http://doi.org/10.37550/tdmu.VJS/2022.05.341

ĐÁNH GIÁ BỘ PHÂN XỬ MỨC ƯU TIÊN CỐ ĐỊNH VÀ ROUND ROBIN TRÊN PHẦN CỨNG FPGA SPARTAN 3E

Trương Thanh Sang[(1)], Phan Hữu Phúc[(1)], Nguyễn Ngô Lâm[(1)], Trương Quang Phúc[(1)], Trịnh Quốc Thanh[(2)]

(1) Trường Đại học Sư phạm kỹ thuật TPHCM; (2) Trường Đại học Thủ Dầu Một

Ngày nhận bài 26/07/2022; Ngày phản biện 30/072022; Chấp nhận đăng 30/8/2022

Liên hệ Email: 18161265@student.hcmute.edu.vn

https://doi.org/10.37550/tdmu.VJS/2022.05.341

Tóm tắt

Trong hệ thống trên chip (System on chip - SoC), việc có sự truy cập đồng thời từ nhiều nguồn (Source) hoặc nhiều Master đến cùng một Slave là việc thường xuyên xảy ra. Tuy nhiên, một Slave không thể đáp ứng đồng thời tất cả các truy cập cùng lúc mà chỉ có thể đáp ứng một cách tuần tự từng truy cập theo một thứ tự nhất định. Việc xác định truy cập nào thực hiện trước, truy cập nào thực hiện sau chính là "phân xử truy cập". Thành phần thực hiện chức năng "phân xử truy cập" thường được gọi là bộ phân xử (arbiter). Trong đề tài này, nhóm tác giả tiến hành thiết kế bộ phân xử mức ưu tiên cố định và bộ phân xử Round Robin thực hiện hoạt động phân xử cho bốn Master và một Slave. Hai bộ phân xử sẽ được tổng hợp thiết kế bằng ngôn ngữ mô tả phần cứng Verilog trên phần mềm Xilinx ISE Design Suite 14.7. Thiết kế của hai bộ phân xử sau khi tổng hợp sẽ được kiểm tra và đánh giá bằng các testcase để so sánh về thuật toán và tốc độ phân xử. Sau cùng, nhóm tác giả tiến hành thực hiện hai bộ phân xử lên phần cứng Xilinx Spartan 3E để kiểm tra kết quả mô phỏng.

Từ khóa: *bộ phân xử, round Robin, spartan 3E*

Abstract

EVALUATION OF FIXED PRIORITY ARBITER AND ROUND ROBIN ON FPGA SPARTAN 3E

In System on chip (SoC), having simultaneous access from multiple Sources or Masters to the same Slave is common. However, one Slave cannot respond to all accesses at the same time but can only respond sequentially to each access in a certain order. Determining which access to execute first, which access to execute after is called "access arbitration". The component that performs the "access arbitration" function is often called an Arbiter. In this paper, the authors design a fixed-priority arbiter and a Round Robin arbiter that performs arbitration for four Masters and one Slave. These two arbiters will be combined and designed using Verilog on Xilinx ISE Design Suite 14.7

software. The design of the two arbiters after synthesis will be checked and evaluated by testcases to compare the algorithm and arbitration speed. Finally, the authors will implement two arbiters on FPGA Xilinx Spartan 3E to check the simulation results.

1. Tổng quan

1.1. Giới thiệu

SoC là một bước tiến lớn đối với ngành vi mạch và điện tử. Tuy nhiên, việc thiết kế gặp rất nhiều vấn đề cần phải được quan tâm đặc biệt là độ phức tạp của hệ thống ngày càng tăng dẫn đến số lượng kết nối từ nhiều nguồn hoặc Master đến Slave tại cùng một thời điểm tăng khiến cho việc truyền nhận dữ liệu có thể bị gián đoạn. Trong trường hợp này, một Slave không thể đáp ứng tất cả các yêu cầu truy cập cùng một lúc mà chỉ có thể đáp ứng các yêu cầu truy cập một cách tuần tự theo một thứ tự trước sau nhất định. Để giải quyết vấn đề trên cần sử dụng một khối xử lý giúp phân phối thứ tự kết nối của nhiều nguồn hoặc Master và bộ phân xử (Arbiter) là một giải pháp hiệu quả giúp phân luồng truy cập một cách hợp lý.

Có nhiều bộ phân xử với các thuật toán phục vụ những mục đích khác nhau tùy vào hệ thống như: Bộ phân xử theo mức ưu tiên cố định (Fixed Priority Arbiter), Bộ phân xử theo hình thức xổ số (Lottery Arbiter), Bộ phân xử sử dụng thuật toán Round Robin,... Bộ phân xử theo mức ưu tiên cố định là ví dụ điển hình cho dạng phân xử theo mức độ ưu tiên. Với việc phân xử theo mức ưu tiên như vậy, nguồn yêu cầu truy cập (request) có mức ưu tiên cao luôn chiếm ưu thế và dễ dàng được cấp quyền (grant) hơn các nguồn request có mức ưu tiên thấp. Nguồn request có mức ưu tiên thấp nhất có thể bị "treo" (không được grant) nếu các nguồn request ưu tiên cao hơn liên tục tích cực. Trong bộ phân xử Round Robin thì các thứ tự các mức ưu tiên của các request được xoay vòng liên tục. Đối với các hệ thống cần sự cân bằng quyền truy cập giữa các nguồn request thì bộ phân xử Round Robin là một lựa chọn hiệu quả.

1.2. Tình hình nghiên cứu

Trong bài báo (Gupta & Goel, 2015), nhóm tác giả đã đề xuất và thiết kế một bộ phân xử có thể được cấu hình cho n người dùng. Thiết kế được thực hiện bằng thuật toán Round Robin, thông qua kết quả mô phỏng giúp so sánh và đánh giá để đưa ra được một thiết kế bộ phân xử phù hợp nhất cho bất kỳ thiết bị, lõi sở hữu trí tuệ (IP), bộ nhớ trên chip (On chip memory) và các nguồn tài nguyên. Đối với bài báo (Deb & Rajrajan, 2013), tác giả đã giới thiệu một số giải pháp chia sẻ tài nguyên giữa các nguồn yêu cầu truy cập bên trong một bộ phân xử. Trong bài báo (Shin, III, & Riley, 2002) và (Weber, 2001), nhóm tác giả đã liệt kê về những điều cần lưu ý trong việc thiết kế một bộ phân xử Round Robin và đề xuất một vài ý tưởng thiết kế giao diện giúp giao tiếp với bộ phân xử nhằm đạt được hiệu năng cao. Tuy nhiên, các bài báo trên chưa trình bày thiết kế chi tiết hoặc các thiết kế đơn giản chưa giải quyết được vấn đề ưu tiên quyền truy cập.

Bên cạnh đó, trong nước cũng có một số nghiên cứu về lĩnh vực vi mạch và SoC như trong bài viết (Ngọc & Bảo, 2020) nhóm tác giả tiến hành thiết kế một bộ giao tiếp đa kênh UART sử dụng FPGA. Đối với bài viết (Anh & Trí, 2021), nhóm tác giả đã đề xuất một thiết kế bộ truyền nhận theo giao thức I2C và tiến hành thực hiện lên phần cứng Spartan 6. Tuy nhiên, cả hai bài viết chỉ tập trung vào đặc điểm và nguyên lý hoạt động của các chuẩn truyền thông nhưng các vấn đề liên quan như việc phân xử truy cập giữa các kết nối lại chưa được đề cập đến.

2. Bộ phân xử

Trong một hệ thống SoC, việc có sự truy cập giữa nhiều nguồn hoặc nhiều Master đến cùng một Slave là việc thường xuyên xảy ra. Tuy nhiên, một Slave không thể đáp ứng đồng thời tất cả các truy cập cùng lúc mà chỉ có thể đáp ứng một cách tuần tự từng truy cập theo một thứ tự nhất định. Việc xác định truy cập nào thực hiện trước, truy cập nào thực hiện sau, chính là "phân xử truy cập". Thành phần thực hiện chức năng "phân xử truy cập" thường được gọi là bộ phân xử (arbiter). Khi số lượng truy cập tăng lên thì lúc này việc thêm vào một bộ phân xử là rất cần thiết. Một bộ phân xử thường có những chức năng chính như:

– Nhận các yêu cầu truy cập từ các nguồn khác nhau, mỗi một yêu cầu gọi là một request.

– Thực hiện thuật toán phân xử nếu có nhiều nguồn truy cập (request) cùng lúc.

– Cấp quyền (grant) cho nguồn (Source/Master) được chọn.

– Việc lựa chọn bộ phân xử sẽ phụ thuộc vào yêu cầu của hệ thống và thuật toán phân xử mà thiết kế hướng đến. Thông thường, thiết kế bộ phân xử sẽ được chia theo hai hướng:

– Phân xử theo mức độ ưu tiên.

– Phân xử cân bằng (Quân, 2019).

Đối với mỗi hướng sẽ có một bộ phân xử tiêu biểu. Mỗi hướng thiết kế có thế mạnh riêng tùy vào mục đích sử dụng của hệ thống. Đối với hệ thống thiết kế theo hướng phân xử theo mức độ ưu tiên thì bộ phân xử theo mức ưu tiên cố định thường được sử dụng do có thiết kế đơn giản, ít tốn tài nguyên. Ngược lại, đối với hệ thống thiết kế theo hướng phân xử cân bằng thì bộ phân xử Round Robin với ưu điểm về sự cân bằng truy cập thường được ưu tiên sử dụng. Qua đó, có một cái nhìn tổng quát về sự khác biệt giữa hai hướng thiết kế.

2.1. Bộ phân xử mức ưu tiên cố định

Bộ phân xử theo mức ưu tiên cố định là bộ phân xử đơn giản và ít tốn tài nguyên nhất. Các yêu cầu truy cập (requesters) sẽ được thiết đặt trước mức ưu tiên từ cao đến thấp. Các yêu cầu này sẽ được phân xử theo thứ tự trước sau theo mức ưu tiên được thiết đặt. Đối với bộ phân xử theo mức ưu tiên cố định, các mức ưu tiên của từng request sẽ là cố định, khác với sự xoay vòng mức ưu tiên ở bộ phân xử Round Robin.

Mỗi nguồn request sẽ được gán một mức ưu tiên khác nhau và các request sẽ được kiểm tra trước. Request có mức ưu tiên cao nhất sẽ được cấp quyền trước và sau đó bộ phân xử sẽ tiếp tục cấp quyền cho request có mức ưu tiên cao tiếp theo. Đối với bộ phân xử mức ưu tiên cố định, mức ưu tiên cao nhất là cố định và không đổi. Chính vì vậy, nếu request có mức ưu tiên cao hơn lại tích cực một lần nữa thì request đó sẽ được cấp quyền trước các request có mức ưu tiên thấp hơn. Quá trình phân xử sẽ được thực hiện cho đến khi không còn request nào có yêu cầu truy cập.

Ưu điểm chính của bộ phân xử mức ưu tiên cố định là chúng ta có thể thiết đặt sẵn các giá trị mức ưu tiên cho bất kỳ nguồn request cụ thể nào. Tuy nhiên, trong trường hợp các nguồn request với mức ưu tiên cao hơn liên tục tích cực thì các nguồn request còn lại với mức ưu tiên thấp hơn sẽ không được cấp quyền dẫn đến tình trạng bị treo (Hu, Chen, Li, & Liu, 2004).

2.2. Bộ phân xử Round Robin

Bộ phân xử Round Robin là một bộ phân xử được thiết kế theo hướng cân bằng mức ưu tiên của các truy cập. Ban đầu, các truy cập này được thiết lập mức ưu tiên từ cao đến thấp như ở bộ phân xử mức ưu tiên cố định. Nhưng sự khác biệt của bộ phân xử Round Robin là các mức ưu tiên này sẽ được xoay vòng liên tục sau mỗi chu kỳ phân xử. Với cách xoay vòng mức ưu tiên này thì khả năng được cấp quyền của từng truy cập sẽ được cân bằng.

Bộ phân xử Round Robin sử dụng một bộ đếm tăng đều hoặc giảm đều để quét qua các nguồn request, mỗi request sẽ ứng với giá trị cố định của bộ đếm. Khi bộ đếm chứa giá trị trùng với một nguồn requewst đang tích cực mức cao thì nguồn request này sẽ được cấp quyền và bộ đếm tăng/giảm để chọn nguồn request tiếp theo. Lúc này, nguồn request đang được cấp quyền sẽ có mức ưu tiên thấp nhất trong chu kỳ phân xử tiếp theo. Round Robin đảm bảo không có tiến trình nào độc quyền, các tiến trình được phục vụ đều đặn mà không phụ thuộc vào thời gian xử lý của tiến trình trước.

Ưu điểm của bộ phân xử Round Robin so với bộ phân xử mức ưu tiên cố định là các request sẽ được phân xử truy cập công bằng, thời gian mỗi request nhận được mức ưu tiên cao nhất là như nhau. Tuy nhiên, việc phân bổ quyền quan trọng hoặc chỉ định quyền truy cập cho một request cụ thể nếu tần số yêu cầu truy cập của request đó thường xuyên là không thể (Quân, [Arbiter] Bài 3 - Bộ phân xử Round Robin đơn giản, 2019).

3. Phần cứng Xilinx Spartan 3E

Báo cáo này được thực hiện trên phần cứng Xilinx Spartan 3E. Đây là phần cứng thí nghiệm sử dụng chip FPGA Spartan 3E XC3S500E của hãng Xilinx. Ngoài ra, phần cứng này còn hỗ trợ nhiều cổng giao tiếp ngoại vi, cổng kết nối mở rộng và nhiều chân I/O khác nhau. Trong bảng 1 dưới đây là một số các tính năng được cung cấp sẵn trên phần cứng Xilinx Spartan 3E:

http://doi.org/10.37550/tdmu.VJS/2022.05.341

Bảng 1. *Một số tính năng của Xilinx Spartan 3E (Xilinx, 2011)*

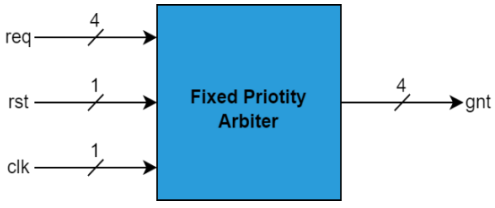
Chip FPGA chính	Xilinx XC3S500E Spartan-3E
Điện áp hoạt động	5 V
Điện áp ngõ ra	3.3 V
Xung nhịp chính cho hệ thống	50 MHz
Số chân I/O	232
Kiểu chân FBGA	320 chân
Số lượng phần tử logic	Hơn 10000
Cấu hình PROM	Flash Platform Xilinx 4 Mbit
DDR SDRAM	64 Mbyte (512 Mbit), giao diện dữ liệu x16
Probe Landing Pads NOR Flash ROM	Intel StrataFlash 16 Mbyte (128 Mbit)
Flash	16 Mbit loại nối tiếp SPI
Cổng giao tiếp thiết bị ngoại vi	VGA
	Cổng kết nối cho bàn phím hoặc chuột PS/2
	Hai cổng RS-232 9 chân (kiểu DTE- và DCE)
Cổng kết nối mở rộng	Đầu nối 100 chân Hirose FX2, 43 kết nối I/O
	3 kết nối ngoại vi 6 chân

Vì là FPGA của Xilinx nên Spartan 3E được hỗ trợ phần mềm lập trình Xilinx ISE Design Suite để thiết kế và cấu hình cho FPGA. Việc giao tiếp vật lý để nạp chương trình cho FPGA dựa trên chuẩn JTAG.

4. Thiết kế hệ thống

4.1. Thiết kế bộ phân xử mức ưu tiên cố định

Bộ phân xử mức tiên cố định được thiết kế để thực hiện hoạt động quét lần lượt qua các nguồn request với mức ưu tiên phân xử giảm dần từ request 0 đến request 3 để cấp quyền cho Master tương ứng. Quá trình phân xử sẽ dựa vào các tín hiệu rst, clk và thứ tự mức ưu tiên được thiết lập trước để phân luồng truy cập. Bộ phân xử mức ưu tiên cố định sẽ ưu tiên cấp quyền cho request có mức ưu tiên cao hơn cho đến khi request đó không còn có nhu cầu truy cập nữa thì các request có mức ưu tiên thấp hơn mới được tiến hành phân xử. Hoạt động phân xử sẽ lặp lại cho đến khi nào không còn request nào yêu cầu truy cập. Hình 1 dưới đây mô tả sơ đồ khối tổng quát và bảng 2 mô tả các chân tín hiệu ngõ vào và ngõ ra của bộ phân xử mức ưu tiên cố định.

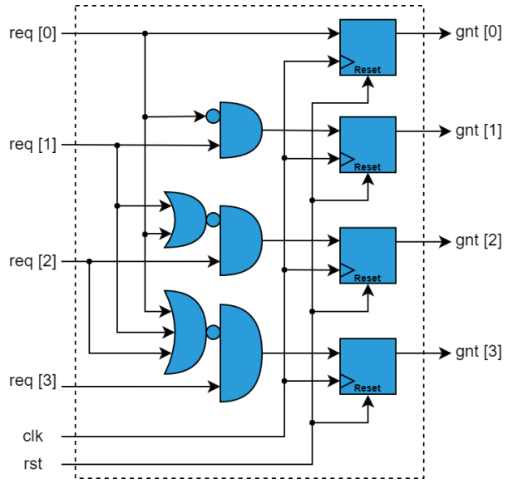


Hình 1. *Sơ đồ khối tổng quát bộ phân xử mức ưu tiên cố định*

Bảng 2. *Mô tả chân tín hiệu bộ phân xử mức ưu tiên cố định*

Tên chân tín hiệu	Loại	Độ rộng (bit)	Chức năng
req	Input	4	Tín hiệu yêu cầu quyền truy cập
clk	Input	1	Tín hiệu xung clock
rst	Input	1	Tín hiệu reset bộ phân xử
gnt	Output	4	Tín hiệu cấp quyền từ bộ phân xử

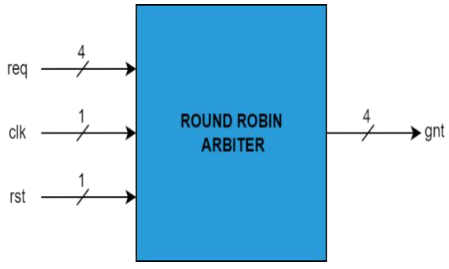
Bộ phân xử mức ưu tiên cố định được tạo thành từ bốn Flip Flop D kết hợp với các cổng logic như AND, NOT, OR để thiết lập thứ tự các mức ưu tiên cho các nguồn request. Các Flip Flop D này sẽ sử dụng cùng một nguồn xung clock và kết nối cùng một chân tín hiệu reset tích cực mức cao. Mỗi ngõ ra Q của từng Flip Flop D sẽ tương ứng với một ngõ ra của bộ phân xử. Hình 2 mô tả sơ đồ kiến trúc của bộ phân xử ưu tiên cố định.



Hình 2. *Sơ đồ kiến trúc bộ phân xử mức ưu tiên cố định*

4.2. Thiết kế bộ phân xử Round Robin

Bộ phân xử Round Robin có các đường tín hiệu ngõ vào và ngõ ra tương tự như ở bộ phân xử mức ưu tiên cố định. Điểm khác biệt giữa hai bộ phân xử nằm ở cách phân phát quyền ưu tiên. Thay vì được giữ cố định thì thứ tự các mức ưu tiên ở bộ phân xử Round Robin sẽ được xoay vòng giữa các request để tạo nên sự cân bằng về quyền truy cập. Quá trình phân xử sẽ dựa vào các tín hiệu rst, clk và thứ tự mức ưu tiên hiện tại để cấp quyền cho request đủ điều kiện. Hình 3 dưới đây mô tả sơ đồ khối tổng quát và bảng 3 mô tả các chân tín hiệu ngõ vào và ngõ ra của bộ phân xử Round Robin.



Hình 3. *Sơ đồ khối tổng quát bộ phân xử Round Robin*

Bảng 3. *Mô tả chân tín hiệu bộ phân xử Round Robin*

Tên chân tín hiệu	Loại	Độ rộng (bit)	Chức năng
req	Input	4	Tín hiệu yêu cầu quyền truy cập
clk	Input	1	Tín hiệu xung clock
rst	Input	1	Tín hiệu reset bộ phân xử
grant	Output	4	Tín hiệu cấp quyền từ bộ phân xử

Hình 4 bên dưới mô tả một kiến trúc của bộ phân xử Round Robin. Sự phối hợp hoạt động của bộ chia xung, Flip Flop D và bộ đếm vòng giúp bộ phân xử Round Robin xoay vòng và thiết lập chu kỳ xoay quyền ưu tiên cho các request ngõ vào. Các khối logic

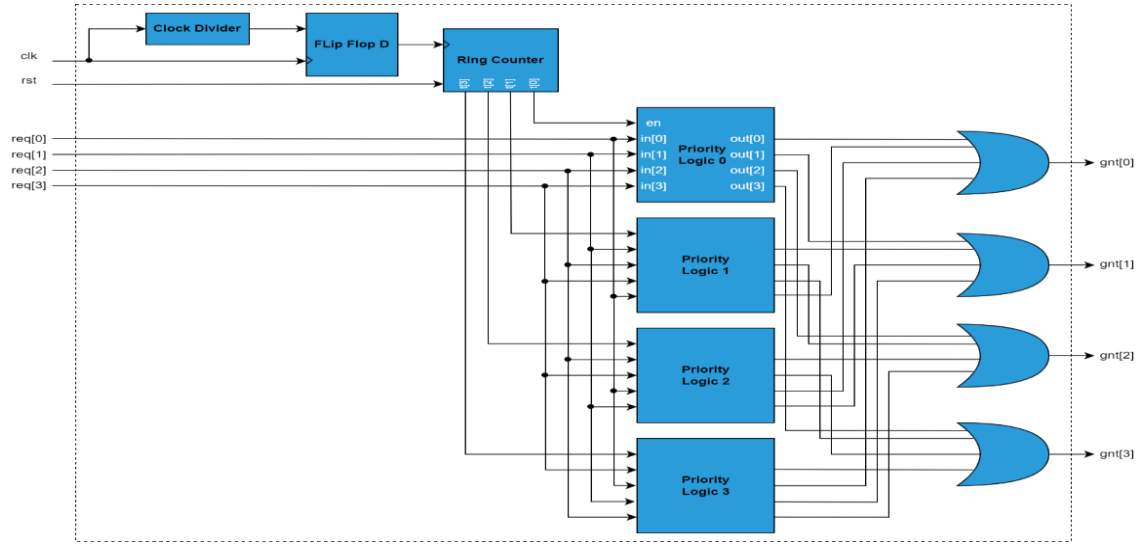
ưu tiên kết hợp với các cổng logic OR giúp bộ phân xử thiết lập các mức ưu tiên ở mỗi chu kỳ xoay và cho kết quả phân xử ở ngõ ra grant. Các chức năng cụ thể của bốn khối chính trong bộ phân xử Round Robin:

– Bộ chia xung (Clock Divider) có chức năng chia nhỏ tần số xung clock để thiết đặt chu kỳ xoay vòng các mức ưu tiên.

– Flip Flop D có chức năng làm một mạch tạo xung kích và giữ tín hiệu cho bộ đếm vòng.

– Bộ đếm vòng (Ring Counter) có chức năng tương tự như một thanh ghi dịch thực hiện hoạt động dịch từng bit để cấp tín hiệu enable cho các khối logic ưu tiên.

– Khối logic ưu tiên (Priority Logic) có chức năng nhận tín hiệu truy cập từ các nguồn request kết hợp với tín hiệu enable nhận từ ngõ ra bộ đếm vòng để cấp quyền theo thứ tự được thiết lập sẵn ở từng khối. Chức năng này gần giống chức năng của một bộ mã hóa ưu tiên (Priority Encoder) tuy nhiên dữ liệu ngõ ra sẽ không bị mã hoá.



Hình 4. *Sơ đồ kiến trúc bộ phân xử Round Robin*

5. Kết quả

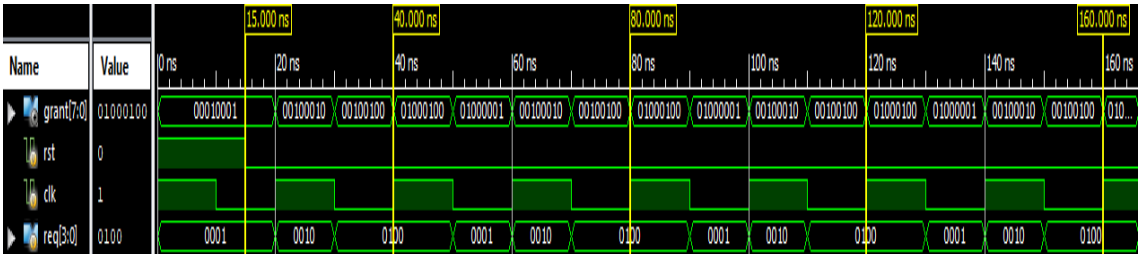
Các kết quả mô phỏng và thực hiện trên phần cứng đều được sử dụng với nguồn xung clock là 50 MHz. Để có một cái nhìn tổng quát về hoạt động của hai bộ phân xử, chúng tôi tiến hành xây dựng bốn testcase với số lượng truy cập đồng thời khác nhau của bốn ngõ vào request. Các testcase được trình bày cụ thể trong bảng 4 dưới đây.

Bảng 4. *Bảng testcase*

Testcase	Nội dung thực hiện
Testcase 1	Kiểm tra hoạt động hệ thống khi có 1 request yêu cầu
Testcase 2	Kiểm tra hoạt động hệ thống khi có 2 request đồng thời yêu cầu
Testcase 3	Kiểm tra hoạt động hệ thống khi có 3 request đồng thời yêu cầu
Testcase 4	Kiểm tra hoạt động hệ thống khi có 4 request đồng thời yêu cầu

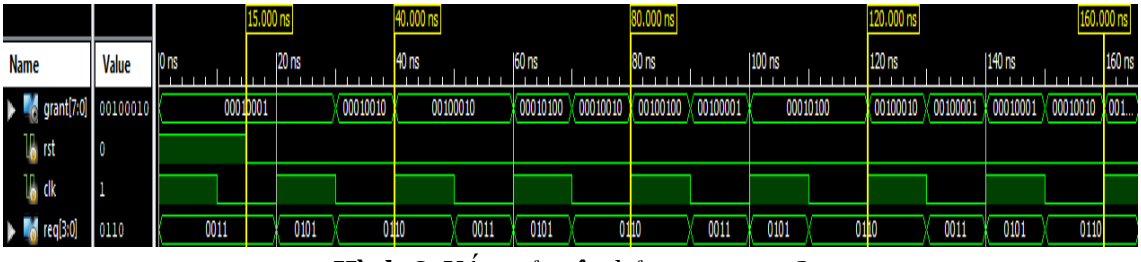
5.1. Kết quả mô phỏng

Chúng tôi tiến hành mô phỏng hai bộ phân xử trong cùng một sơ đồ dạng sóng để so sánh thuật toán và tốc độ phân xử. Ngõ ra grant lúc này sẽ có độ rộng là 8 bit với 4 bit grant[7:4] ứng với ngõ ra grant của bộ phân xử mức ưu tiên cố định, 4 bit grant[3:0] ứng với ngõ ra grant của bộ phân xử Round Robin.



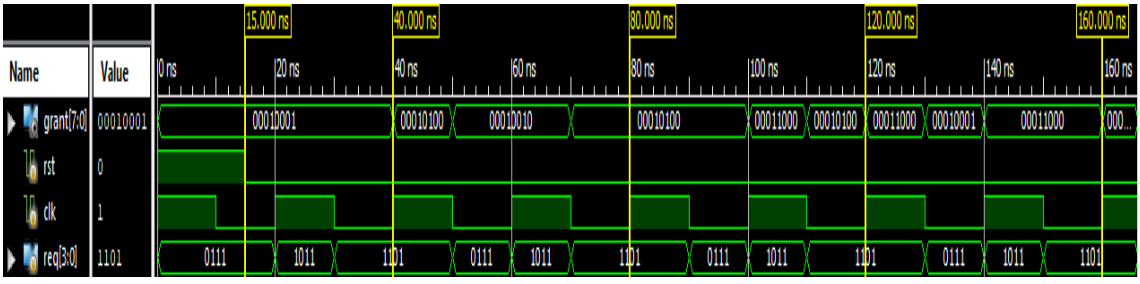
Hình 5. *Kết quả mô phỏng testcase 1*

Hình 5 phía trên mô tả kết quả mô phỏng testcase 1 của hệ thống. Trong khoảng thời gian 0 ns < t < 15 ns, giá trị chân tín hiệu rst = 1 nên thứ tự mức ưu tiên ở bộ phân xử Round Robin được đưa về mặc định là req[0] → req[1] → req[2] → req[3]. Nghĩa là req[0] sẽ có mức ưu tiên cao nhất tiếp đến là req[1], req[2] và cuối cùng là req[3]. Đến thời điểm t > 15 ns giá trị chân tín hiệu rst = 0, hệ thống tiến hành phân xử theo mức ưu tiên cố định cho ngõ ra grant[7:4] và phân xử theo thuật toán Round Robin cho ngõ ra grant[3:0]. Trong khoảng thời gian 30 ns < t < 40 ns, ngõ vào req = 0100 ứng với req[2] tích cực, giá trị ngõ ra grant là 00100100 do vẫn chưa đến thời điểm xung clock cạnh lên nên bộ phân xử ưu tiên cố định vẫn chưa thực hiện hoạt động phân xử. Ở thời điểm t = 40 ns, có xung clock cạnh lên thì giá trị ngõ ra grant lúc này mới chuyển thành 01000100 ứng với req[2] đã được cấp quyền ở cả hai bộ phân xử. Trong khoảng thời gian 40 ns < t < 80 ns, thứ tự mức ưu tiên ở bộ phân xử Round Robin đã được chuyển thành req[1] → req[2] → req[3] → req[0]. Tại t = 50 ns, ngõ vào req có giá trị 0001 ứng với req[0] có yêu cầu truy cập nhưng giá trị ngõ ra (grant [7:4]) của bộ phân xử mức ưu tiên cố định vẫn giữ nguyên là 0100 do vẫn chưa đến thời điểm xung clock tích cực cạnh lên. Sau mỗi 40 ns tương đương hai chu kỳ xung clock thì thứ tự mức ưu tiên ở bộ phân xử Round Robin sẽ xoay vòng một mức tạo nên sự cân bằng về quyền truy cập giữa các request. Bộ phân xử mức ưu tiên cố định có thứ tự các mức ưu tiên được giữ cố định là req[0] → req[1] → req[2] → req[3] và chỉ tiến hành phân xử khi đến thời điểm cạnh lên xung clock.



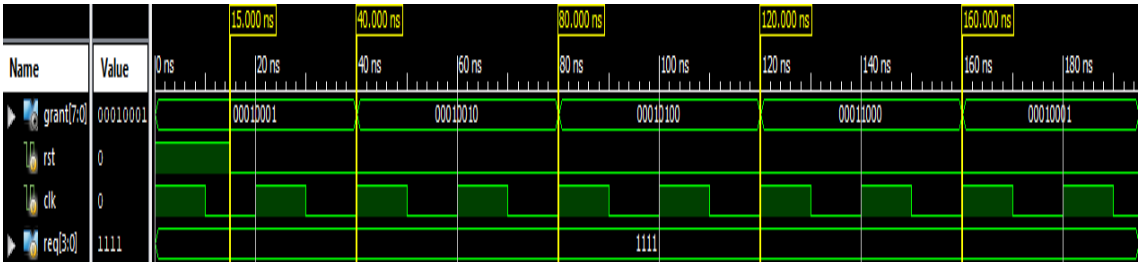
Hình 6. *Kết quả mô phỏng testcase 2*

Hình 6 phía trên mô tả kết quả mô phỏng testcase 2 của hệ thống. Trong khoảng thời gian 0 ns < t < 15 ns, giá trị chân tín hiệu rst = 1 nên thứ tự mức ưu tiên ở bộ phân xử Round Robin được đưa về mặc định là req[0] → req[1] → req[2] → req[3]. Khi t > 15 ns chân tín hiệu rst = 0 nên hệ thống tiến hành phân xử theo mức ưu tiên cố định cho ngõ ra grant[7:4] và phân xử theo thuật toán Round Robin cho ngõ ra grant[3:0]. Tại thời điểm t = 20 ns, giá trị ngõ vào req là 0101 ứng với req[2] và req[0] đang cùng có yêu cầu truy cập và đồng thời có xung clock tích cực cạnh lên nên ngõ ra grant lúc này có giá trị là 00010001 ứng với req[0] đang được cấp quyền ở cả hai bộ phân xử. Trong khoảng thời gian 40 ns < t < 80 ns, mức ưu tiên cao nhất ở bộ phân xử Round Robin đã được chuyển cho req[1], lúc này req[0] sẽ có mức ưu tiên thấp nhất. Tại t = 50 ns, ngõ vào req có giá trị là 0011 ứng với req[1] và req[0] cùng có yêu cầu truy cập nhưng chỉ có ngõ ra của bộ phân xử Round Robin (grant [3:0]) thay đổi giá trị thành 0010 do vẫn chưa đến thời điểm xung clock cạnh lên. Tại thời điểm t = 60 ns, có xung clock tích cực cạnh lên và ngõ vào req thay đổi giá trị thành 0101. Ngõ ra grant lúc này có giá trị là 00010100 ứng với req[0] được cấp quyền ở của bộ phân xử mức ưu tiên cố định và req[2] được cấp quyền ở bộ phân xử Round Robin do req[2] có mức ưu tiên cao hơn req[0]. Kết quả ngõ ra grant ở mỗi bộ phân xử sẽ được phân xử theo thuật toán tương ứng ở những chu kỳ sau tương tự như ở testcase 1.



Hình 7. *Kết quả mô phỏng testcase 3*

Hình 7 mô tả kết quả mô phỏng testcase 3 của hệ thống. Trong khoảng thời gian từ 0 ns đến 15 ns, giá trị chân rst = 1 nên thứ tự mức ưu tiên ở hai bộ phân xử được đưa về mặc định là req[0] → req[1] → req[2] → req[3]. Khi t > 15 ns, chân tín hiệu rst = 0 nên hệ thống tiến hành phân xử theo mức ưu tiên cố định cho ngõ ra grant[7:4] và phân xử theo thuật toán Round Robin cho ngõ ra grant[3:0]. Trong khoảng thời gian 15 ns < t < 40 ns, req[0] vẫn đang giữ mức ưu tiên cao nhất ở bộ phân xử Round Robin. Đồng thời, các ngõ vào lúc này đều có req[0] tích cực nên ngõ ra grant luôn có giá trị 00010001 ứng với req[0] được cấp quyền ở cả hai bộ phân xử. Trong khoảng thời gian 40 ns < t < 80 ns, thứ tự mức ưu tiên ở bộ phân xử Round Robin đã được xoay vòng thành req[1] → req[2] → req[3] → req[0]. Tại thời điểm t = 40 ns có cạnh lên xung clock, ngõ vào req có giá trị là 1101 ứng với req[1] không tích cực nên req[2] sẽ được ưu tiên cấp quyền trước ở bộ phân xử Round Robin. Ngõ ra grant lúc này sẽ có giá trị là 00010100 ứng với req[0] được cấp quyền ở bộ phân xử mức ưu tiên cố định và req[2] được cấp quyền ở bộ phân xử Round Robin. Kết quả ngõ ra grant ở mỗi bộ phân xử sẽ được phân xử theo thuật toán tương ứng ở những chu kỳ sau tương tự như ở testcase 1 và 2.

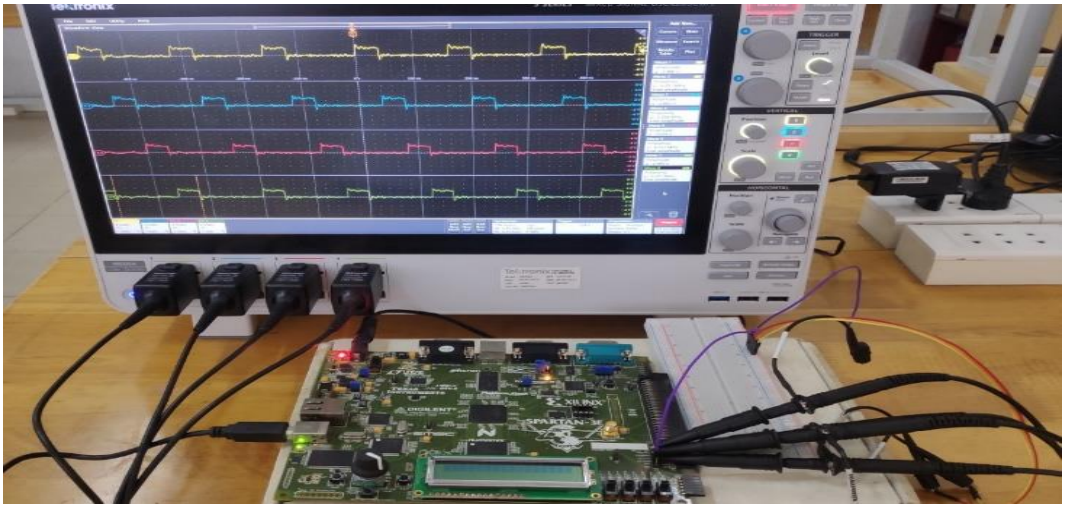


Hình 8. *Kết quả mô phỏng testcase 4*

Hình 8 mô tả kết quả mô phỏng testcase 4 của hệ thống. Trong khoảng thời gian từ 0 ns đến 15 ns thì giá trị chân tín hiệu rst = 1, thứ tự mức ưu tiên ở bộ phân xử Round Robin được đưa về mặc định là req[0] → req[1] → req[2] → req[3], lúc này req[0] sẽ có mức ưu tiên cao nhất nên req[0] sẽ được grant ở cả hai bộ phân xử. Khi t > 15 ns thì chân tín hiệu rst = 0, hệ thống tiến hành phân xử theo mức ưu tiên cố định cho ngõ ra grant[7:4] và phân xử theo thuật toán Round Robin cho ngõ ra grant[3:0]. Trong testcase 4, cả bốn request cùng tích cực tại mọi thời điểm nhưng tại mỗi thời điểm chỉ có một request được grant, các request khác bắt buộc phải chờ đến thời điểm mức ưu tiên của nó là cao nhất thì mới được grant. Do hoạt động phân xử khác nhau ở hai bộ phân xử nên việc cấp quyền cũng khác nhau. Ở bộ phân xử mức ưu tiên cố định thì thứ tự mức ưu tiên luôn là req[0] → req[1] → req[2] → req[3] nên req[0] sẽ luôn được cấp quyền tại mọi thời điểm. Ngược lại, thứ tự mức ưu tiên ở bộ phân xử Round Robin luôn được xoay vòng sau mỗi hai chu kỳ xung clock nên các request sẽ được grant lần lượt tại thời điểm mức ưu tiên của nó là cao nhất.

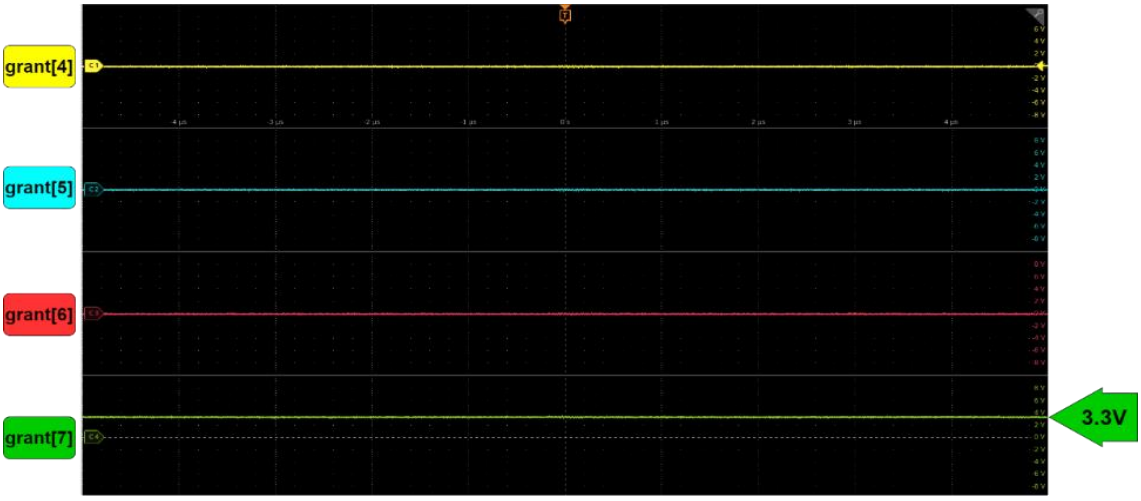
5.2. Kết quả thực hiện trên phần cứng

Trong phần này, chúng tôi thực hiện hệ thống trên phần cứng Xilinx Spartan 3E để kiểm tra kết quả mô phỏng. Các testcase sẽ lần lượt được kiểm tra ở cả hai bộ phân xử mức ưu tiên cố định và bộ phân xử Round Robin. Để quan sát kết quả dạng sóng thực tế của hai bộ phân xử, nhóm tác giả tiến hành kết nối các ngõ ra của phần cứng với máy hiển thị sóng hỗn hợp Oscilloscope 5 Series MS054 như hình 9.

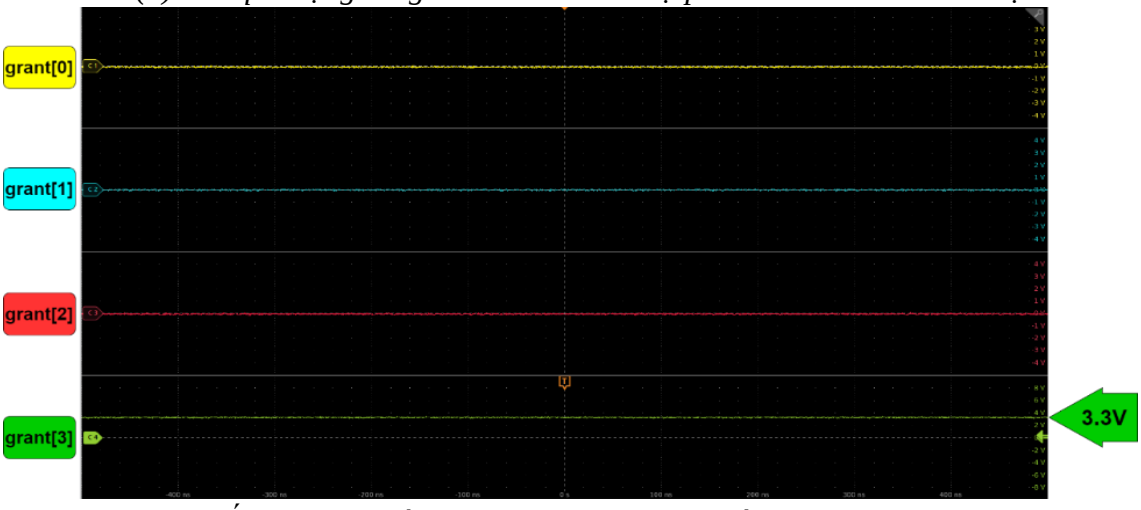


Hình 9. *Hình thực tế phần cứng FPGA Xilinx Spartan 3E và Oscilloscope 5 Series MS054*

Hình 10 mô tả dạng sóng của testcase 1 của hai bộ phân xử khi trạng thái ngõ vào là req = 1000 ứng với req[3] đang có yêu cầu truy cập. Hình 10 (a) là kết quả dạng sóng ngõ ra grant[7:4] tương ứng với ngõ ra của bộ phân xử mức ưu tiên cố định. Hình 10 (b) là kết quả dạng sóng ngõ ra grant[3:0] ứng với ngõ ra của bộ phân xử Round Robin. Kết quả ngõ ra grant ở ngõ ra ở cả hai bộ phân xử đều là grant = 1000 ứng với req[3] được cấp quyền. Tại mỗi thời điểm chỉ có một trong bốn request yêu cầu truy cập nên request đó sẽ được cấp quyền. Kết quả dạng sóng thực tế của testcase 1 đúng với kết quả dạng sóng đã mô phỏng.



(a) Kết quả dạng sóng của testcase 1 ở bộ phân xử mức ưu tiên cố định

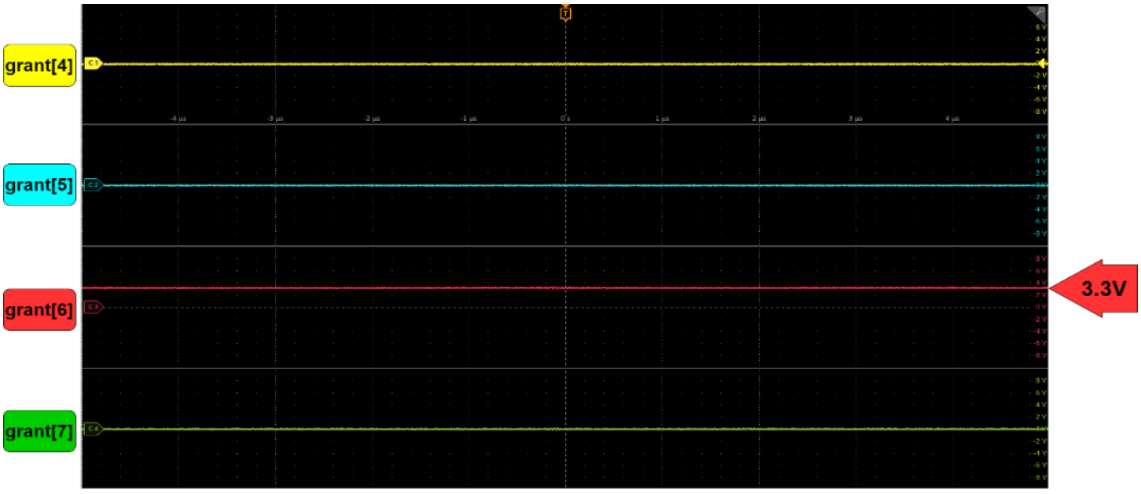


(b) Kết quả dạng sóng của testcase 1 ở bộ phân xử Round Robin

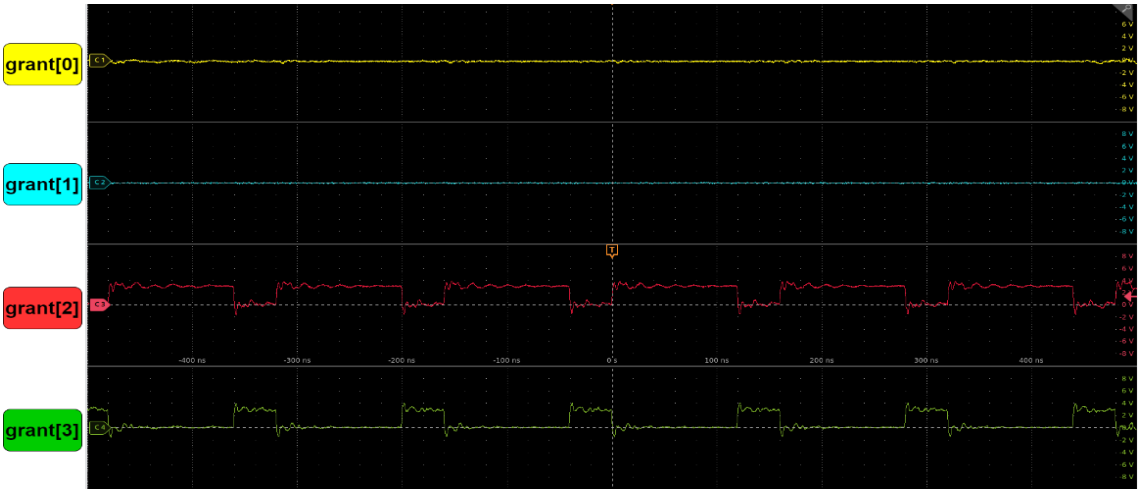
Hình 10. *Kết quả dạng sóng thực tế của testcase 1*

Hình 11 mô tả dạng sóng của testcase 2 của hai bộ phân xử khi trạng thái ngõ vào là req = 1100 ứng với req[3] và req[2] đang có yêu cầu truy cập. Dạng sóng ngõ ra ở hình 11 (a) là grant[7:4] = 0100 ứng với req[2] được cấp quyền ở bộ phân xử mức ưu tiên cố định. Đối với dạng sóng ngõ ra ở bộ phân xử Round Robin (grant[3:0]), bởi vì req[2] có mức ưu tiên cao hơn req[3] trong hai chu kỳ mà req[0] và req[1] giữ mức ưu tiên cao

nhất. Nên khi req[0] và req[1] không có yêu cầu truy cập thì req[2] sẽ được cấp quyền. Điều này giải thích cho dạng sóng tích cực mức cao của tín hiệu grant[2] kéo dài ba chu kỳ được mô tả trong hình 11 (b). Trong khi đó, dạng sóng tích cực mức cao của tín hiệu grant[3] chỉ kéo dài trong một chu kỳ tại thời điểm req[3] mang mức ưu tiên cao nhất. Kết quả dạng sóng thực tế của testcase 2 ở cả hai bộ phân xử đều đúng với mức ưu tiên thiết đặt trước và đúng với kết quả đã mô phỏng.



(a) Kết quả dạng sóng của testcase 2 ở bộ phân xử mức ưu tiên cố định

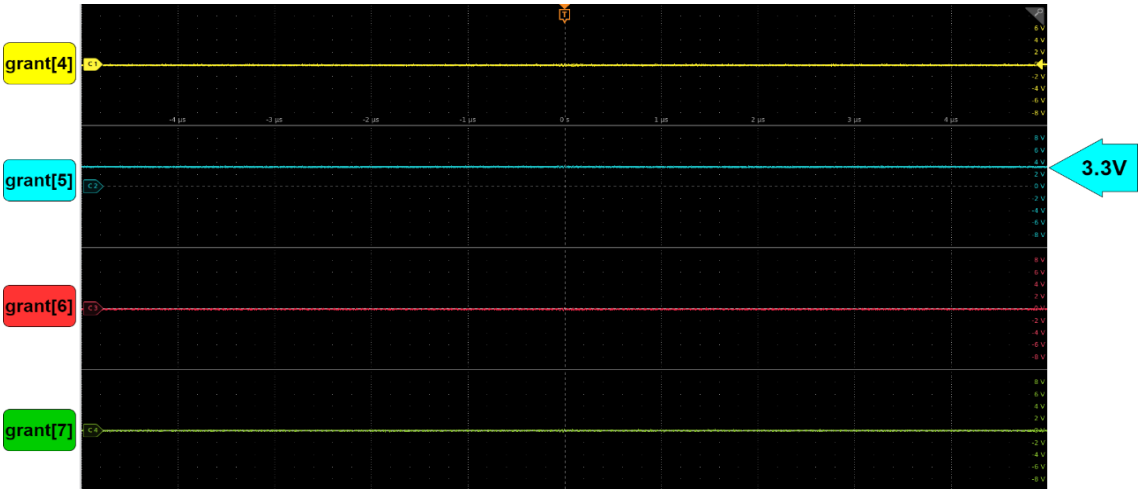


(b) Kết quả dạng sóng của testcase 2 ở bộ phân xử Round Robin

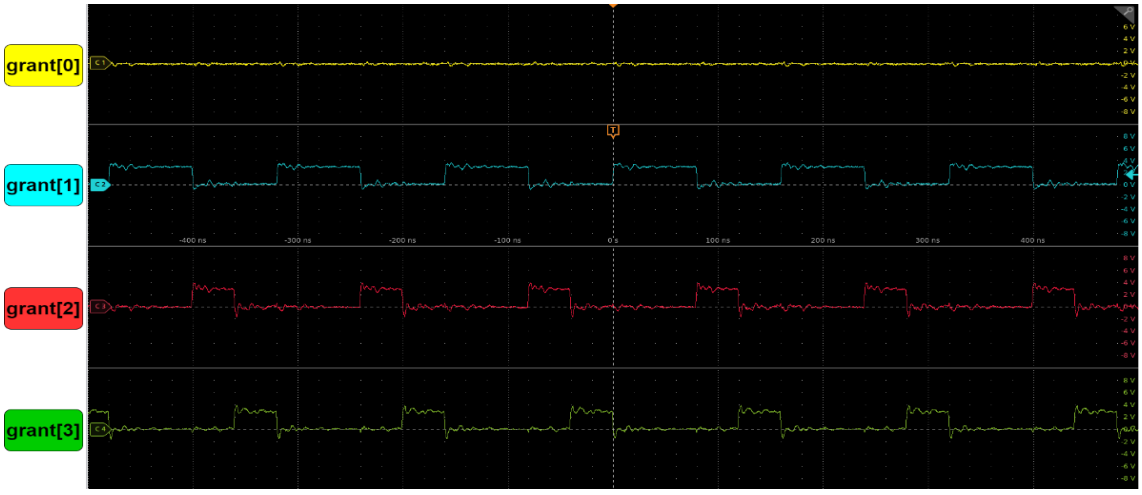
Hình 11. *Kết quả dạng sóng thực tế của testcase 2*

Hình 11 mô tả dạng sóng của testcase 2 của hai bộ phân xử khi trạng thái ngõ vào là req = 1100 ứng với req[3] và req[2] đang có yêu cầu truy cập. Dạng sóng ngõ ra ở hình 11 (a) là grant[7:4] = 0100 ứng với req[2] được cấp quyền ở bộ phân xử mức ưu tiên cố định. Đối với dạng sóng ngõ ra ở bộ phân xử Round Robin (grant[3:0]), bởi vì req[2] có mức ưu tiên cao hơn req[3] trong hai chu kỳ mà req[0] và req[1] giữ mức ưu tiên cao nhất. Nên khi req[0] và req[1] không có yêu cầu truy cập thì req[2] sẽ được cấp quyền. Điều này giải thích cho dạng sóng tích cực mức cao của tín hiệu grant[2] kéo dài ba chu

kỳ được mô tả trong hình 11 (b). Trong khi đó, dạng sóng tích cực mức cao của tín hiệu grant[3] chỉ kéo dài trong một chu kỳ tại thời điểm req[3] mang mức ưu tiên cao nhất. Kết quả dạng sóng thực tế của testcase 2 ở cả hai bộ phân xử đều đúng với mức ưu tiên thiết đặt trước và đúng với kết quả đã mô phỏng.



(a) Kết quả dạng sóng của testcase 3 ở bộ phân xử mức ưu tiên cố định

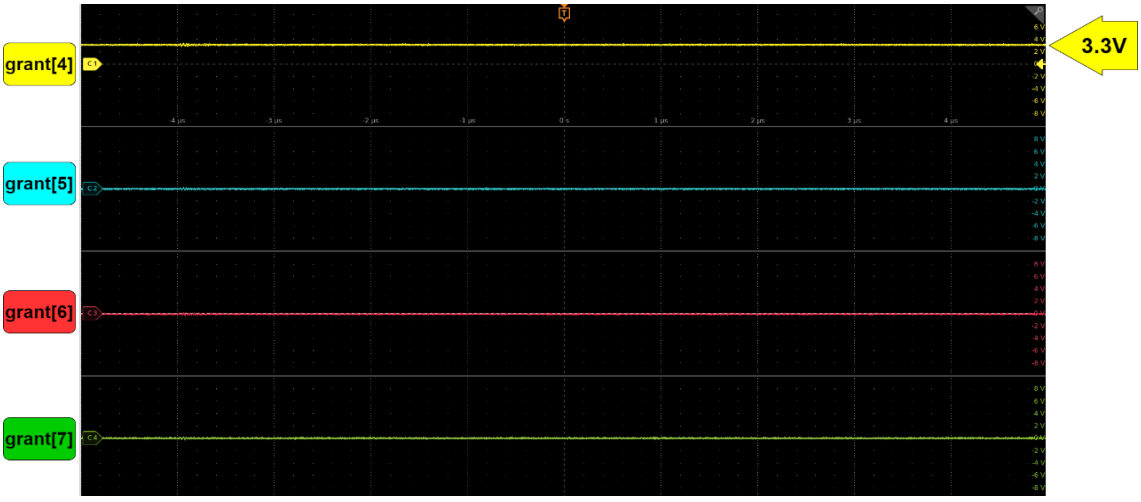


(b) Kết quả dạng sóng của testcase 3 ở bộ phân xử Round Robin

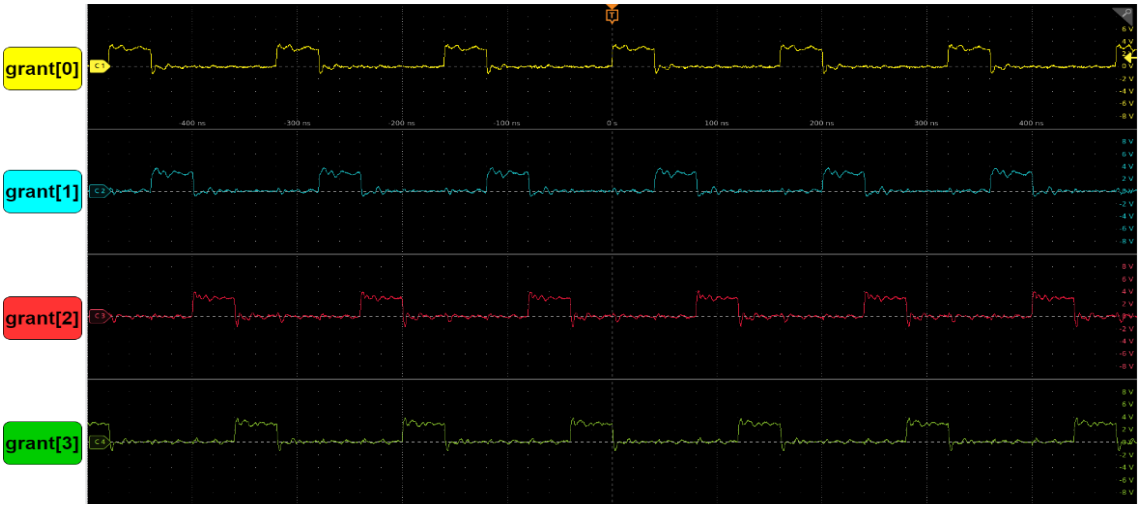
Hình 12. *Kết quả dạng sóng thực tế của testcase 3*

Hình 12 mô tả dạng sóng của testcase 3 của hai bộ phân xử khi trạng thái ngõ vào là req = 1110 ứng với req[3], req[2] và req[1] đang có yêu cầu truy cập. Dạng sóng ngõ ra ở hình 12 (a) là grant[7:4] = 0010 ứng với req[1] được cấp quyền ở bộ phân xử mức ưu tiên cố định. Đối với dạng sóng ngõ ra hình 12 (b) của bộ phân xử Round Robin, bởi vì req[1] có mức ưu tiên cao hơn req[2] và req[3] trong chu kỳ req[0] giữ mức ưu tiên cao nhất nên khi req[0] không có yêu cầu truy cập thì req[1] sẽ được cấp quyền,. Điều này giải thích cho dạng sóng tích cực mức cao của tín hiệu grant[1] kéo dài trong hai chu kỳ. Trong khi đó, dạng sóng tích cực mức cao của tín hiệu grant[2] và grant[3] chỉ kéo dài trong một chu kỳ tại thời điểm mà mức ưu tiên cao nhất được xoay vòng tương ứng cho

req[2] và req[3]. Kết quả dạng sóng thực tế của testcase 3 ở cả hai bộ phân xử đều đúng với mức ưu tiên được thiết đặt trước và đúng với kết quả mô phỏng.



(a) Kết quả dạng sóng của testcase 4 ở bộ phân xử mức ưu tiên cố định



(b) Kết quả dạng sóng của testcase 4 ở bộ phân xử Round Robin

Hình 13. *Kết quả dạng sóng thực tế của testcase 4*

Hình 13 mô tả dạng sóng của testcase 4 của hai bộ phân xử khi trạng thái ngõ vào là req = 1111 ứng với cả bốn request đang có yêu cầu truy cập. Dạng sóng ngõ ra ở hình 13 (a) là grant[7:4] = 0001 ứng với req[0] được cấp quyền ở bộ phân xử mức ưu tiên cố định. Đối với dạng sóng ngõ ra ở bộ phân xử Round Robin (grant[3:0]) được mô tả trong hình 13 (b), cả bốn tín hiệu request đồng thời có yêu cầu truy cập tại mọi thời điểm nên cả bốn request sẽ đều được cấp quyền lần lượt ở từng chu kỳ mà request đó mang mức ưu tiên cao nhất. Điều này lý giải cho việc chu kỳ dạng sóng tích cực của mỗi tín hiệu request đều như nhau và sẽ lập lại khi quyền ưu tiên cao nhất được xoay vòng lại cho chính nó. Kết quả dạng sóng thực tế của testcase 4 ở cả hai bộ phân xử đều đúng với mức phân xử được thiết đặt trước và đúng với kết quả đã mô phỏng trên phần mềm.

6. Kết luận

Trong bài báo này, chúng tôi đã thiết kế hai bộ phân xử mức ưu tiên cố định và bộ phân xử Round Robin với bốn Master và một Slave sử dụng ngôn ngữ miêu tả phần cứng Verilog. Đồng thời, chúng tôi cũng xây dựng các testcase để kiểm tra kết quả mô phỏng của cả hai bộ phân xử về mặt thuật toán và tốc độ phân xử. Kết quả cho thấy tốc độ phân xử của bộ phân xử mức ưu tiên cố định phụ thuộc hoàn toàn xung clock, nếu ngõ vào thay đổi nhanh hơn xung clock thì sẽ bị bỏ qua. Tốc độ phân xử ở bộ phân xử Round Robin không chỉ phụ thuộc vào xung clock mà còn phụ thuộc vào chu kỳ xoay các mức ưu tiên và tốc độ thay đổi của các request. Bên cạnh đó, kết quả thực hiện trên phần cứng Spartan 3E đã cho kết quả tương đồng với kết quả mô phỏng. Sự cân bằng trong việc cấp phát quyền truy cập ở bộ phân xử Round Robin không thực sự quá cao, nguồn request chưa được grant có thể bị bỏ qua nếu request này tích cực trong khoảng thời gian không được cấp quyền ưu tiên cao nhất. Hệ thống chỉ đang sử dụng tần số xung clock nội của phần cứng là 50 MHz, chưa được thực hiện trên đa dạng các nguồn xung với tần số khác nhau. Thông qua những kết quả đạt được trên mô phỏng và phần cứng thực tế, nhóm tác giả xin đề xuất một vài hướng phát triển cho đề tài như: mở rộng số lượng Master và Slave để sử dụng trong các hệ thống thực tế, thay đổi tần số để có thể sử dụng trong những hệ thống hoạt động ở tần số cao, tích hợp bộ phân xử vào các hệ thống với ngõ vào là các chuẩn truyền thông như I2C, UART hoặc SPI.

TÀI LIỆU THAM KHẢO

[1] Anh, N. H., & Trí, D. M. (2021). *Thiết Kế và Thi Công Bộ Truyền Nhận Theo Giao Thức I2C*. Trường Đại học Sư phạm kỹ thuật Thành phố Hồ Chí Minh.

[2] Deb, R., & Rajrajan, D. (2013). Speed efficient implementation of round robin arbiter design using VERILOG. *International Journal of Enhanced Research in Science Technology & Engineering, 2*(9), 1-9.

[3] Gupta, J., & Goel, N. (2015). *Efficient Bus Arbitration Protocol for SoC Design*. 396-400. DOI:10.1109/ICSTM.2015.7225449

[4] Hu, C., Chen, X., Li, W., & Liu, B. (2004). Fixed-Length Switching vs. Variable-Length Switching in Input-Queued IP Switches. *In 2004 IEEE International Workshop on IP Operations and Management*, pp. 117-122. DOI:10.1109/IPOM.2004.1547602

[5] Ngọc, P. T., & Bảo, H. N. (2020). *Thiết Kế Bộ Giao Tiếp Multichannel UART Sử Dụng FPGA*. Trường Đại học Sư phạm kỹ thuật Thành phố Hồ Chí Minh.

[6] Quân, N. (2019). [Arbiter] *Bài 1 - Phân xử theo mức ưu tiên cố định*. (VLSI Technology) Retrieved May 21, 2022, from https://nguyenquanicd.blogspot.com/2019/08/arbiter-bai-1-phan-xu-theo-muc-uu-tien.html

[7] Quân, N. (2019). [Arbiter] *Bài 3 - Bộ phân xử Round Robin đơn giản*. (VLSI Technology) Retrieved May 21, 2022, from https://nguyenquanicd.blogspot.com/2019/08/arbiter-bai-3-bo-phan-xu-round-robin-on.html.

[8] Shin, E. S., III, V. J., & Riley, G. F. (2002). Round-robin Arbiter Design and Generation. In *15th International Symposium on System Synthesis*, pp. 243-248.

[9] Weber, M. (2001). *Arbiters: Design Ideas and Coding Styles*. Boston: Synopsys Users Group (SNUG).

[10] Xilinx. (2011). *Spartan-3E FPGA Starter Kit Board User Guide*. Xilinx, Inc.