_2_o	1	Ngõ ra	Tín hiệu cho phép bộ đếm (Counter Enable) cho kênh 2.
16	arr_preload_2_o	16	Ngõ ra	Giá trị ARR (preload) cho kênh 2.

17	psc_preload_2_o	16	Ngõ ra	Giá trị PSC (preload) cho kênh 2.
18	cmp_ch2_start_o	16	Ngõ ra	Giá trị CMP khi bắt đầu chu kỳ cho kênh 2.
19	cmp_ch2_end_o	16	Ngõ ra	Giá trị CMP khi kết thúc chu kỳ cho kênh 2.
20	cfg_reg_ch2	8	Ngõ ra	Thanh ghi cấu hình 8-bit cho kênh 2.
21	dtg_ch2_o	8	Ngõ ra	Giá trị DTG 8-bit cho kênh 2.
22	cen_3_o	1	Ngõ ra	Tín hiệu cho phép bộ đếm cho kênh 3.
23	arr_preload_3_o	16	Ngõ ra	Giá trị ARR (preload) cho kênh 3.
24	psc_preload_3_o	16	Ngõ ra	Giá trị PSC (preload) cho kênh 3.
25	cmp_ch3_start_o	16	Ngõ ra	Giá trị CMP khi bắt đầu chu kỳ cho kênh 3.
26	cmp_ch3_end_o	16	Ngõ ra	Giá trị CMP khi kết thúc chu kỳ cho kênh 3.
27	cfg_reg_ch3	8	Ngõ ra	Thanh ghi cấu hình 8-bit cho kênh 3.
28	dtg_ch3_o	8	Ngõ ra	Giá trị DTG 8-bit cho kênh 3.
29	cen_4_o	1	Ngõ ra	Tín hiệu cho phép bộ đếm cho kênh 4.
30	arr_preload_4_o	16	Ngõ ra	Giá trị ARR (preload) cho kênh 4.
31	psc_preload_4_o	16	Ngõ ra	Giá trị PSC (preload) cho kênh 4.
32	cmp_ch4_start_o	16	Ngõ ra	Giá trị CMP khi bắt đầu chu kỳ cho kênh 4.
33	cmp_ch4_end_o	16	Ngõ ra	Giá trị CMP khi kết thúc chu kỳ cho kênh 4.
34	cfg_reg_ch4	8	Ngõ ra	Thanh ghi cấu hình 8-bit cho kênh 4.
35	dtg_ch4_o	8	Ngõ ra	Giá trị DTG 8-bit cho kênh 4.
36	cen_5_o	1	Ngõ ra	Tín hiệu cho phép bộ đếm cho kênh 5.
37	arr_preload_5_o	16	Ngõ ra	Giá trị ARR (preload) cho kênh 5.

38	psc_preload_5_o	16	Ngõ ra	Giá trị PSC (preload) cho kênh 5.
39	cmp_ch5_start_o	16	Ngõ ra	Giá trị CMP khi bắt đầu chu kỳ cho kênh 5.
40	cmp_ch5_end_o	16	Ngõ ra	Giá trị CMP khi kết thúc chu kỳ cho kênh 5.
41	cfg_reg_ch5	8	Ngõ ra	Thanh ghi cấu hình 8-bit cho kênh 5.
42	dtg_ch5_o	8	Ngõ ra	Giá trị DTG 8-bit cho kênh 5.
43	cen_6_o	1	Ngõ ra	Tín hiệu cho phép bộ đếm cho kênh 6.
44	arr_preload_6_o	16	Ngõ ra	Giá trị ARR (preload) cho kênh 6.
45	psc_preload_6_o	16	Ngõ ra	Giá trị PSC (preload) cho kênh 6.
46	cmp_ch6_start_o	16	Ngõ ra	Giá trị CMP khi bắt đầu chu kỳ cho kênh 6.
47	cmp_ch6_end_o	16	Ngõ ra	Giá trị CMP khi kết thúc chu kỳ cho kênh 6.
48	cfg_reg_ch6	8	Ngõ ra	Thanh ghi cấu hình 8-bit cho kênh 6.
49	dtg_ch6_o	8	Ngõ ra	Giá trị DTG 8-bit cho kênh 6.
50	cen_7_o	1	Ngõ ra	Tín hiệu cho phép bộ đếm cho kênh 7.
51	arr_preload_7_o	16	Ngõ ra	Giá trị ARR (preload) cho kênh 7.
52	psc_preload_7_o	16	Ngõ ra	Giá trị PSC (preload) cho kênh 7.
53	cmp_ch7_start_o	16	Ngõ ra	Giá trị CMP khi bắt đầu chu kỳ cho kênh 7.
54	cmp_ch7_end_o	16	Ngõ ra	Giá trị CMP khi kết thúc chu kỳ cho kênh 7.
55	cfg_reg_ch7	8	Ngõ ra	Thanh ghi cấu hình 8-bit cho kênh 7.
56	dtg_ch7_o	8	Ngõ ra	Giá trị DTG 8-bit cho kênh 7.
57	cen_8_o	1	Ngõ ra	Tín hiệu cho phép bộ đếm cho kênh 8.
58	arr_preload_8_o	16	Ngõ ra	Giá trị ARR (preload) cho kênh 8.

59	psc_preload_8_o	16	Ngõ ra	Giá trị PSC (preload) cho kênh 8.
60	cmp_ch8_start_o	16	Ngõ ra	Giá trị CMP khi bắt đầu chu kỳ cho kênh 8.
61	cmp_ch8_end_o	16	Ngõ ra	Giá trị CMP khi kết thúc chu kỳ cho kênh 8.
62	cfg_reg_ch8	8	Ngõ ra	Thanh ghi cấu hình 8-bit cho kênh 8.
63	dtg_ch8_o	8	Ngõ ra	Giá trị DTG 8-bit cho kênh 8.

Bảng 3.14 Bảng mô tả các tín hiệu của khối pwm_register

Module pwm_register là khối giao tiếp cấp cao nhất ở phía nội bộ của chip, chịu trách nhiệm:

- Lưu trữ cấu hình: Lưu trữ tất cả các thông số cấu hình mà Master I²C đã ghi vào thông qua bus địa chỉ (addr_i).
- Phân phối tham số: Cung cấp các giá trị cấu hình tương ứng ra các tín hiệu đầu ra để điều khiển kênh PWM.
- Hỗ trợ đọc: Khi rd_en_i được kích hoạt, nó trả về dữ liệu của thanh ghi được chọn trên rd_data_o.

Đây là khối trung tâm kết nối giữa giao tiếp I²C và logic tạo xung PWM.

3.4.2. Bảng địa chỉ thanh ghi

Thanh ghi Điều khiển Chung (General Control Registers)				
Địa chỉ	**Bit Tác động**	**Kích thước**	**Kênh**	**Mô tả**
8'd0	0	1 bit	1	Tín hiệu Counter Enable cho các kênh 1.
	1	2 bit	2	Tín hiệu Counter Enable cho các kênh 2.
	2	3 bit	3	Tín hiệu Counter Enable cho các kênh 3.
	3	4 bit	4	Tín hiệu Counter Enable cho các kênh 4.
	15:04	Bỏ trống		

Bảng 3.15 Bảng mô tả địa chỉ của thanh ghi điều khiển chung

Thanh ghi Prescaler và Chu kỳ (PSC/ARR Registers)				
Địa chỉ	**Thanh ghi Tác động**	**Kích thước**	**Kênh**	**Mô tả**
8'd1	psc_preload_1_o	16 bit	1	Giá trị Prescaler cho Kênh 1.
8'd2	arr_preload_1_o	16 bit	1	Giá trị Auto-Reload cho Kênh 1.
8'd3	psc_preload_2_o	16 bit	2	Giá trị Prescaler cho Kênh 2.
8'd4	arr_preload_2_o	16 bit	2	Giá trị Auto-Reload cho Kênh 2.
8'd5	psc_preload_3_o	16 bit	3	Giá trị Prescaler cho Kênh 3.
8'd6	arr_preload_3_o	16 bit	3	Giá trị Auto-Reload cho Kênh 3.
8'd7	psc_preload_4_o	16 bit	4	Giá trị Prescaler cho Kênh 4.
8'd8	arr_preload_4_o	16 bit	4	Giá trị Auto-Reload cho Kênh 4.

Bảng 3.16 Bảng mô tả địa chỉ của các thanh ghi Prescaler và ARR

Thanh ghi Kênh PWM (Channel-specific Registers)				
Địa chỉ	**Thanh ghi Tác động**	**Kích thước**	**Kênh**	**Mô tả**
8'd10	cmp_ch1_start_o	16 bit	1	Giá trị Compare (Bắt đầu) cho Kênh 1.
8'd11	cmp_ch1_end_o	16 bit	1	Giá trị Compare (Kết thúc) cho Kênh 1.
8'd12	Dead Time_ch1_o	8 bit	1	Giá trị Dead Time cho Kênh 1.
8'd13	cfg_reg_ch1	8 bit	1	Thanh ghi Cấu hình (CFG) cho Kênh 1.
8'd14	cmp_ch2_start_o	16 bit	2	Giá trị Compare (Bắt đầu) cho Kênh 2.
8'd15	cmp_ch2_end_o	16 bit	2	Giá trị Compare (Kết thúc) cho Kênh 2.
8'd16	Dead Time_ch2_o	8 bit	2	Giá trị Dead Time cho Kênh 2.
8'd17	cfg_reg_ch2	8 bit	2	Thanh ghi Cấu hình (CFG) cho Kênh 2.
8'd18	cmp_ch3_start_o	16 bit	3	Giá trị Compare (Bắt đầu) cho Kênh 3.
8'd19	cmp_ch3_end_o	16 bit	3	Giá trị Compare (Kết thúc) cho Kênh 3.
8'd20	Dead Time_ch3_o	8 bit	3	Giá trị Dead Time cho Kênh 3.
8'd21	cfg_reg_ch3	8 bit	3	Thanh ghi Cấu hình (CFG) cho Kênh 3.
8'd22	cmp_ch4_start_o	16 bit	4	Giá trị Compare (Bắt đầu) cho Kênh 4.
8'd23	cmp_ch4_end_o	16 bit	4	Giá trị Compare (Kết thúc) cho Kênh 4.
8'd24	Dead Time_ch4_o	8 bit	4	Giá trị Dead Time cho Kênh 4.
8'd25	cfg_reg_ch4	8 bit	4	Thanh ghi Cấu hình (CFG) cho Kênh 4.
8'd26	cmp_ch5_start_o	16 bit	5	Giá trị Compare (Bắt đầu) cho Kênh 5.
8'd27	cmp_ch5_end_o	16 bit	5	Giá trị Compare (Kết thúc) cho Kênh 5.
8'd28	Dead Time_ch5_o	8 bit	5	Giá trị Dead Time cho Kênh 5.
8'd29	cfg_reg_ch5	8 bit	5	Thanh ghi Cấu hình (CFG) cho Kênh 5.
8'd30	cmp_ch6_start_o	16 bit	6	Giá trị Compare (Bắt đầu) cho Kênh 6.
8'd31	cmp_ch6_end_o	16 bit	6	Giá trị Compare (Kết thúc) cho Kênh 6.
8'd32	Dead Time_ch6_o	8 bit	6	Giá trị Dead Time cho Kênh 6.

8'd33	cfg_reg_ch6	8 bit	6	Thanh ghi Cấu hình (CFG) cho Kênh 6.
8'd34	cmp_ch7_start_o	16 bit	7	Giá trị Compare (Bắt đầu) cho Kênh 7.
8'd35	cmp_ch7_end_o	16 bit	7	Giá trị Compare (Kết thúc) cho Kênh 7.
8'd36	Dead Time_ch7_o	8 bit	7	Giá trị Dead Time cho Kênh 7.
8'd37	cfg_reg_ch7	8 bit	7	Thanh ghi Cấu hình (CFG) cho Kênh 7.
8'd38	cmp_ch8_start_o	16 bit	8	Giá trị Compare (Bắt đầu) cho Kênh 8.
8'd39	cmp_ch8_end_o	16 bit	8	Giá trị Compare (Kết thúc) cho Kênh 8.
8'd40	Dead Time_ch8_o	8 bit	8	Giá trị Dead Time cho Kênh 8.
8'd41	cfg_reg_ch8	8 bit	8	Thanh ghi Cấu hình (CFG) cho Kênh 8.

Bảng 3.17 Bảng mô tả địa chỉ của các thanh ghi chức năng PWM

CHƯƠNG 4: MÔ PHỎNG VÀ KIỂM THỬ

4.1. Xây dựng testbench

4.1.1. Module test

Testbench cho module test được xây dựng nhằm xác minh chức năng của từng khối riêng lẻ trong thiết kế trước khi tích hợp hệ thống.

Mỗi testbench bao gồm các thành phần chính như bộ sinh xung clock và tín hiệu reset, khối DUT (Device Under Test), bộ sinh stimulus để tạo dữ liệu đầu vào theo các kịch bản kiểm thử, khối monitor để theo dõi tín hiệu đầu ra và trạng thái nội bộ.

Việc kiểm thử được thực hiện độc lập cho từng module, độc lập, không phụ thuộc trạng thái module khác, bao phủ cả các trường hợp hoạt động bình thường. Thông qua mô phỏng, kết quả được đánh giá dựa trên độ đúng chức năng và dạng sóng, qua đó đảm bảo mỗi module đáp ứng yêu cầu thiết kế trước khi chuyển sang giai đoạn kiểm thử ở mức hệ thống.

Để thể hiện chức năng của từng module, bảng dưới đây chỉ trình bày một số testcase kiểm thử chính, mang tính đại diện. Trên thực tế, quá trình kiểm thử đã được thực hiện với đầy đủ các testcase theo kế hoạch. Các testcase kiểm thử chính cho từng module thể hiện trong báo cáo gồm:

Tên module	**Trường hợp kiểm thử**
pwm_prescaler	psc_preload = 0 psc_preload thay đổi từ 0 lên 1 psc_preload thay đổi từ 2 lên 4
pwm_counter	psc_preload = 0, arr_reload = 3 psc_preload = 1, arr_reload =3 arr_reload chuyển từ 3 lên 4
pwm_comparator	cmp_start = 3, cmp_end =7, arr_reload = 7 cmp_start = 2, cmp_end =5, arr_reload = 7
pwm_oc_refgen	Mode = 0 Mode = 1
pwm_oc_deadtime	dead-time = 0 dead-time = 1 dead-time từ 1 lên 2

Bảng 4.1 Các testcase kiểm thử theo từng module

4.1.2. System test

Kiểm thử hệ thống (System test) được thực hiện sau khi tích hợp toàn bộ các module thành một thiết kế hoàn chỉnh nhằm đánh giá tính đúng đắn và sự phối hợp giữa các khối chức năng.

Testbench ở mức hệ thống được xây dựng tại mức top-level, trong đó mô phỏng đầy đủ các khối chính như giao tiếp I²C, các lõi PWM và các thanh ghi cấu hình, đồng

thời giả lập thành phần ngoại vi (I²C Master)để tạo môi trường hoạt động gần với thực tế.

Các kịch bản kiểm thử được thiết kế để kiểm tra quá trình khởi tạo hệ thống, giao tiếp đọc/ghi dữ liệu, ảnh hưởng của các tham số cấu hình đến ngõ ra PWM, cũng như khả năng xử lý các tình huống biên và điều kiện bất thường. Kết quả kiểm thử được đánh giá thông qua tính đúng của chức năng hệ thống, sự ổn định trong vận hành và các chỉ số coverage tổng thể, từ đó đảm bảo thiết kế đáp ứng yêu cầu đề ra trước khi hoàn tất quá trình xác minh.

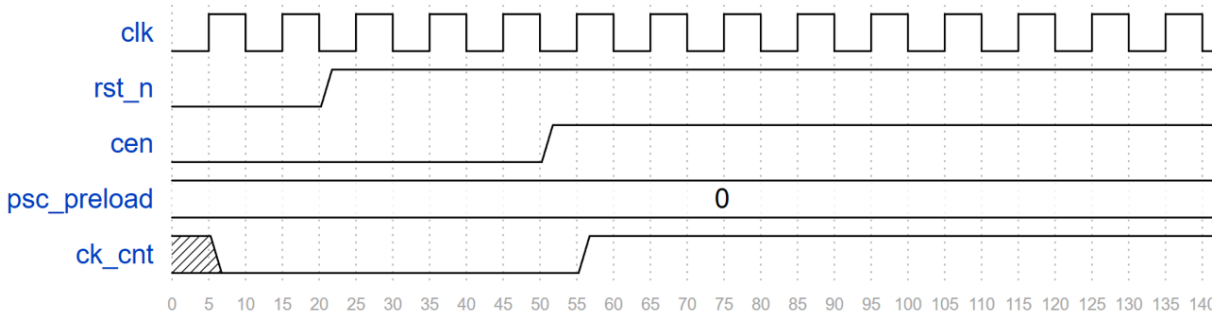
Các testcase kiểm thử cho system test gồm:

Tên testcase	Trường hợp kiểm thử	Kết quả mong muốn
I²C_frame_chk	Ghi 1 byte. Ghi 2 byte. Ghi 3 byte.	Chỉ khi nào ghi 3 byte đúng chuẩn thì dữ liệu mới được thay đổi, còn lại giữ nguyên.
I²C_rw_chk	Ghi – đọc dữ liệu với địa chỉ thanh ghi từ 0 đến 42.	Dữ liệu đọc được đúng với dữ liệu ghi.
	Ghi – đọc dữ liệu với địa chỉ thanh ghi từ 0 đến 42.	Dữ liệu đọc được là 0.
I²C_slv_addr_chk	Thay đổi địa chỉ vật lý của slave.	Chỉ khi nào Master truyền đúng địa chỉ của Slave thì Slave mới phản hồi.
pwm_psc_chk	Hệ số psc lần lược bằng 0, 1, 3, 65535.	Tần số pwm ở đầu ra thay đổi theo giá trị psc được cài đặt.
pwm_arr_chk	Hệ số arr lần lược bằng 9, 65535.	Tần số pwm ở đầu ra thay đổi theo giá trị psc được cài đặt.
pwm_oc_chk	Thay đổi chế độ đầu ra (cực tính, mode, oc_sel, dead-time).	Giá trị đầu ra đúng với cài đặt

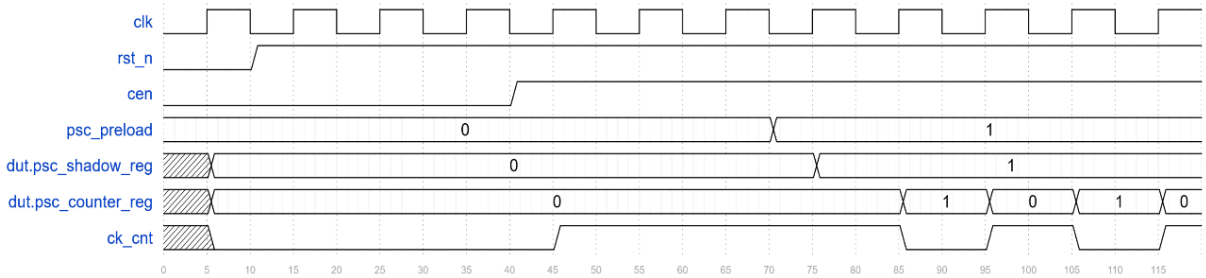
Bảng 4.2 Các testcase kiểm thử cho system test

4.2. Kết quả dạng sóng mô phỏng

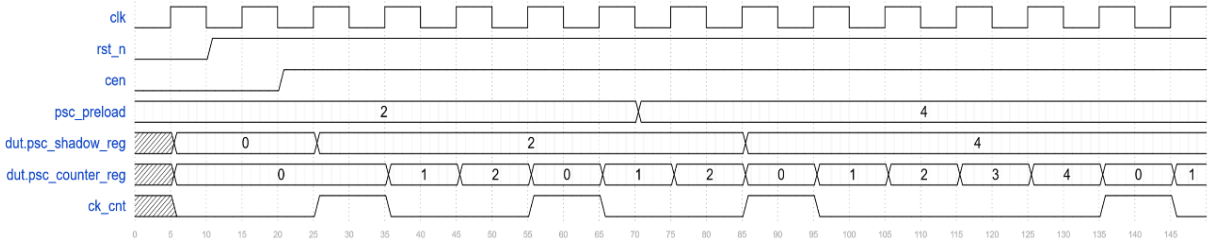
4.2.1. Khối pwm_prescaler



Hình 4.1 Dạng sóng của khối pwm_prescaler khi hệ số chia bằng 0

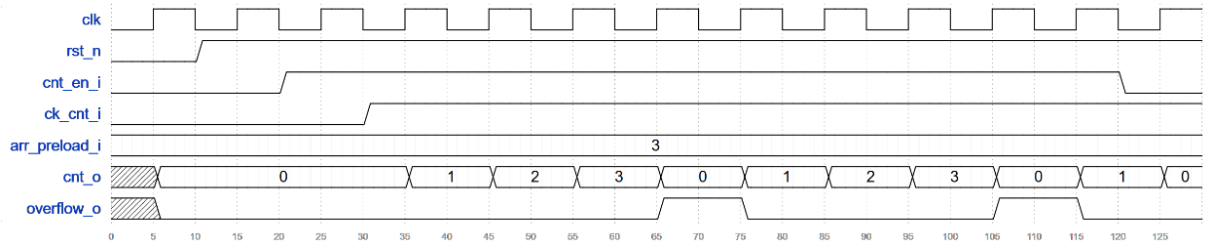


Hình 4.2 Dạng sóng của khối pwm_prescaler khi hệ số chia thay đổi từ 0 lên 1

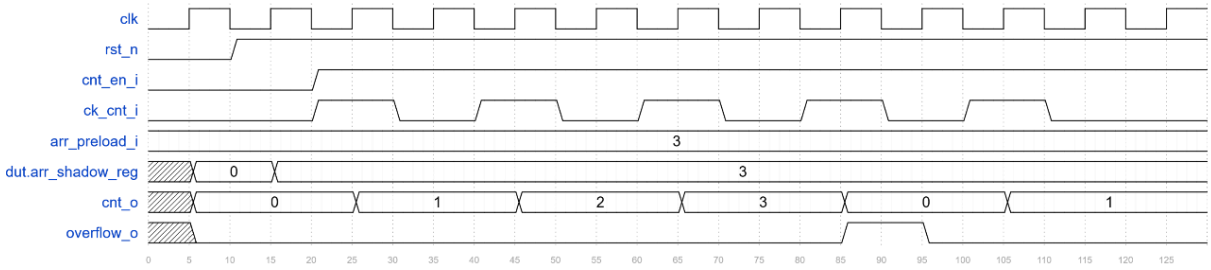


Hình 4.3 Dạng sóng của khối pwm_prescaler khi hệ số chia thay đổi từ 2 lên 4

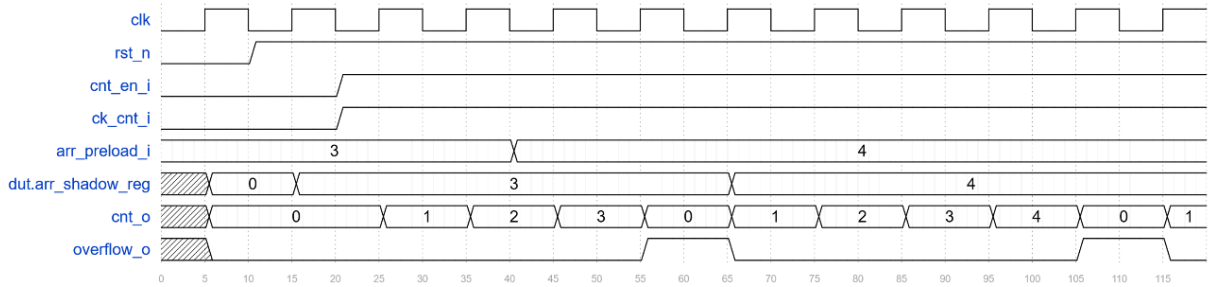
4.2.2. Khối pwm_counter



Hình 4.4 Dạng sóng của khối pwm_counter khi psc_preload = 0, arr_reload = 3

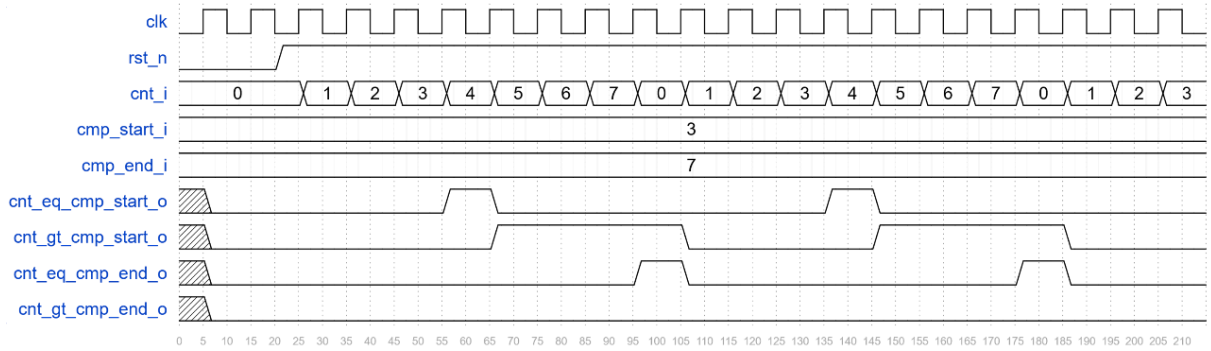


Hình 4.5 Dạng sóng của khối pwm_counter khi psc_preload = 1, arr_reload =3

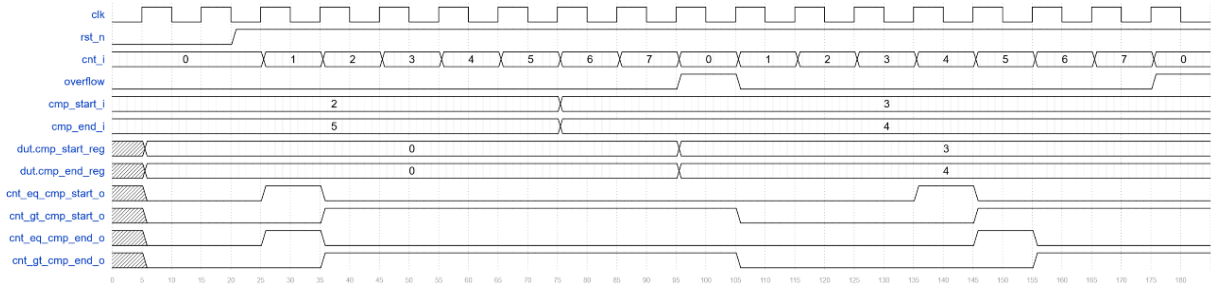


Hình 4.6 Dạng sóng của khối pwm_counter khi arr_reload chuyển từ 3 lên 4

4.2.3. Khối pwm_comparator

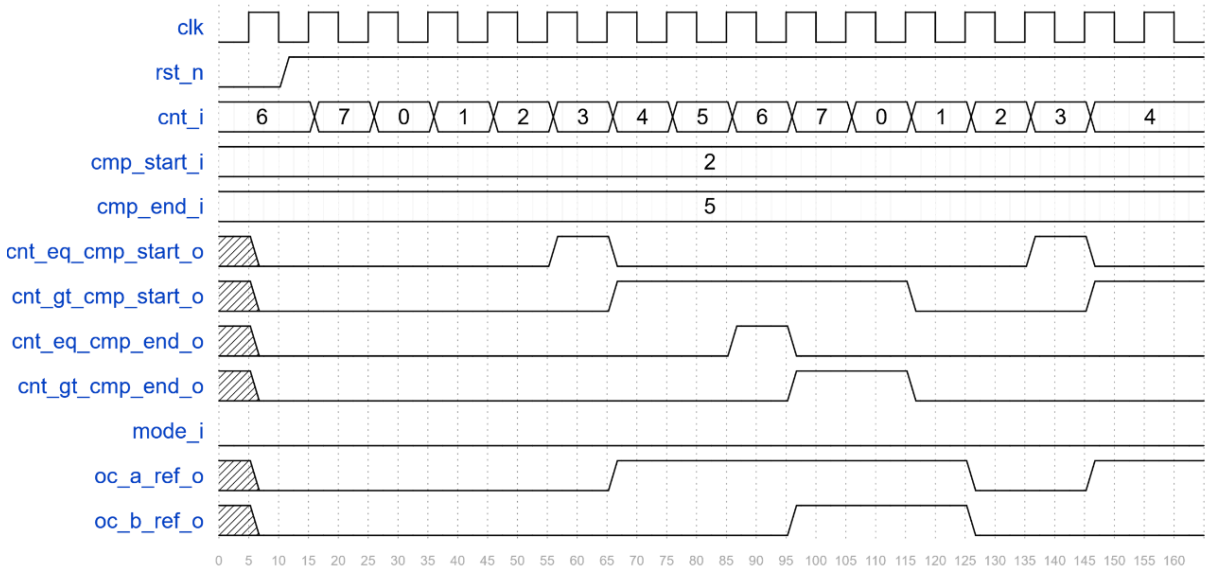


Hình 4.7 Dạng sóng của khối pwm_comparator khi cmp_start = 3, cmp_end =7, arr_reload = 7

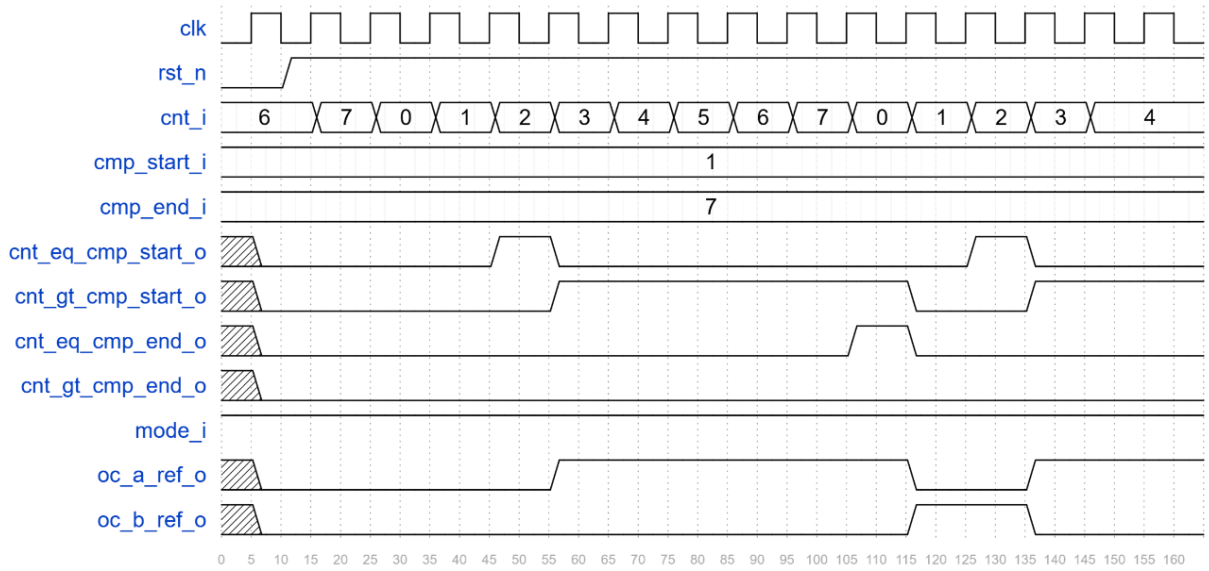


Hình 4.8 Dạng sóng của khối pwm_comparator khi cmp_start = 2, cmp_end =5, arr_reload = 7

4.2.4. Khối pwm_oc_refgen

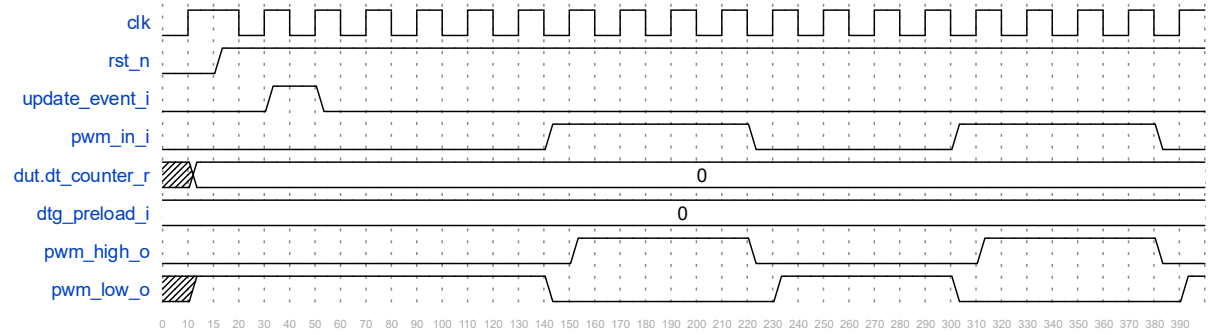


Hình 4.9 Dạng sóng của khối pwm_oc_refgen khi Mode = 0

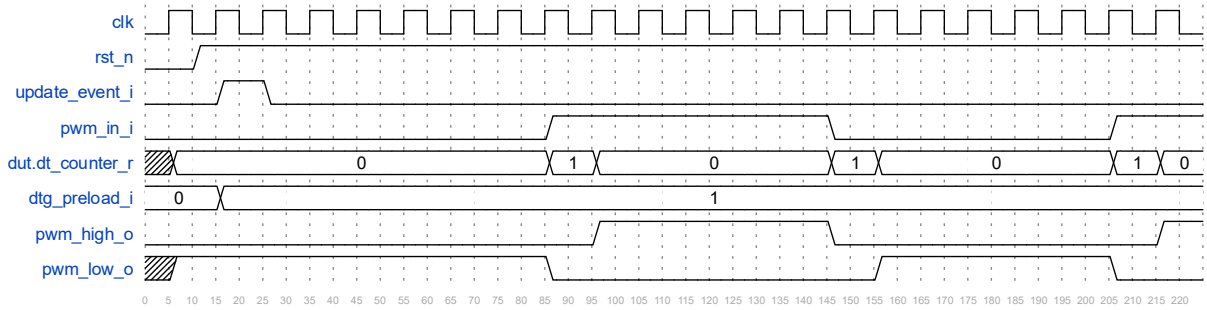


Hình 4.10 Dạng sóng của khối pwm_oc_refgen khi Mode = 1

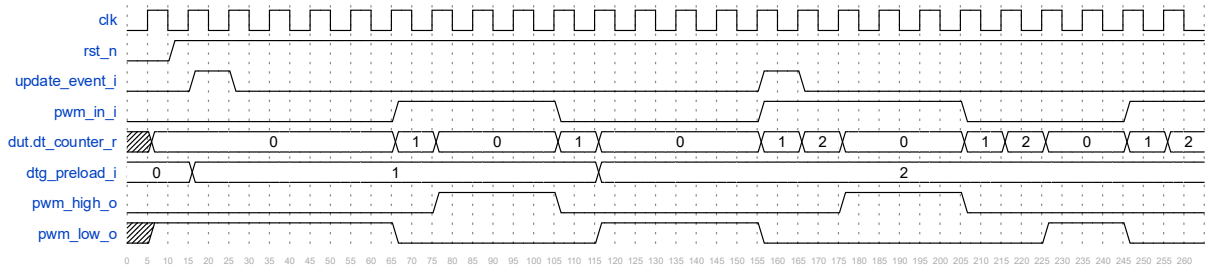
4.2.5. Khối pwm_oc_deadtime



Hình 4.11 Dạng sóng của khối pwm_oc_deadtime khi dead-time = 0



Hình 4.12 Dạng sóng của khối pwm_oc_deadtime khi Dead-time 1



Hình 4.13 Dạng sóng của khối pwm_oc_deadtime khi Dead-time từ 1 lên 2

4.3. Báo cáo code coverage

Code coverage là một chỉ số dùng để đánh giá mức độ mà các đoạn mã trong thiết kế đã được kích hoạt và kiểm tra trong quá trình mô phỏng. Thông qua việc thu thập code coverage, có thể xác định những phần logic đã được kiểm thử đầy đủ cũng như những nhánh, điều kiện hoặc trạng thái chưa được kích hoạt, từ đó làm cơ sở để bổ sung test case và nâng cao độ tin cậy của hệ thống.

Một số loại code coverage chính bao gồm:

- Statement coverage: đo lường tỷ lệ các câu lệnh trong thiết kế được thực thi ít nhất một lần trong quá trình mô phỏng.
- Branch coverage: kiểm tra mức độ bao phủ của các nhánh rẽ (if/else, case), đảm bảo mỗi nhánh logic đều được kích hoạt.
- Condition coverage: đánh giá các điều kiện con trong biểu thức logic được kiểm tra với cả giá trị đúng và sai.
- Expression coverage: phản ánh mức độ bao phủ của các biểu thức logic trong thiết kế.
- FSM coverage: bao gồm FSM state coverage và FSM transition coverage, dùng để kiểm tra việc các trạng thái và chuyển trạng thái của máy trạng thái hữu hạn đã được kích hoạt đầy đủ hay chưa.
- Toggle coverage: theo dõi sự thay đổi giá trị của các tín hiệu (0→1, 1→0), nhằm đánh giá mức độ hoạt động thực tế của các đường tín hiệu trong thiết kế.

Do số lượng module lớn và mỗi module có 3 đến 5 loại độ bao phủ nên kết quả code coverage chỉ thống kê theo từng module chính. Gồm:

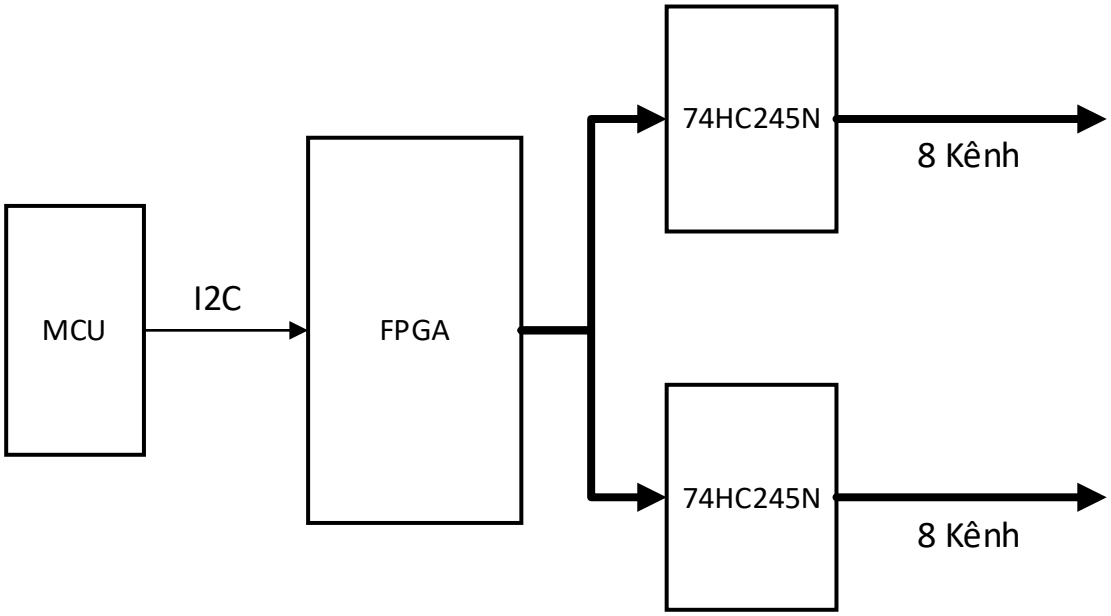
Tên module	**Loại**	**Phần trăm bao phủ**
I²C_top	Branches	100.00%
	Statements	100.00%
	Toggles	93.93%
pwm_register	Branches	100.00%
	Statements	100.00%
	Toggles	93.98%
pwm_core_1	Toggles	96.66%
pwm_core_2	Toggles	96.66%
pwm_core_3	Toggles	96.66%
pwm_core_4	Toggles	96.66%

Bảng 4.3 Bảng tổng hợp độ bao phủ mã

CHƯƠNG 5: TRIỂN KHAI VÀ ĐO ĐẠC TRÊN FPGA

5.1. Cấu trúc tổng thể, tổng hợp và nạp cấu hình lên FPGA

5.1.1. Cấu trúc tổng thể



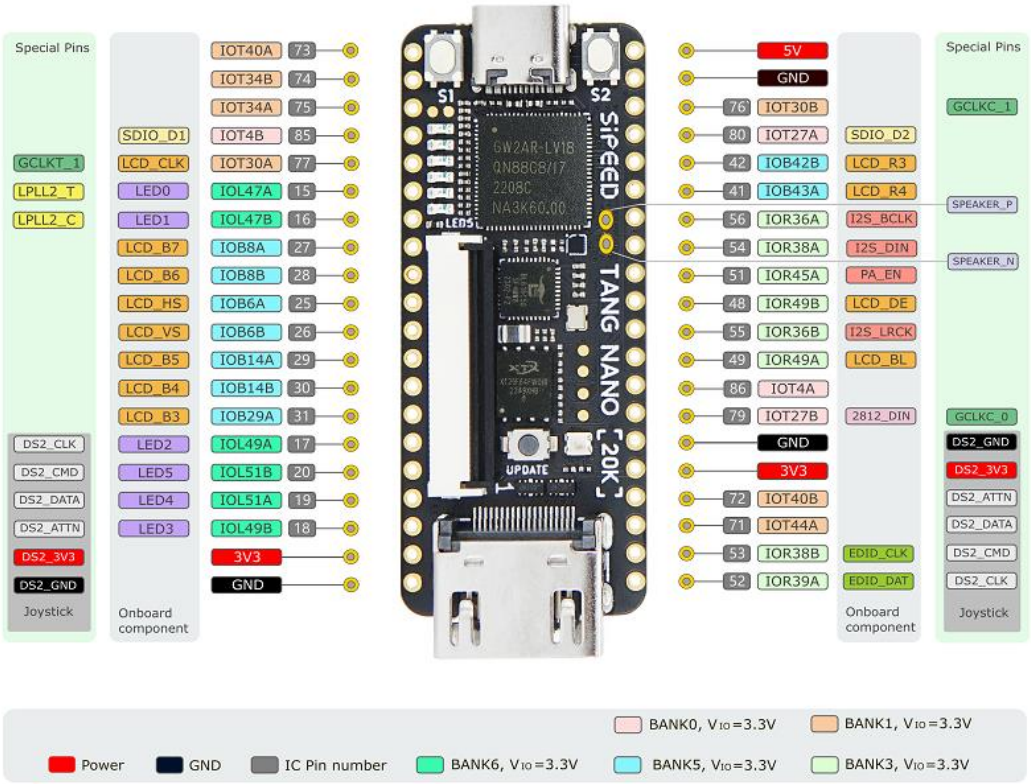
Hình 5.1 Sơ đồ khối mô tả cách MCU (vi điều khiển) giao tiếp với FPGA và cách FPGA điều khiển các đầu ra.

Sơ đồ trên bao gồm các thành phần sau:

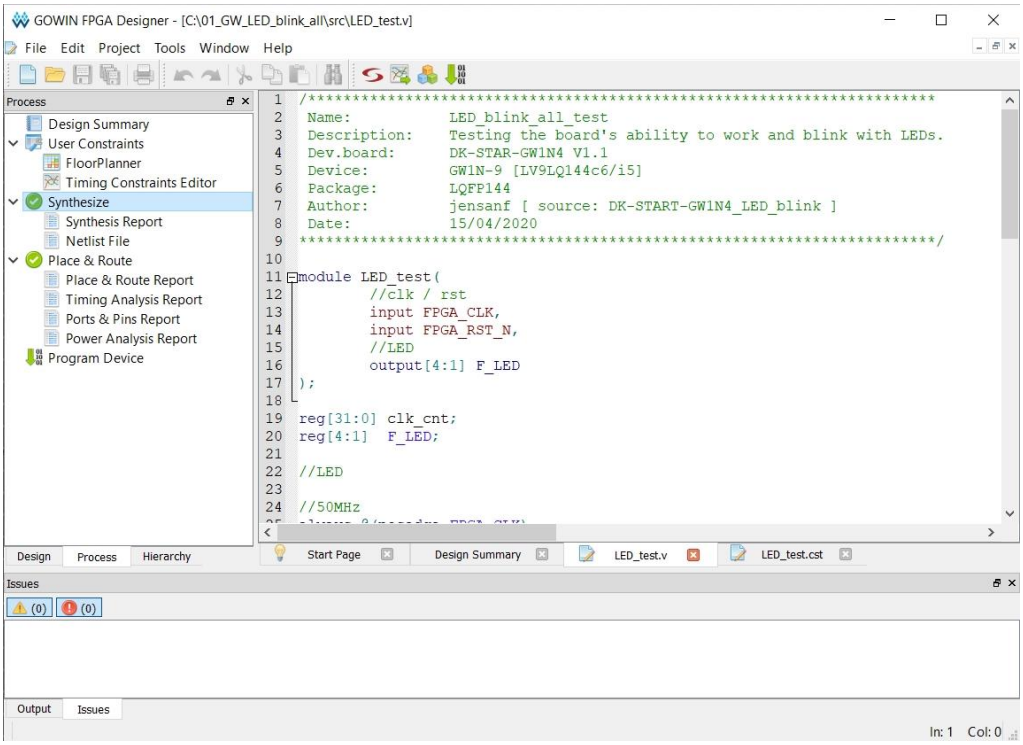
- MCU (Master): MCU sử dụng giao tiếp I²C để cấu hình và điều khiển hệ thống bên trong FPGA.
- FPGA (Slave & Controller): Nhận lệnh từ MCU qua I²C. Thực hiện các chức năng tạo PWM.
- 74HC245N (Bộ Đệm): Đây là chip thu phát bus 8-bit (8-bit bus transceiver). Đóng vai trò là bộ đệm công suất (power buffer) để tăng khả năng truyền tải của tín hiệu ra từ FPGA và tách biệt điện áp bus giữa FPGA và tải (load).

5.1.2. Tổng hợp và nạp cấu hình lên FPGA

Do tiêu chí chính là nhỏ gọn, chi phí thấp, dễ tiếp cận, IDE tương đối đơn giản, Tang Nano 20K là lựa chọn hợp lý với các kit tham chiếu của Xilinx/AMD hay Altera/Intel. Tang Nano 20K là một bo mạch phát triển sử dụng FPGA GW2AR-18 QN88, chứa 20736 khối logic LUT4 và 15552 Flip-Flop. FPGA này có 2 PLL và nhiều khối DSP hỗ trợ phép nhân 18 bit × 18 bit.Tích hợp chip JTAG sẵn trên bo cho việc nạp bitstream vào FPGA. Thạch anh 27 MHz cung cấp xung clock chính cho FPGA.

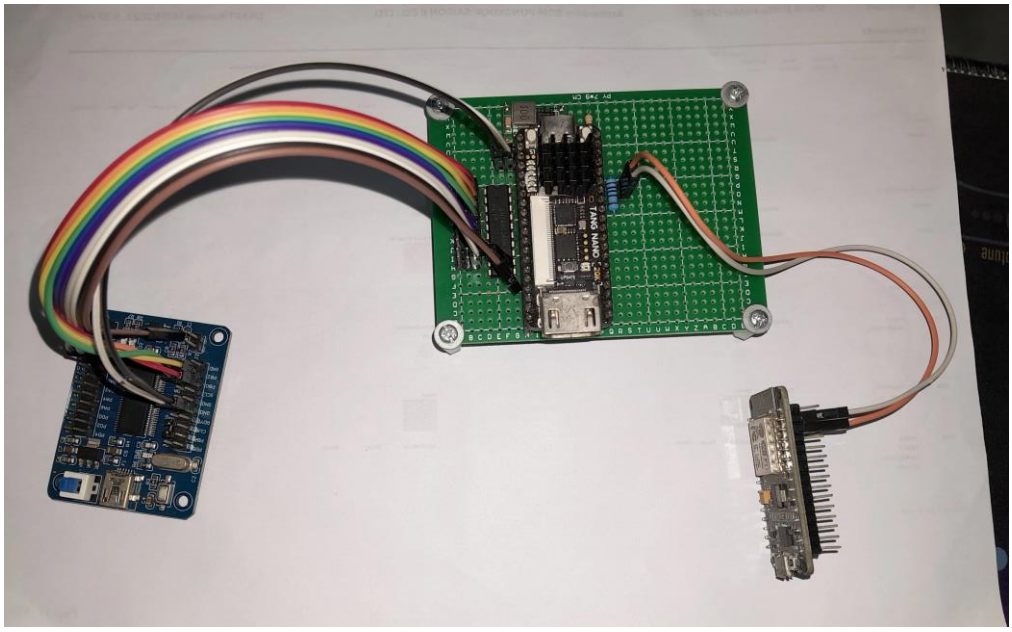


Hình 5.2 Hình ảnh thực tế của Tang Nano 20k



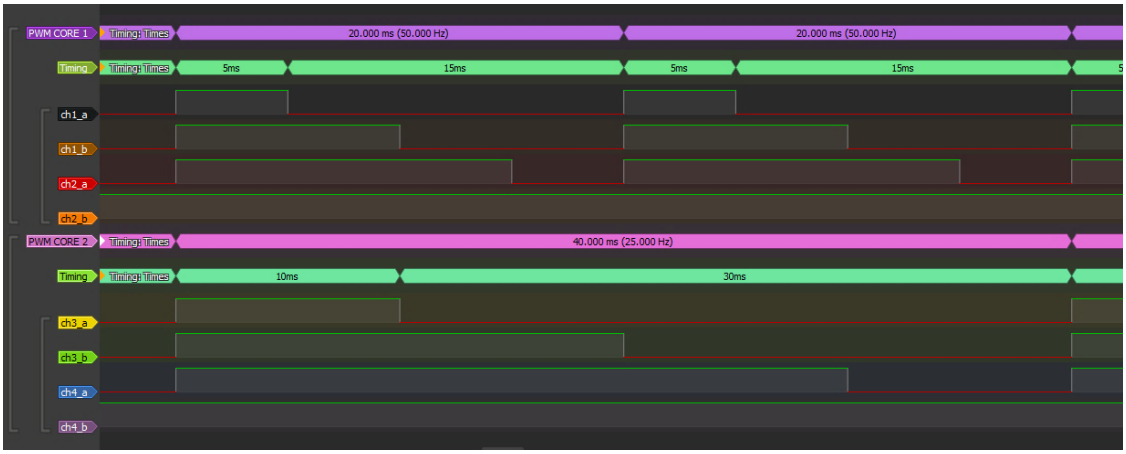
Hình 5.3 Phần mềm Gowin IDE

5.2. Kết quả đo bằng logic analyzer và so sánh với mô phỏng



Hình 5.4 Kết nối phần cứng thực tế

Các thiết bị dùng trong ảnh từ trái sang phải lần lượt là: Logic Analyzer, FPGA Tang Nano 20K, Vi điều khiển ESP8266.



Hình 5.5 Dạng sóng đo được trên logic analyzer

Dạng sóng trên đo được với các cài đặt như sau:

- PWM CORE 1:
  - + PSC = 2399 (giá trị mong muốn là 2400), ARR = 999 (giá trị mong muốn là 1000)
  - + Ch1_cmp_start = 250, Ch1_cmp_end = 500
  - + Ch2_cmp_start = 750, Ch2_cmp_end = 1000

Từ thông số psc và arr ở trên, tính ra được tần số PWM đầu ra của PWM CORE 1 là:

$$\frac{f_{pwm}}{PSC_{val} * ARR_{val}} = \frac{120x10^6}{2400 * 1000} = 50Hz \quad (5.1)$$

- PWM CORE 2:
    + PSC = 4799 (giá trị mong muốn là 4800), ARR = 999 (giá trị mong muốn là 1000),
    + Ch3_cmp_start = 250, Ch3_cmp_end = 500
    + Ch4_cmp_start = 750, Ch4_cmp_end = 1000

Từ thông số psc và arr ở trên, tính ra được tần số PWM đầu ra của PWM CORE 1 là:

$$\frac{f_{pwm}}{PSC_{val} * ARR_{val}} = \frac{120x10^6}{4800 * 1000} = 25Hz \quad (5.2)$$

Dựa vào hình ảnh dạng sóng đo được trên logic analyzer, thấy được tần số của PWM CORE 1 là 50Hz và tần số của PWM CORE 2 là 25Hz. Đồng thời độ rộng xung của từng tín hiệu đầu ra đều đúng với thông số đã cài đặt (25%, 50%, 75%, 100%).

CHƯƠNG 6: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

6.1. Kết quả đạt được:

Các mục tiêu thiết kế chính đã được hoàn thành với các kết quả cụ thể sau:

- Thiết kế giao tiếp I²C Slave: Các chức năng Đọc (Read), Ghi (Write) đã được mô hình hóa và kiểm tra. Có thể thay đổi địa chỉ Slave thông qua chân vật lý bên ngoài.
- Thiết kế bộ điều khiển PWM đa kênh: Các khối logic điều khiển PWM với tổng cộng 16 Kênh đầu ra, bao gồm các module cốt lõi (Prescaler, Counter, Register) và logic đầu ra (Comparator, Deadtime, Output Compare).
- Về phần mềm: Kết quả xác minh (Verification) có mức bao phủ tổng thể (Total Coverage By Instance) đạt 85.89%. Các module cốt lõi như bộ lọc I²C, bộ đếm PWM (pwm_counter, pwm_prescaler) và thanh ghi (pwm_register) đạt 100% Code Coverage đối với Branches và Statements.
- Về phần cứng: Đã triển khai thành công logic I²C Slave trên FPGA (sử dụng chip GW2AR-18 của Tang Nano 20K) để nhận lệnh và cấu hình từ MCU Master.Các tín hiệu từ các kênh PWM đầu ra được đưa qua các bộ đệm 74HC245N, sau đó được được quan sát và kiểm tra bằng logic analyzer, xác nhận biên dạng, chu kỳ và timing phù hợp với thiết kế.

6.2. Đánh giá và kết luận:

Thiết kế là ổn định về mặt chức năng cốt lõi và sẵn sàng cho việc triển khai, nhưng cần tăng cường các bài kiểm tra xác minh (đặc biệt cho các kênh đầu ra PWM và các chuyển đổi trạng thái I²C) để đạt mức độ tin cậy tối đa.

6.3. Khó khăn và bài học kinh nghiệm:

Các khó khăn chính đã gặp khi làm gồm:

- Đảm bảo bao phủ Toggles (Toggle Coverage): Khó khăn lớn nhất là khiến các tín hiệu đầu ra của bộ so sánh PWM (pwm_comparator) chuyển đổi đầy đủ (từ 0 sang 1 và ngược lại) trong môi trường mô phỏng.
- Xác minh chuyển đổi FSM I²C: Khó khăn trong việc thiết lập kịch bản kiểm tra kích hoạt tất cả các chuyển đổi trạng thái (Transitions) của FSM (Finite State Machine) trong module I²C Slave.
- Thiết kế Testbench hiệu quả: Môi trường Testbench (/tb) chưa đủ linh hoạt để bao phủ tất cả các điều kiện logic.

Bài học kinh nghiệm rút ra được là:

- Tập trung vào Edge Cases: Các bài kiểm tra phải được thiết kế tập trung vào các trường hợp biên của tham số (ví dụ: Duty Cycle 0% và 100% cho PWM) thay vì chỉ các giá trị trung bình để kích hoạt Toggles và các nhánh mã ít dùng.

- Kiểm tra trạng thái và lỗi: Cần ưu tiên thiết kế các kịch bản lỗi bus I²C hoặc các giao dịch phức tạp (như Repeated Start) để đảm bảo tất cả các chuyển đổi FSM được kiểm tra.
- Tối ưu hóa cấu trúc Testbench: Cấu trúc của Testbench cần được tối ưu hóa để dễ dàng mở rộng và tạo ra các tổ hợp đầu vào phức tạp nhằm kích hoạt các điều kiện logic bị bỏ sót.

6.4. Hạn chế và hướng phát triển tương lai:

Một số hạn chế hiện tại gồm:

- Khoảng trống Coverage (Verification Gap): Tổng mức bao phủ chưa đạt 100%, có rủi ro tiềm ẩn ở các nhánh mã và điều kiện logic chưa được kiểm tra.
- Thiếu tính năng I²C nâng cao: I²C Slave hiện tại có thể chỉ hỗ trợ các giao dịch cơ bản mà chưa bao gồm các tính năng nâng cao như Interrupt.

Các hướng phát triển tương lai có:

- Hoàn thiện Xác minh (Verification Completion):
  + Mục tiêu 100%: Thiết kế các Testcase tập trung vào các khu vực yếu kém để đạt 100% Branches/Statements Coverage.
  + Tăng cường Toggles: Thiết kế các kịch bản thay đổi Duty Cycle liên tục và đột ngột để đẩy Toggles Coverage của PWM lên mức chấp nhận được.

Thực nghiệm trên các phần cứng khác: Triển khai Bitstream lên một số loại bo FPGA khác và thực hiện đo lường các tín hiệu PWM/I²C bằng máy hiện sóng (Oscilloscope) để xác nhận tần số, độ chính xác thời gian (timing).

Nâng cấp giao tiếp: Nâng cấp module I²C Slave để hỗ trợ các tính năng nâng cao như Interrupts để thông báo sự kiện (event) về cho MCU Master mà không cần Master liên tục dò hỏi (polling).

PHỤ LỤC

Folder source code:

https://github.com/mikoyangenisysvn/Final_Years_Proj/tree/main/RTL_15_12/code

DANH MỤC TỪ VIẾT TẮT

Từ viết tắt	Thuật ngữ Tiếng Anh	Giải thích Chức năng
I²C	Inter-Integrated Circuit	Giao thức truyền thông nối tiếp tốc độ thấp được sử dụng để cấu hình và điều khiển chip.
PWM	Pulse Width Modulation	Điều chế độ rộng xung, phương pháp điều khiển công suất bằng cách thay đổi độ rộng của xung điện.
PLL	Phase-Locked Loop	Vòng khóa pha, tạo ra các xung clock ổn định và có tần số khác từ xung clock cơ bản.
ARR	Auto-Reload Register	Thanh ghi tự động nạp lại, giá trị giới hạn của bộ đếm, xác định chu kỳ (period) của xung PWM.
PSC	Prescaler	Bộ chia tần số, module chia xung clock đầu vào để tạo ra xung clock cho bộ đếm.
CMP	Compare	Ám chỉ logic so sánh giá trị đếm với ngưỡng cài đặt để xác định thời điểm bật/tắt xung.
DTG	Dead Time Generator	Bộ tạo thời gian chết, chèn một khoảng thời gian trễ giữa xung chính và xung bù để bảo vệ mạch công suất.
OC	Output Compare	Điều khiển đầu ra, logic chịu trách nhiệm tạo ra xung PWM cuối cùng dựa trên các cờ so sánh.
CEN	Counter Enable	Tín hiệu cho phép bộ đếm, điều khiển bật/tắt hoạt động của từng lõi PWM Core.
CFG	Configuration	Cấu hình, ám chỉ các thanh ghi chứa các cài đặt.
SCL	Serial Clock Line	Đường xung đồng hồ của giao tiếp I²C.
SDA	Serial Data Line	Đường dữ liệu hai chiều của giao tiếp I²C.
Fm+	Fast Mode Plus	Chế độ tốc độ cao của I²C, thường hỗ trợ lên đến 1 MHz.
RTL	Register Transfer Level	Mức thanh ghi, phong cách mô tả phần cứng được sử dụng trong mã Verilog.
TB	Testbench	Môi trường kiểm tra mô phỏng được tạo ra để xác minh chức năng của module thiết kế.
DUT	Design Under Test	Thiết kế đang được kiểm tra.

TÀI LIỆU THAM KHẢO

[1] ewskills, "ewskills," ewskills, [Online]. Available: https://www.ewskills.com/arduino/pwm. [Accessed 03 09 2025].

[2] K. Magdy, "deepbluembedded," deepbluembedded, [Online]. Available: https://deepbluembedded.com/stm32-pwm-example-timer-pwm-mode-tutorial/. [Accessed 10 10 2025].

[3] N. Semiconductors, "I²C-bus specification and user manual," 1 10 2021. [Online]. Available: https://www.nxp.com/docs/en/user-guide/UM10204.pdf. [Accessed 30 10 2025].

[4] S. Afzal, "I²C Primer: What is I²C? (Part 1)," Analog Devices, 2 9 2016. [Online]. Available: https://www.analog.com/en/resources/technical-articles/I²C-primer-what-is-I²C-part-1.html. [Accessed 30 9 2025].

[5] J. Wu, "A Basic Guide to I²C," 11 2022. [Online]. Available: https://www.ti.com/lit/an/sbaa565/sbaa565.pdf. [Accessed 11 11 2025].

[6] sipeed, "Tang Nano 20K," sipeed, 27 02 2023. [Online]. Available: https://wiki.sipeed.com/hardware/en/tang/tang-nano-20k/nano-20k.html. [Accessed 11 11 2025].

[7] sipeed, "Tang Nano 20K Blink led," sipeed, 24 05 2023. [Online]. Available: https://wiki.sipeed.com/hardware/en/tang/tang-nano-20k/nano-20k.html. [Accessed 20 11 2025].