THIẾT KẾ VÀ MÔ HÌNH HOÁ KHỐI GIAO TIẾP TRUYỀN THÔNG NỐI TIẾP ĐỒNG BỘ DÙNG TRONG VI ĐIỀU KHIỂN

Phan Hải Phong*, Hoàng Lê Hà,Nguyễn Văn Ân, Hồ Đức Tâm Linh
Khoa Điện tử - Viễn thông, Trường Đại học Khoa học – Đại học Huế
**Email:phongph@husc.edu.vn*

TÓM TẮT

Các dòng vi điều khiển thế hệ mới luôn yêu cầu phải có một giao thức truyền thông phù hợp, cho phép các vi điều khiển này có thể giao tiếp với các thiết bị ngoại vi hoặc với các nền tảng khác. Serial Peripheral Interface(SPI) là một giao thức truyền thông nối tiếp đồng bộ cho phép các thiết bị có thể kết nối với nhau dễ dàng mà không yêu cầu quá nhiều tài nguyên của hệ thống. Bài báo này tập trung vào việc nghiên cứu, thiết kế và mô hình hoá một khối điều khiển giao tiếp theo giao thức SPI để xây dựng một khối giao tiếp cho hệ vi điều khiển đang được thiết kế. Các kết quả mô phỏng hoạt động ở mức logic của khối giao tiếp này cũng được trình bày cụ thể trong bài báo.

Từ khóa: *SPI, vi điều khiển, vi xử lý.*

1. MỞ ĐẦU

Ngày nay, với sự phát triển của công nghệ bán dẫn thì việc nghiên cứu và phát triển các thế hệ vi mạch tích hợp (Integrated Circuit - IC) mới là một sự phát triển tất yếu của ngành công nghiệp điện tử hiện đại. Các vi mạch mới ra đời ngày càng có hiệu năng cao, chất lượng vượt trội và giá thành giảm. Với kích thước các transistor càng được thu nhỏ thì các vi mạch càng được tích hợp thêm nhiều tính năng trên cùng một đế silic nhưng vẫn đảm bảo được tốc độ hoạt động cao và kích thước nhỏ gọn. Bên cạnh đó, sự phát triển của các hệ thống điện tử, tự động hoá, các hệ thống nhúng đã đặt ra yêu cầu về việc cần phải có các thế hệ vi điều khiển (Microcontroller Unit – MCU) mới có tốc độ hoạt động cao và tích hợp thêm nhiều thiết bị ngoại vi. Điều này đặt ra cho lĩnh vực thiết kế vi mạch điện tử một hướng nghiên cứu quan trọng, đó là phát triển các thế hệ vi điều khiển mới có tốc độ cao hơn, hoạt động ổn định hơn và đặc biệt là phải tích hợp được thêm nhiều tính năng tiên tiến hơn.

Nghiên cứu, thiết kế và chế tạo các thế hệ vi điều khiển mới luôn là một hướng nghiên cứu nhận được nhiều sự chú ý không chỉ của các nhà nghiên cứu mà còn có cả các công ty thiết kế và sản xuất vi mạch. Các dòng vi điều khiển mới không chỉ được nâng cao về tốc độ, số bit xử lý song song như các dòng PIC16, PIC24, PIC32 của Microchip [1][2], mà còn phải được tích hợp thêm nhiều khối chức năng đặc biệt khác như các dòng PSOC (Programmbale System-on-Chip) của Cypress [3]. Bên cạnh đó, các nghiên cứu mới không chỉ hướng tới việc tích hợp thêm nhiều tính năng đặc biệt cho vi điều khiển như: điều khiển fuzzy logic [4], tích hợp các bộ

giao tiếp không dây Radio-frequency identification (RFID) [5][6], mà những nghiên cứu này còn hướng đến việc thiết kế vi điều khiển với những công nghệ mới để giảm thiểu năng lượng tiêu thụ [7][8][9]...

Các thế hệ MCU mới ra đời đồng thời cũng đặt ra yêu cầu cần phải có các phương thức giao tiếp tương ứng để hỗ trợ kết nối giữa MCU với những thiết bị ngoại vi khác. Để có thể xây dựng nên một lõi vi điều khiển có khả năng ứng dụng cao thì một khối giao tiếp truyền thông như thế cần phải được tích hợp vào trong lõi vi xử lý (Central Processing Unit - CPU) nhằm giúp lõi đó giao tiếp với các thiết bị ngoại vi thông dụng. Hiện nay, hai phương thức truyền thông phổ biến hiện đang được áp dụng nhiều cho các dòng MCU đó chính là Inter-Integrated Circuit (I2C) và Serial Peripheral Interface (SPI).

Chuẩn giao tiếp I²C [10] được giới thiệu lần đầu tiên vào năm 1982 với mục đích là cung cấp một phương thức thuận tiện để cho phép kết nối giữa CPU với các vi mạch ngoại vi của ti-vi. Trước đây, một phương pháp phổ biến để thực hiện việc kết nối này đó là sử dụng các bus địa chỉ và dữ liệu song song. Nhưng cùng với sự phát triển của các thiết bị ngoại vi thì số lượng các thiết bị cần kết nối tăng lên, điều này dẫn đến việc phải tăng thêm số lượng dây dẫn bus trên mạch in, cũng như cần thêm các tín hiệu điều khiển khác. Để tiết kiệm số lượng chân của MCU, cũng như để đơn giản hoá việc thiết kế mạch in thì phòng thí nghiệm Philips (Eindhoven – Hà Lan) đã phát minh ra chuẩn giao tiếp I²C cho phép kết nối với các thiết bị ngoại vi chỉ với hai đường tín hiệu và tốc độ truyền thông đạt từ 100Kbps cho đến 3.4 Mbps (đối với các ngoại vi yêu cầu tốc độ cao). SPI cũng là một chuẩn truyền thông nối tiếp đồng bộ được Motorola phát triển và lần đầu tiên được giới thiệu vào năm 1979 (tích hợp trên vi điều khiển Motorola 68000) [11]. Khác với chuẩn I^2C, chuẩn giao tiếp SPI không có một đặc tả chính xác mà chỉ là các đặc tả tham chiếu và việc phát triển ứng dụng là hoàn toàn phụ thuộc vào người thiết kế. Chính vì vậy mà người dùng hoàn toàn có thể thiết kế và tuỳ biến chuẩn giao tiếp này theo ý của mình để đạt được hiệu quả sử dụng cao nhất.

Trong bài báo này, chúng tôi tập trung vào việc nghiên cứu về chuẩn giao tiếp SPI, từ đó xây dựng mô hình và mô hình hoá một khối điều khiển giao tiếp SPI bằng ngôn ngữ mô tả phần cứng VHDL. Mô hình của khối giao tiếp này sau khi được mô hình hoá sẽ được mô phỏng để kiểm tra hoạt động ở mức logic trên phần mềm Model Sim.

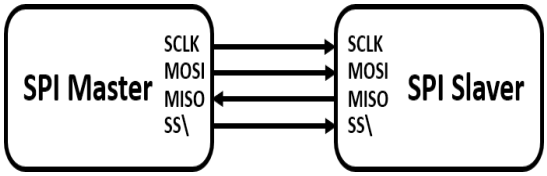
2. THIẾT KẾ VÀ MÔ HÌNH HÓA LÕI GIAO TIẾP NỐI TIẾP SPI

2.1 Chuẩn giao tiếp truyền thông nối tiếp đồng bộ SPI

Phương thức truyền thông SPI được thực hiện thông qua giao tiếp bốn đường tín hiệu như ở 0[12]. Các tín hiệu này được mô tả cụ thể như sau:

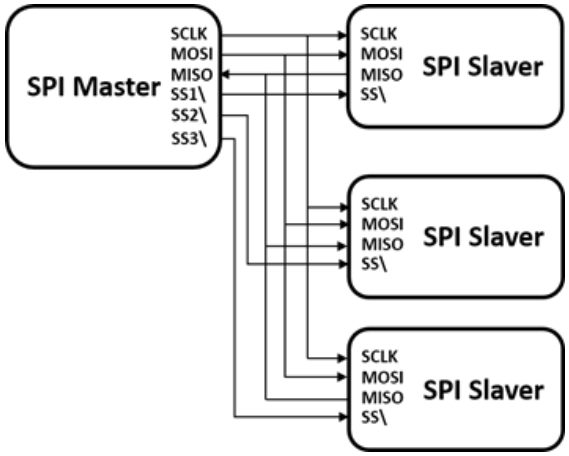
- Tín hiệu xung nhịp (SCLK): Tín hiệu này được gửi từ thiết bị chủ (Master) đến tất cả các thiết bị tớ (Slaver). Tất cả các tín hiệu khác trong SPI phải đồng bộ theo tín hiệu xung nhịp này.

- Tín hiệu dữ liệu nối tiếp từ thiết bị Master đến thiết bị Slaver - Master Out Slaver In (MOSI).
- Tín hiệu dữ liệu nối tiếp từ thiết bị Slaver đến thiết bị chủ - Master In Slaver Out (MISO).
- Tín hiệu lựa chọn thiết bị Slaver SS\: đây là tín hiệu được sử dụng để chọn ra thiết bị Slaver nào sẽ kết nối với thiết bị Master.



Hình 1. Mô hình giao tiếp giữa hai thiết bị theo chuẩn SPI.

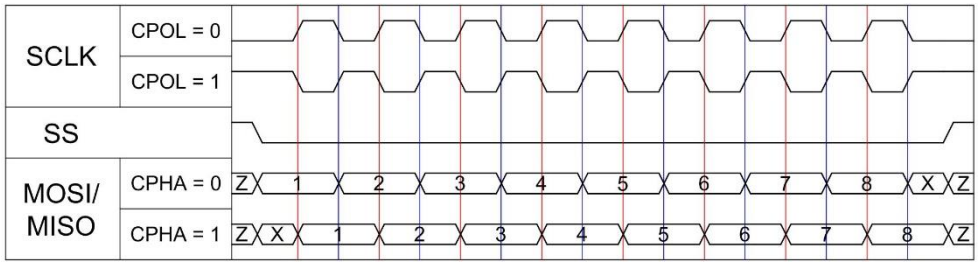
Chuẩn SPI là một chuẩn giao tiếp nối tiếp đồng bộ cho phép kết nối giữa hai thiết bị theo mô hình Master – Slaver (master – slave). Theo mô hình này, một thiết bị trung tâm (thiết bị Master) sẽ có thể kết nối được với nhiều thiết bị Slaver. Có nhiều kỹ thuật khác nhau để một thiết bị Master có thể bắt tay với thiết bị Slaver, tuy nhiên phương thức phổ biến nhất đó là mỗi thiết bị Slaver sẽ có một tín hiệu lựa chọn SS\ khác nhau như 0. Các tín hiệu SCLK, MOSI và MISO sẽ được sử dụng chung để tiết kiệm tài nguyên. Để bắt tay với một thiết bị Slaver, thiết bị Master sẽ phát ra tín hiệu lựa chọn thiết bị bằng cách đưa tín hiệu lựa chọn (SS\) tương ứng với thiết bị đó xuống mức thấp để báo hiệu thiết bị Slaver nào được chọn. Đồng thời nó cũng kích hoạt tín hiệu xung nhịp để đồng bộ giữa thiết bị Master và thiết bị Slaver đó. Thiết bị Master sẽ tiến hành gửi dữ liệu nối tiếp cho thiết bị Slaver trên đường tín hiệu MOSI và nhận lại dữ liệu trên đường tín hiệu MISO.



Hình 2. Mô hình giao tiếp Master - Slaver của chuẩn giao tiếp SPI.

Chuẩn SPI có tổng cộng bốn chế độ hoạt động, được phân biệt dựa trên hai thông số: CPOL (chỉ thị mức của tín hiệu xung) và CPHA (chỉ thị pha của tín hiệu xung). Các thiết bị Master và thiết bị Slaver phải sử dụng chung một chế độ để giao tiếp được chính xác. Nếu CPOL có giá trị "không" thì tín hiệu SCLK thường ở mức thấp và xung đầu tiên được tính tại

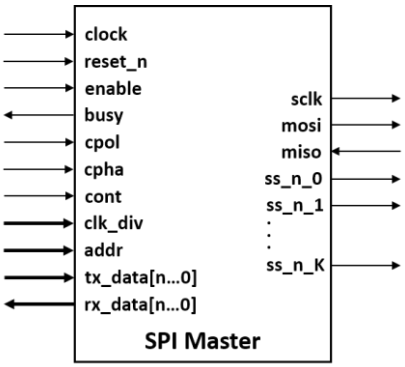
cạnh lên của tín hiệu. Nếu CPOL có giá trị "một" thì tín hiệu SCLK thường ở mức cao và xung khởi đầu được tính tại cạnh xuống của tín hiệu. Nếu CPHA có giá trị "không", bit đầu tiên của dữ liệu sẽ được ghi lên bus tại thời điểm cạnh xuống của xung SS\, và sẽ được đọc tại sườn đầu tiên của xung SCLK. Nếu CPHA có giá trị "một", dữ liệu sẽ được ghi tại sườn đầu tiên của xung SCLK và được đọc tại sườn thứ hai của xung đó. Giản đồ thời gian của bốn chế độ hoạt động được minh hoạ như trong 0.



Hình 3. Giản đồ xung của bốn chế độ hoạt động của giao thức SPI.

2.2 Xây dựng mô hình khối giao tiếp SPI Master (SPI Master - SPIm)

Để thiết bị Master có thể giao tiếp được với các thiết bị Slaver qua giao thức SPI, một khối điều khiển truyền thông SPI Master phải được xây dựng để điều khiển quá trình truyền thông này. Khối này có nhiệm vụ nhận các yêu cầu truyền thông và dữ liệu cần gửi đi ở dạng song song từ thiết bị Master. Khối SPIm sẽ xác định chế độ truyền và thực hiện việc bắt tay cũng như điều khiển quá trình truyền thông với thiết bị tớ. Dữ liệu từ thiết bị Slaver gửi về cũng sẽ được khối này chuyển từ dạng nối tiếp thành song song và chuyển lại cho thiết bị Master. Để có thể giao tiếp với thiết bị Master một cách thuận tiện nhất, trong mô hình của khối SPIm, chúng tôi đưa ra các tín hiệu bắt tay và tín hiệu dữ liệu được mô tả như ở trong 0.



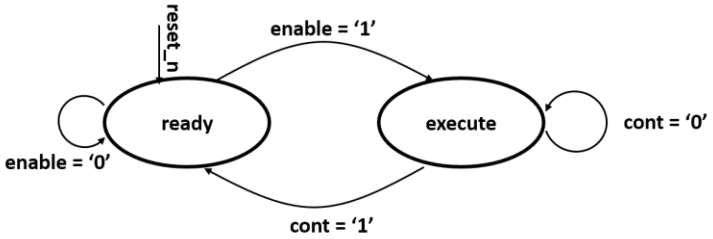
Hình 4. Tín hiệu vào ra trên khối SPI Master (SPIm).

Chức năng và mô tả cụ thể của từng tín hiệu trong khối này được chỉ ra như ở trong 0.

Bảng 1. Mô tả các tín hiệu trong khối SPIm

Tín hiệu	Độ rộng bit	Chế độ vào/ra	Mô tả chức năng
clock	1	in	Xung nhịp hệ thống
reset_n	1	in	Tín hiệu reset không đồng bộ, kích hoạt ở mức thấp
enable	1	in	Mức cao "H": chốt các thiết lập, địa chỉ, dữ liệu để sẵn sàng cho quá trình truyền. Mức thấp "L": không có quá trình truyền nào được yêu cầu.
cpol	1	in	Thiết lập thông số CPOL.
cpha	1	in	Thiết lập thông số CPHA.
cont	1	in	Cờ báo chế độ liên tục.
clk_div	32	in	Thiết lập tốc độ truyền. Giá trị đưa vào sẽ là số chu kỳ xung nhịp chia cho ½ chu kỳ của SCKL.
addr	32	in	Địa chỉ của thiết bị tớ. Các thiết bị Slaver được đánh địa chỉ bắt đầu từ giá trị 0.
tx_data	n	in	Dữ liệu cần truyền.
miso	1	in	Tín hiệu truyền dữ liệu nối tiếp MISO
sclk	1	buffer	Xung nhịp đồng bộ cho SPI.
ss_n	K	buffer	Tín hiệu lựa chọn thiết bị tớ.
mosi	1	out	Tín hiệu nhận dữ liệu nối tiếp MOSI.
busy	1	out	Tín hiệu báo dữ liệu đã sẵn sàng hay chưa.
rx_data	n	out	Dữ liệu nhận được.

Hoạt động của khối SPIm được mô tả bằng một máy trạng thái với hai trạng thái chính là: *ready* và *execute* (0). Khi có tín hiệu **reset_n** của hệ thống, khối SPIm chuyển vào trạng thái *ready* để chờ một quá trình truyền. Trong trạng thái này, các tín hiệu lựa chọn thiết bị Slaver SS\ sẽ được chuyển lên mức cao để không một thiết bị Slaver nào được chọn. Tín hiệu MOSI cũng được treo ở trạng thái có trở kháng cao (mức Z). Khi tín hiệu enable có giá trị "1", báo hiệu chuẩn bị một quá trình truyền. Khối SPIm sẽ thực hiện việc chốt địa chỉ, chuyển dữ liệu nhận được từ thiết bị Master vào trong bộ nhớ đệm, gửi các tín hiệu báo hiệu đến thiết bị tớ. Sau đó, trạng thái *ready* sẽ được chuyển thành trạng thái *execute* để thực hiện việc truyền dữ liệu với thiết bị tớ.



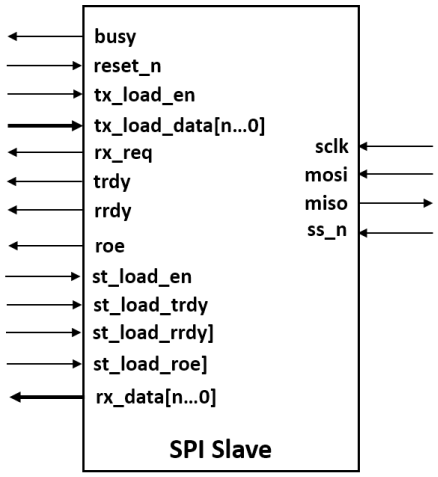
Hình 5. Mô hình máy trạng thái để thực hiện quá trình truyền thông trên SPIm.

Đoạn mã VHDL để mô tả thực thể của khối SPIm được trình bày như bên dưới.

```
ENTITY spi_master IS
  GENERIC(
    slaves  : INTEGER := 4;
    d_width : INTEGER := 2);
  PORT(
    clock   : IN     STD_LOGIC;
    reset_n : IN     STD_LOGIC;
    enable  : IN     STD_LOGIC;
    cpol    : IN     STD_LOGIC;
    cpha    : IN     STD_LOGIC;
    cont    : IN     STD_LOGIC;
    clk_div : IN     INTEGER;
    addr    : IN     INTEGER;
    tx_data : IN     STD_LOGIC_VECTOR(d_width-1 DOWNTO 0);
    miso    : IN     STD_LOGIC;
    sclk    : BUFFER STD_LOGIC;
    ss_n    : BUFFER STD_LOGIC_VECTOR(slaves-1 DOWNTO 0);
    mosi    : OUT    STD_LOGIC;
    busy    : OUT    STD_LOGIC;
    rx_data : OUT    STD_LOGIC_VECTOR(d_width-1 DOWNTO 0));
END spi_master;
```

2.3 Xây dựng mô hình khối giao tiếp SPI Slaver (SPI Slaver - SPIs)

Với thiết bị tớ, một khối điều khiển giao tiếp SPI Slaver - SPIs cũng cần phải được xây dựng để thực hiện quá trình bắt tay với thiết bị Master, cũng như đảm nhận việc giao tiếp dữ liệu với thiết bị tớ. Khối SPIs sẽ thực hiện việc bắt tay với thiết bị Master khi được lựa chọn giao tiếp. Khối này sẽ đảm nhận việc nhận dữ liệu nối tiếp từ thiết bị Master, chuyển dữ liệu này thành song song và gửi cho thiết bị tớ. Đồng thời, khối SPIs cũng sẽ đọc dữ liệu ở dạng song song từ thiết bị Slaver để chuyển thành dữ liệu nối tiếp và gửi cho thiết bị chủ khi có yêu cầu đọc dữ liệu. Để có thể bắt tay giao tiếp với các thiết bị tớ, các tín hiệu của khối SPIs được đề xuất theo như mô hình ở 0.



Hình 6. Tín hiệu vào ra trên khối SPI Slaver (SPIs).

Chức năng và mô tả cụ thể của từng tín hiệu trên khối SPIs được trình bày như ở trong 0.

Bảng 2. Mô tả các tín hiệu trong khối SPIs

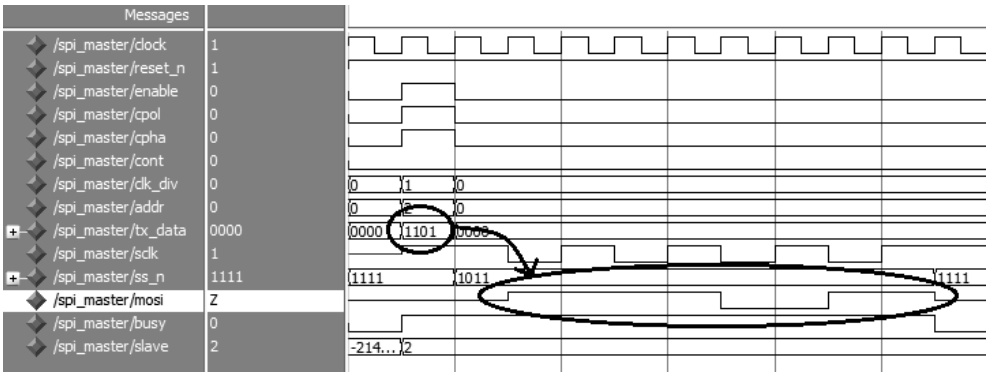
Tín hiệu	Độ rộng bit	Chế độ vào/ra	Mô tả
sclk	1	in	Xung nhịp cho SPI
mosi	1	in	Đường dữ liệu nối tiếp MOSI
miso	1	out	Đường dữ liệu nối tiếp MISO
ss_n	1	in	Tín hiệu lựa chọn thiết bị tớ, kích hoạt ở mức thấp.
busy	1	out	Tín hiệu báo bận: có giá trị '1' khi đang truyền dữ liệu với thiết bị chủ, có giá trị '0' khi sẵn sàng giao tiếp với thiết bị tớ.
reset_n	1	in	Tín hiệu khởi động lại hệ thống, kích hoạt ở mức thấp
tx_load_en	1	in	Tín hiệu chốt dữ liệu cần truyền vào thanh ghi đệm truyền
tx_load_data	n	in	Dữ liệu từ thiết bị tớ, gửi đến thanh ghi đệm truyền
rx_req	1	in	Yêu cầu gói dữ liệu nhận được cuối cùng từ thiết bị chủ.
rx_data	n	out	Dữ liệu cuối cùng nhận được từ thiết bị chủ.
trdy	1	buffer	Tín hiệu báo sẵn sàng truyền. Có giá trị '1' khi khối SPIs đã đưa dữ liệu vào thanh ghi đệm truyền nhưng vẫn chưa truyền đi.
rrdy	1	buffer	Tín hiệu báo sẵn sàng nhận. Có giá trị '1' khi SPIs đã nhận đủ dữ liệu từ thiết bị chủ nhưng thiết bị Slaver vẫn chưa yêu cầu nhận.
roe	1	buffer	Tín hiệu báo lỗi nhận tràn. Có giá trị '1' khi có thêm dữ liệu từ thiết bị chủ truyền đến, nhưng dữ liệu cũ vẫn chưa được nhận xong bởi thiết bị tớ.
st_load_en	1	in	Tín hiệu báo kích hoạt chốt trạng thái.
st_load_trdy	1	in	Tín hiệu báo giá trị của trdy đã được chốt
st_load_rrdy	1	in	Tín hiệu báo giá trị của rrdy đã được chốt
st_load_roe	1	in	Tín hiệu báo giá trị của roe đã được chốt

3. KẾT QUẢ MÔ PHỎNG VÀ NHẬN XÉT

Để đánh giá hoạt động của các khối SPIm và SPIs ở mức logic sau khi mô hình hoá bằng ngôn ngữ VHDL, chúng tôi sử dụng phần mềm mô phỏng ModelSim phiên bản 6.5 để mô phỏng và kiểm chứng các hoạt động của từng khối. Một chương trình đánh giá (testbench) sẽ được xây dựng để tạo ra các tín hiệu điều khiển và dữ liệu tương tự như các thiết bị chủ và thiết bị Slaver được gắn với các khối SPIm và SPIs. Chương trình này sẽ tạo ra các dữ liệu ngẫu nhiên để gửi và nhận giữa hai thiết bị này. Bằng cách quan sát giản đồ xung của quá trình mô phỏng, chúng ta sẽ đánh giá hoạt động của các khối được thiết kế có đúng với yêu cầu đặt ra hay không.

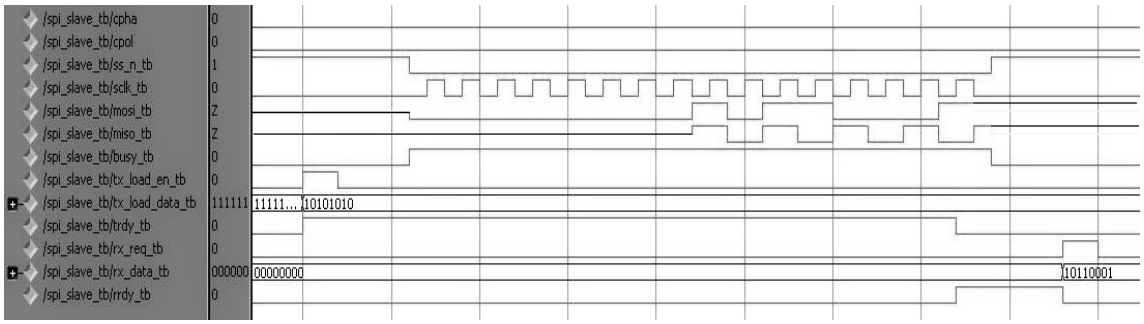
Đối với mô phỏng cho khối SPIm, chúng tôi thử mô phỏng truyền một khối dữ liệu 4 bit với giá trị "1101" (được đánh dấu như trên 0). Thiết lập khối SPIm hoạt động ở chế độ "00" và kiểu truyền là không liên tục (cont = '0'). Sau khi thiết lập các giá trị cho tín hiệu điều khiển và

tiến hành chạy mô phỏng, chúng tôi nhận định tín hiệu trên đường truyền MOSI có giá trị đúng như giá trị của dữ liệu cần truyền.



Hình 7. Mô phỏng truyền một khối dữ liệu ở chế độ không liên tục trên khối SPIm.

Đối với khối SPIs, chúng tôi tiến hành mô phỏng khối này hoạt động ở chế độ "00" như ở 0. Khi khối SPIs đang ở trạng thái rỗi, thiết bị Slaver sẽ đưa tín hiệu **tx_load_en** lên mức cao đồng thời gửi dữ liệu "10101010" lên bus **tx_load_data**. Tín hiệu **trdy** chuyển lên mức cao báo hiệu một dữ liệu đã gửi đến và đang đợi truyền đi. Thiết bị chủ sẽ khởi động một quá trình truyền với lệnh "00000000", để chỉ thị khối SPIs nhận dữ liệu trên đường MOSI và lưu vào thanh ghi đệm nhận, đồng thời gửi dữ liệu trong thanh ghi đệm truyền lên đường MISO. Khi bit dữ liệu cuối cùng được truyền xong, tín hiệu **tdry** chuyển về mức thấp để báo hiệu dữ liệu trong thanh ghi đệm truyền đã được gửi đến thiết bị chủ. Tương tự, khi khối SPIs nhận được bit dữ liệu cuối cùng và lưu vào thanh ghi đệm nhận, tín hiệu **rrdy** sẽ được đưa lên mức cao để báo hiệu một dữ liệu mới đã được nhận hoàn chỉnh. Lúc này, thiết bị Slaver sẽ phát tín hiệu **rx_req** và khối SPIs sẽ gửi dữ liệu nhận được từ thiết bị chủ "10110001" lên bus **rx_data**.



Hình 8. Mô phỏng truyền dữ liệu trên khối SPIs.

4. KẾT LUẬN

Trong bài báo này, chúng tôi đã tiến hành nghiên cứu, tìm hiểu về chuẩn SPI, một chuẩn truyền thông được sử dụng phổ biến cho các MCU để giúp những MCU này giao tiếp với các thiết bị ngoại vi một cách thuận tiện. Sau một thời gian nghiên cứu và tìm hiểu các đặc tả mô tả hoạt động của chuẩn giao tiếp này, chúng tôi đã đề xuất được một mô hình cho các khối

giao tiếp SPIm và SPIs. Đây là các khối chức năng sẽ đảm nhận việc bắt tay với các thiết bị Master và thiết bị Slaver, sau đó thực hiện quá trình truyền thông theo giao thức SPI để truyền nhận dữ liệu giữa hai thiết bị Master - Slaver này. Mô hình được đề xuất cho các khối SPIm và SPIs đã được mô hình hoá thành công bằng ngôn ngữ mô tả phần cứng VHDL.

Sau khi mô hình hoá, chúng tôi đã tiến hành mô phỏng hoạt động của hai khối này trên phần mềm mô phỏng ModelSim. Kết quả mô phỏng thu được đã chứng tỏ hoạt động ở mức logic của các khối được thiết kế hoàn toàn đúng với các yêu cầu đặt ra ban đầu. Các khối SPIm và SPIs hoàn toàn đáp ứng với các đặc tả hoạt động của giao thức truyền thông SPI và có thể ứng dụng để xây dựng thành các khối chức năng đảm nhận vai trò truyền thông cho các MCU.

LỜI CẢM ƠN

Bài báo này được thực hiện trong khuôn khổ đề tài nghiên cứu khoa học cấp Đại học Huế - mã số: DHH2014-01-54.

TÀI LIỆU THAM KHẢO

[1]. Microchip (2013). *Software & Hardware Solutions for the 32-bit Designer*.

[2]. Microchip (2013). *PIC24 Microcontroller Family*.

[3]. Cypress (2013). *PSoC - Programmable System-On-Chip*.

[4]. Khan. S.A, Hossain.M.I (2010). Design and implementation of microcontroller based fuzzy logic control for maximum power point tracking of a photovoltaic system, *Electrical and Computer Engineering (ICECE)*, 2010 *International Conference*, pp.322-325.

[5]. Vuza. D.T, Frosch. R, Koeberl. H (2007). A Long Range RFID Reader Based on the Atmel AT91SAM7S64 Micro-Controller", *Electronics Technology, 30th International Spring Seminar*, pp.445-450.

[6]. Guang-li Long (2011). Design of reader device for radio frequency Card based on microcontroller unit STC89C52, *Multimedia Technology (ICMT), 2011 International Conference*, pp.3493-3496.

[7]. Erdas. N, Gunduzalp. M (1999). Design of an 8 bit general purpose microcontroller with sea-of-gates design style, *Microelectronics, 1999. ICM '99. The Eleventh International Conference*, pp.177-180.

[8]. Krauss. R, Croxell. J (2012). A low-cost microcontroller-in-the-loop platform for controls education, *American Control Conference (ACC)*, pp.4478-4483.

[9]. Mabuchi. Y, Nakamura. A, Ohmae. A, Uno. T, Ichikawa. K, Mizuno. H (2009). Development of a Low EMI Micro-controller Package for Automobile Applications, *Electromagnetic Compatibility, 2009 20th International Symposium*, Zurich, pp.377-380.

[10]. Philips Semiconductors (2000). *The I2C-Bus Spectification*, Ver 2.1.

[11]. Motorola, Inc (2000). *SPI Block guide*, Ver 3.06.

[12]. STMicroelectronics (2013). *SPI Protocol*.

DESIGNING AND MODELING A SYNCHORONOUS SERIAL COMMUNICATION INTERFACE MODULE USING IN MICROCONTROLLER

Phan Hai Phong*, Hoang Le Ha, Nguyen Van An, Ho Duc Tam Linh

Department of Electronics - Telecommunications, Hue University College of Sciences

**Email:phongph@husc.edu.vn*

ABSTRACT

Communication protocol for new – generation microcontroller families is always required so that these microcontrollers are able to communicate with peripherals or other platforms. Serial Peripheral Interface (SPI) is a synchronous serial communication protocol allowing devices to connect easily with each other with low-cost system resources. This article focuses on the investigation, design, and modeling of a SPI controller to make up an interface module for a designed microcontroller. Results of modeling which refer to the operation of this module in logical level are shown in this article as well.

Keywords: *SPI, microcontroller, microprocessor.*