DBpia

저전력 소모 임베디드 프로세서 코어 자동생성 시스템의 설계

준회원 김 동 원*, 정회원 황 선 영*

Design of an Automatic Generation System for Embedded Processor Cores with Minimal Power Consumption

Dong-Won Kim* *Associate Member*, Sun-Young Hwang* *Regular Member*

요 약

본 논문은 머신 기술 언어를 이용하여 전력 소모가 최소화된 임베디드 코어 자동 생성 시스템의 설계에 대해 기술한다. 머신 기술 언어를 사용하여 어플리케이션에 적합한 임베디드 코어를 빠른 시간에 설계하기 위해 어플리케이션 실행시 동적 전력 소모가 최소화된 코어를 생성하는 시스템을 구축하였다. 제안된 시스템은 각 인스트럭션의 파이프라인 스테이지의 행위 정보로부터 파이프라인 해저드를 찾아내며 처리하는 능력을 가진 임베디드 코어를 생성한다. 생성된 코어는 파워 소모가 최소화되게 만들어져 있다. 자동 생성 시스템의 검증을 위해 ARM9, MIPS R3000에 대해 SMDL로 기술하고 전력 최적화 과정을 거쳐 HDL 코드를 자동 생성하였으며, 어플리케이션에 대해 시뮬레이션을 수행하여 소모되는 전력을 측정하였다. 실험 결과로 생성된 프로세서는 정확한 동작을 수행하였고, 어플리케이션과 생성된 코어에 따라 동적 전력 소모가 20%~40% 줄어드는 것을 확인하였다.

Key Words : MDL, ASIP, Automatic Generation, Spurious Switching Activity, Low Power

ABSTRACT

This paper describes the system which automatically generates power-minimized embedded cores from MDL descriptions. An automatic generation system is constructed which generated embedded cores which consumes less power for application programs. From the usage information on pipeline stages for each instruction, the proposed system generates embedded cores with the capability of detecting/resolving pipeline hazards. The generated cores are configured such that the power consumption is minimized. The proposed system has been tested by generating HDL codes for ARM9, MIPS R3000 architectures. Experimental results show functional accuracy of the generated cores, and show that power reduction of 20%~40% has been observed for benchmark programs.

Ⅰ. 서 론

최근 반도체 기술의 발전으로 인해 VLSI 설계 방법이 회로 수준을 넘어 시스템 수준으로 진화하고 있다. 시스템 설계 부분에서 유저의 요구는 기능과 실행 속도 뿐 아니라 전력 소모에의 높은 성능을 요구한다. 복잡한 시스템 개발에서 다양한 요구사항을 tradeoff까지 고려하며 진행하기에는 time-to-

※ 본 논문은 2007년도 「서울시 산학연 협력 사업」 의 「나노 IP/SoC 설계 기술 혁신 사업단」의 지원으로 이루어졌으며 IDEC에서 제공한 CAD tool을 이용해 simulation과 synthesis를 수행 하였습니다.
* 서강대학교 전자공학과 CAD&ES 연구실 (hwang@sogang.ac.kr)
논문번호 : KICS2007-06-276, 접수일자 : 2007년 6월 14일, 최종논문접수일자 : 2007년 10월 2일

www.dbpia.co.kr

market 요구를 만족시키기 어렵다. 이러한 문제점을 해결하기 위해 최근의 설계방식은 설계물의 유연성과 재사용률을 높이고 있는 추세이다. 이러한 유연성, 재사용률 그리고 어플리케이션의 최적화를 위해 고정된 범용 프로세서 대신 ASIP (Application Specific Instruction-set Processors)이 사용되고 있다[1][2]. 기존의 SoC 설계가 기능 구현과 고속 동작에 초점이 맞추어져 있었으나, 최근에는 휴대용 기기의 수요 증가로 인해 한정된 배터리 사용시간 동안 장시간 동작할 수 있는 저전력 설계기술 개발이 활발히 진행되고 있다.

지금까지 MDL 기반의 프로세서 코어 자동 생성은 최근까지 많은 연구가 진행되어 왔다. 이 중 nML[3], LISA[4], EXPRESSION[5], Micro Operation Description[6] 등을 이용한 프로세서 생성에 대한 연구가 대표적인 예이다. 베를린 대학에서 개발한 nML은 언어 자체에서 파이프라인을 지원하지 않고 간단한 컨트롤러만을 생성할 수 있기 때문에, 파이프라인 기능이 있는 프로세서 생성에 어려움이 있고 전력을 줄이는 측면에서는 특별한 정책을 취하지 않고 있다. 독일의 Aachen 대학에서 개발한 LISA는 nML의 단점을 개선하여 파이프라인 기술이 가능하며, TMS320c620x등의 DSP 계열의 프로세서의 기술이 가능하다. 그러나 프로세서 코어 생성 측면에서 데이터패스와 디코딩 정보만을 추출하여 고정된 파이프라인 컨트롤러에 맵핑하는 방식을 취하고 있어, 다양한 머신의 구조를 생성하는 데에는 한계가 있고 전력을 줄이는 방법으로 gated clock을 사용하고 있어 최근 이슈가 되고 있는 동적전력 감소에 적극적으로 대응할 수 없다는 문제가 있다[7]. 미국의 UCI에서 개발한 EXPRESSION은 ADL(Architecture Description Language)를 이용하여 설계자가 컨트롤러, 해저드 검출방법과 전력을 줄일 수 있는 모듈에 대한 정보를 자세히 기술해야 하기 때문에 프로세서 개발 초기부터 타겟 프로세서에 대해 구체적이고 정확한 기술이 요구된다. 일본 오사카 대학에서는 Micro Operation Description을 통해 프로세서 코어를 자동 생성하였으나 언어 특성상 추상화 수준이 낮은 RTL 수준의 기술이 필요하기 때문에, 타겟 프로세서의 컴파일러 개발에 제약이 있으며, 블록 데이터 전송을 위한 멀티사이클 인스트럭션, 파이프라인 해저드 처리와 전력을 고려하지 않았다.

ASIP을 포함하는 프로세서의 매 클럭 사이클마다 소모되는 전력의 편차는 상당히 크며, 전력 소모의 편차는 수행되는 명령어, 어드레스 모드, 사용되는 레지스터 파일 넘버, 레지스터에 저장되는 값과 프로그램 카운터의 변화 등에 의해 생긴다[8]. 이런 전력 소모의 편차를 줄이기 위해서 컴파일러를 이용한 코드 최적화[9] 또는 프로세서 모듈의 bit-switching에 의한 동적전력[10]를 줄이는 방안이 모색되고 있다.

본 연구에서는 설계자가 MDL 기반의 설계 방식으로 전력에 최적화된 임베디드 프로세서를 설계할 경우 머신 기술 언어 SMDL를 사용하여 타겟 프로세서를 기술하여 프로세서 코어를 생성하되[11], 어플리케이션 실행시 소모되는 동적 전력을 줄일 수 있는 코어를 생성하는 시스템을 구축하였다. 구축된 시스템은 RISC 타입의 ASIP을 자동 생성하며, 별도의 파이프라인 해저드 정보의 기술없이 해저드 감지/처리가 가능하고, 메모리 엑세스 등의 효율적인 처리를 위해 다중 사이클 인스트럭션을 지원한다. 생성된 코어는 spurious 연산자의 bit-switching을 최소화하여 동적전력 소모를 최소화한다. 구축된 시스템은 SMDL 시스템의 retargetable 컴파일러 및 인스트럭션 셋 시뮬레이터와 연동하여 ASIP의 생성과 검증에 사용된다. 제안된 시스템의 평가를 위해 상용화된 임베디드 프로세서 중 ARM9과 MIPS R3000을 SMDL로 기술하고 구축된 시스템을 통해 프로세서 모델을 생성한 뒤, 각각의 모델을 실제 프로세서의 동작과 비교하여 생성된 프로세서 모델의 정확성을 검증하였고, 어플리케이션을 실행하여 파워 최소화 모듈을 추가하지 않은 SMDL 생성 프로세서 모델과의 동적전력을 비교하였다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 연구 발표된 코어 생성기 및 임베디드 프로세서 코어의 자동 생성 과정을 제시하고 개선된 사항을 기술한다. Ⅲ장에서는 제안된 시스템에서 동적 전력 소모를 최소화하기 위한 코어의 생성 방법을 제시하고, Ⅳ장에서는 제안된 시스템의 검증을 위해 파워 최소화 모듈을 생성하지 않는 코어 생성기의 결과물과 제안된 생성기의 결과물의 전력 소모를 어플리케이션을 이용하여 비교하였다. Ⅴ장에서는 결론 및 추후 과제를 제시한다.

Ⅱ. 임베디드 코어 생성기 개관

구축된 임베디드 코어 생성기는 SMDL을 파싱하여 구조 정보 추출과 메모리 모델을 생성하는 전처리 과정, 중간형태인 CDFG (Control Data Flow

www.dbpia.co.kr

Graph) 합성 과정, 프로세서의 컨트롤러, 데이터패스와 전력 컨드롤 유닛을 생성하는 코어 모듈 생성 과정, 생성된 모델을 합성 가능한 HDL 코드로 변환하는 HDL 생성 과정으로 이루어진다[11].

2.1 전 처리 과정

SMDL로 기술된 타겟 프로세서를 SMDL 파서가 AST로 생성하고, 생성된 AST중 인스트럭션은 연산자/레지스터와 그들 간의 연결구조로 구성된 CDFG로 변환한다. 또한 중간형태의 구조와 메모리 정보를 이용하여 타겟 프로세서의 버스연결, 핀 할당, 파이프라인 정보로 이루어진 구조 정보와 타겟 프로세서의 주 메모리로 사용될 메모리 모델을 생성한다.

2.2 인스트럭션 합성과 연결 최적화

인스트럭션 합성과 연결 최적화 과정은 전처리 과정에서 생성된 각 인스트럭션의 CDFG를 하나의 CDFG로 조합하여 타겟 프로세서의 데이터패스 모듈을 생성하기 위한 과정으로, 모듈간의 연결을 최소화하여 전력을 줄일 수 있는 역할을 수행한다. 조합된 CDFG의 노드는 연산자, 레지스터와 멀티플렉서를 나타내며 에지는 버스를 나타낸다. 인스트럭션 합성과 연결 최적화는 다음과 같은 과정으로 이루어진다.

모든 CDFG의 노드들을 각 스테이지에 맞게 배열하고 중복되는 레지스터와 연산자를 제거한다. 이후 인스트럭션별 CDFG들로부터 노드들 간의 연결정보를 추출하고, 각 연결구조의 특성에 따라 최소화 과정을 거친다. 동일 시작점-동일 종점의 연결구조는 하나의 연결구조 만을 남기고 제거한다. 다른 시작점-동일 종점으로 연결되는 연결구조들은 종점 연산자의 교환법칙 가능 여부에 따라 종점의 특정 포트에 연결하거나, 컬러링 알고리즘을 이용하여 최소의 연결을 가진 연결구조 및 최소 크기의 멀티플렉서를 추출한다. 이 후 각 리소스를 해당 파이프라인 스테이지에 맞도록 스케줄링을 수행한다.

2.3 코어 모듈 생성 과정

코어 모듈 생성 과정은 IF(Instruction Fetch)/PC (Program Counter) 블록을 생성하는 IF/PC 블록 생성기와 데이터패스, 컨트롤러, 파이프라인 해저드 처리 모듈, 멀티 싸이클처리 모듈을 생성하는 코어 합성기와 전력 제어의 역할을 하는 모듈을 생성하는 전력 컨트롤 생성기로 구성된다. IF/PC 블록 생성기는 전 처리 과정에서 생성한 구조 정보를 이용하여 PBR (Program Buffer Register), PAR (Program Address Register)를 가진 PC 체인 구조[12]의 IF/PC 블록 모델을 생성한다. 데이터패스는 전처리 과정 및 연결 최적화 과정에서의 파이프라인 스케줄링된 CDFG를 이용하여 구성하게 되고, 각 노드들이 실행될 스테이지와 동작에 대한 정보를 가진 ACT(Active Component Table)[13]와 기존에 구성된 CDFG의 연결 정보와 함께 구성하여 해저드 처리 모듈과 전력 컨트롤 모듈을 제외한 데이터패스와 컨트롤러를 구성한다. 기존 SMDL 파이프라인 해저드 처리 회로 생성 과정은 데이터 및 컨트롤 해저드 전부 스톨만을 이용하여 해저드 처리를 수행한다. 멀티 사이클 인스트럭션 실행 시는 ACT를 분석하여 처리에 필요한 사이클 정보를 멀티 사이클 룩 업 테이블에 저장하여 해저드 스톨 회로를 이용하여 처리한다.

2.4 외부 입출력 포트 생성 과정

포트 기술로부터 생성되는 외부 입출력 포트는 외부 구조 정보 중간형태 중 유저 정의 포트의 이름, 속성 그리고 비트 폭을 이용하여 유저가 기술한 타겟 프로세서의 외부 포트를 생성한다. 포트 기술 없이 생성되는 외부 입출력 포트는 타겟 프로세서가 동작하기위해 반드시 필요한 기본적인 포트나 외부 디바이스와 동기를 맞추는데 필요한 포트들을 유저 기술 없이 자동 생성한다.

Ⅲ. 생성된 코어의 전력소모 감소를 위한 모듈 생성

제안된 전력 소모 감소를 위한 방식은 많은 전력을 소비하는 모듈을 찾아내는 전력 컨트롤 모듈 스케줄링 과정, 전력 컨트롤 유닛을 생성하는 전력 컨트롤 모듈 바인딩 과정, 인스트럭션의 수행 사이클 중 동작에 필요 없는 부분을 정지시키는 gated clock 모듈 생성 과정, 전력 컨트롤 모듈 및 gated clock을 제어하는 컨트롤러 모듈 생성 과정 그리고 실행 사이클을 줄이기 위한 해저드 처리 모듈 과정으로 이루어진다.

3.1 전력 컨트롤 모듈 스케줄링

SMDL의 일련의 생성과정을 거친 CDFG에는 전력 소모를 줄이기 위한 방법으로 spurious 연산자의 bit-switching을 줄이기 위한 컨트롤 모듈이 추가된다. 이 경우 전력 컨트롤을 위한 모듈 생성을 모든 연산 모듈의 입력단에 생성하는 방식이 있으나, 컨트롤러의 신호 생성의 오버헤드가 커질 가능성이 있다. 위와 같은 오버헤드를 줄이고 어플리케이션에

www.dbpia.co.kr

적합한 전력 컨트롤 모듈 생성을 위해 SMDL 코어 생성기에서는 다음과 같은 생성 방식을 사용한다.

SMDL 시스템의 retargetable 컴파일러는 저전력 코드 생성을 지원한다. 이 컴파일러는 Profile-Based Optimization을 수행하여 타겟 프로세서에 최적화된 저전력 코드를 생성하며, 이는 어플리케이션에 대한 profile 정보를 이용하여 최적화된 코드를 생성하는 기법이다[18]. 그림 1은 Profile-Based Optimization을 수행하여 저전력 VHDL 코드를 생성하는 구조를 나타낸다. 컴파일러는 C 소스 코드와 타겟 프로세서에 대한 SMDL에서 파생된 IF(Intermediate Form)를 입력 받아 binary화 된 타겟 코드를 생성하며, 인스트럭션 셋 시뮬레이터 생성기는 타겟 프로세서에 대한 SMDL 기술을 이용하여 profile 정보를 추출할 수 있는 cycle-accurate 인스트럭션 셋 시뮬레이터를 생성한다. 생성된 시뮬레이터는 컴파일러로부터 binary화 된 타겟 코드를 입력으로 받아 시뮬레이션을 수행하면서 각 인스트럭션 수행 횟수, switching activity 등의 power profile 정보를 생성한다. CDFG 파워 분석기는 인스트럭션 셋 시뮬레이터로부터 생성된 전력 profile 정보를 피드백 받고 IF의 CDFG를 파워 정보가 입력된 CDFG로 업데이트한다. 위의 과정을 특정 어플리케이션이 각 인스트럭션의 수행 횟수가 수렴할 때 까지 반복 수행 후, 해당 어플리케이션에 대한 각 인스트럭션의 수행 횟수를 계산해낸다. 이 정보와 모듈 라이브러리에 포함된 각 모듈의 파워 소모값을 이용하여 생성하려는 ASIP에서 많이 사용되는 연산 모듈들의 빈도수와 파워값을 계산해낸다. 이 값을 이용하여 최종적으로 많은 전력을 소모하는 모듈들의 동적 파워를 컨트롤 할 수 있는 CDFG를 얻게 된다.

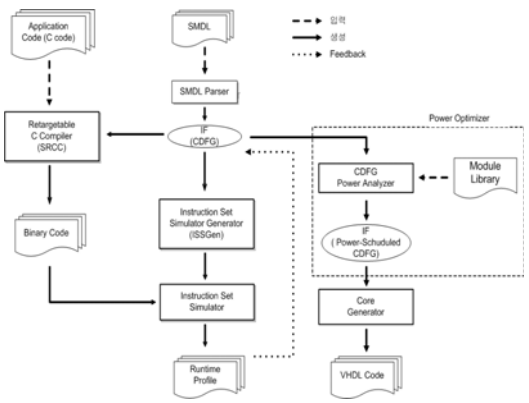


그림 1. Profile based optimization을 수행한 전력 컨트롤 모듈 생성 과정.

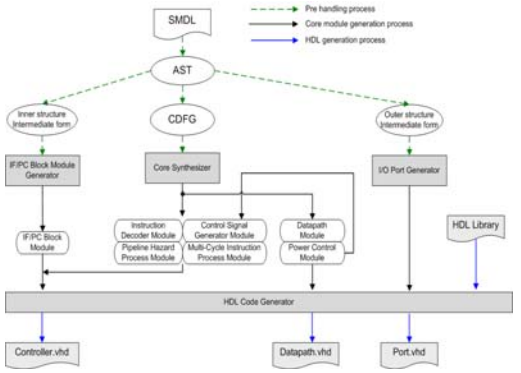


그림 2. 저전력 소모 코어 생성기 시스템의 개관.

표 1. MIPS R3000에서 어플리케이션의 각 인스트럭션의 실행 횟수 및 모듈 사용 여부를 나타낸 표의 일부

연산모듈 / instruction		ALU		Multiplier		Comparator		Barrel Shifter		…
Name	실행 횟수	A_in	B_in	A_in	B_in	A_in	B_in	A_in	B_in	
ADD	10	on	on	pass	pass	pass	pass	pass	pass	…
SHL	4	pass	on	pass	pass	pass	pass	on	on	…
JMP	5	pass	on	pass	pass	on	on	on	on	…
MUL	6	pass	on	on	on	pass	pass	pass	pass	…
:	:	:		:		:		:		

연구 발표된 SMDL 코어 생성 과정은 SMDL 기술만을 이용하여 코어를 생성하기 때문에 전력 최적화 과정이 포함되어 있지 않다. 생성된 ASIP의 동적 전력 소모를 최소화하기 위해 파워 프로파일 정보가 들어간 CDFG에 bit-switching에 의해 생기는 동적 전력을 줄일 수 있는 모듈 생성부분을 추가하였다. 그림 2는 이러한 전력 컨트롤 모듈 생성 부분을 추가로 인해 개선된 코어 생성기의 개관을 보이고, 표 1은 MIPS R3000을 SMDL로 기술한 경우 파워 프로파일 정보가 들어간 CDFG를 이용하여 각 인스트럭션의 수행횟수와 spurious operator를 나타낸다.

3.2 전력 컨트롤 모듈 바인딩

SMDL 생성 과정에서는 전력을 고려하지 않고 유저가 기술한 대로 프로세서가 생성되는 방식을 사용한 시스템에 동적 전력 소모 감소를 위해 전력 컨트롤 모듈을 추가한 파워 스케줄링된 CDFG를 이용하여 모듈에 대한 우선순위를 정한다. 해당 모듈이 특정 인스트럭션이 실행될 경우 spurious 연산자일 경우에 대비하여 모듈 전방에 아래와 같은 모듈을 자동으로 배치하여 연산모듈의 동적 전력 소모를 줄인다. 이 경우 다음과 같은 세 가지 방식으로 나눌 수 있다.

www.dbpia.co.kr

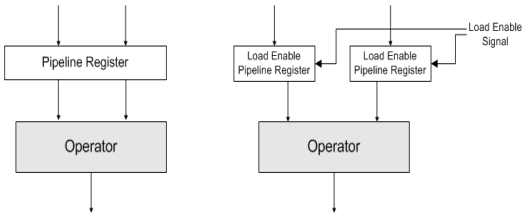


그림 3. 로드 인에이블 레지스터를 이용한 bit-switching 컨트롤.
(a) 전력 컨트롤 모듈 스케줄링 적용 전.
(b) 전력 컨트롤 모듈 스케줄링 적용 후.

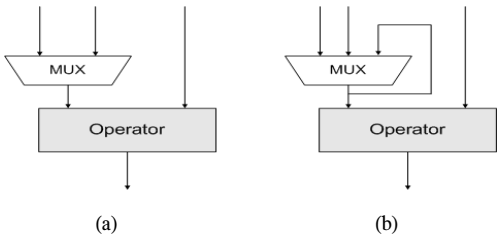


그림 4. Retentive 멀티플렉서를 이용한 bit-switching 컨트롤.
(a) 전력 컨트롤 모듈 스케줄링 적용 전.
(b) 전력 컨트롤 모듈 스케줄링 적용 후.

3.2.1 로드 인에이블 레지스터 바인딩

파이프라인 레지스터가 N-1 스테이지와 N 스테이지 사이에 존재하고, 연산자가 N 스테이지에 존재하면서 해당 파이프라인 레지스터에 직접 연결되어 있는 경우, 그림 3과 같이 파이프라인 스테이지 레지스터를 load enable register로 대체하여 spurious 연산자의 bit-switching을 막는다. 이 경우에 회로의 면적에 큰 영향을 주지 않는 다는 장점이 있다.

3.2.2 Retentive 멀티블렉서 바인딩

파이프라인 레지스터가 N-1 스테이지와 N 스테이지 사이에 존재하고, 연산자가 N 스테이지에 존재하면서 해당 파이프라인 레지스터에 직접 연결되어 있지 않은 경우, 그림 4와 같이 retentive 멀티플렉서를 연결하여 spurious 연산자의 bit-switching을 막는다. 이를 위해 retentive 멀티플렉서가 추가되나 각 인스트럭션의 CDFG 스케줄링 과정에서 2개의 operand가 하나의 목적지를 가리킬 경우 멀티플렉서를 추가해주어야 하기 때문에 실질적으로 면적이나 전력에 부담을 주지 못한다.

3.2.3 랫치 바인딩

파이프라인 레지스터가 N-1 스테이지와 N 스테이지 사이에 존재하고, 연산자가 N 스테이지에 존재하며 그림 5와 같이 멀티 패스가 존재할 경우에 load

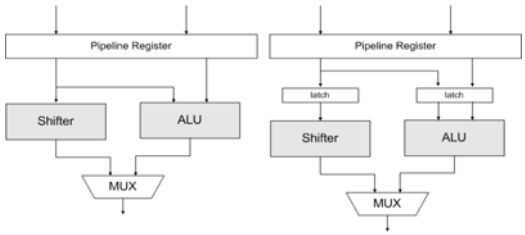


그림 5. Load enable 랫치를 이용한 guarded logic.
(a) 전력 컨트롤 모듈 스케줄링 적용 전.
(b) 전력 컨트롤 모듈 스케줄링 적용 후.

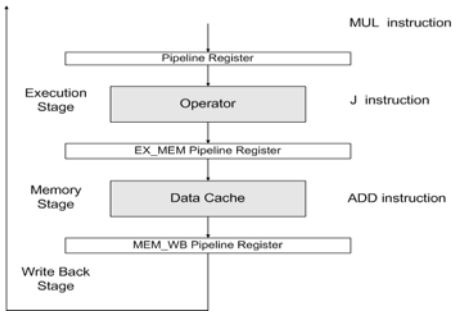
enable 랫치를 이용하여 bit-switching을 막도록 한다. 랫치를 사용할 경우 값이 enable 되지 않으면 enable 되었을 때 보다 데이터 bit-switching의 파워 소모가 0.1% 미만으로 떨어지게 되는 강점이 존재한다[14]. 물론 RTL 회로에 랫치 사이즈만큼의 면적부담이 작용하나, 실험 결과 회로 전체적인 면에서 볼 경우에는 0.5% 미만으로 면적이 증가하게 된다.

3.3 Gated Clock 제어 모듈 생성

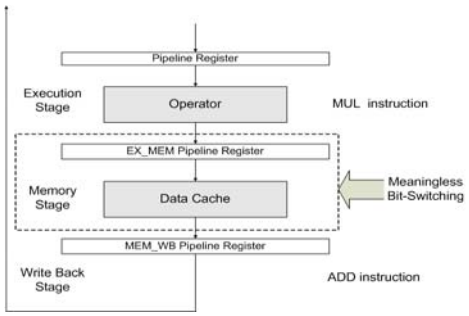
현재 ASIP 설계에서 클럭 신호는 파워 소비의 큰 부분을 차지하고 있다. 이 원인은 첫째로 신호가 항시 변하고 있다는 점이고, 두 번째는 클럭 트리로 인하여 실제 회로에서 많은 면적을 차지하고 있는 데에 있다. 이러한 클럭에 의한 파워 소모를 줄이기 위해 가장 많이 사용되는 방법이 gated clock이다[14].

Gated clock을 사용하기 위해서는 전력 스케줄링된 CDFG로부터 각 인스트럭션의 파이프라인 동작 정보를 추출하여 각 사이클 당 사용되는 스테이지와 그렇지 않은 스테이지를 구분해 낸 후에 인스트럭션 패치 스테이지를 제외한 각 파이프라인 레지스터에 gated clock을 사용하도록 한다. 그림 6은 MIPS R3000 코어에서 파이프라인 레지스터를 gated clock을 이용하여 컨트롤하는 예를 나타낸다. 그림 6 (a)에서는 인스트럭션이 실행되는 순서를 나타낸다. MIPS R3000이 실행 스테이지에서 JUMP 인스트럭션의 어드레스를 계산할 경우에, 그림 6 (b)와 같은 형태로 인스트럭션이 각 스테이지에서 실행된다. 이때 메모리 스테이지와 EX_MEM 파이프라인 레지스터에 아무런 의미가 없는 JUMP 인스트럭션의 쓰레기 값이 들어가 bit-switching을 일으키게 된다. 이러한 점을 방지하기 위해 생성되는 코어의 컨트롤러는 각 스테이지의 파이프라인 레지스터에 연결된 gated clock을 이용하여 컨트롤러가 그림 6 (c)와 같이 제어하는 방식을 사용한다.

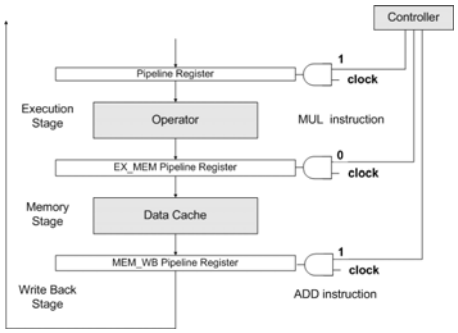
www.dbpia.co.kr



(a)



(b)



(c)

그림 6. MIPS R3000에서 gated clock의 컨트롤 예.
(a) 인스트럭션 순서.
(b) 의미없는 파이프라인 스테이지의 수행 예.
(c) Gated clock을 이용한 제어 예시.

3.4 전력 컨트롤 모듈 컨트롤러 생성

전력 스케줄링된 CDFG로부터 추가된 전력 컨트롤 모듈을 구성된 ACT에 추가한다. 추가된 부분은 데이터 패스를 바꾸지 않고 여기에 모듈을 추가시키는 방식을 사용하기 때문에 추가모듈에 의한 ACT 확장에는 문제가 없다. 이 후 각 인스트럭션에 해당하는 CDFG에서 에지 정보를 추출하고 스케줄링된 CDFG에서 추출한 에지와 동일한 루트를 가지는 에지를 찾아낸다. 이 루트에 포함되지 않는 루트의 연산 모듈 입력에 해당하는 전력 컨트롤 모듈을 전부 인스트럭션이 실행되는

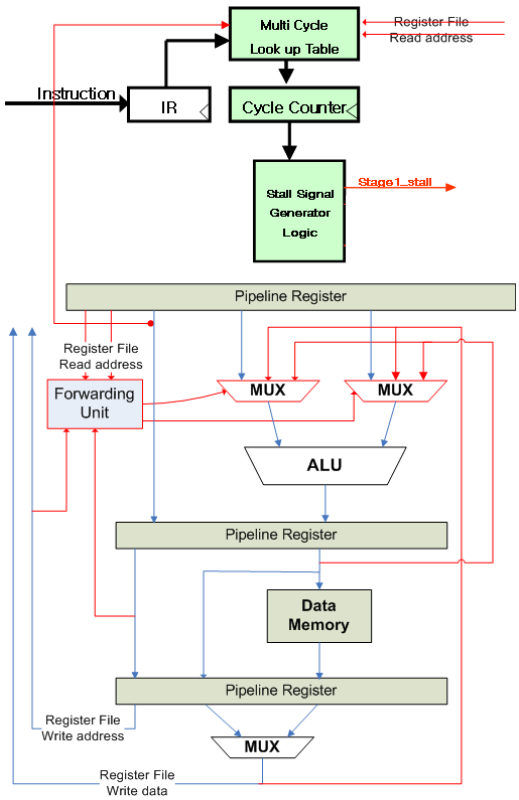


그림 7. 개선된 파이프라인 해저드 처리 생성 회로.

스테이지에서 disable시켜 spurious 연산자의 bit-switching을 막아 동적 전력을 줄일 수 있게 한다.

3.5 해저드 처리 모듈 생성

SMDL 생성 과정에서는 스톨만을 이용하여 해저드 처리를 수행할 경우 어플리케이션을 실행할 경우 실행 속도가 늘어나게 되며, 이로 인해 생기는 추가 스톨에 의해 파워도 소모되는 구조를 가지고 있다. 코어 생성기에서는 이러한 부분을 방지하기 위해 인터널 포워딩이 연산 유닛과 저장 유닛 사이에서 수행될 수 있도록 필요한 버스와 모듈을 자동 생성할 수 있도록 한다. 그림 7은 MIPS R3000에서 생성되는 인터널 포워딩 처리 유닛을 나타낸 것이다. 메모리 로드 후에 이 로드 값을 다음 인스트럭션에서 연산에 사용할 경우, 반드시 인스트럭션 패치 부분을 스톨시켜 주어야 한다. 이 때 메모리 스테이지와 실행 스테이지의 차이가 있을 경우에는 그 차이 수만큼 스톨시켜 주는 방식을 사용한다. 스톨은 multi cycle lookup table에 read address와 write address를 넣어 비교하고 동시에 instruction sequence를 분석할 수 있는 기능을 추가하는 방식으로 수행된다.

www.dbpia.co.kr

Ⅳ. 실험 결과

제안된 시스템에서 타겟 프로세서의 생성을 검증하기 위해 상용화된 임베디드 프로세서 코어인 ARM9[15], MIPS R3000[16][17]을 타겟 프로세서로 정하고, SMDL을 사용하여 '내부 구조 기술' 부분과 '외부 구조 기술' 부분을 기술하였다. 기술된 타겟 프로세서 기술들을 제안된 시스템에 입력하여 각각의 타겟 프로세서 코어들을 생성하였고, 생성된 타겟 프로세서 코어의 동작의 정확성을 여러 벤치마크를 통해 검증하였다. 동시에 전력을 측정하여 기존 자동생성 방식에서 생성된 코어의 소비 전력과 제안된 자동생성 방식의 생성된 코어의 소비 전력과 면적 증가를 비교하였다. 각 실험에 사용된 어플리케이션은 retargetable 컴파일러 SRCC를 이용하여 컴파일된 실행 코드를 사용하였다. 제안된 시스템은 UNIX 기반의 SUN-Sparc Workstation에서 구현되었으며, 합성에 사용된 툴은 Synopsys사의 Design Compiler이다. 합성 환경으로 공정은 매그너칩 0.25㎛을 사용하였고, 동작 주파수는 70 MHz, 동작 온도는 25℃, 동작 전압은 3.3 V 로 셋팅하였다.

4.1 ARM9 코어의 생성 실험

ARM9 모델을 생성하는 실험에서는 실제 프로세서와 마찬가지로 하버드 아키텍처에 파이프라인은 5단이며, 클럭은 2-phase non-overlapping 클럭을 사용하였다. 표 2는 8-point 2-radix FFT와 4x4 매트릭스 곱셈의 기존 코어 생성기와 제안된 코어 생성기의 전력과 면적 측면에서의 차이를 보인다. 표 2와 같이 각 어플리케이션에 대한 동적 전력소모가 전력컨트롤 회로를 포함한 코어가 이러한 회로를 포함하지 않은 회로에 비해 동적 전력소모가 크게 줄어듦을 알 수 있다.

생성된 모델은 5단 파이프라인에 인스트럭션 버스와 데이터 버스가 분리되었으며 버스나 리소스 충돌은 발생하지 않았다. FFT 연산 실행시는 4,056 사이클, 매트릭스 곱셈 연산시 1,908 사이클이 소요되었다. 이 때 사용된 전력은 사용된 합성 결과 타겟 프로세서는 16,231게이트로, 전력 컨트롤 모듈 추가시는 17,689게이트로 설계되어 8.9%가 증가되었다. 이 때 증가된 면적은 대부분 전력 컨트롤 모듈을 컨트롤하는 컨트롤러와 인터널 포워딩으로 인해 생긴 부분이다.

표 2. ARM9에서의 어플리케이션 실행 결과

	기존 방법 (uW)	제안된 방법 (uW)	파워 소모 비교 (%)	면적 비교 (%)
8-point 2-radix FFT	32.2	18.4	- 42.9	+8.9
4x4 Multiplication	21.2	15.2	- 28.3	

4.2 MIPS R3000 코어의 생성 실험

MIPS R3000 모델을 생성하는 실험에서는 실제 프로세서와 마찬가지로 하버드 아키텍처에 파이프라인은 5단으로, 클럭은 2-phase non-overlapping 클럭을 사용하였다. 표 3은 8-point 2-radix FFT와 4x4 매트릭스 곱셈의 기존 코어 생성기와 제안된 코어 생성기의 전력과 면적 측면에서의 차이를 보인다. 표 3과 같이 각 어플리케이션에 대한 동적 전력소모가 전력컨트롤 회로를 포함한 코어가 이러한 회로를 포함하지 않은 회로에 비해 동적 전력소모가 크게 줄어듦을 알 수 있다.

생성된 모델은 5단 파이프라인에 인스트럭션 버스와 데이터 버스가 분리되었으며 버스나 리소스 충돌은 발생하지 않았다. FFT 연산 실행시는 4,502 사이클, 매트릭스 곱셈 연산시 2,011 사이클이 소요되었다. 이 때 사용된 전력은 사용된 합성 결과 타겟 프로세서는 21,653게이트로, 전력 컨트롤 모듈 추가시는 24,021게이트로 설계되어 10.9%가 증가되었다. MIPS R3000이 ARM9에 비해 제안된 방법에 의한 동적 전력소모율이 적은 이유는 ARM9에 비해 기본적으로 필요한 모듈의 양이 적기 때문이다.

표 3. MIPS R3000에서의 어플리케이션 실행 결과

	기존 방법 (uW)	제안된 방법 (uW)	파워 소모 비교 (%)	면적 비교 (%)
8-point 2-radix FFT	42.2	30.3	- 28.2	+10.9
4x4 Multiplication	30.1	23.9	- 21.1	

Ⅵ. 결론 및 추후과제

본 논문은 머신 기술 언어 SMDL을 이용하여 임베디드 프로세서 코어를 자동 생성하는 시스템인 임베디드 코어 생성기에서 spurious 연산자의 bit-switching 전력을 줄일 수 있는 방법을 제안하였다. 제안된 시스템은 기존 시스템의 전 처리 과정, 코어 모듈 생성 과정, HDL 코드 생성 과정에 전력 컨트롤 모듈을 추가하여 타겟 프로세서의 코어를 생성한다. 제안된 시스템이 생성한 프로세서는 SMDL retargetable 컴파일러에 의해 생성된 만들어진 어플리케이션을 실행할 시, 기존 생성과정에서 만들어진 코어보다 20% 이상 동적전력이 줄어듦을 확인할 수 있었다.

www.dbpia.co.kr

실험은 상용화된 임베디드 프로세서중 ARM9, MIPS R3000을 SMDL로 기술하고, 제안된 시스템을 이용하여 프로세서 모델을 생성하였다. 생성된 모델의 전력 소비를 비교하기 위해 기존 코어 생성기에서 생성된 프로세서 모델과 전력 컨트롤 모듈이 포함되어 생성된 프로세서 모델의 전력을 비교하고 그 결과를 제시하였다.

추후 과제로는 파워를 많이 소비하는 부분인 버스, 메모리 및 캐쉬를 컨트롤 할 수 있는 회로와 상위 레벨의 파워 관리 방식인 DVS 및 DVM과 연동할 수 있는 회로 생성이 필요하다.

참 고 문 헌

〔1〕 N. Dutt and K. Choi, “Configurable Processor for Embedded Computing”, IEEE Computer, Vol. 36, No. 1, pp. 120-123, Jan. 2003.

〔2〕 최기영, 조영철, “SoC 설계방법의 최근 동향”, 대한 전자공학회지, 30권 9호, pp. 17-27, 2003년 9월.

〔3〕 A. Fauth, M. Fredericks, and A. Knoll, “Generation of Hardware Machine Models from Instruction Set Descriptions”, in Proc. IEEE Workshop VLSI Signal Processing, Veldhoven, Netherlands, pp. 242-250, Oct. 1993.

〔4〕 O. Schliebusch et al, “A Novel Methodology for the Design of Application-Specific Instruction- Set Processors (ASIPs) Using a Machine Description Language”, IEEE Trans. CAD of Int. Circuits and Systems, Vol. 20, No. 11, pp. 1338-1354, Nov. 2001.

〔5〕 P. Mishra, A. Kejariwal, and N. Dutt, “Rapid Exploration of Pipelined Processors through Automatic Generation of Synthesizable RTL Model”, in Proc. IEEE Int. Workshop on Rapid System Prototyping, San Diego, CA, pp. 226-232, Jun. 2003.

〔6〕 M. Itoh et al. “Synthesizable HDL Generation for Pipelined Processors from a Micro-Operation Description”, IEICE Trans., Vol. E83-A, No. 3, pp. 394-400, Mar. 2000.

〔7〕 A. Chattopadhyay, D. Kammler, E. Witte, 0. Schliebusch, H. Ishebabi, and B. Geukes, “Automatic Low Power Optimizations during ADL-driven ASIP Design”, in Proc. Int. Symp. VLSI Design, Automation and Test, pp. 1-4, Apr. 2006.

〔8〕 N. Chang and K. Kim, “Real-time Per-cycle Energy Consumption Measurement of Digital Systems”, IEE Electronics Letters, Vol. 36, No. 13, pp. 1169-1170, Jun. 2000.

〔9〕 M. Lee, V. Tiwari, S. Malik, and M. Fujita, “Power Analysis and Minimization Techniques for Embedded DSP Software”, IEEE Trans. on VLSI Systems, Vol. 5, No. 1, pp. 123-135, Mar. 1997

〔10〕 H. Lee, J. Lee, and S. Hwang, “A Novel High Level Synthesis Algorithm for Low Power ASIC Design”, Journal of Microelectronic Systems Integration, Vol. 4, No. 4, pp. 219-232, Dec. 1996.

〔11〕 조재범, 유용호, 황선영, “임베디드 프로세서 코어 자동생성 시스템의 구축”, 한국통신학회 논문지, Vol. 30, No. 6A, pp. 526-533, Jun. 2005.

〔12〕 TEMIC Semiconductors, “TSC691E Integer Unit User’s Manual”, 1996.

〔13〕 H. Lee and S. Hwang, “Design of a High-Level Synthesis System for Automatic Generation of Pipelined Datapath”, Journal of KITE, Vol. 31-A, No. 4, pp. 53-67, Mar. 1994.

〔14〕 V. Tiwari, R. Donnelly, S. Malik and R. Gonzalez, “Dynamic Power Management for Microprocessor : A Case Study”, IEEE 10th International Conference on VLSI Design, Jan. 1997.

〔15〕 ARM, ARM922T Technical Reference Manual (rev 0), 2001.

〔16〕 J. Hennessy and D. Patterson, Computer Architecture : A Quantitative Approach, Morgan Kaufmann Publishers Inc., 1990.

〔17〕 G. Kane, MIPS RISC Architecture, Prentice-Hall, 1998.

〔18〕 S. Pees, V. Zivojnovic, A. Hoffmann, and H. Meyr, “Retargetable Timed Instruction Set Simulation of Pipelined Processor Architectures”, in Proc. Int. Conf. Signal Processing Applications and Technology, Toronto, Canada, pp. 595-599, Sept. 1998.

www.dbpia.co.kr

김 동 원 (Dong-Won Kim) 준회원

2005년 2월 서강대학교 전자공학과 졸업
2007년 8월 서강대학교 전자공학과 공학석사 취득
2007년 9월~현 재 알파칩스
<관심분야> 프로세서 및 Controller 자동 설계, 저전력 회로 설계, 머신 기술 언어 등

황 선 영 (Sun-Young Hwang) 정회원

1976년 2월 서울대학교 전자공학과 졸업
1978년 2월 한국과학원 전기 및 전자공학과 공학석사 취득
1986년 10월 미국 Stanford 대학교 전자공학 박사학위 취득
1976년~1981년 삼성 반도체(주) 연구원, 팀장
1986년~1989년 Stanford대학 Center for Integrated Systems 연구소 책임 연구원 및 Fairchild Semiconductor, Palo Alto Research Center 기술자문
1989년~1992년 삼성전자(주) 반도체 기술자문
2002년 4월~2004년 3월 서강대학교 정보통신대학원장
1989년 3월~현 재 서강대학교 전자공학과 교수
<관심분야> SoC 설계 및 framework 구성, CAD 시스템, Embedded System, DSP System 설계 등

www.dbpia.co.kr