Synthesizable ARM9 호환 CPU의 설계

서 보 익, 배 영 돈, 박 인 철
한국과학기술원 전자전산학과 전기 및 전자공학 전공
전화 : 042-869-8023 / 팩스 : 042-869-4410

Design of a Synthesizable ARM9 Compatible CPU

Bo-Ik Seo, Young-Don Bae, In-Cheol Park
Dept. of Electrical Engineering and Computer Science, KAIST
E-mail : biseo@ics.kaist.ac.kr

Abstract

In this paper, we describes the design of a CPU compatible with ARM9 processor. The CPU is fully synthesizable and described in Verilog-XL. Starting from the synthesizable ARM7 compatible CPU we developed earlier, we modified its pipeline to five stages. For this we first partition the behaviors of each instruction into five stage pipeline operations. Then we designed the controller and the datapath considering the forwarding or interlock schemes. Finally the compatibility of the designed CPU is verified by comparing the results of every instruction executed in test programs with those of the reference simulator developed for the ARM7 compatible CPU.

I. 서론

ARM7은 영국 Advanced RISC Machine 사에서 mobile phone과 같은 application에 적용하기 위해 개발한 저전력 embedded CPU로서 널리 쓰이고 있다[1]. 최근 application이 복잡해지면서 보다 고성능의 CPU를 필요로 함에 따라 명령어 수준의 호환성을 유지하면서 pipeline 구조를 개선한 ARM9 CPU가 발표되었다[4,5]. 그러나 이 ARM9 CPU는 반도체 제작 공정별로 제공되므로 공정이 바뀔 때마다 막대한 비용의 로열티를 지불하고 사용해야만 한다. 따라서 synthesizable하면서도 이 CPU와 호환되는 새로운 CPU의 개발이 필요하다.

본 논문에서는 이러한 맥락에서 기존에 개발된 synthesizable한 ARM7 CPU core [6]를 개량하여 5 stage pipeline의 구조를 갖는 ARM9 호환 CPU를 개발하였다. 이를 위해 pipeline 상에서 각 명령어의 동작을 정의하고 이를 토대로 controller와 datapath의 구조를 정하였다. 이렇게 CPU의 구조를 설계한 후 합성이 가능하도록 verilog로 기술하고 그 시뮬레이션으로 동작과 호환성을 검증하였다.

II. 명령어의 동작

2.1 파이프라인 동작

ARM9의 명령어 구조는 ARM architecture V4이며 이것은 ARM7과 동일하다[3]. 그러나 파이프라인 구조는 ARM7이 그림1과 같이 3단계인데 반해 ARM9은 그림2에서 보듯이 fetch, decode, execute, memory, writeback의 5단계의 pipeline 구조로 되어 있다. 따라서 우리는 ARM7에서 쓰이는 명령어들의 동작을 5단계의 파이프라인의 구조에 맞추어 다시 정의하였다.

또한 ARM9과의 호환성을 실현하기 위하여, 명령어가 실행되는 clock cycle 수를 ARM 사에서 발행한 ARM9TDMI Technical Reference Manual (Rev 1)[4]에 수록되어 있는 Instruction cycle time과 같도록 하였다. 그림3은 예로서 load multiple data(LDM) 명령어의 파이프라인 동작을 보인 것이다.

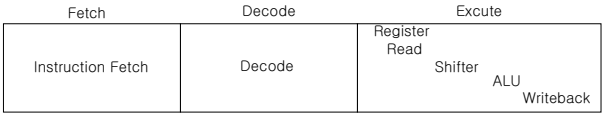

그림1. ARM7의 명령어 파이프라인 구조

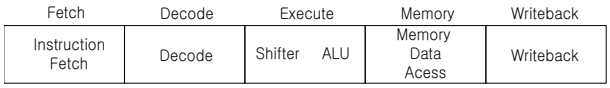

그림2. ARM9의 명령어 파이프라인 구조

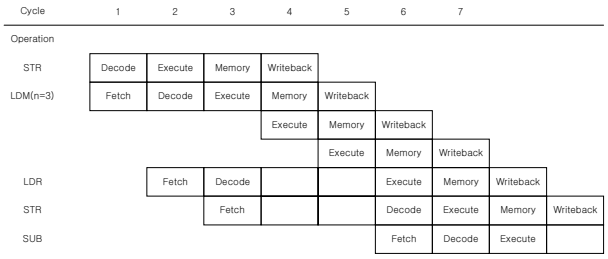

그림3. 명령어 파이프라인의 예

2.2 Forwarding 및 interlock 처리

(1) Forwarding

Forwarding 경로는 그림6에 나타나 있듯이, execute stage에서 decode stage로 가는 경로, memory stage에서 decode stage로 가는 경로, 그리고 writeback stage에서 decode stage로 가는 경로의 세가지를 구현하였다. 컨트롤 블록은 연속해서 실행되는 명령어 사이의 데이터 의존성을 파악하여 의존성이 있는 경우, 레지스터 뱅크에 데이터를 저장하지 않고 forwarding 경로를 데이터를 바로 전달하게 된다. 그림4는 예로서 n번째 명령어의 연산 결과를 n+1 번째 명령어가 operand로 읽는 경우의 파이프라인 동작을 보인 것이다.

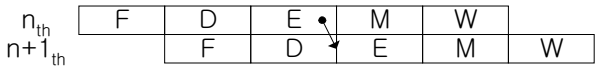

그림4. Forwarding하는 경우의 예

(2) Interlock

앞에서 서술한 forwarding 경로만으로는 명령어 사이의 데이터 의존성으로 인하여 발생하는 문제를 완전히 해결할 수 없는 경우가 발생한다. 이 경우에는 forwarding 경로를 통하여 문제를 해결할 수 있을 때까지 나중에 오는 명령어의 진행을 멈추어 interlock을 발생하도록 하였다. 이것은 주로 load 명령어의 결과값을 바로 다음 명령어가 쓰는 경우 발생한다. 그림5는 예로서 n 번째 명령어가 memory에서 load한 값을 n+1 번째 명령어가 operand로 읽는 경우에 1 cycle의 interlock을 발생시키고 forwarding하는 파이프라인 동작을 보인 것이다.

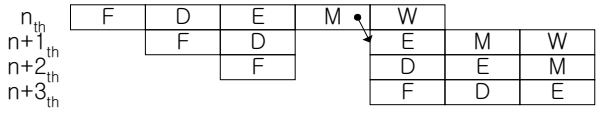

그림5. Interlock이 발생하는 경우의 예

III. 프로세서 구조

3.1 데이터패스 구조

데이터패스의 구조는 그림6과 같다. 데이터패스는 32비트 레지스터 뱅크, 32비트 배럴 쉬프터, ALU, 32×8 곱셈기, memory access를 위한 address 계산 블록, store할 데이터 혹은 load한 데이터의 재정렬을 위한 데이터 aligner, PC, 그리고 파이프라인 단계를 구분하기 위한 레지스터 등으로 구성되어 있다.

또한 forwarding을 구현하기 위하여 execute, memory, writeback stage의 출력값을 decode stage로 입력할 수 있는 fowarding 패스와 이러한 패스들 중 하나를 선택하여 execute stage의 연산에 사용하기 위한 멀티플렉서가 있다.

Decode stage와 execute stage 사이에 있는 파이프라인 레지스터는 특별히 두 개가 있는데, 이것은 그림8과 같은 forwarding을 해야 할 경우 forwarding 할 데이터를 별도로 저장해 놓기 위한 것이다.

3.2 컨트롤러 구조

컨트롤 블록은 multi-cycle 동작에 필요한 FSM과 데이터패스와 파이프라인을 제어하는 신호를 생성하기 위한 컨트롤 신호 생성기, 그리고 forwarding과 interlock을 판단하기 위한 블록 등으로 구성되어 있다.

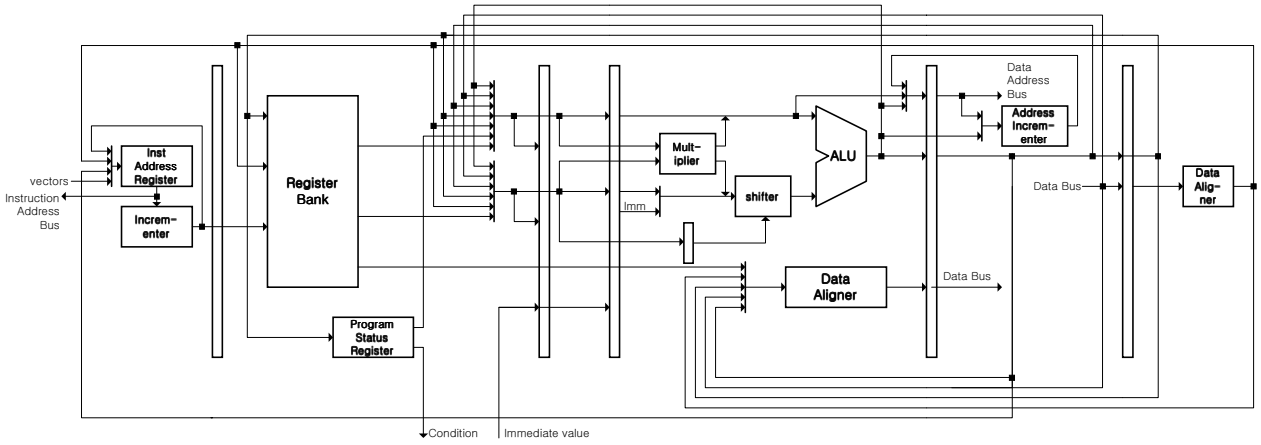

그림 6. 데이터패스 구조

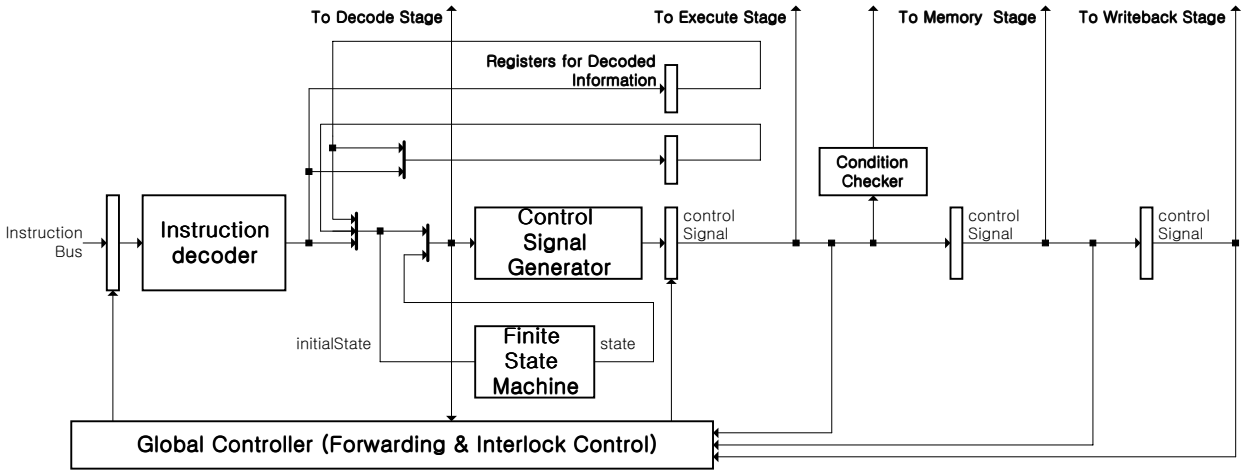

그림 7. 컨트롤러 구조

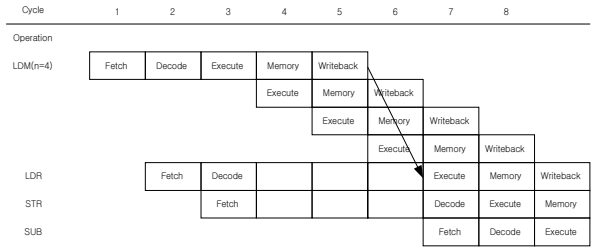

그림 8. 특별한 Forwarding의 예

Ⅳ. 설계 및 검증

전체 설계는 기본적으로 Verilog를 사용하여 기술하였으며, 컨트롤러의 instruction decoder와 control signal generator 중 일부는 PLA 형식으로 기술하고 espresso를 사용하여 optimize하였다. 모든 블록은 single falling edge clock 구조로 설계하였다.

본 논문에서는 ARM Software Develoment Toolkit에서 제공하는 ARM 프로세서 시뮬레이터 Armulator를 reference 모델로 사용하여 명령어 단위의 호환성과 Verilog 시뮬레이션을 동시에 수행할 수 있도록 하였다[2].

별개의 프로그램인 Armulator와 Verilog-XL의 결과를 비교하기 위하여 IPC (Inter-process Communication)를 사용하였다[7]. IPC는 UNIX에서 기본적으로 지원한다. 그리고 Verilog-XL의 인터페이스는 PLI (Program Language Interface)를 사용하였다. 호환성 검증을 위한 시뮬레이션 환경은 그림 9와 같다.

각 명령어 단위로 Verilog-XL 시뮬레이션 결과(레지스터, 상태 레지스터, 프로그램 카운터 값, 명령어 코드)를 PLI(Program Language Interface)루틴을 통하여 전달하고 동일한 명령어에 대하여 Armulator에서 수행한 결과를 IPC 메시지 큐를 통하여 비교하였다.

호환성의 검증은 RT(Register Transfer) 레벨에서

수행하였다. 시뮬레이션 입력으로는 다양한 어셈블리 코드를 어셈블러를 통하여 만든 바이너리 코드를 사용하였으며 올바로 수행이 됨을 확인하였다.

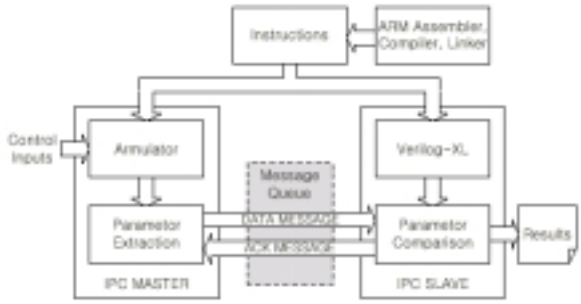

그림 9. 명령어 단위의 호환성 검증

V. 결론

저전력 embedded CPU를 요구하는 분야는 계속하여 성장하고 있으므로, 이러한 분야에서 쓰일 수 있는 CPU를 개발하는 일은 중요하다고 할 수 있을 것이다.

본 논문에서는 ARM9 호환 프로세서를 개발하기 위하여 ARM7의 명령어의 동작과 pipeline 구조의 동작을 재정의 하였다. 이를 바탕으로 controller와 datapath의 구조를 제시하고 설계하여 합성 가능한 형태로 verilog로 기술하였다. 설계된 CPU의 동작과 호환성은 IPC를 사용하여 Armulator와 비교함으로써 검증하였다.

본 논문에서 설계된 프로세서는 synthesizable하므로 휴대용 단말기 등 저전력 embedded core가 필요한 ASIC의 설계에 공정에 관계없이 쓰일 수 있을 것이라 사료된다.

참고문헌

[1] Advanced RISC Machines, ARM7TDMI Datasheet, August 1995.

[2] Advanced RISC Machines, ARM Software Development Toolkit, Version 2.50, 1998

[3] D.V Jaggar, Advanced RISC Machines Architectural Reference Manual, Prentice Hall, London, 1996

[4] Advanced RISC Macines, ARM9TDMI Techinical Reference Manual, Rev 1, 1998

[5] http://www.arm.com

[6] 배영돈, 서보익, 이용석, 박인철, “ARM7 호환 32비트 RISC 프로세서의 설계 및 검증”, 대한전자공학회, 1999년

[7] W. R. Stevens, Advanced Programming in the UNIX Environment, Addison Wesley, 1992