PHƯƠNG PHÁP HIỆN THỰC VI MẠCH BẤT ĐỒNG BỘ TRÊN FPGA

Đinh Đức Anh Vũ

Trường Đại học Bách khoa, ĐHQG-HCM

(Bài nhận ngày 27 tháng 04 năm 2011, hoàn chỉnh sửa chữa ngày 29 tháng 11 năm 2011)

***TÓM TẮT**: FPGA đã và đang chiếm ưu thế trong việc làm phương tiện hiệu quả để cung cấp khả năng làm mẫu và hiện thực nhanh chóng các mạch số với chi phí kỹ thuật thấp. Tuy vậy, hầu hết các loại FPGA cũng như quy trình thiết kế và hiện thực FPGA hiện tại không hỗ trợ cho việc hiện thực mạch bất đồng bộ bởi vì thiếu những phần tử cơ bản của mạch bất đồng bộ như cổng Muller. Nghiên cứu này sẽ trình bày hai phương pháp hiệu quả để hiện thực vi mạch bất đồng bộ trên các loại FPGA dạng Look-Up Table (LUT). Hai phương pháp này được xây dựng dựa trên kỹ thuật xây dựng những cổng Muller không nhiễu trong hai thư viện ở hai mức HDL và EDIF. Các ràng buộc về thời gian và/hoặc ràng buộc về vị trí được sinh ra tự động để bắt buộc công cụ hiện thực FPGA ánh xạ các phần tử này lên các khối luận lý thích hợp của FPGA. Các họ FPGA dạng LUT của Xilinx và Altera có thể được dùng để hiện thực mạch bất đồng bộ bằng hai phương pháp này.*

***Từ khóa**: Mạch bất đồng bộ, FPGA, LUT.*

1. GIỚI THIỆU

Thiết kế bất đồng bộ đang được đầu tư và phát triển bởi vì những ưu điểm rõ ràng có nó so với thiết kế đồng bộ như: không lệch xung nhịp, tiêu thụ năng lượng thấp, hiệu suất được tính trong trường hợp trung bình, khả năng chuyển đổi công nghệ tốt hơn và có khả năng mô-đun hóa [[1], [2]]. Có rất nhiều công trình nghiên cứu xây dựng các phương pháp luận để thiết kế các hệ thống bất đồng bộ lớn một cách hiệu quả. Mặc dù các hệ thống bất đồng bộ có thể được hiện thực trên các chip tích hợp mật độ cao thủ công (custom VLSI), nhưng thời gian chế tạo theo phương pháp này là quá dài để có thể làm mẫu và kiểm thử hệ thống.

Trong khi đó, FPGA là một phương tiện cho phép phát triển nhanh chóng các hệ thống mới có cấu trúc mạch phù hợp với nó [[3], [4], [5]]. FPGA đang là phương tiện chiếm ưu thế trong việc làm mẫu các mạch số. Tuy nhiên, kiến trúc của các loại FPGA thông dụng cũng như các công cụ hỗ trợ thiết kế (CAD) hiện tại không hỗ trợ mạch bất đồng bộ. Do đó, một phương pháp hỗ trợ hiện thực mạch bất đồng bộ trên các loại FPGA thông thường là một đòi hỏi cấp thiết để có thể phát triển các hệ thống bất đồng bộ.

Các công trình liên quan đến vấn đề này đã giới thiệu những kiến trúc FPGA mới hỗ trợ cả mạch đồng bộ lẫn bất đồng bộ như là MONTAGE [[6]], PHCB [[7]], PCA-1 hay PLB [[8], [9]], PGA-STC [[10]]. Nghiên cứu trong [[11]] xem xét đánh giá các kiến trúc FPGA thế hệ đầu tiên này trong lĩnh vực phát triển mạch bất đồng bộ trên FPGA. Tuy nhiên, cách tiếp cận này chỉ phù hợp với những phòng

thí nghiệm và những trường đại học lớn được đầu tư trang thiết bị đầy đủ.

Một cách tiếp cận khác để giải quyết bài toán hiện thực mạch bất đồng bộ trên FPGA là phát triển các thư viện các phần tử chuyên biệt cho mạch bất đồng bộ và sử dụng chúng trong khi hiện thực mạch bất đồng bộ trên FPGA truyền thống. Năm 1993, tại Đại học Utah, Erik Brunvand đã thiết kế một thư viện các phần tử để hiện thực các mạch tự định thời (self-timed circuits) trên FPGA của Actel [[12]]. Tuy nhiên, trong cách tiếp cận này, tác giả đã không xem xét đến vấn đề nhiễu của các phần tử khi hiện thực trên FPGA. Năm 1995, tại Đại học U.C.Davis, Kapian Masheswaran đã trình bày phương pháp hiện thực mạch bất đồng bộ trên họ FPGA Xilinx XC4000 bằng cách sử dụng các phần tử chuyên biệt cho mạch bất đồng bộ được định nghĩa trước với các hàm ràng buộc thời gian trễ của các tín hiệu [[10]]. Bởi vì các họ FPGA khác nhau sẽ có thời gian trễ tính toán trên các khối luận lý cũng như trên các kênh dây dẫn là khác nhau nên phương pháp này không thể dùng cho những họ FPGA khác. Ru. R. Mocho và các đồng sự đã giới thiệu một phương pháp sử dụng ngôn ngữ đặc tả phần cứng VHDL để hiện thực mạch bất đồng bộ trên các loại FPGA [[13]]. Để hỗ trợ phương pháp của mình các tác giả đã đặc tả trước các phần tử dùng riêng cho mạch bất đồng bằng VHDL. Tuy nhiên, vấn đề nhiễu của các phần tử khi hiện thực trên FPGA đã không được quan tâm.

Nghiên cứu của phòng thí nghiệm TIMA, Pháp, đã giới thiệu phương pháp hiện thực mạch bất đồng bộ trên FPGA thông thường. Đóng góp chính của nghiên cứu là đã chứng minh các cổng Muller [[14], [15], [16]] sẽ không gây ra nhiễu nếu nó được hiện thực trên các họ FPGA dạng LUT trong một số điều kiện ràng buộc [[16]]. Phương pháp này sử dụng tập tin định dạng XNF (Xilinx Netlist Format) để tạo các cổng Muller, nhưng định dạng này đã không còn được hỗ trợ nữa.

Trong nghiên cứu này, chúng tôi giới thiệu hai cách tiếp cận để hiện thực mạch bất đồng bộ trên FPGA thông thường. Cách tiếp cận đầu tiên là sử dụng quy trình thiết kế FPGA truyền thống kết hợp với sử dụng các cổng Muller không nhiễu được định nghĩa trước trong một thư viện xây dựng bằng HDL. Cách tiếp cận thứ hai là kết hợp phương pháp luận thiết kế mạch bất đồng bộ PAiD (Project Asynchronous circuits Design) được phát triển bởi trường Đại học Bách Khoa với công cụ hiện thực FPGA truyền thống. Trong cả hai cách tiếp cận này, chúng tôi sử dụng các ràng buộc trong tập tin .ucf (user constraints file) [[17]] để chắc chắn các cổng Muller được ánh xạ vào khối luận lý thích hợp nhằm bảo đảm không xảy ra nhiễu trên các cổng này.

Phần 2 của bài báo giới thiệu những khái niệm cơ bản về mạch bất đồng bộ. Phần 3 sẽ trình bày cách hiện thực cổng Muller không nhiễu trên FPGA của Xilinx. Hai cách tiếp cận để hiện thực mạch bất đồng bộ trong nghiên cứu này được chúng tôi mô tả chi tiết trong phần 4. Phần 5 sẽ trình bày kết quả thực nghiệm đã đạt được. Cuối cùng phần 6 sẽ trình bày kết luận và định hướng sắp tới.

2. MẠCH BẤT ĐỒNG BỘ

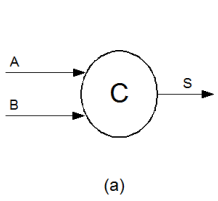
Hệ thống bất đồng bộ [[2]] bao gồm một điều khiển dựa trên sự kiện (hoặc mức) và một sự phân chia dữ liệu. Không giống như mạch đồng bộ sử dụng một tín hiệu xung nhịp toàn cục điều khiển tất cả các hoạt động, các hoạt động trong mạch bất đồng bộ được điều khiển cục bộ bởi các tín hiệu bắt tay (handshaking signal). Giao thức bắt tay này được định nghĩa trên một cặp tín hiệu yêu cầu hành động (req) và xác nhận hành động đã hoàn thành (ack) như trong Hình 1. Để đảm bảo tính đúng đắn và thứ tự, các mô-đun giao tiếp với nhau phải đảm bảo các quy tắc:

Quy tắc 1: Bên gửi không được gửi tín hiệu yêu cầu mới cho đến khi yêu cầu cũ trước đó được phản hồi.

Quy tắc 2: Bên nhận không được gửi tín hiệu phải hồi trừ khi nó nhận được tín hiệu yêu cầu.

Hình 1. Giao tiếp yêu cầu và xác nhận

Bởi vì các giao thức bắt tay dựa trên sự thay đổi mức các tín hiệu nên nhiễu không được phép xảy ra. Để hiện thực các giao thức bắt tay này, ngoài các cổng cơ bản người ta còn phải dùng thêm cổng Muller. Các cổng Muller đóng vai trò rất quan trọng trong việc hiện thực mạch bất đồng bộ. Nó được dùng để hiện thực các mạch điều khiển bắt tay và kiểm tra việc hoàn thành tính toán. Cổng Muller đối xứng 2 ngõ nhập, được gọi là cổng MULLER2, được mô tả trong Hình 2. Hình 3 trình bày hiện thực cổng MULLER2 không nhiễu bằng cổng AND và OR.



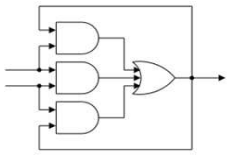
(a)

1: if A=B then S:=A
2: S = AB + S(A+B)
3: Truth table

A	B	S
0	0	0
0	1	No Change
1	0	No Change
1	1	1

(b)

Hình 2. (a) ký hiệu; (b) đặc tả cổng MULLER2



S^{-1}

AB	00	01	11	10
0	0	1	0	
0	1	1	1	

Hình 3. Hiện thực cổng MULLER2 không nhiễu bằng các cổng AND và OR

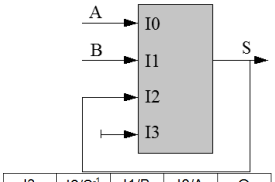
3. HIỆN THỰC CỔNG MULLER TRÊN FPGA DẠNG LUT

Trong phần này chúng tôi sẽ trình bày cách hiện thực các cổng Muller không nhiễu trên họ FPGA Xilinx Spartan-3. Tuy nhiên, phương pháp này hoàn toàn có thể được áp dụng để hiện thực cho các loại FPGA dạng LUT khác như Altera Cyclone II, Cyclone III. Điểm khác biệt chính giữa Altera và Xilinx là kích thước của LUT và cấu trúc tập tin ràng buộc .ucf.

Trong Xilinx Spartan-3 các khối luận lý khả cấu hình (Configurable Logic Blocks - CLB) cung cấp tài nguyên chủ yếu để hiện thực các mạch đồng bộ cũng như các mạch tổ hợp. Mỗi CLB bao gồm 4 Slice được kết nối với nhau. Mỗi Slice gồm 2 LUT 4 ngõ nhập. Mỗi LUT có 4 ngõ nhập là I1-I4 và một ngõ ra O. Điều này cho phép bất kỳ hàm Boolean 4 biến

nào cũng có thể được hiện thực dùng 1 LUT. Bài báo [[10]] đã chứng minh rằng khi hiện thực hàm bằng LUT thì nhiễu do hiện thực sẽ không xảy ra.

Một cổng Muller 2 ngõ nhập có thể được hiện thực bằng một LUT-4 (LUT có 4 ngõ nhập) và đảm bảo không nhiễu. Hình 4 mô tả hiện thực của cổng MULLER2 và bảng thực trị của nó. Trong cách hiện thực này các ngõ vào A, B và ngõ ra S của MULLER2 được ánh xạ vào các ngõ vào I0, I1 và ngõ ra O của LUT-4 theo thứ tự đó. Tín hiệu hồi tiếp S^{-1} được ánh xạ vào ngõ vào I2 của LUT-4. Biểu thức cho tín hiệu S là S = I0·I1 + I1·I2 + I2·I0. Trong đó I0, I1, I2 và I3 lần lượt là các tín hiệu lựa chọn của LUT. Bảng thực trị trong Hình 4 sẽ được ghi vào bộ nhớ của LUT-4 hiện thực cổng MULLER2.



I3	I2/S^{-1}	I1/B	I0/A	O
x	0	0	0	0
x	0	0	1	0
x	0	1	0	0
x	0	1	1	1
x	1	0	0	0
x	1	0	1	1
x	1	1	0	1
x	1	1	1	1

Hình 4. Ánh xạ cổng MULLER2 vào LUT-4 và bảng thực trị của LUT-4

Đối với các cổng Muller chiếm nhiều hơn một LUT như MULLER3R (cổng Muller 3 ngõ nhập có tín hiệu reset) do thời gian trễ trên các dây nối giữa các LUT nên nhiễu có thể xảy ra. Bài báo [[16]] đã chứng minh không hình thức rằng nếu cổng Muller được hiện thực trên 1 CLB thì nhiễu do hiện thực sẽ không xảy ra.

Để giải quyết vấn đề các cổng Muller sử dụng nhiều hơn một LUT phải được ánh xạ vào một CLB, nghiên cứu này sẽ dùng ràng buộc vị trí. Mặc dù các Slice trong CLB của Xilinx có hai LUT-4 nhưng bộ công cụ P&R (Place and Route) không tự động sắp đặt hai LUT hiện thực cho một cổng Muller vào một Slice. Các ràng buộc vị trí sẽ được sinh ra và đặt vào trong tập tin ràng buộc .ucf. Các ràng buộc này sẽ điều khiển bộ P&R sắp đặt các LUT hiện thực một cổng Muller vào cùng một Slice. Cấu trúc của ràng buộc vị trí được mô tả trong Hình 5 [[17]]. Trong đó **"<Name LUT>"** là tên của LUT, tên này được gán tự động bởi công cụ tổng hợp, **RLOC=X#Y#** chỉ ra mối quan hệ vị trí giữa các LUT; tức là nếu hai LUT có cùng giá trị **X#Y#** thì nó sẽ được đặt trên cùng một Slice của Xilinx CLB, nếu hai LUT có giá trị **X#Y#** tương ứng lần lượt là **X0Y0** và **X0Y1** thì hai LUT sẽ được ánh xạ vào hai Slice kế nhau trên cùng một hàng.

INST	"<Name_LUT>"	RLOC=X#Y#

Hình 5. Cấu trúc ràng buộc của Xilinx FPGA

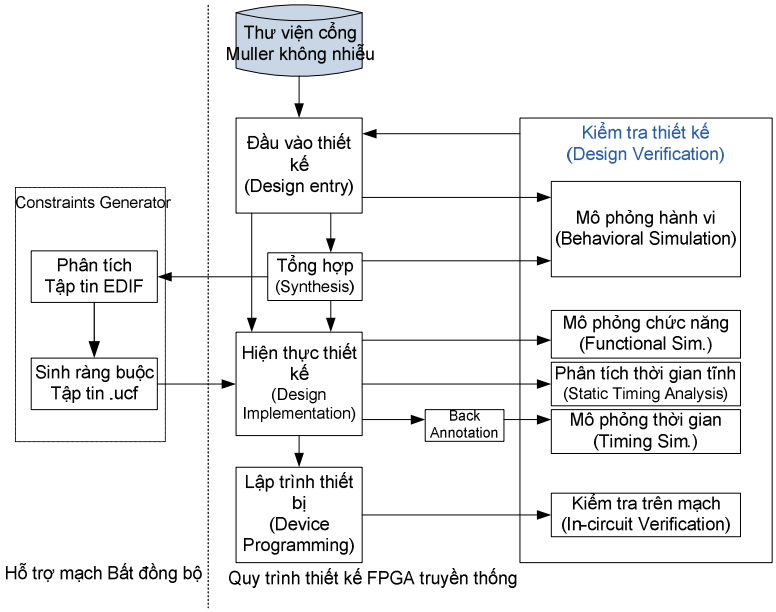
4. CÁC PHƯƠNG PHÁP HIỆN THỰC MẠCH BẤT ĐỒNG BỘ TRÊN FPGA

4.1. Phương pháp dùng quy trình thiết kế FPGA truyền thống

Từ phương pháp hiện thực cổng Muller không nhiễu như ở phần 0 và quy trình thiết kế FPGA truyền thống, nghiên cứu này đề nghị một phương pháp (PP1) hiện thực mạch bất

đồng bộ trên FPGA có thể tóm tắt trong hình 6.

Đầu vào của quy trình thiết kế này là mạch bất đồng bộ dạng QDI (Quasi-Delay Insensitive) được đặt tả bằng mô hình cấu trúc sử dụng những cổng Muller không nhiễu đã được định nghĩa trước. Mạch bất đồng bộ này có thể đã được thiết kế bằng tay hay bằng phương pháp thiết kế nào đó như Tangram của Philip hay Balsa của Đại học Manchester…. Do đó mạch bất đồng bộ này đã được bảo đảm tính đúng đắn cũng như thỏa mãn các ràng buộc của mạch bất đồng bộ. Các cổng Muller tham gia hiện thực mạch bất đồng bộ này đã được thiết kế dùng những phần tử cơ bản của FPGA là các LUT và các Multiplexer nên sẽ không bị thay đổi bởi bộ tổng hợp. Netlist kết quả tổng hợp được sẽ bao gồm các LUT và các Multiplexer hiện thực các cổng cơ bản và các cổng Muller.



Hình 6. Quy trình hiện thực mạch bất đồng bộ trên FPGA trong PP1

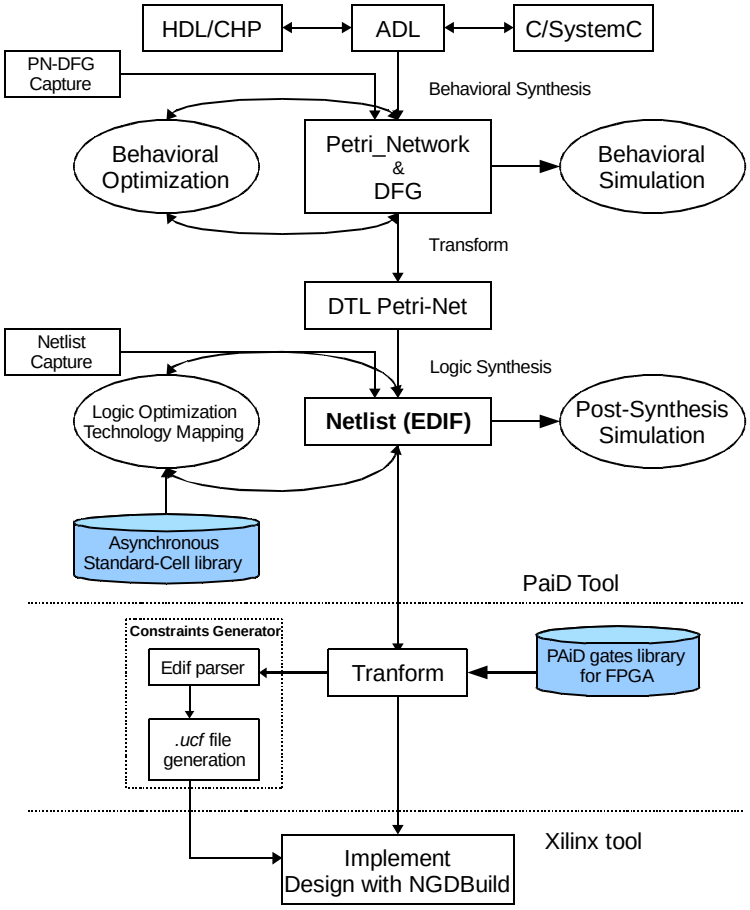
Như đã đề cập ở trên các cổng Muller tham gia trong mạch bất đồng bộ phải được sắp đặt trên cùng một CLB, điều này có thể làm được bằng cách xây dựng các ràng buộc. Do đó cần có một quy trình phụ cũng như một công cụ hỗ trợ để sinh tự động các ràng buộc này. Nghiên cứu này phát triển một công cụ sinh ràng buộc tự động (Constraints Generator) nhằm yêu cầu công cụ P&R sắp đặt các LUT hiện thực cho từng cổng Muller thuộc cùng một CLB của FPGA. Đầu vào của công cụ này là mạch bất đồng bộ đã được tổng hợp và được đặc tả ở định dạng EDIF. Constraints Generator sẽ phân tích và tìm ra những LUT dùng hiện thực cho một cổng Muller để sinh ra ràng buộc về vị trí theo định dạng của tập tin .ucf cho từng họ FPGA.

Mạch bất đồng bộ sau khi tổng hợp và tập tin ràng buộc sẽ được tiếp tục hiện thực trong bước hiện thực thiết kế. Quá trình mô phỏng

sau P&R sẽ được thực hiện để bảo đảm tính đúng đắn của mạch sau khi lập trình trên FPGA thực tế.

4.2. Phương pháp kết hợp PAiD và phần mềm hỗ trợ hiện thực FPGA

PAiD [[14]] là phương pháp luận để thiết kế vi mạch bất đồng bộ dạng QDI được phát triển bởi trường Đại học Bách Khoa – ĐHQG-TPHCM năm 2008. Phần đầu Hình 7 trình bày quy trình thiết kế của phương pháp PAiD. Xuất phát từ đặc tả cấp cao theo ngôn ngữ ADL (Asynchronous Design Language) hoặc HDL/CHP/SystemC của vi mạch, quá trình tổng hợp hành vi sẽ biến đổi đặc tả này thành dạng mô phỏng trung gian của vi mạch bất đồng bộ bằng sự kết hợp giữa Petri-Net và DFG (Data Flow Graph). Biểu diễn trung gian này sau khi được tối ưu và mô phỏng tính đúng đắn sẽ được chuyển đổi sang dạng DTL Petri-Net khả tổng hợp. Cuối cùng quá trình tổng hợp luận lý sẽ sinh ra cấu trúc netlist của mạch bất đồng bộ theo định dạng chuẩn EDIF. Netlist mạch bất đồng bộ tạo ra bởi PAiD sẽ có được tính đúng đắn cũng như được tối ưu ở nhiều mức luận lý khác nhau.



Hình 7. Quy trình thiết kế mạch bất đồng bộ trong PP2

Trong nghiên cứu này, chúng tôi kết hợp PAiD với công cụ hiện thực FPGA truyền thống để tạo ra quy trình thiết kế và hiện thực mạch bất đồng bộ trên FPGA hoàn chỉnh (PP2). Do mỗi họ FPGA sẽ có cấu trúc khác nhau và quy trình thiết kế khác nhau, cho nên nghiên cứu này sẽ trình bày chi tiết quy trình thiết kế sử dụng kết hợp công cụ PAiD và công cụ hỗ trợ hiện thực FPGA NGDBuild [[18]] của Xilinx (Hình 7). Việc kết hợp PAiD và các phần mềm hỗ trợ hiện thực khác có thể được thực hiện tương tự.

Các cổng trong netlist mạch bất đồng bộ sinh ra bởi PAiD được hiện thực trong thư viện các phần tử mạch bất đồng bộ cơ bản (bao gồm các cổng cơ bản và cổng Muller). Tuy nhên, FPGA lại sử dụng các LUT để hiện thực các cổng Muller và các mạch tổ hợp. Do đó, để netlist sinh ra bởi PAiD có thể được hiện thực trên Xilinx FPGA thì nó cần được chuyển đổi về dạng kết nối giữa các LUT. Trong nghiên cứu này chúng tôi xây dựng một công cụ được gọi là công cụ Transform. Công cụ này có nhiệm vụ thực hiện việc chuyển đổi cấu trúc netlist sinh ra bởi PAiD về dạng kết nối giữa các phần tử cơ bản của FPGA (LUT, bộ phân kênh…. [[19]]). Các cổng Muller được định nghĩa trước trong thư viện theo chuẩn EDIF để bảo đảm không xảy ra nhiễu khi hiện thực trên FPGA. Kết quả của netlist sau khi được chuyển đổi sẽ phù hợp để hiện thực trên FPGA.

Ngoài ra, cũng như ở phương pháp thứ nhất, để đảm bảo các cổng Muller không gây ra nhiễu khi hiện thực trên FPGA cần phải có các ràng buộc về vị trí cho các LUT tạo thành cổng Muller. Công cụ Constraints Generator sẽ được sử dụng để thực hiện công việc này. Cuối cùng, netlist định dạng EDIF sau khi đã chuyển đổi phù hợp với FPGA và tập tin ràng buộc sẽ được biến đổi trong công cụ NGDBuild và được hiện thực trên FPGA Xilinx.

4.3. So sánh giữa hai phương pháp

Trong phương pháp thứ nhất, người thiết kế sử dụng ngôn ngữ đặc tả phần cứng HDL để đặc tả mạch bất đồng bộ và dùng công cụ tổng hợp đi kèm FPGA để tổng hợp mạch thành cấu trúc các LUT. Do đó, số lượng các LUT sử dụng trong phương pháp này sẽ ít hơn phương pháp thứ hai. Tuy nhiên do phương pháp này sử dụng mô hình cấu trúc để đặc tả mạch bất đồng bộ nên rất khó có thể hiện thực những vi mạch tinh vi có kích thước lớn, hơn nữa việc kiểm tra mô phỏng đối với đặc tả cấu trúc phải được tiến hành rất cẩn thận và tốn nhiều chi phí.

Ngược lại với phương pháp thứ nhất, phương pháp thứ hai cho phép hiện thực được những vi mạch có kích thước lớn bởi vì mạch bất đồng được đặc tả bằng ngôn ngữ cấp cao chuyên biệt cho dạng mạch QDI. Mạch bất động bộ được tổng hợp sinh ra từ PAiD sẽ bảo đảm tính đúng đắn và tối ưu. Ngoài ra, định dạng EDIF cũng cho phép mạch QDI này có thể được hiện thực trên nhiều loại FPGA khác nhau. Tuy vậy khi tiếp cận với phương pháp hiện thực này cần phải sử dụng thêm công cụ Transform để chuyển đổi mạch bất đồng bộ theo cấu trúc của PAiD về dạng LUT nên số lượng LUT sẽ không nhiều hơn trong phương pháp 1.

5. KẾT QUẢ THỰC NGHIỆM

Sau quá trình nghiên cứu và thử nghiệm chúng tôi đã xây dựng hai thư viện các cổng Muller không nhiễu ở hai mức HDL và EDIF để hiện thực mạch bất đồng bộ trên FPGA Xilinx Spartan-3 như trong Bảng 1.

Bảng 1. Các cổng Muller đã hiện thực

Cổng Muller	Đặc tả	# LUT
	Hai ngõ nhập	
MULLER2	MULLER2	1
MULLER2R	reset	1
MULLER3	Ba ngõ nhập	1
MULLER3R	MULLER3	2
MULLER4	reset	2
	Bốn ngõ nhập	

Chúng tôi đã hiện thực các mạch bất đồng bộ Buffer, Comparator và Selector [[20]] bằng hai phương pháp trên với FPGA Xilinx Spartan-3 xc3s200-5ft256. Kết quả mô phỏng cho thấy hai phương pháp đã trình bày là đúng đắn. Bảng 2 trình bày tổng kết về kết quả hiện thực các mạch bất đồng bộ trên xc3s200-5ft256 để kiểm tra phương pháp

Bảng 2. Kết quả hiện thực các mạch bất đồng bộ trên xc3s200-5ft256

Mạch	# LUT/Slice xc3s200-5ft256	
	PP1	PP2
Buffer	3/2	3/2
Comparator	6/3	6/3
Selector	20/10	21/11

6. KẾT LUẬN VÀ CÔNG VIỆC TIẾP THEO

Báo cáo này đã trình bày hai phương pháp hiện thực mạch bất đồng bộ trên FPGA dạng LUT. Các phương pháp này dựa trên việc xây dựng cổng Muller không nhiễu. Qua những kết quả mô phỏng, có thể kết luận rằng hai phương pháp đúng đắn. Báo cáo này cũng đã trình bày cách xây dựng các cổng Muller không nhiễu trên các LUT.

Với những kết quả đã đạt được, chúng tôi tin tưởng rằng, nghiên cứu mạch bất đồng bộ sẽ được đẩy mạnh trong cộng đồng nghiên cứu và trong Đại học Bách Khoa nhờ khắc phục được một trong những hạn chế của mạch bất đồng bộ là thiếu phương tiện hiện thực. Trong giai đoạn sắp tới, chúng tôi sẽ thực hiện việc kết hợp PAiD và các phần mềm hỗ trợ hiện thực FPGA khác như Quartus II của Altera. Ngoài ra, chúng tôi cũng có kế hoạch phát triển các phương pháp này cho FPGA không dựa trên LUT như FPGA dạng Flash của Actel.

METHOD FOR IMPLEMENTING ASYNCHRONOUS CIRCUITS ON FPGA

Dinh Duc Anh Vu

University of Technology, VNU-HCM

ABSTRACT*: FPGA device is a dominant implementation medium for digital circuits. Unfortunately, they do not support asynchronous circuits because of the lack of asynchronous circuit elements such as Muller gates, etc. In this paper, new efficient approaches are proposed to prototype asynchronous circuits on Look-Up Table-based (LUT) FPGA rapidly. The developed techniques are based on building of elements which play an important role in asynchronous circuits. The hazard-free elements are predefined in libraries in HDL and EDIF format. Timing and/or area constraints for place&route tool are automatically generated to map the asynchronous elements on suitable FPGA's logic blocks. Several FPGA devices such as Altera, Xilinx and Actel could be used as target for the implementation.*

Keywords*: Asynchronous circuits, FPGA, LUT.*

TÀI LIỆU THAM KHẢO

[1] Scott Hauck, *Asynchronous Design Methodology: An Overview*, Proceeding of the IEEE, Vol. 83, No.1, 69-93 (1995).

[2] Jen Sparso, Steve Furber, *Principles of Asynchronous Circuit Design – A system Perspective*, Springer Publisher (2001).

[3] K. Compton, S. Hauck, *Reconfigurable Computing: A Survey of Systems and Software*, ACM Computing Surveys, Vol. 34, No. 2, 171-210 (2002).

[4] Scott Hauck, André DeHon, *Reconfigurable Computing: The Theory and Practice of FPGA-Based Computation*, Morgan Kaufmann Publisher (2007).

[5] Stephen D.Brown, Jonathan Rose, Robert J.Francis, Zvonko G.Vranesic, *Field-Programmable Gate Arrays*, Kluwer Academic Publisher (1992).

[6] Scott Hauck, Steven Burns, Gaetano Borriello, Carl Ebeling, *An FPGA for Asynchronous Circuits*, IEEE Design & Test of Computer, Vol. 11, No. 3, 60-69 (1994).

[7] Wong, C.G. Martin, A.J. Thomas, P., *An architecture for asynchronous FPGAs*, In Field-Programmable Technology (FPT), Proceedings IEEE International Conference, 170-177, 15-17 (2003).

[8] Fesquest L., Renaudin R., *A Programmable Logic Architecture for Prototyping Clockless Circuit*, TIMA Lab. Research Report (2005).

[9] N.Huot, H.Dubreuil, L.Fesquet and M.Renaudin, *FPGA Architecture for*

Multiple-style Asynchronous Logic, DATE, 32-33 (2005).

[10] Kapian Maheswaran, *Implementing Self-Timed Circuits in Field Programmable Gate Arrays*, Master Thesis, U.C.Davis (1995).

[11] R.Payne, *Asynchronous FPGA Architectures*, In Computers and Digital Techniques, in IEEE Proceedings, Vol. 143, No. 5, 282-286 (1996).

[12] Erik Brunvard, *Using FPGAs to Implement Self-Timed Systems*, In Journal of VLSI Signal Processing, Vol. 6, No. 2, 173-190 (1993).

[13] R.U.R.Mocho, G.H.Sartori, R.P.Ribas, A.I.Reis, *Asynchronous Circuits design on reconfigurable devices*, Proceedings of the 19th annual symposium on Integrated circuits and systems design, Brazil, 20-25 (2006).

[14] A.V. Dinh-Duc et al., *A Methodology for implementing QDI Asynchronous circuits*, Technical report, VNU-HCM, HCMUT (2008).

[15] Kapian Maheswaran , Jonathan Lipsher, *A Cell Set for Seft-Timed Design Using Xilinx XC4000 Series FPGA*, Technical report, Electrical and Computer Engineering Dept. UC Davis.

[16] Quoc Thai Ho, J.-B.Rigaud, L.Fesquet, M.Renaudin, and R.Rolland, *Implementing Asynchronous Circuits on LUT Based FPGAs*, Proceedings of the Reconfigurable Computing Is Going Mainstream, 12th International Conference on Field-Programmable Logic and Applications, 36-46 (2002).

[17] *Constraints Guide*, Xilinx tutorial, (2002-2008).

[18] *NGDBuild tool*, Xilinx doc, available at http://www.xilinx.com/itp/xilinx4/data/docs/dev/ngdbuild.html.

[19] *Spartan-3 Libraries Guide for HDL Design*, Xilinx tutorial, available at http://www.xilinx.com (2008).

[20] Phạm Quốc Cường, *Nghiên cứu và xây dựng phương pháp hiện thực vi mạch bất đồng bộ trên FPGA*, Luận văn Thạc sĩ, Đại học Bách Khoa (2009).