THIẾT KẾ KIẾN TRÚC TẬP LỆNH VÀ LÕI BỘ DSP 16-BIT DẤU CHẤM CỐ ĐỊNH

Lê Đức Hùng[1], Huỳnh Hữu Thuận[1], Santiago de Pablo[2]
(1) Trường Đại Học Khoa Học Tự Nhiên, ĐHQG – HCM
(2) Đại học Walladolidy, Tây Ban Nha
(Bài nhận ngày 17 tháng 8 năm 2005, hoàn chỉnh sửa chữa ngày 29 tháng 11 năm 2005)

TÓM TẮT: Hiện nay các bộ xử lý tín hiệu số (DSP) được sử dụng khá nhiều trong các ứng dụng thực tế và nghiên cứu khoa học. Việc thiết kế chế tạo các bộ DSP vượt ngoài khả năng công nghệ của ta. Tuy nhiên, trong những năm gần đây, với sự xuất hiện của các thiết bị logic khả trình như FPGA vấn đề trở nên khả thi. Mục tiêu ở đây là thiết kế cấu trúc tập lệnh và kiến trúc lõi một bộ DSP 16-bit dấu chấm cố định rồi thực hiện trên một FPGA để kiểm nghiệm [1], sử dụng ngôn ngữ lập trình Verilog HDL. Các bộ DSP 16-bit dấu chấm cố định hiện cũng đang còn rất thông dụng (tham khảo các hãng Texas Instruments, Analog Devices, National Semiconductor . . .), hơn nữa FPGA thích hợp với tính toán dấu chấm cố định. Cũng nên nhớ là phần mềm có thể biến bộ DSP dấu chấm cố định thành dấu chấm động. Dù rằng bộ DSP nhận được không thể so sánh với các bộ DSP thực thụ trên thị trường nhưng đối với chúng tôi đây là bước khởi đầu cho các thiết kế cao cấp hơn.

1. CẤU TRÚC TẬP LỆNH DSP 16-BIT DẤU CHẤM CỐ ĐỊNH

Do DSP có kích thước Bus 16-bit, nên tập lệnh được thiết kế dựa trên các lệnh 16-bit và theo cấu trúc RISC (Reduced Instruction Set Computer – máy tính có tập lệnh được rút giảm), có độ linh hoạt và dễ sử dụng.

Cấu trúc tập lệnh như sau: 4 bit đầu là mã lệnh Opcode, 12 bit sau làm Operand, trong đó chia làm 3 thanh ghi rS, rT, rD, mỗi thanh ghi 4 bit. Từng thanh ghi được tổ hợp thành 2^4 gồm 16 thanh ghi đánh số từ r0 đến r15, có thể được sử dụng làm toán hạng trong tất cả các lệnh (ngoại trừ thanh ghi r0 dùng cho các mục đích riêng):

Opcode (4-bit)		Operand (12-bit)		
		rS	rT	rD
B15	B12	B11		B0

Chú thích: rD: thanh ghi đích, rS: thanh ghi nguồn 1, rT: thanh ghi nguồn 2

Trong một lệnh, mỗi thanh ghi được mã hoá bằng 4 bit: DDDD được ký hiệu cho rD, tương tự là SSSS cho thanh ghi rS và TTTT cho thanh ghi rT. Ngoài ra hằng số K được dùng khi lệnh thao tác với một hằng số (hằng số này thường nằm ngay sau mã lệnh).

Gần như tất cả lệnh đều dùng 3 toán hạng (rD=rS op rT). Chúng có thể là các thanh ghi (ví dụ r12 = r6 + r15), hằng số (ví dụ r12 = r6 + 0.9327), giá trị rỗng (ví dụ r12 = 0+ r15, r12 = 0 + 0.9327).

Có 2 loại phép nhân: thứ nhất là dạng chuẩn hóa (rD = rS * rT) và thứ hai là dạng số nguyên (rD = rS x rT). Cả hai đều có cùng bộ nhân vật lý như nhau và tạo ra trì hoãn 1 chu kỳ lệnh như nhau, nhưng kết quả khác nhau. Dạng chuẩn hóa dùng dạng <1.15> cho cả

hai toán hạng (1 bit nguyên – bit dấu, 15 bit thập phân, khoảng giá trị từ -1.00 đến 0.999969), dạng số nguyên dùng trong các phép dịch có hai toán hạng đều <8.8>, khoảng giá trị từ -128.00 đến +127,9961.

DSP có hai thanh ghi đặc biệt trong cấu trúc là thanh ghi lệnh IR 16 bit và thanh ghi đếm chương trình PC 12 bit.

Vai trò của thanh ghi r0: Thanh ghi r0 không dùng trong các phép toán và được dùng với nhiều mục đích khác nhau:

Khi *rT = r0* thì *rD = rS op K* với K là hằng số nằm ngay sau mã lệnh **cho phép** thao tác với các toán hạng là hằng số. Ví dụ: r3=r2+K, biến lệnh cộng hai thanh ghi thành lệnh cộng thanh ghi với hằng số.

Khi *rS = r0* thì *rD = 0 op rT* cho phép thao tác với các toán hạng là hằng số Zero. Ví dụ: r3=r2+0, lệnh cộng biến thành lệnh chuyển dời dữ liệu.

Khi *rS =r0 và rT* = r0 thì *rD = 0 op K* cho phép thao tác với các toán hạng là hằng số Zero và K. Ví dụ: r2=0+K, lệnh cộng biến thành lệnh chuyển hằng số vào thanh ghi.

Khi rD = r0 thì lệnh trở thành rS op rT, phản ánh kết quả lên cờ, đây là *lệnh so sánh, và như vậy ta có nhiều dạng lệnh so sánh khác nhau, và trong tập lệnh không cần lệnh so sánh tường minh.*

Như vậy, tất cả các lệnh đều được xây dựng dựa trên 16 lệnh cơ bản, dựa trên 4 bit của mã lệnh có dạng như sau:

0000 NOP/ RET	**0100** MUL (1)	**1000** Phép gán có điều kiện	**1100** AND
0001 IN/ OUT	**0101** MUL (2)	**1001** Phép gán có điều kiện, nhưng đảo dấu	**1101** OR
0010 JP	**0110** MAC (+)	**1010** ADD	**1110** NOR/NOT
0011 GOTO/CALL	**0111** MAC (-)	**1011** SUB	**1111** XOR

1.1. Phần thực hiện lệnh NOP/RET

Lệnh NOP không làm gì cả.

Lệnh RET (rS) với rS là một trong 15 thanh ghi (từ r0 đến r15). Khi gặp lệnh này trong IR, đơn vị điều khiển sẽ nạp giá trị rS vào thanh ghi PC.

1.2. Phần thực hiện lệnh IN/OUT

– *Lệnh IN rD, Pn hoặc IN rD, rS:* khi gặp lệnh này trong IR, đơn vị điều khiển gởi ra DataAddr giá trị của rS (IN rD, rS) hoặc 6 bit của Pn (IN rD, Pn) và ghi dữ liệu từ DataIn ghi vào rD.

– *Lệnh OUT Pn, rS, Pn hoặc OUT rT, rS:* tương tự như lệnh IN nhưng gởi giá trị trong rS ra Data Out.

1.3. Phần thực hiện lệnh JP/WAIT

– *Lệnh "JP (flag) nn":* lệnh nhảy có điều kiện bị giới hạn ở các vị trí gần, bởi vì số bit giảm thiểu (8-bits) được dùng cho địa chỉ. Bất kỳ phép dịch nào từ -128 tới 127 đều có thể thực hiện được. Các cờ dùng trong lệnh này là:

- eq (0000), ne (0001): tương ứng kiểm tra bằng zero hoặc khác zero
- v (0010), nv (0011): tương ứng với kiểm tra cờ tràn trong phép tính cộng và trừ (không phải trên MAC).

- lt (0100), le (0101): tương ứng kiểm tra nhỏ hơn zero và nhỏ hơn hay bằng zero.
- gt (0110), ge (0111): tương ứng kiểm tra lớn hơn zero và lớn hơn hay bằng zero.
- Các dạng khác (1xxx) dùng để giao tiếp.

– *Lệnh WAIT (flag)*: tương tự như lệnh JP, nếu điều kiện đúng xảy ra DSP sẽ dừng thực hiện cho đến khi gặp một điều kiện bên ngoài (READY).

1.3. Phần thực hiện lệnh gán có điều kiện

– *if (flag) rD = rT:* Điều kiện thực hiện giống như lệnh JP (căn cứ vào 3 cờ ZF, SF, OF), đơn vị điều khiển (Control Unit) sẽ lấy dữ liệu từ thanh ghi rT đưa vào rD.

– *if (flag) rD = K:* Tương tự như trên, nhưng lúc này hằng số K ngay sau mã lệnh được ghi vào rD.

1.4. Nhóm lệnh gán có điều kiện, nhưng đảo dấu

if (flag) rD = -rT
if (flag) rD = -K
Thực hiện tương tự như lệnh gán có điều kiện nhưng đảo dấu.

1.5. Các lệnh nhảy tuyệt đối "GOTO nn" và "CALL nn"

Chỉ dùng 8-bit cho địa chỉ đích: điều này có nghĩa là mã thi hành chỉ có 256-words, kích thước này đủ cho các ứng dụng nhỏ của bộ xử lý. Tuy nhiên có thể mở rộng tới 4K bằng cách dùng các MODEL. Mô hình A có thể thực hiện chương trình 256-words, mô hình B là 512-words, nhảy 4. Mô hình D và E là 2Kx16 và 4Kx16, cả hai đều nhảy 8 hoặc 16. Lúc này trình biên dịch thêm các lệnh NOP cho các mô hình A, B, C, D, E tương ứng với bộ xử lý đang dùng mô hình đó.

1.6. Các lệnh gọi chương trình con "CALL (rD) nn"

Giữ địa chỉ trả về trong thanh ghi rD (tổng quát là r0), sau đó, lệnh "RET (rS)" được dùng để thay đổi thanh ghi PC và trả lại trạng thái trước đó. Vì không có Stack nên các chương trình con xếp lồng vào nhau được quản lý bởi người lập trình: nhiều thanh ghi được dùng cho chương trình hoặc dùng Stack ngoài. Lệnh GOTO không khác lệnh CALL, chỉ thay đổi giá trị thanh ghi r0.

1.7. Phần thực hiện các nhóm lệnh MUL, MAC (+), MAC (-), ADD, SUB, AND, OR, NOR, XOR

Được thực hiện nhờ ALU. Có ba bộ phận chuyên biệt gồm MAC (có Opcode 01xx) dùng cho các phép nhân (có hoặc không có tích lũy), ARITHMETIC (có Opcode 10xx) dùng cho các lệnh ADD/SUB (có kèm theo điều kiện hoặc không), và LOGIC dùng cho các lệnh Logic (có Opcode 11xx). Mỗi khối có một vùng đệm ngõ ra tùy thuộc vào Opcode.

2. KIẾN TRÚC LÕI DSP 16-BIT DẤU CHẤM CỐ ĐỊNH

2.1 Đặc điểm

– Cấu trúc Havard, bao gồm hai kiến trúc BUS, một dùng cho mã lệnh (gồm Code Addr, Code Data) và một dùng cho dữ liệu (Data Addr, Data In, Data Out).

– Bus dữ liệu: 16 bit, Bus địa chỉ: 12 bit.

– Tần số hoạt động: 50 Mhz, điện thế vào ra 3.3V. Do tần số của DSP là 50MHz nên chu kỳ xung Clock là 20ns (1/50MHz) và một lệnh thực hiện trong 4 chu kỳ xung Clock nên chu kỳ lệnh là 20ns x 4 = 80ns.

– Có 16 thanh ghi gồm 15 thanh ghi (r1-r15), 1 thanh ghi r0 dành riêng, và các thanh ghi đặc biệt như PC, IR ...

– Bộ nhân/tích lũy MAC 16/24 bit dấu chấm cố định có dấu và dạng chuẩn hóa trong 1 chu kỳ lệnh. Bộ ALU 24-bit thực hiện các phép toán logic, số học, điều kiện. Có các cờ Zero, cờ dấu, cờ tràn.

– Chế độ định vị địa chỉ: tức thời (toán hạng đích là thanh ghi hay ô nhớ, toán hạng nguồn là hằng số) và thanh ghi (dùng các thanh ghi trong bộ xử lý để chứa dữ liệu cần thao tác).

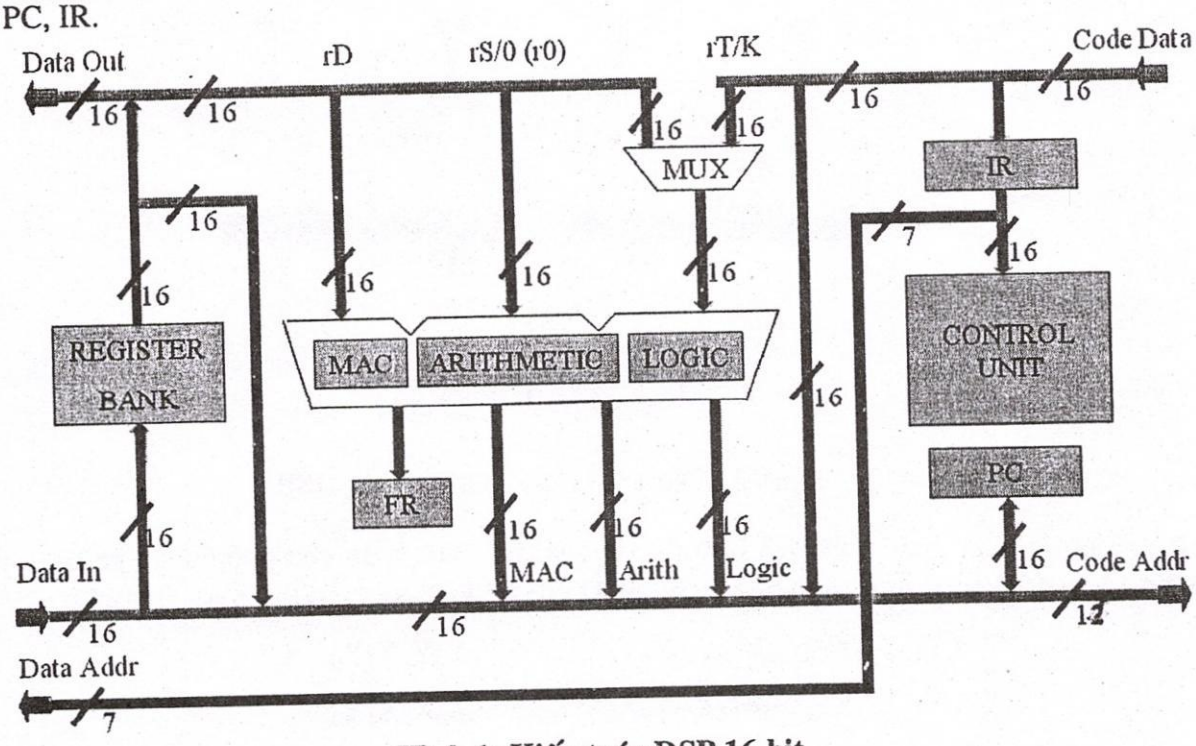
– Bộ nhớ mã lệnh thi hành 256x16 bits-4Kx16 bits.

– Giao tiếp bộ nhớ và giao tiếp vào ra: 128 cổng I/O (2^7).

2.2. Kiến trúc bên trong DSP 16-bit

Sơ đồ khối của DSP (hình 1) tương ứng với tập lệnh cần thiết kế:

Thiết kế bên trong DSP dựa trên cấu trúc gồm hai bus, một bus dùng cho các toán hạng và một bus dành cho kết quả. Những thành phần chính là băng thanh ghi gồm 16 thanh ghi, ALU (gồm MAC, Arithmetic, Logic), MAC, và các thanh ghi đặc biệt như PC, IR.

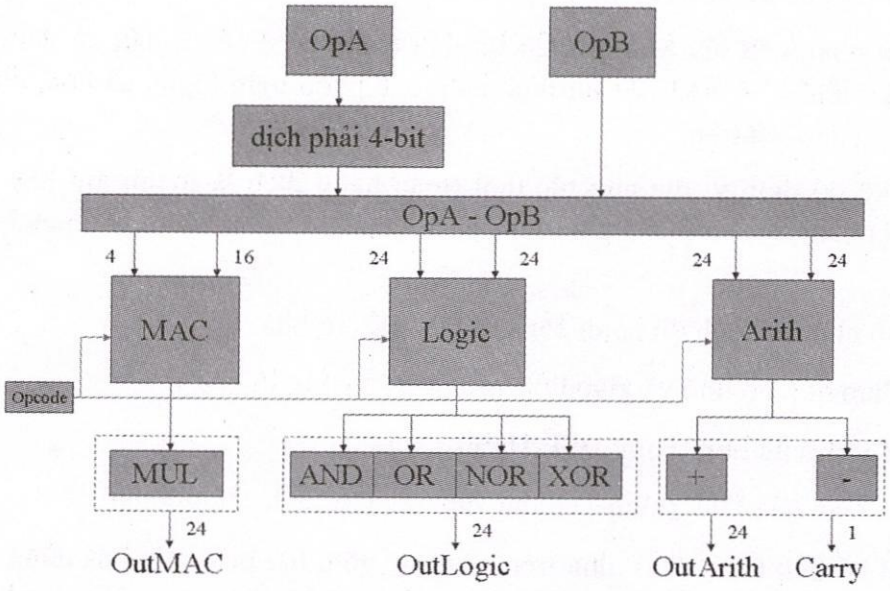


Hình 1: Kiến trúc DSP 16-bit

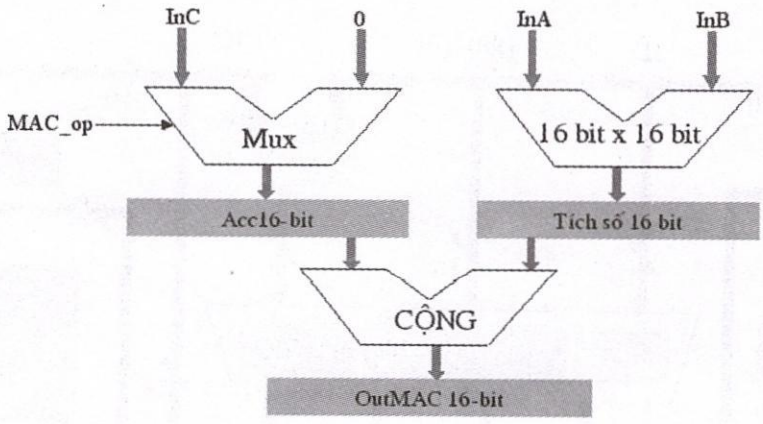
– Băng thanh ghi (r0-r15) được xem như là một bộ nhớ SRAM kích thước 16x24 với ghi đồng bộ và đọc không đồng bộ. FPGA các hãng Altera, Xilinx có hỗ trợ nguồn tài nguyên này.

– Đơn vị số học và Logic (ALU) (hình 2) thực hiện hầu như tất cả các phép toán, có các thành phần chuyên dụng và ba bộ đệm ngõ ra, từng bộ đệm tương ứng cho từng cột của tập lệnh.

– Tất cả các lệnh được thực hiện trong một chu kỳ lệnh (kể cả lệnh MAC) gồm 4 Clock. Trong thiết kế phần cứng, để đạt được điều này và tăng tốc độ xử lý tác giả đưa vào một số kiến trúc song song bên trong thiết kế sao cho đảm bảo các phép toán thực hiện trong 4 Clock và sử dụng ít số tài nguyên bên trong FPGA



Hình 2 : Cấu trúc ALU trong DSP



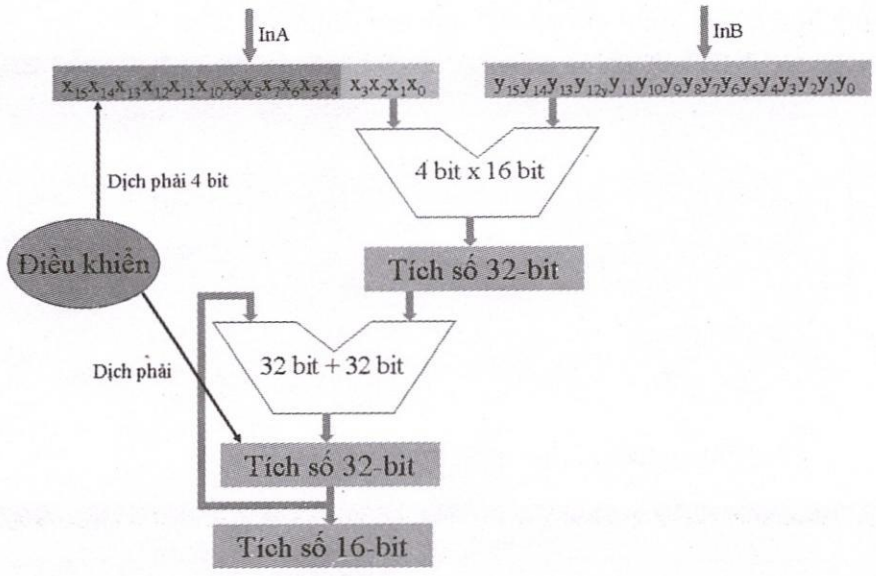
Hình 3 : Thực hiện bộ MAC trong DSP

Hình 3 là bộ MAC (nhân và tích lũy). Ví dụ quá trình tính toán cho phép nhân 16 bit x 16 bit (không dấu) được thực hiện bên trong ALU với 4 Clock như sau (hình 4):

$$Y = Y_{15}Y_{14}Y_{13}Y_{12}Y_{11}Y_{10}Y_9Y_8Y_7Y_6Y_5Y_4Y_3Y_2Y_1Y_0.$$

$$X = X_{15}X_{14}X_{13}X_{12}X_{11}X_{10}X_9X_8X_7X_6X_5X_4X_3X_2X_1X_0.$$

– Sau mỗi Clock, giá trị X được dịch 4 bit, 4 bit thấp của X sẽ được đưa vào bộ nhân 4x16. Bộ nhân 4x16 sẽ được thực hiện song song tạo ra kết quả 32 – bit (từ kết quả 20-bit) với thời gian trễ của 1 cổng AND, 2 bộ cộng (sử dụng bộ cộng tính toán trước cờ nhớ Carry Look Ahead) giảm tối đa thời giạn tính toán. Kết quả 32-bit này sẽ được cộng dồn sau đó dịch chuyển sang phải bốn bit. Do đó phép nhân 16x16 sẽ được thực hiện trong 4 Clock.



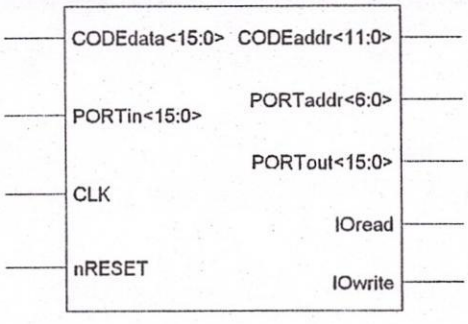
Hình 4 : Thực hiện bộ MUL 16 bit x 16 bit trong DSP

3. THỰC HIỆN THIẾT KẾ DSP TRÊN FPGA

Sử dụng chương trình Quartus II 3.0 của hãng Altera và Kit FPGA Stratix EP1S25 của hãng Parallax [4], [5]. Toàn bộ chương trình DSP 16-bit được viết bằng ngôn ngữ lập trình Verilog HDL và cài đặt trên FPGA Stratix EP1S25 của hãng ALTERA

Cấu trúc chương trình DSP có cấu trúc phân cấp theo ba khối chức năng: DSP -> ALU -> MUL

3.1. Mô hình DSP sau khi tổng hợp (hình 5)

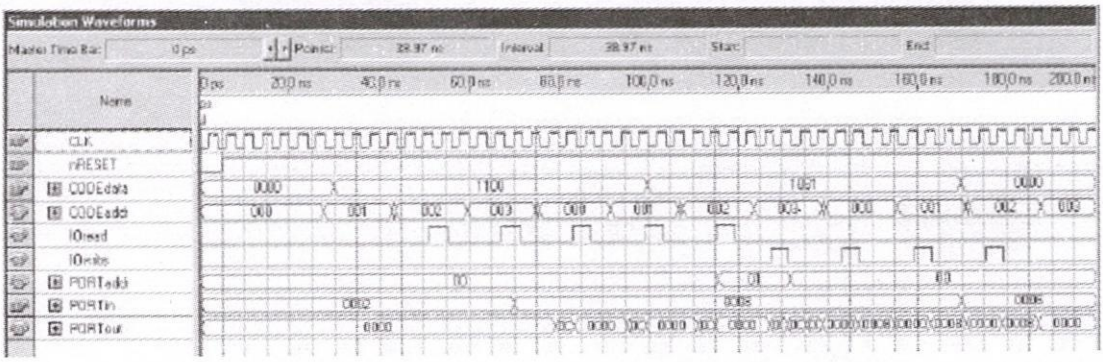


Hình 5: Mô hình DSP sau khi tổng hợp

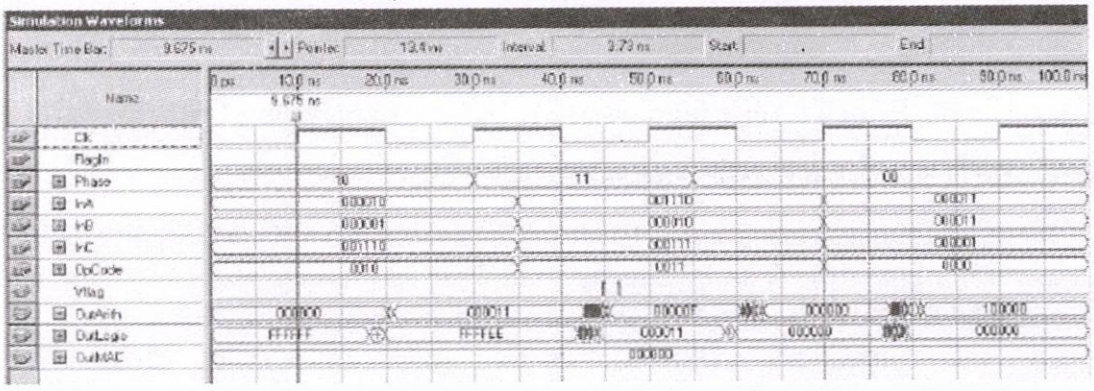
Mô tả chân tín hiệu:

- CODEdata[15..0] (ngõ vào): mã lệnh DSP 16-bit
- CODEaddr[11..0] (ngõ ra): 12-bit địa chỉ, 2^{12} = 4K bộ nhớ
- PORTaddr[6..0] (ngõ ra): 7-bit địa chỉ, tương ứng 2^7 = 128 cổng bộ nhớ vào/ra
- PORTIN[15..0] (ngõ vào): Giá trị ngõ vào 16-bit
- PORTOUT[15..0] (ngõ ra): Giá trị ngõ ra 16-bit
- CLK (ngõ vào): Ngõ vào xung Clock cho DSP
- nRESET (ngõ vào): tín hiệu Reset DSP, kích hoạt mức 0
- IOREAD (ngõ ra): Tín hiệu ngõ ra chỉ thị lệnh đọc dữ liệu, tác động ở mức 1
- IOWRITE (ngõ ra): Tín hiệu ngõ ra chỉ thị lệnh ghi dữ liệu, tác động ở mức 1

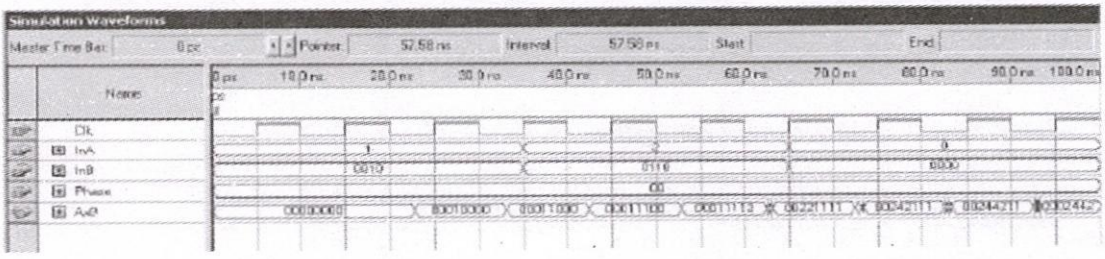
Sau khi tổng hợp kết quả mô phỏng của như sau (hình 6, 7, 8):



Hình 6: Mô phỏng hoạt động của DSP

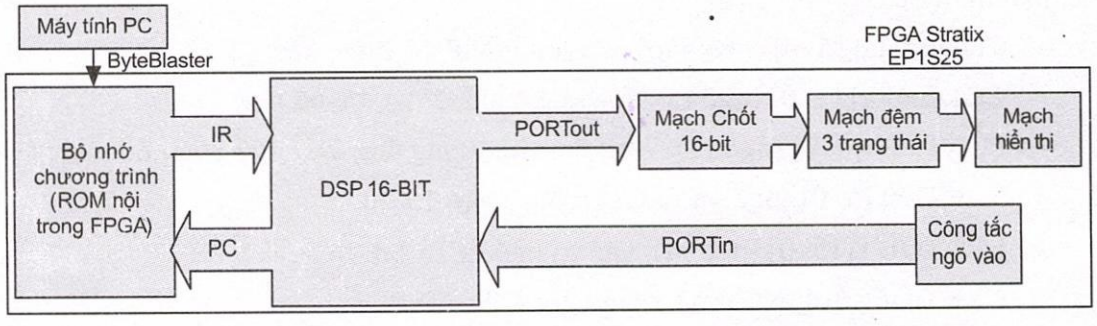


Hình 7: Mô phỏng bộ xử lý toán học ALU



Hình 8: Mô phỏng bộ nhân MUL

3.2 Sơ đồ cấu trúc của DSP trên FPGA, kết nối với máy tính (hình 9)



Hình 9: Sơ đồ cấu trúc DSP trên FPGA

4. MỘT SỐ CHƯƠNG TRÌNH KIỂM NGHIỆM

Sau khi đã thiết kế bộ DSP và lập trình FPGA để thực thi sơ đồ logic của bộ DSP chúng tôi thực hiện nhiều chương trình kiểm nghiệm như: cộng 2 số, nhân 2 số, cộng và tích lũy (MAC), lọc, tính toán công suất $P = UI\cos\phi$ . . . Tất cả đều hoạt động như dự kiến.

5. KẾT LUẬN

Ý đồ của công trình là thiết kế kiến trúc một bộ DSP 16-bit dấu chấm cố định độc lập với cấu trúc của FPGA rồi cài kiến trúc bộ DSP lên FPGA như một môi trường logic. Cách này không tận dụng tối đa và tối ưu tài nguyên của FPGA nên bộ DSP nhận được có nhiều hạn chế. Còn nếu dựa vào tài nguyên cụ thể của một FPGA để biến nó thành một bộ DSP thì bộ DSP sẽ cao cấp hơn nhiều, nhưng công trình không đi theo hướng này vì mục đích ở đây là thiết kế bộ DSP theo các nguyên lý cơ bản còn FPGA chỉ là môi trường để kiểm nghiệm (thay vì tiến đến thiết kế IC và đưa đi chế tạo). Trên sự thành công này chúng tôi có cơ sở để nâng thiết kế lên cấp cao hơn.

DESIGN OF INSTRUCTION SET AND CORE ARCHITECTURE FOR A FIXED-POINT 16-BIT DSP

Le Duc Hung[1], Huynh Huu Thuan[1], Santiago de Pablo[2]
(1) University of Natural Sciences, Viet Nam National University Ho Chi Minh city
(2) Walladolidy University, Spain

ABSTRACT: Today, Digital Signal Processors (DSP) are used widely in pratical applications and scientific research. Designing and manufacturing DSPs are beyond our technological level. However, with the appearance of programmable logic devices, such work becomes possible. The project is to design the instruction set and core architecture of a 16-bit DSP and then implemented on a FPGA as a logic environment [1], using Verilog HDL programming language. The DSP in this paper cannot compare with real DSPs on the market, but for us this is a step for more advanced designs.

TÀI LIỆU THAM KHẢO

[1]. *Chương trình hợp tác với Santiago de Pablo*, Khoa Kỹ thuật Điện tử, Đại học Valladolid, Tây Ban Nha, http://www.dte.eis.uva.es/OpenProjects/OpenDSP.

[2]. Jeremy Barsten, Jeremy Stickwell, *Digital Signal Prossesor using FPGA and VLSI technology*, http://cegt201.bradley.edu/projects/proj2003/dsppproj/Memo.htm

[3]. B Venkataramani, M Bhaskar, *Digital Signal Processors,* McGraw-Hill, 2002.

[4]. *Stratix Programmable Logic Device Family Datasheet*, Altera Corp, 2002.

[5]. KIT Smartpack Stratix EP1S25, Parallax, http://www.parallax.com.