32 비트 DSP RISC 프로세서를 위한 ALU 설계 및 테스트

최대봉, 문병인, 유동열 , 이태영, 이용석
연세대학교 전자공학과 VLSI & CAD 연구실
서울시 서대문구 신촌동 134 번지
cdai@dubiki.yonsei.ac.kr

ALU Design & Test for 32-bit DSP RISC Processors

Daibong Choi, Byungin Moon, Dongryul Yu, Taeyoung Lee, Yongsurk Lee
VLSI & CAD Lab., Elec. Eng., Yonsei University
134, Shinchon-dong, Seodaemun-gu, Seoul
cdai@dubiki.yonsei.ac.kr

Abstract

We designed an ALU(Arithmetic Logic Unit) with BIST(Built-In Self Test), which is suitable for 32-bit DSP RISC processors. We minimized the area of this ALU by allowing different operations to share several hardware blocks. Moreover, we applied DFT(Design for Testability) to ALU and offered BIST(Built-In Self-Test) function. BIST is composed of pattern generation and response analysis. We used the reseeding method and testability design for the high fault coverage. These techniques reduce the test length. Chip's reliability is improved by testing and the cost of testing system can be reduced.

1. 서 론

최근에 멀티미디어와 디지털 신호 처리는 내장 시스템을 응용한 분야에서 중요한 시장으로 부각되고 있다. 또한 전화통신과 멀티미디어 분야에서 빠르고 쉽게 응용 할 수 있는 새로운 내장 프로세서가 요구되고 있다. 이러한 내장 프로세서는 하드웨어가 비교적 간단한 RISC 형태를 가지고 DSP 연산에 알맞은 MAC(Multiplier-Accumulator) 등과 같은 하드웨어를 내장하게 되었다. 또한, 성능이 우수하고 작은 면적과 효율면에서 우수성을 가지고 있어야 한다. 여러 가지 기능과 하드웨어를 부가함에 따라 회로의 집적이 증가하고 복잡하게 되어 테스팅이 더욱 어려워지고 있다. 그 결과 생산비용에 있어서 테스팅 비용이 차지하는 비율이 급속하게 증가하였다.

본 연구에서는 이러한 멀티미디어와 디지털 신호처리를 하는 내장용 RISC 프로세서에 적합한 ALU를 설계하였다. 그리고, 테스팅을 위한 자체 내장 테스트 회로를 부가하였다. 또한, ALU 면적과 지연 증가가 최소가 되도록 회로를 설계하고 fault coverage 를 높이기 위해서 테스팅이 용이한 설계를 하고 패턴 발생기에 reseeding 방법을 적용하였다.

2. DSP RISC 프로세서의 구조

DSP RISC 프로세서는 디지털 신호 처리에서 많은 양의 데이터를 처리하거나 FFT 같은 특수한 알고리즘을 빠르게 처리하기 위해 DSP 기능과 일반적인 범용 RISC 마이크로프로세서의 특징을 가지고 있다.

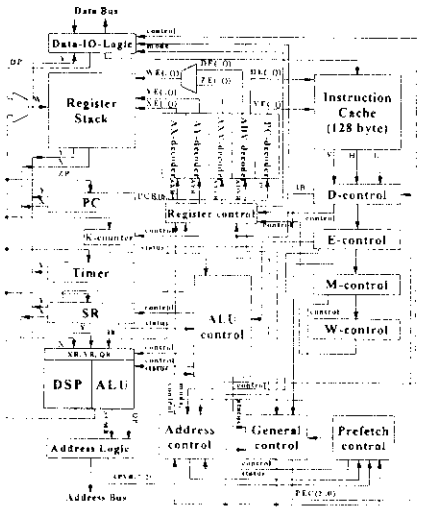

그림 1. DSP/RISC 내장 프로세서 전체 구조

이 프로세서가 가지고 있는 특징을 살펴보면 다음

*본 연구는 1998 년도 한국 학술진흥재단 대학부설 연구소과제 연구비에 의하여 연구되었음.

과 같다. 파이프라인 즉, fetch, decode, execution, memory, write-back 으로 구성된 5 단 형태를 가지고 있다. 또한 데이터 포워딩과 지연 분기, 조건분기 등을 지원하고 명령어 캐쉬를 가지고 있다.

일반적으로 RISC 프로세서는 고정길이를 갖지만 본 논문의 RISC 프로세서는 16, 32, 48 비트의 3 가지 길이 명령어를 지원하여 페치할 때 필요한 메모리 대역폭이 절약될 수 있고 최적화된 코드를 사용할 수 있게 한다. 그리고 대부분의 명령어를 1 사이클에 처리한다.

DSP RISC 프로세서 구조의 구성은 그림 1 에서 보듯이 특수 레지스터, PC, ALU, DSP 유닛, 명령어 해석기, 명령어 캐쉬, 버스 제어기 등으로 이루어져 있다.

3. 회로의 구성과 구현

본 연구에서 설계하는 ALU 는 기본적으로 32 비트 연산을 수행하며 5 단 파이프라인 중에서 execution 단계에 속한다. ALU 에서 필요한 기능은 덧셈, 뺄셈, 나눗셈과 같은 산술연산, AND, XOR 과 같은 논리연산, 쉬프트, leading zero 검사등 많은 기능이 필요하다. 기능별로 블록을 사용하지 않고 몇 개의 기능 블록만을 만들고 동작이 이 기능 블록들을 공유하도록 하였다. 그래서 ALU 의 컨트롤러가 복잡하지만 많은 하드웨어를 줄일 수 있었다. 그리고, 자체 내장 테스트 회로를 가지고 있다. 내부에서 패턴을 발생하여 회로의 고장 유무를 검사한다.

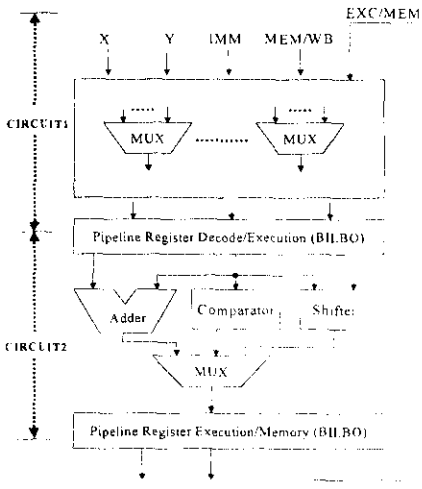

그림 2. ALU 전체 구조

본 ALU 의 구체적인 구성 요소를 나열하면 다음과 같다. 덧셈기, 쉬프트, 비교기, 32 비트 레지스터, 제어기, BIST 제어기 등이 있다. 그림 2 에서 이러한 구성요소의 전체적인 구조를 나타냈다. 이 ALU 가 지원하는 연산은 산술연산, 논리연산, 나눗셈 등의 연산을 할 수 있다. 여기에서 덧셈기는 면적의 오버헤드가 있지만 속도가 빠른 CLA(Carry-Look-Ahead) 구조를 사용하였다. 논리연산을 위한 논리회로를 따로 구현하지 않고 덧셈기 회로 일부를 사용하여 AND, OR 그리고, XOR 연산을 구현하였다. 이렇게 함으로써 지연시간이 더 생겼으나 면적은 논리 연산을 위한 회로를 따로 구현하는 것 보다 16%를 줄일 수 있었다.

쉬프터는 배럴 쉬프트 기능과 동일하며 비록 제어기가 복잡하지만 면적을 줄이도록 설계되었다. 쉬프터의 구성은 2 단계로 구성되어 있다. 2 개의 32 비트 입력을 필요로 한다. 그래서 32 비트 단일 워드(single word)뿐만 아니라 이중 워드(double word)도 쉬프트할 수 있고 회전도 가능하다. 그림 3 은 쉬프터 구조이다.

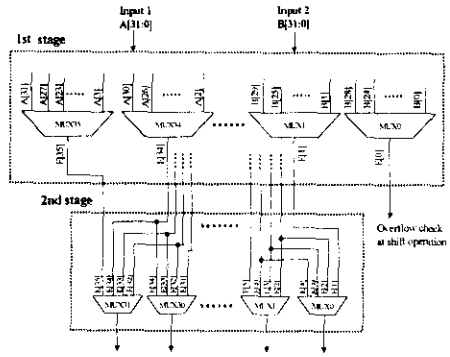

그림 3. 쉬프터 구조

1 단계는 쉬프터에서 4 비트씩 좌로 쉬프트 하며 2 단계는 1 비트씩 좌로 쉬프트 한다. 여기서 좌 방향으로만 쉬프트 하는데 이것은 우로 n 비트 쉬프트하는 것은 좌로 32 - n 비트 쉬프트하는 것과 동일하게 생각할 수 있다. 그리고 단일 워드를 우로 쉬프트할 때에는 좌측 방향과는 달리 주의가 필요하다. 이유는 부호를 고려해야 하기 때문이다. 이중 워드에서 좌/우 방향으로 쉬프트할 때에는 데이터를 로드할 때 유의하여야 한다. 그림 4 는 이러한 단일 워드, 이중 워드 쉬프트시에 로드하는 방법을 나타낸 것이다.

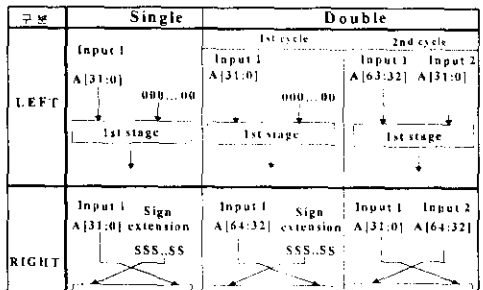

그림 4. 쉬프트시에 로드 방법

다음으로, 비교기에서는 leading zero 검사와 IEEE 부동 소수점 표준 754 에서의 NaN(Not a Number)를 검사하기 위한 회로, 레지스터 값이 모두 '0' 혹은 '1'인지를 검사하기 위한 회로와 32 비트 중에서 어느 바이트가 '0'인지를 검사하는 회로를 가지고 있다.

디지털 신호처리에서 leading zero 연산이 많이 필요하지 않기 때문에 본 연구의 ALU는 고정 소수점을 지원하는 프로세서에 적합하도록 면적을 줄이는 방향으로 설계 하였다. 그래서 leading zero 검사를 위해서 2 사이클이 필요하다. 첫번째 사이클에서는 연산자를 8 개의 4 비트 단위로 끊어 어떤 부분에 먼저 0 이 아닌 1 이 나오는지를 검사한다. 다음으로 두번째 사이클에서는 첫번째 단계에서 구한 8 개의 부분 중 일부를 그룹 쉬프터를 사용하여 쉬프트 한다. 그리고, 쉬프트 한 결과에서 상위 3 비트만을 이용하여 쉽게 1 이 나오는 위치를 알 수 있다. 그림 5 에 leading zero 를 검사하는 블록이 나와 있다.

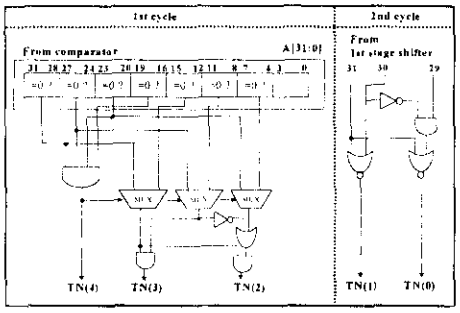

그림 5. Leading zero 검사 회로

나눗셈 연산은 ALU 내부에 나눗셈을 위한 별도의 블록을 가지고 있지 않다. 연산은 non-restoring 방식으로 덧셈/뺄셈과 쉬프트를 반복하여 수행한다. 따라서 피제수가 이중 워드인 64 비트이고 제수가 워드인 32 비트 나눗셈인 경우 수행 사이클은 몫과 나머지 보정 단계를 포함하여 36 사이클이 걸린다.

자체적으로 ALU 회로를 검사하기 위해서 BIST 를 내장 하였다. 이렇게 함으로써 테스팅 비용과 검사 시간을 절약할 수 있고 칩에 대한 신뢰성을 높일 수 있다. 먼저 ALU 회로에 이러한 테스팅을 위한 회로를 부가하기 위해서 다음과 같은 사항을 고려하였다. 본 연구의 ALU 유닛 구성이 대부분 조합회로와 4 개의 파이프라인 레지스터로 구성되어 있다. 또한, ALU 는 파이프라인 단계 중 decode, execution, memory 3 단계에 걸쳐 있다. 그리고, 일반적인 방식으로 ALU 제어기와 ALU 를 따로 테스팅할 때에는 관찰점이 증가하고 이에 따른 패턴 생성기가 추가되고 정상 동작시에 지연시간이 증가하는 부담을 가중시키게 되며 테스팅시에는 각 블록별로 따로 테스트를 해야 하므로 상호 관련성이 높은 블록에 대해서는 테스팅이 힘들게 된다. 먼저 이러한 점들을 고려하여 상호 작용하는 구조에 적합한 집적화된 테스팅 구조를 택하였다. 그리고 파이프라인에 적합한 BIST 의 일종인 BILBO(Built-in Logic Block Observer) 구조를 적용하였다. 그래서 파이프라인을 고려하여 ALU 를 테스팅에 적합하도록 분할하면 파이프라인 레지스터들을 중심으로 CIRCUIT1 과 CIRCUIT2 로 분리하였다. CIRCUIT1 은 파이프라인 decode 단계에 속하며 CIRCUIT2 는 파이프라인 execution 단계에 속한다.

BILBO 구조를 적용할 때 테스팅을 위한 면적이 최소가 되도록 하기 위해서 응답분석기와 패턴 발생기로 적합한 레지스터인 파이프라인 레지스터들을 BILBO 구조로 하였다.

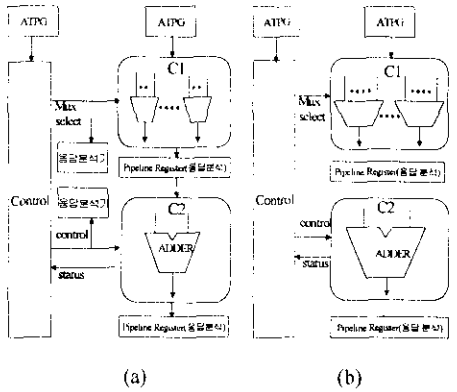

그림 6. 일반적인 구조(a)와 집적화된 구조(b)

테스팅을 조절하기 위해서 BIST 제어기 부분은 테스트가 시작되면 상태에 따라 동작하게 된다. 그림 7 은 BIST 제어기의 상태도이다. 만약 ALU 에 테스트 시작 신호가 오면, 먼저 BILBO 구조의 플립-플롭을 초기화 한다. 두번째 단계는 ALU 블록인 CIRCUIT1 과 CIRCUIT2 의 스캔 패스를 연결하여 BILBO 구조가 정상인지를 검사하는 것과 동시에 패턴발생을 위해서 특정 seed 값을 갖도록 하며 에러가 발생하면 BILBO_ERROR 를 1 로 세트한다.

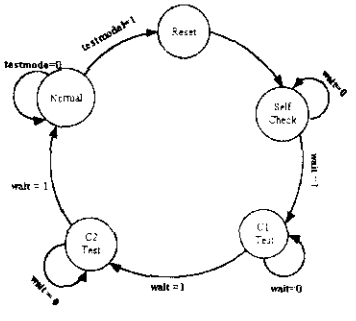

그림 7. BIST 제어기 상태도

세번째 단계는 CIRCUIT1 의 테스팅을 위해서 파이프라인 decode/execution 레지스터는 응답 분석 모드로 동작하고 파이프라인 execution/memory 레지스터는 패턴 발생 모드로 동작한다. 네번째 단계는 CIRCUIT2 를 테스팅하기 위해서 execution/memory 레지스터는 응답 분석 모드로 동작하고 파이프라인 decode/execution 레지스터들은 패턴 발생 모드로 동작하게 된다. 세번째와 네번째 단계에서 응답 분석 결과를 이미 저장되어

있는 ROM의 데이터와 비교한다. 그리고 값이 다르면 출력으로 PASS/FAIL 신호를 발생하여 외부로 출력한다. 여기서 패턴 발생기에서 패턴을 만들 때 단일 seed 값에 의해서는 원하는 fault coverage를 얻는데 한계가 있다. 또한, 자체 내장 테스팅을 적용하는 것도 한계가 있다. 이를 보완하기 위해서 테스팅이 용이한 설계로 변경하였다. 예를 들어 stuck-at과 같은 고장을 발생이 적은 방향으로 설계를 변경하였다. 그리고, 고장이 관찰점으로 전파되기가 용이하도록 게이트의 순서를 바꾸거나 설계를 변경하였다. 패턴 발생기에서는 단일 seed 값을 사용하지 않고 일정한 수만큼 패턴을 발생하여 회로 블록을 테스팅 후에 다음 seed 값을 사용하여 패턴을 발생하여 다시 테스팅할 수 있도록 BIST 제어기를 변경하였다.

4. 검증

본 연구에서 설계한 회로는 HDL(Hardware Description Language)로 기술하였다. 그리고, HDL 시뮬레이션을 통해 각 연산에 대한 결과를 검증하였다. 그리고, 자동 합성 툴을 사용하여 0.6u 3.3v 3 메탈 공정을 사용하여 합성 하였다.

표 1. ALU 각 회로에 대한 고장 시뮬레이션 결과
(* Coverage는 %이다.)

Design	Detect	Aband	Tied	Redun	Untest	Probl	Unpro	Total	Co-verage
CIR-CUIT1	4607	0	0	0	7	0	0	4614	99.85
CIR-CUIT2	10011	0	0	0	357	0	0	10368	96.56

Fault coverage를 측정하기 위하여 HDL 기술을 자동 합성한 후에 그 합성 결과로 나온 회로를 이용하여 테스트 시뮬레이션 툴을 사용하여 고장 시뮬레이션을 하였다. 만족할 만한 fault coverage가 나올 수 있도록 패턴 초기값을 바꾸어 가면서 높은 fault coverage가 되는 패턴 값을 추출하였고 테스트가 용이한 ALU 디자인으로 변경하였다. 이렇게 회로 블록인 CIRCUIT1과 CIRCUIT2에 대해서 각각 고장 시뮬레이션을 통하여 구한 fault coverage 결과는 표 1과 같다.

이 결과는 회로 블록 CIRCUIT1에 대해서는 10000개의 패턴을 발생하여 테스팅을 하였고 이때의 fault coverage는 99.85%가 나왔다. 그리고, 회로 블록 CIRCUIT2에 대해서는 50000개의 패턴을 입력하여 96.56%의 fault coverage를 얻을 수 있었다. 그림 8은 이러한 패턴 수에 대한 fault coverage를 나타낸 그림이다.

5. 결 론

본 연구에서는 DSP RISC 구조의 내장 프로세서에 적합한 작게 내장 테스팅 기능을 가진 ALU를 설계하였다. 설계는 HDL로 기술하여 자동합성과 자동 레이아웃을 사용하여 최종적인 레이아웃을 만들었다. 설계 공정은 0.6u 3.3v 3 메탈 라이브러리를 사용하였다. 그리고, 최대 동작 가능 주파수는 최악 조건하에서 66MHz이다. 트랜지스터는 17788개이다. 코어 사이즈는 3956.05 x 3900.9 (mm)이다. 이와 같이 면적이 커진 것은 응답 분석기에서 쓰게 될 비교 결과를 포함하고 있는 ROM과 BIST 제어기가 ALU에 내장되어 있기 때문이다.

이때 생성하는 패턴 수는 6만개이며 66MHz로 동작할 경우에 9ms가 걸린다. 테스팅 회로를 내장으로 인한 면적의 증가는 13.37%이다(ROM 제외). 비록, 면적의 증가가 있었지만 자체 내장 테스팅에 의한 fault coverage는 98.4%을 얻을 수 있었다.

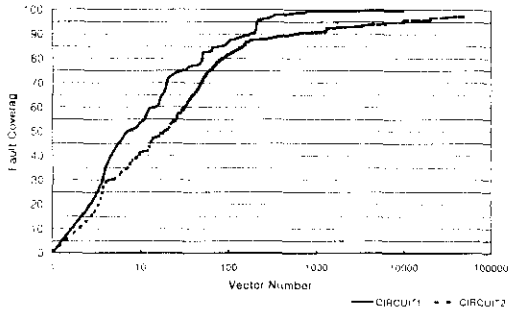

그림 8. 패턴 수(log scale)에 따른 CIRCUIT1과 CIRCUIT2의 fault coverage(%)

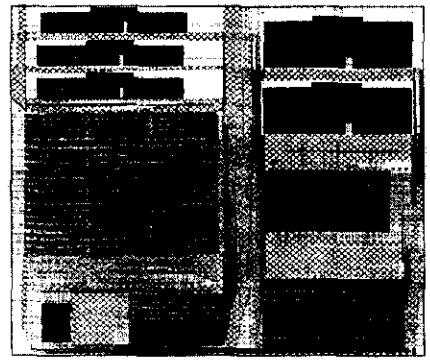

그림 9. ALU 코어 레이아웃(ROM 포함)

6. 참 고 문 헌

[1] Miron Abramovici, Melvin A. Breuer, Arthur D. Friedman, "*Digital System Testing and Testable Design*," IEEE Press, 1990

[2] 홍성제, 박은세, 강성호, 최호용, 장훈 공저, "*테스팅 및 테스팅을 고려한 설계*," 홍릉과학출판사, 1998

[3] Michael Dolle, Manfred Schlett, "*A Cost-Effective RISC/DSP Microprocessor for Embedded Systems*," IEEE Micro, pp. 32-40, Oct. 1995

[4] 이용석, "*ALU와 Register File의 구조*," 비디오 강좌 시리즈(*http://mpu.yonsei.ac.kr*), 1998