MÔ HÌNH C/C++ CHO LÕI CPU 32-BIT CÓ ĐƠN VỊ QUẢN LÝ BỘ NHỚ

NGUYỄN HOÀNG NGỌC NGÂN, LÊ HÀ THIÊN
Trung tâm Nghiên cứu và Đào tạo thiết kế vi mạch (ICDREC)

Một lõi vi xử lý 32 bit đã được thiết kế và thực hiện với nhiều kỹ thuật tinh tế như đường ống, thực hiện lệnh không theo thứ tự, có đơn vị quản lý bộ nhớ (MMU - Memory Management Unit), dự đoán lệnh rẽ nhánh... Nhằm hỗ trợ việc kiểm tra chức năng cho lõi vi xử lý có MMU bằng cách tạo ra các mẫu ngẫu nhiên, mô hình chức năng lõi đã được xây dựng bằng ngôn ngữ C/C++ dùng để kiểm soát các luồng lệnh, hoạt động và chức năng của lõi vi xử lý. Bài viết trình bày cấu trúc chức năng và hoạt động của mô hình CPU 32 bit (viết tắt là MH-C) bao gồm mô phỏng tập lệnh, mô phỏng MMU và cách thức giao tiếp với môi trường kiểm tra.

***Từ khóa:* CPU, đơn vị quản lý bộ nhớ, dự đoán lệnh rẽ nhánh, đường ống, bộ đếm chương trình (PC), BIU, bộ nhớ đệm.**

AC/C++ MODEL FOR THE 32-BIT MICRO-PROCESSOR WITH A MEMORY MANAGEMENT UNIT

Summary

A 32-bit microprocessor is designed and built using various sophisticated techniques, such as pipelining, out of order execution, memory management unit (MMU), branch prediction, and so on. To promote the verification for the core with the MMU using the random sample, the functional model is built to control the flow of instruction, operation and function of the microprocessor in C/C++ languages. This paper is dedicated to introduce the structures and the functions of the CPU 32-bit model including the instruction simulation, the MMU simulation and the interface with the testing environment.

***Key words:* CPU, Memory Managerment Unit (MMU), Branch Prediction, pipeline, Program counter (PC), BIU, Cache.**

Giới thiệu

Cấu trúc của vi xử lý 32 bit ngày càng trở nên phức tạp. Ở Việt Nam, ICDREC đã chế tạo thử nghiệm thành công vi xử lý 32 bit đầu tiên (chip VN1632) với cấu trúc đường ống 5 tầng, chỉ có một luồng lệnh, không có MMU [1]. Hiện tại, ICDREC đang thiết kế và gần hoàn thiện vi xử lý 32 bit VN32LP [2] có MMU với độ phức tạp lớn hơn lõi vi xử lý tiền thân. Việc kiểm tra trực tiếp từng luồng lệnh dựa trên các chức năng mô tả trong thiết kế tỏ ra không hiệu quả, mất nhiều thời gian. Trong khi đó, yêu cầu rút ngắn thời gian nhưng vẫn đảm bảo tính chính xác của các chức năng đòi hỏi việc kiểm tra vi xử lý phải hữu hiệu hơn. Đó là đổi từ phương pháp kiểm tra trực tiếp sang phương pháp tạo các mẫu kiểm tra ngẫu nhiên dưới sự hỗ trợ của VMM (Verification Methodolgy Manual) [3].

Tuy nhiên, trong việc kiểm tra vi xử lý bằng cách tạo các mẫu ngẫu nhiên, làm sao biết các lệnh được xử lý đúng hay không và kết quả do vi xử lý tạo ra có đúng hay không? Để giải quyết vấn đề này, một kỹ thuật đơn giản là xây dựng một mô hình chuẩn cho vi xử lý [4]. Mô hình được xây dựng bằng một ngôn ngữ khác bởi một nhóm thiết kế độc lập. Trong bài báo này, MH-C được xây dựng trên ngôn ngữ C/C++ có tính năng tương đương với vi xử lý. Trong môi trường kiểm tra, lệnh sẽ được đưa đến cho vi xử lý và MH-C.

Kết quả trả về của hai mô hình này sẽ được so sánh với nhau. Những sai biệt của hai kết quả được xem như là lỗi.

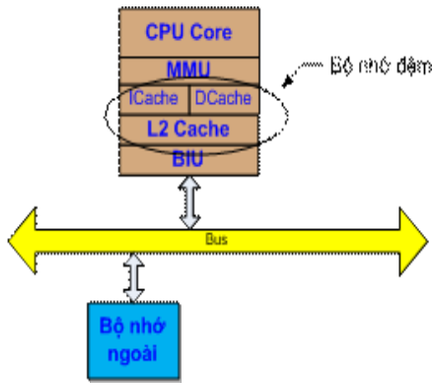
Trong [5], tác giả đã đề cập đến việc xây dựng tập lệnh và xử lý ngắt. Tuy nhiên, đối với vi xử lý có MMU, thì cần phải giải quyết thêm vấn đề phức tạp về chuyển đổi địa chỉ và cách thức xác định khi nào cần lấy dữ liệu ở bộ nhớ đệm và khi nào cần lấy dữ liệu ở bộ nhớ ngoài.

MH-C xây dựng cho vi xử lý có MMU có các chức năng như xử lý tập lệnh, chức năng phát hiện ngắt ngoại, phát hiện các ngắt nội của vi xử lý như cờ báo tràn, lỗi địa chỉ, lỗi chuyển đổi địa chỉ, lỗi đường bus… Ngoài ra, tương tự như MMU trong vi xử lý, MH-C cũng sẽ có bộ nhớ đệm. Dữ liệu truy xuất bộ nhớ sẽ nằm trong các bộ nhớ đệm hoặc các bộ nhớ ngoài, việc lấy dữ liệu từ một trong hai bộ nhớ này là do khối Điều khiển bộ nhớ quyết định. Kết quả xử lý của MH-C trả về cho môi trường kiểm tra qua chuẩn System Verilog DPI (Direct Programming Interface) [6].

Bên cạnh đó, bài viết này cũng sẽ trình bày kết quả ứng dụng MH-C trong việc kiểm tra thiết kế vi xử lý VN32LP của ICDREC.

Vi xử lý 32 bit có MMU

Trong vi xử lý 32 bit như được mô tả khái quát ở hình 1, CPU (Central Processing Unit) đọc lệnh từ bộ nhớ ngoài thông qua BIU (Bus Interface Unit), sau đó lưu lệnh vào trong các bộ nhớ đệm (L2 Cache, ICache). Trong quá trình hoạt động của mình, nếu CPU muốn sử dụng lại lệnh đó thì có thể truy cập vào trong các bộ nhớ đệm. Tương tự như



Hình 1: mô hình vi xử lý 32 bit có MMU

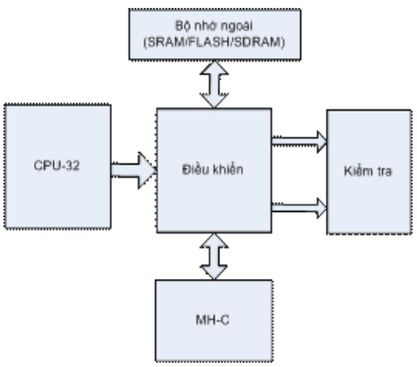
vậy, các dữ liệu của các lệnh truy xuất bộ nhớ cũng sẽ được lưu trong các bộ nhớ đệm (L2 Cache và DCache). MMU sẽ chịu trách nhiệm chuyển đổi địa chỉ ảo của CPU thành các địa chỉ thật dùng để truy xuất các bộ nhớ.

Xây dựng mô hình CPU

Phần sau đây trình bày các thành phần trong MH-C, lưu đồ xử lý các lệnh truy xuất dữ liệu và phương thức kết nối với môi trường kiểm tra System Verilog. Mặc dù MH-C được xây dựng và mở rộng dựa trên mô hình [5], nhưng cũng gặp không ít khó khăn trong việc xây dựng giải thuật truy xuất dữ liệu cho MMU; xác định dữ liệu truy xuất là ở trong bộ nhớ đệm hay bộ nhớ ngoài. Ngoài ra, vì vi xử lý hoạt động theo chu kỳ xung và kiến trúc đường ống, còn MH-C lấy lệnh và trả kết quả tức thời. Ban đầu, việc lấy lệnh cho MH-C dựa vào những tầng xử lý đầu tiên. Trong khi đó, vi xử lý được chỉnh sửa và thay đổi cấu trúc sau khi phát hiện ra lỗi. Điều này buộc phải thường xuyên điều chỉnh phương thức nhận lệnh để gửi đến MH-C. Để tránh tình trạng này, lệnh được lấy trực tiếp từ trong các bộ nhớ ngoài dựa vào PC do MH-C đưa ra sau khi chuyển từ địa chỉ ảo sang thật. Do vậy, lệnh đưa cho MH-C độc lập với cấu trúc vi xử lý.

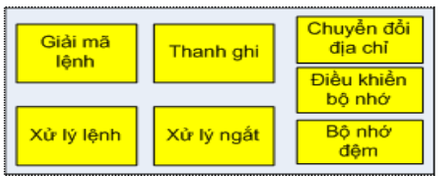
Kiến trúc tổng quát của môi trường kiểm tra

Trong môi trường kiểm tra ở hình 2, khối điều khiển có nhiệm vụ thu thập các kết quả xử lý của lệnh từ CPU. Tùy theo CPU xử lý 1 hoặc 2 lệnh cùng một thời điểm, khối điều khiển sẽ thu thập kết quả của lệnh trong các thanh ghi chung, thanh ghi đồng xử lý và PC. Đồng thời, khối điều khiển cũng sẽ lấy các kết quả do MH-C xử lý và đưa tất cả qua khối kiểm tra. Khối kiểm tra sẽ so sánh hai kết quả này. Nếu sai, thì môi trường sẽ dừng mô phỏng và thông báo các sai khác này. Dựa vào đó, người kiểm tra sẽ tìm ra lỗi của thiết kế.



Hình 2: môi trường kiểm tra

Cấu trúc bên trong MH-C bao gồm các khối như trong hình 3. Trong đó, khối giải mã lệnh có chức năng tách các trường lệnh để nhận biết lệnh nào đã được đưa vào MH-C. Khối xử lý lệnh có nhiệm vụ xử lý các lệnh được nhận biết bởi khối giải mã lệnh. Nếu lệnh đó không gây ra ngắt nội thì kết quả xử lý sẽ được lưu vào các thanh ghi của MH-C. Nếu có ngắt nội thì khối này báo cho khối xử lý



Hình 3: cấu trúc của MH-C

ngắt nhận biết loại ngắt nào để xử lý. Khối xử lý ngắt có chức năng xử lý cả ngắt nội lẫn ngắt ngoại. Khối chuyển đổi địa chỉ đổi địa chỉ ảo thành địa chỉ thật, mô phỏng chức năng quản lý bộ nhớ của vi xử lý. Bộ nhớ đệm trong MH-C mô phỏng bộ nhớ đệm truy xuất dữ liệu (DCache) của CPU. MH-C có thêm khối điều khiển bộ nhớ quyết định dữ liệu của lệnh nào có thể được lưu vào hay trả về từ bộ nhớ đệm, lệnh nào không.

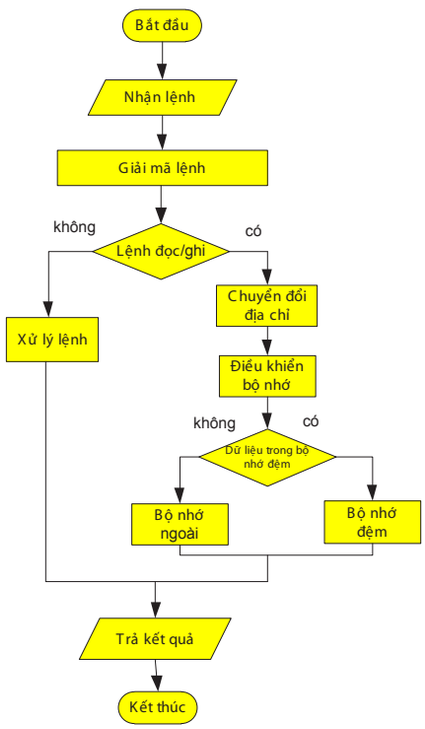
Mô hình tập lệnh

Kết quả của một lệnh được ghi vào trong các thanh ghi của MH-C để sử dụng cho các lệnh kế tiếp và trả về qua giao thức DPI [6]. Trị giá của PC được quyết định bởi lệnh và khi có các ngắt PC sẽ trỏ tới vùng địa chỉ ngắt. Đối với các lệnh tính toán (cộng, trừ, nhân, chia, dịch), PC của lệnh kế tiếp là PC hiện tại cộng cho 4. Đối với các lệnh nhảy, PC lệnh kế tiếp là vị trí mà lệnh nhảy muốn trỏ tới.

Mô hình quản lý bộ nhớ dữ liệu

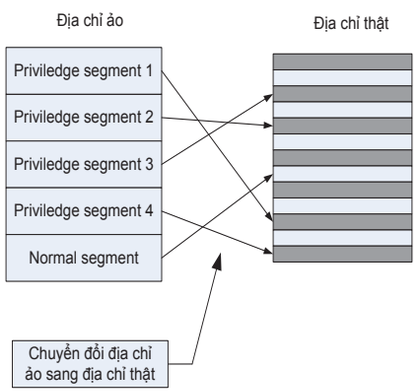
Việc xử lý dữ liệu được thể hiện qua lưu đồ giải thuật ở hình 4 với các bước sau:

- Lệnh chương trình sẽ được đưa qua khối giải mã. Nếu lệnh thuộc loại truy xuất dữ liệu thì sẽ được thực hiện theo luồng phía bên phải. Đối với các loại lệnh khác như tính toán, lệnh rẽ nhánh sẽ được xử lý theo từng chức năng của lệnh đó.



Hình 4: lưu đồ xử lý lệnh truy xuất bộ nhớ

- Địa chỉ ảo của dữ liệu cần truy xuất sẽ được chuyển đổi sang địa chỉ thật tương ứng bởi khối chuyển đổi địa chỉ. Việc quy định chuyển đổi địa chỉ như thế nào tùy thuộc vào cách thức phân vùng địa chỉ của vi xử lý và cách điều khiển của hệ điều hành thông qua các lệnh cấu hình như hình 5.



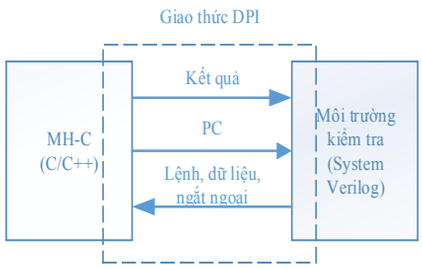
Hình 5: chuyển đổi địa chỉ ảo sang địa chỉ thật

- Tiếp theo khối điều khiển bộ nhớ sẽ xác định dữ liệu truy xuất là nằm ở bộ nhớ ngoài hay bộ nhớ đệm. Việc truy xuất này theo nguyên tắc: khi ghi, chỉ có những địa chỉ ảo được phép ghi vào vùng nhớ tạm thì dữ liệu tương ứng với địa chỉ đó mới được ghi; khi đọc, chỉ địa chỉ ảo được phép đọc dữ liệu và dữ liệu đó phải được ghi vào bộ nhớ đệm trước đó thì kết quả của lệnh đọc trả về là dữ liệu trong bộ nhớ đệm; ngược lại thì lấy từ bộ nhớ ngoài.

- Trả kết quả: với bộ nhớ đệm mô phỏng tương tự như bộ nhớ đệm trong vi xử lý đã giải quyết được vấn đề trả đúng kết quả đối với các lệnh truy xuất dữ liệu mà dữ liệu không nằm trong bộ nhớ ngoài.

Giao tiếp với môi trường kiểm tra

Phần giao tiếp giữa MH-C và môi trường kiểm tra (xây dựng bằng ngôn ngữ System Verilog) được thực hiện qua giao thức DPI trong khối điều khiển (hình 6). Lệnh đưa cho MH-C được lấy từ các bộ nhớ ngoài dựa vào PC do MH-C đưa ra sau khi chuyển từ địa chỉ ảo sang thật. Khối điều khiển đưa cho MH-C các tín hiệu ngắt ngoại để xử lý ngắt và dữ liệu từ bộ nhớ ngoài cho các lệnh đọc bộ nhớ mà dữ liệu không có trong bộ nhớ đệm.



Hình 6: giao tiếp giữa MH-C và môi trường System Verilog

Kết quả trả về được sử dụng để so sánh với kết quả tính toán của CPU. Hình 7 mô tả kết quả trả về của lệnh đọc bộ nhớ có địa chỉ 0x0040000 với phần bù là 0x16. Qua bộ chuyển đổi địa chỉ, địa chỉ thật để truy xuất bộ nhớ là 0x40000016. Vì dữ liệu đã có sẵn trong bộ nhớ đệm nên dữ liệu 0x1010ffff trả về sẽ được lấy trong bộ nhớ đệm. PC của MH-C cho lệnh kế tiếp là 0x1C.

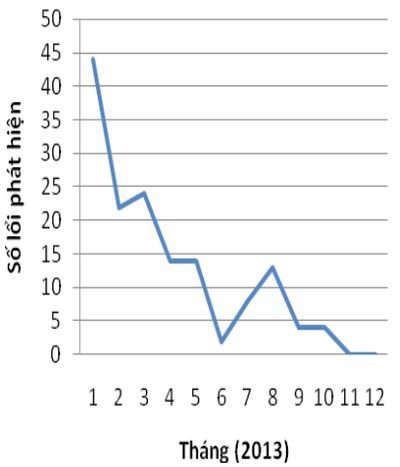
```
[R_INFO] : LDW R3, R1, 0x16
[R_INFO] : Read A Word
[R_DEBUG]:  Decode instruction
        Value of RS = 0x00400000
        Value of OFFSET = 0x16
[R_DEBUG]: Reading Memory at
Physical Address 0x40000016 in
Cache memory.
[R_DEBUG]: Data = 1010ffff,
[R_DEBUG]: Next PC Value = 1c
```

Hình 7: kết quả của MH-C

Ứng dụng và kết quả

Trong kiểm tra thiết kế vi xử lý VN32LP, MH-C đã kiểm soát tất cả các luồng lệnh, người kiểm tra chỉ tập trung vào việc tạo nhiều mẫu ngẫu nhiên càng nhiều càng tốt. Từ đó, thời gian tìm ra lỗi nhanh và hiệu quả hơn so với phương pháp kiểm tra trực tiếp. Theo số liệu thống kê về lỗi ở biểu đồ hình 8; trong tháng 1, người kiểm tra tập trung vào việc tạo mẫu ngẫu nhiên, còn MH-C hỗ trợ việc tìm ra lỗi của thiết kế cho nên lỗi được phát hiện nhanh chóng và nhiều nhất. Các tháng sau, số lượng lỗi giảm dần (ở tháng 8, 9 có sự gia tăng lỗi là do thay đổi cấu trúc vi xử lý). Dựa vào phần mềm VCS của Công ty Synopsys đánh giá mức độ kiểm tra thiết kế thì hơn 99% mã của thiết kế đã được kiểm tra (hình 9).

Ngoài ra, việc tạo mẫu ngẫu



Hình 8: biểu đồ phát hiện lỗi VN32LP

SCORE	LINE	COND	TOGGLE	FSM	BRANCH	NAME
99.52	100.00	100.00	97.60	100.00	100.00	top

Hình 9: mức độ kiểm tra thiết kế

nhiên đã làm giảm số lượng mẫu kiểm tra. Với môi trường không có hỗ trợ của mô hình chuẩn CPU như công việc kiểm tra VN1632 trước đây, người kiểm tra phải viết từng mẫu kiểm tra cho từng lệnh cho CPU và tự kiểm tra kết quả của lệnh đó. Như vậy vi xử lý có 65 lệnh thì sẽ có 65 mẫu. Ứng dụng MH-C thì chỉ cần một mẫu bao gồm 65 lệnh được tạo ra ngẫu nhiên. Kiểm tra lệnh rẽ nhánh là một công việc đầy khó khăn cho người tạo mẫu. Nếu không có MH-C, người kiểm tra phải tạo ra nhiều mẫu ứng với các trường hợp luồng lệnh khác nhau và phải kiểm soát luôn cả trình tự của luồng lệnh đó. Với MH-C và phương pháp tạo mẫu ngẫu nhiên với hỗ trợ của VMM, người kiểm tra viết ít mẫu hơn nhưng nhiều trường hợp hơn và không cần kiểm soát luồng lệnh. Kết quả là việc kiểm tra VN32LP chỉ cần 272 mẫu, ít hơn nhiều so với hơn 1.000 mẫu tạo ra trong việc kiểm tra 32 đầu tiên.

Kết luận

Bài viết đã trình bày một mô hình mô phỏng chức năng của vi xử lý có MMU và đã được ứng dụng vào việc kiểm tra vi xử lý VN32LP của ICDREC. Với việc mô phỏng chức năng MMU của vi xử lý, MH-C đã giải quyết các vấn đề truy cập bộ nhớ đệm, về chuyển đổi địa chỉ thật và ảo, từ đó trả về đúng dữ liệu cần thiết cho các lệnh truy xuất bộ nhớ. MH-C góp phần làm giảm thời gian kiểm tra thiết kế VN32LP, tạo tiền đề để xây dựng các mô hình so sánh với thiết kế vi xử lý 32 bit có độ phức tạp lớn hơn trong tương lai ✍

Tài liệu tham khảo

[1] ICDREC, "Vn1632 Risc Microprocessor User's Manual", Tư liệu nội bộ, ICDREC, Ho Chi Minh City, 2009.

[2] ICDREC, "Hướng dẫn sử dụng vi điều khiển VN32", Tư liệu nội bộ, ICDREC, Ho Chi Minh City, 2013.

[3] J.C. Chen, "Applying CRV to Microprocessor Verification", 10.2007. [Online]. Available: http://www.eetimes.com/document.asp? doc_id=1276112.

[4] K. Khan, "http://www.eetimes.com/document.asp? doc_id=1277649", 2.5.2002. [Online].

[5] Le Ha Thienaqn", in Proceedings of the 1st-VLSIs & Related Technologies (4S-2010), Ho Chi Minh City, Viet Nam, June 17-18, 2010.

[5] A. Organization, "Direct Programming Interface (DPI)," in SystemVerilog 3.1a Language Reference Manual, Napa, California, USA, 2004, pp. 347-358.