논문 2004-41SD-4-11

가변길이 명령어 모드를 갖는 Embedded Microprocessor의 설계

(A Design of an Embedded Microprocessor with Variable Length Instruction Mode)

박 기 현*, 오 민 석*, 이 광 엽*, 한 진 호**, 김 영 수**, 배 영 환**, 조 한 진**

(Ki Hyun Park, Min Seok Oh, Kwang Youb Lee, Jin Ho Han, Young Soo Kim, Young Hwan Bae ,and Han Jin Cho)

요 약

본 논문은 메모리 크기의 제약을 많이 받는 내장형 마이크로프로세서의 문제를 해결하기 위해 32-bit 명령어와 24-bit, 16-bit 명령어를 혼합 사용하여 3가지 명령어 모드를 갖는 새로운 명령어 셋(X32V ISA)을 제안하였으며, 이를 기반으로 32-bit 5 stage pipeline RISC 마이크로프로세서를 설계하였다. 이를 검증하기 위해서 X32V ISA 전용 시뮬레이터를 이용하여 멀티미디어 프로그램의 프로그램 코드 사이즈를 산출하였다. 그 결과로 Light mode와 Ultra light mode는 Default mode에 비해 각각 최소 8%, 27%의 프로그램 코드 사이즈 감소를 확인하였으며, Xilinx FPGA를 이용하여 33MHz 동작 환경에서 X32V ISA의 모든 명령어 수행을 검증하였다.

Abstract

In this paper, we proposed a new instruction set(X32V ISA) with 3 different types of instruction mode. The proposed instruction set organizes 32-bit, 24-bit, 16-bit instruction in order to solves a problem of memory size limitation in an embedded microprocessor. We designed a 32-bit 5 stage pipeline RISC microprocessor based on the X32V ISA. To verify the proposed the X32V ISA and a microprocessor, we estimated a program code size of multimedia application programs using a X32V simulator. In result, we verified that the Light mode and the Ultra Light mode obtains 8%, 27% reduction of a program code size through comparison with the Default mode. The proposed microprocessor was verified all X32V instructions execution at Xilinx FPGA with 33MHz operating frequency.

Keywords: X32V ISA, RISC microprocessor, program code size

I. 서 론

최근 내장형 마이크로프로세서를 탑재한 소형 정보기기나 가전기기 같은 내장형 시스템(Embedded System)이 급속도로 발전하면서 내장형 시스템의 핵심적 역할을 하는 내장형 마이크로프로세서의 중요도가 높아졌다. 내장형 마이크로프로세서는 PC의 CPU와 같이 고성능의 구조나 강력한 처리기능 보다는 응용분야에 적합한 구조와 기능을 갖는 것이 중요하다. 내장형 시스템은 소형화, 휴대화 추세로 발전하고 있으며, 이로 인해 내장형 마이크로프로세서는 소프트웨어를 저장하기 위한 ROM, RAM의 크기가 한정되어 있으므로 응용 프로그램은 되도록 작은 크기의 코드를 가져야한다. 이는 마이크로프로세서의 명령어 세트와 밀접한 관계를 갖는다. 내장형 마이크로프로세서의 대부분이 명령어의 길이가 같고 단일 명령어로 단일 동작을 수행하여 명령어 실행 및 해석이 간편한 RISC(Reduce Instruction Set Computer) 형태의 구조를 채용하고 있다[1]. 그러나

* 정회원, 서경대학교 컴퓨터공학과
(Department of Computer Engineering, Seokyeong University)
** 정회원, 한국전자통신연구원 시스템IC설계팀
(System IC Design Team, ETRI)
※ 본 연구는 ETRI와 IT_SoC 사업단 지원으로 수행되었습니다.
접수일자: 2003년11월13일, 수정완료일: 2004년3월26일

RISC 형태의 구조는 명령어 기능이 단순하여 이 때문에 프로그램 코드의 크기가 증가하는 경향이 있다. 이런 문제를 해결하고자 대표적인 32-bit 내장형 마이크로프로세서인 ARM 마이크로프로세서와 MIPS 마이크로프로세서는 16-bit으로 압축된 명령어 세트인 Thumb 명령어와 MIPS16 명령어를 사용하여 프로그램 코드의 크기를 줄이고 있다[1][2].

본 논문은 32-bit 명령어를 사용하는 Default 모드와 32-bit 명령어와 16-bit 명령어를 같이 사용하는 Light 모드, 32-bit 명령어와 24-bit 명령어, 16-bit 명령어를 같이 사용하는 Ultra light 모드를 갖는 X32V ISA(Instruction Set Architecture)를 제안하였으며, 이를 기반으로 가변 길이의 명령어 모드를 갖는 32-bit 5-stage pipeline RISC 마이크로프로세서를 설계하였다. 제안된 X32V ISA와 마이크로프로세서를 검증하기 위하여 X32V 전용 시뮬레이터로 멀티미디어 어플리케이션 프로그램의 프로그램 코드 사이즈를 산출하여 비교하였으며[5], Xilinx FPGA를 통해 X32V ISA의 모든 명령어 수행을 검증하였다. 본 논문의 구성은 서론에 이어 II장에서는 제안된 X32V ISA에 대하여 기술하였고, III장에서는 운영모드와 예외상황 처리 방법에 대해 기술하였으며, IV장에서는 가변 길이 명령어를 처리하는 명령어 버퍼 및 decompressor에 대해 기술하였다. IV장에서는 설계된 마이크로프로세서의 데이터 패스 설계에 대해 기술하였고 V장에서는 각 모드의 프로그램 코드 사이즈 비교와 회로 검증에 대해 기술하였으며, VI장에서 결론을 맺었다.

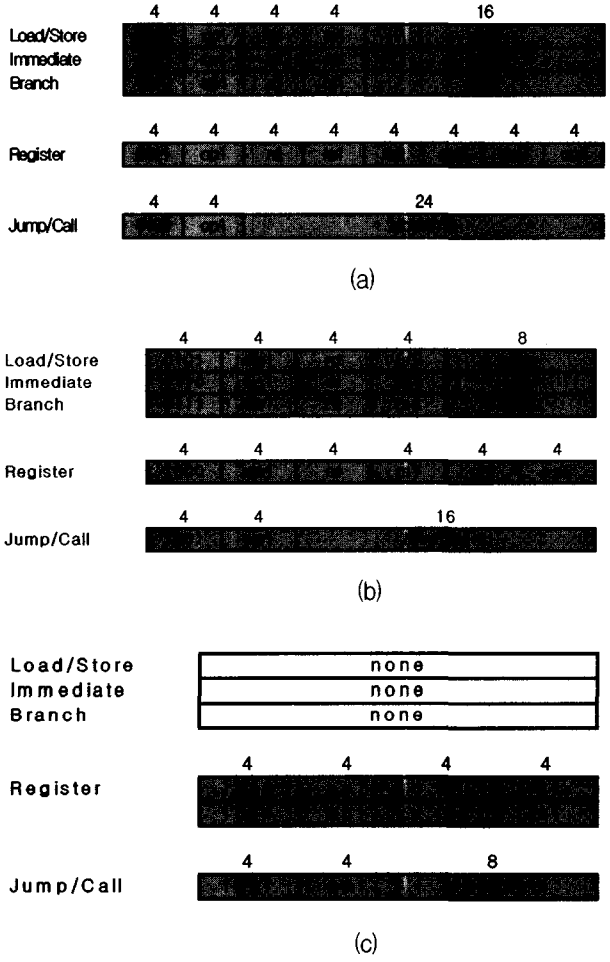


그림 1. 제안된 X32V ISA 형식: (a) 32-bit 명령어 형식; (b) 24-bit 명령어 형식; (c) 16-bit 명령어 형식;
Fig. 1. The proposed X32V ISA format.: (a) 32-bit instruction format; (b) 24-bit instruction format; (c) 16-bit instruction format;

II. 본 론

1. 가변길이 모드 명령어 셋 제안

새롭게 제안된 명령어 셋 X32V ISA는 RISC 형식의 명령어로 32-bit, 24-bit, 16-bit 포맷이 있으며 Load/Store, Immediate, Branch, Register, Jump/Call 명령어로 크게 5가지로 분류된다. 그림 1은 X32V 명령어 형식으로 최상위 4-bit은 명령어 타입 필드로 명령어 타입과 명령어 길이를 구분하는 역할을 한다. 그림 1a, 1b는 32 bit 명령어 포맷과 24-bit 명령어 포맷으로 목적 레지스터와 소스 레지스터 2개를 표현하는 3-address 형식으로 되어있다. 그림 1c는 16 bit 명령어 포맷으로 목적 레지스터와 소스 레지스터의 1개를 표현

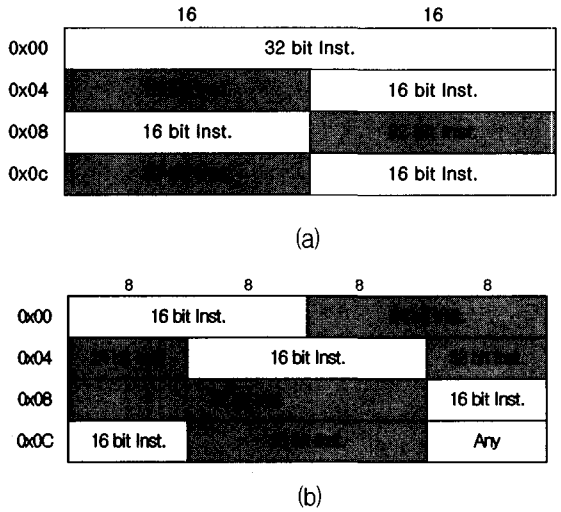


그림 2. 명령어 메모리 구성: (a) Light mode; (b) Ultra light mode;
Fig. 2. Instruction memory composition.: (a) Light mode; (b) Ultra light mode;

하는 2-address 형식으로 되어있으며 5개의 타입 중 Register 타입과 Jump/Call 타입이 지원된다.

제안된 X32V ISA는 명령어 실행 시 3개의 모드를 갖고 있으며, 32-bit 명령어를 사용하는 Default 모드와 32-bit 명령어와 16-bit 명령어를 같이 사용하는 Light 모드, 32-bit 명령어와 24-bit 명령어, 16-bit 명령어를 같이 사용하는 Ultra light 모드로 구분되어 진다. 이는 24-bit, 16-bit로 압축되어있는 명령어를 사용하여 기능이나 성능의 감소 없이 프로그램 코드의 크기를 줄이기 위함이다. 그림 2는 Light 모드와 Ultra light 모드의 명령어 메모리 구성을 나타낸다. 24-bit 명령어와 16-bit 명령어는 32-bit 명령어에 비해 메모리 접근 범위나 레지스터 오퍼랜드의 활용에서 성능이 뒤떨어지나, 좁은 범위의 메모리 접근이나 레지스터의 누적 연산이 요구되는 명령어 실행 시 사용되면 기능이나 성능의 감소가 없이 명령어 코드 밀도를 높일 수 있다[3][4][5].

제안된 X32V 명령어는 Signed/Unsigned 산술, 비교, 곱셈 명령어와 논리, 쉬프트 명령어 등 28개의 데이터 처리 명령어를 갖고 있으며, 다양한 조건 분기 명령어와 무조건 분기 명령어, procedure call과 return 명령어 등의 제어 명령어 13개를 갖고 있다. 메모리 전송 명령어 8개로 Signed/Unsigned 형태로 byte, half word, word 단위의 데이터를 데이터 메모리에 전송이 가능하다. 표 1은 X32V 명령어를 기능별로 분류하였으며, 16-bit 명령어 형식의 Load/Store, Branch, Immediate 타입을 제외하고 32-bit, 24-bit, 16-bit 명령어 포맷에서 모두 사용이 가능하다.

표 1. X32V ISA 명령어 분류
Table 1. X32V ISA instruction group.

명령어 기능	명령어 이름	명령어 타입
메모리 전송	L_W, L_B, L_BU, L_H, L_HU, S_W, S_B, S_H	Load/Store
데이터 처리	LG_AND, LG_OR, LG_XOR, ADD, ADD_U, SUB, SUB_U, SFT_LL, SFT_RL, SFT_RA, SET_LT, SET_LTU, MUL, MUL_U, MOV	Register
	LG_ANDi, LG_ORi, LG_XORi, ADDi, ADD_Ui, SUBi, SUB_Ui, SFT_LLi, SFT_RLi, SFT_RAi, SET_LTi, SET_LTUi, MOV_UPi	Immediate
제어	B_EQ, B_NE, B_EQZ, B_NEZ, B_LTZ, B_GTZ, B_LTEZ, B_GTEZ	Branch
	J_AR, J_PR	Register
	J_A, J_P, J_R	Jump/Call

2. 가변 길이 명령어 처리

제안된 X32V ISA는 Default 모드, Light 모드, Ultra light 모드로 3가지의 명령어 수행 모드를 갖는다. Light 모드와 Ultra light 모드는 32bit, 24bit, 16bit 명령어를 혼합 사용함으로 이를 처리하기위한 과정이 명령어 인출 단계에서 필요하다. 그림 3은 명령어 인출 단계의 가변 길이 명령어 처리 유닛을 나타내며, 명령어 파이프라인 버퍼와 명령어 변환기로 구성된다.

64-bit 쉬프트 레지스터로 구성된 명령어 파이프라인 버퍼는 명령어 메모리로부터 32-bit 워드 단위로 명령어를 전달받으며, 최상위 4-bit을 통해 명령어의 길이를 분석하여 매 클럭 사이클마다 분석된 명령어의 길이만큼 명령어 변환기로 전달한다. 명령어 파이프라인 버퍼는 버퍼의 여유 bit을 확인하여 명령어의 흐름을 순차적으로 유지하기위해 PC 레지스터를 제어함으로서 명령어 메모리의 주소 전달을 조절한다. Light 모드와 Ultra light 모드는 명령어의 명령어 메모리 구성 위치가 word boundary를 벗어날 수 있으며, 분기명령어 실행 시 분기 target 명령어가 word boundary를 벗어나게 될 경우 1 사이클의 수행 페널티를 갖는다. 명령어 변환기는 명령어 파이프라인 버퍼로부터 명령어를 전달받아 16-bit, 24-bit 모든 명령어를 해당하는 32-bit 명령어로 변환시켜 명령어 디코드 단계로 명령어를 전달

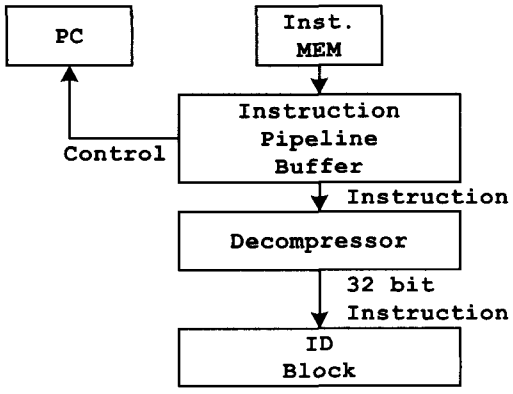


그림 3. 명령어 인출 단계의 구성
Fig. 3. The block of instruction fetch.

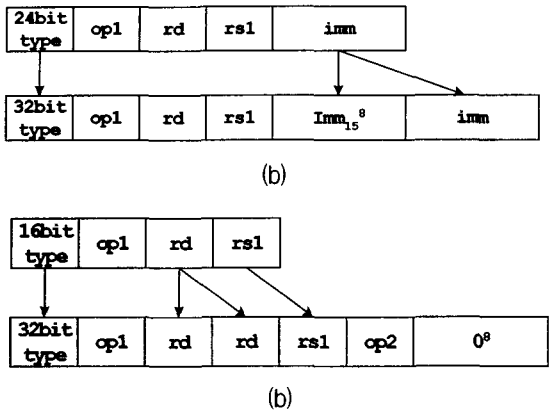


그림 4. 24-bit, 16-bit 명령어의 32-bit 명령어 변환: (a) 24-bit 명령어 변환; (b) 16-bit 명령어 변환;
Fig. 4. 32-bit instruction conversion of 24-bit, 16-bit instruction.: (a) 24-bit instruction conversion; (b)16-bit instruction conversion;

한다. 그림 4a는 24-bit 명령어를 수행동작이 같은 32-bit 명령어로 변환하는 과정을 나타낸다. 레지스터 타입의 24-bit 명령어는 하위 8 bit에 0으로 확장함으로서 32-bit 명령어로 변환하며, 레지스터 타입을 제외한 명령어 타입은 하위의 즉시 값(immediate value)을 signed/unsigned 확장하여 32-bit 명령어로 변환한다. 그림 4b는 16-bit 명령어를 수행동작이 같은 32-bit 명령어로 변환하는 과정을 나타낸다. 16-bit 명령어는 32, 24-bit 명령어와 다르게 2-address 형식이므로, 목적 레지스터 필드가 첫 번째 소스 레지스터 필드로 확장된다.

3. 데이터 패스 설계

본 논문이 제안한 X32V ISA를 기반으로 설계한 마이크로프로세서는 5-stage 파이프라인 구조로 IF(Inst -ruction Fetch), ID(Instruction Decode), EX(Exe cuti -on), MEM(Memory), WB(Write Back) 단계로 구성된다. IF 단계는 파이프라인으로 동작하는 마이크로프로세서를 위해 매 사이클마다 명령어 메모리로부터 명령어를 인출하며, 명령어의 주소를 저장하는 PC 레지스터와 명령어 메모리, 순차적인 word 메모리 주소를 발생하는 명령어 메모리 주소 발생기로 구성되어있다[7]. ID 단계는 명령어를 해석하여 제어신호를 생성하고 분기 주소 계산, 분기 조건 판단, 레지스터 파일로부터 오퍼랜드(operand) 인출 동작을 하며, 명령어의 즉시 값을 32-bit 데이터로 Signed /Unsigned 확장하는 명령어 필드 확장 유닛과 분기 주소를 계산하는 32-bit BCLA (Block Carry Look-ahead Adder), 16개의 32-bit 레지

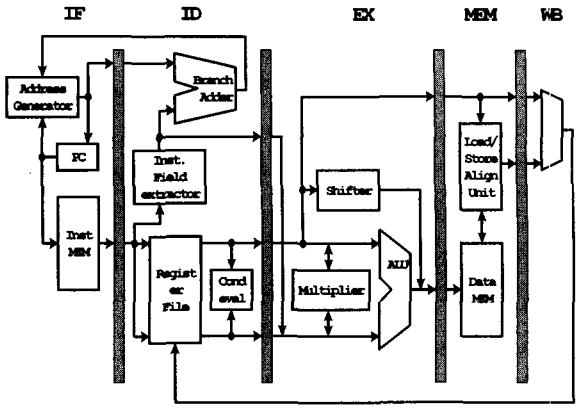


그림 5. X32V 마이크로프로세서의 데이터 패스
Fig. 5. The data-path of the X32V microprocessor.

스터로 구성된 레지스터 파일, 조건 분기문의 조건을 판단하는 조건 판단 회로로 데이터 패스가 구성된다. EX 단계는 명령어 실행 시 연산 동작을 수행하며, 논리, 산술 쉬프트 연산을 하는 배럴 쉬프터와 논리, 산술 연산을 수행하는 ALU(Arithmetic Logic Unit), Signed/Unsigned 곱셈을 실행하는 곱셈기로 데이터 패스가 구성된다. MEM 단계는 데이터 메모리의 데이터 입출력을 수행하며, 데이터 메모리와 word(32- bit), half word(16-bit), byte(8-bit) 단위로 메모리와 데이터를 주고받는 정렬 회로로 구성된다. WB 단계는 EX 단계의 연산 결과 데이터와 메모리 데이터를 선택하여 레지스터 파일에 저장하며, 멀티플렉서로 구성된다[8]. 그림 5는 X32V 마이크로프로세서의 5-stage 파이프라인 데이터 패스 구성을 나타낸다.

가. ALU의 설계

마이크로프로세서의 핵심적인 역할을 수행하는 ALU는 논리, 산술 연산을 담당하는 EX단계의 유닛으로 데이터 연산 명령어의 실제 연산 동작을 수행하며, 메모리 전송 명령어 수행 시 데이터 메모리의 주소를 계산한다. ALU의 동작 속도, 소모 전력과 면적은 마이크로프로세서의 속도, 면적, 소모 전력 등 성능에 크게 영향을 미친다[6]. 본 논문이 제안한 X32V 마이크로프로세서의 ALU는 타 덧셈기보다 속도, 면적, 소모 전력에서 성능이[7] 뛰어난 BCLA를 기반으로 설계하였다.

그림 6은 ALU의 내부 구성도로 제어 신호를 통해 입력 발생 회로와 출력 발생 회로를 제어함으로서 표 2와 같은 기능을 수행한다. GP발생기는 입력 발생 회로로부터 데이터를 전달받아 G와 P를 발생하며[8], 여러 블록으로 구성된 캐리(Carry) 예측기로 전달된다. 캐리

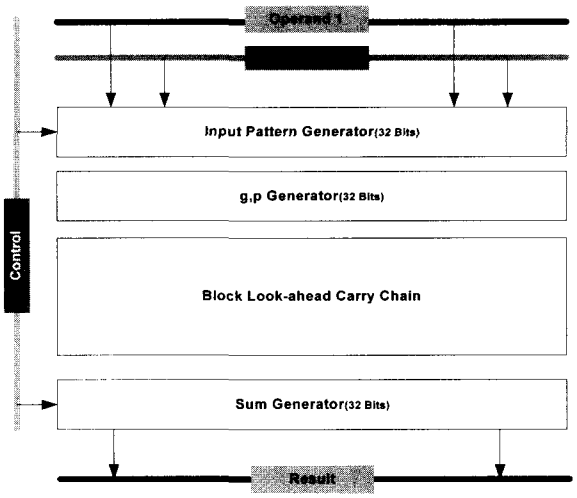


그림 6. ALU의 내부 구성
Fig. 6. The block diagram of ALU.

표 2. ALU의 기능
Table 2. The function of ALU.

명령어 성격	ALU 기능
Logical Operation	AND, OR, XOR, INV
Arithmetic Operation	ADD, ZERO, SUB, CMP
Transfer Operation	MOV

예측기는 예측된 캐리를 합 발생기로 전달하며, 합 발생기는 이를 기반으로 연산의 결과를 발생한다.

나. 곱셈기의 설계

타 내장형 마이크로프로세서에서 곱셈 연산을 ALU 반복 방식으로 매우 많은 수행 사이클을 소모하며 처리하는 경우가 있다. 이는 작은 면적이 요구되는 내장형 마이크로프로세서는 큰 면적을 갖는 고속 32-bit 전용 곱셈기의 하드웨어 부담이 크기 때문이다[2][9]. 제안하는 X32V 마이크로프로세서의 곱셈기는 작은 면적을 갖으면서 적은 수행 사이클을 갖는다. 제안된 곱셈기는 한 사이클 당 32 * 8 곱셈 연산을 하며, 최대 4번 반복 수행을 한다. 이는 32 * 32 곱셈기에 비해 적은 부분곱 발생으로 인해 1/4에 해당하는 면적을 갖는다. 표 3은 곱셈기의 수행 사이클 수로 두 번째 오퍼랜드 상태에 따라 2회에서 5회로 가변적이다. 곱셈기의 곱셈 연산 수행 과정은 두 개의 오퍼랜드를 곱셈하여 합 벡터와 캐리 벡터를 발생하는 곱셈 사이클과 곱셈기에서 전달된 합 벡터와 캐리 벡터를 덧셈 연산하여 최종 결과를 MEM 단계에 전달하는 ALU 사이클로 나뉜다.

그림 7은 곱셈기의 내부 구성으로 radix-4 booth 알

표 3. 곱셈기의 수행 사이클 수
Table 3. The number of multiplier execution cycles.

오퍼랜드2의 상태	다중 수행 사이클 회수
[31:8] = 1 또는 0	1(Multiplier cycle)+1(ALU cycle)
[31:16] = 1 또는 0	2(Multiplier cycle)+1(ALU cycle)
[31:24] = 1 또는 0	3(Multiplier cycle)+1(ALU cycle)
Others	4(Multiplier cycle)+1(ALU cycle)

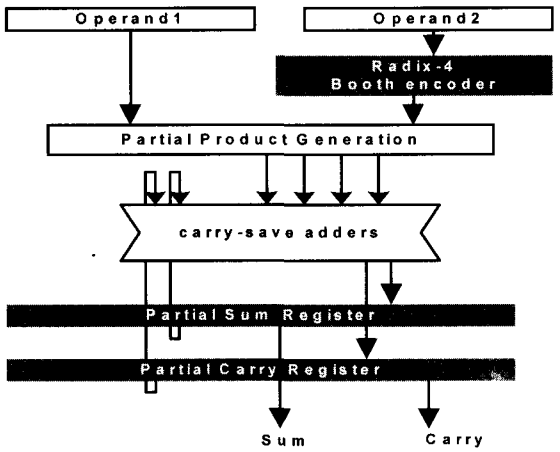


그림 7. 곱셈기의 내부 구성
Fig. 7. The block diagram of a multiplier.

고리듬 적용하여 한 사이클 당 32 * 8 곱셈 연산을 수행한다. 곱셈기는 다중 사이클 수행 시 합 레지스터와 캐리 레지스터에 저장된 이전 곱셈 결과 값을 CSA(Carry Save Adder)로 다시 전송하여 현재의 부분곱과 덧셈 연산한다. 곱셈기의 출력은 carry save 형식으로 64-bit 합 벡터와 64-bit 캐리 벡터로 ALU에 전달하며, ALU는 2개의 곱셈의 결과를 덧셈 연산하여 MEM 단계로 전달한다.

다. 쉬프터의 설계

X32V 마이크로프로세서의 쉬프터는 ALU와 직렬로 연결되어 있는 ARM 마이크로프로세서의 쉬프터와 달리 ALU와 병렬로 연결되어있다. 쉬프터와 ALU가 직렬로 연결되어 있는 경우 쉬프트 연산의 결과를 ALU가 전달 받아야 함으로 쉬프터는 매우 고속으로 동작해야 한다. 이로 인해 ARM 마이크로프로세서의 쉬프터는 최대 지연이 3상태 버퍼 1개인 32by32 크로스바 스위치 구성되어있다[2]. 32by32 크로스바 스위치는 고속으로 동작하나 하드웨어 부담이 매우 크다. ALU와 병렬로 연결되어있는 X32V 마이크로프로세서의 쉬프터

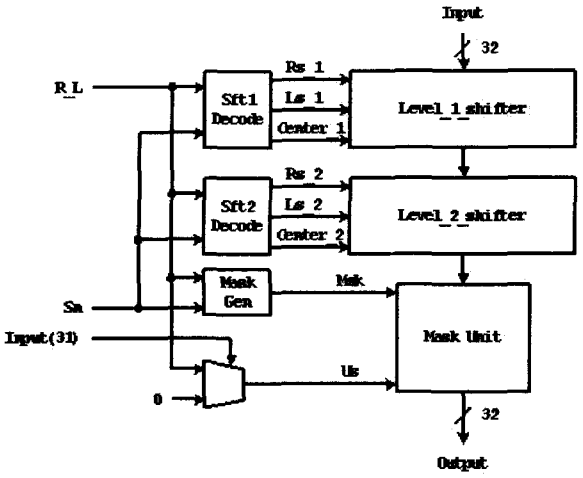


그림 8. 쉬프터의 내부 구성
Fig. 8. The block diagram of a shifter.

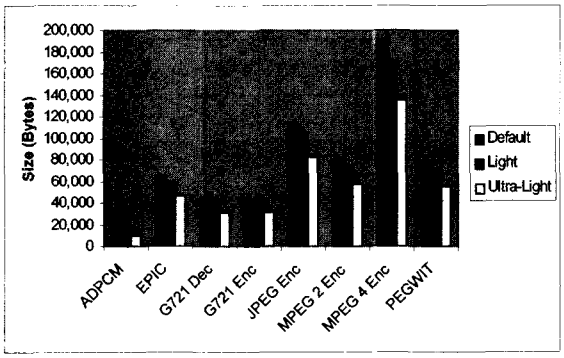


그림 9. 벤치마크 프로그램들의 코드 사이즈
Fig. 9. The code size of benchmark programs.

는 ARM 마이크로프로세서의 쉬프터와 같이 큰 하드웨어 부담을 갖는 고속의 32by32 크로스바 스위치 구조는 적당하지 않다. 제안된 X32V 마이크로프로세서의 쉬프터는 최대 지연이 3상태 버퍼 1개인 8-bit 4by4 크로스바 스위치와 최대 지연이 3상태 버퍼 1개인 좌우 8-bit 범위로 1-bit 단위로 쉬프트 연산을 하는 쉬프터로 2단계 구성이 되어있다. 이는 1단계로 구성된 ARM의 32by32 크로스바 스위치구조 쉬프터에 비해 3상태 버퍼 1개의 지연 시간을 갖으나, 하드웨어 부담은 절반으로 감소된다. 제안된 쉬프터는 ID 단계에서 전달된 두 번째 오퍼랜드의 하위 5-bit을 쉬프트 양으로 전달받아 첫 번째 오퍼랜드를 논리, 산술 쉬프트 연산한다. 그림 8은 배럴 쉬프터의 내부 구성도로 2단계의 쉬프트 유닛으로 구성된다. 1단계 쉬프트 유닛은 8-bit 4by4 Crossbar switch로 구성되어 있으며, 쉬프트 양과 좌우 제어 신호를 입력으로 하는 sht1_decoder에 의해 제어된다. 2단계 쉬프트 유닛은 좌우 8-bit 범위로 1-bit 단위로 쉬프트 연산 동작을 수행하며, sht2_decoder에 의해 제어된다. 2단계의 쉬프트 유닛은 논리 및 산술 쉬프트에서 요구되는 0또는 부호 확장이 지원되지 않으므로, 외부에서 이를 제어하는 마스크 발생기와 마스크 유닛이 추가되어 있다.

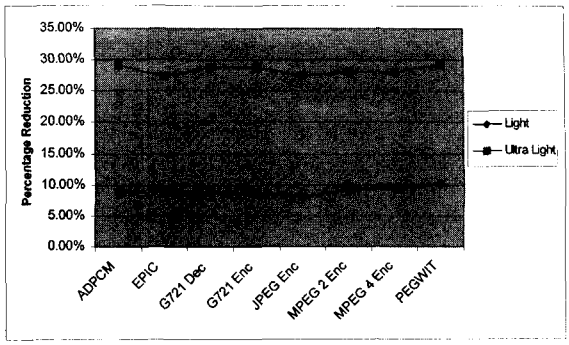


그림 10. 코드 사이즈 감소율
Fig. 10. A reduction rate of code size.

III. 비교 및 검증

1. 프로그램 코드 사이즈 비교

X32V 전용 시뮬레이터를 이용하여 JPEG, MPEG과 같은 멀티미디어 어플리케이션 프로그램의 코드 사이즈를 Default, Light, Ultra light 모드 별로 산출하여 비교하였다. 그림 9는 벤치마크 프로그램의 코드사이즈를 나타내며, Default, Light, Ultra Light 모드에서의 명령어 메모리 코드크기와 데이터 메모리 코드크기를 더해 계산하였다.

Light, Ultra light 모드는 Default 모드에 비해 모든 벤치마크 프로그램에서 코드사이즈 감소를 확인하였으며, 그림 10은 Default 모드에 대한 Light, Ultra light 모드의 코드사이즈 감소율을 나타낸다. Light 모드는 모든 벤치마크에서 최소 8%가 감소하고, Ultra Light 모드는 최소 27%가 감소한다.

Word boundary를 벗어나는 명령어로 분기하는 분기 명령어로 인한 수행 사이클 오버헤드는 CPI(Cycle Per Instruction)를 계산하여 비교하였다. 그림 11은 Default 모드에 대한 Light, Ultra Light 모드의 수행 사이클 오버헤드를 나타내며, Light 와 Ultra Light 모드는 Default 모드와 비교하여 약 3%의 사이클 오버헤드를 가진다.

2. X32V 마이크로프로세서의 검증

제안된 X32V 마이크로프로세서는 PCI 버스 타입 Xilinx FPGA 보드인 iProve(XCV1000E-iPROVE)를 이용하여 33MHz 동작 환경에서 수행 검증을 하였으며,

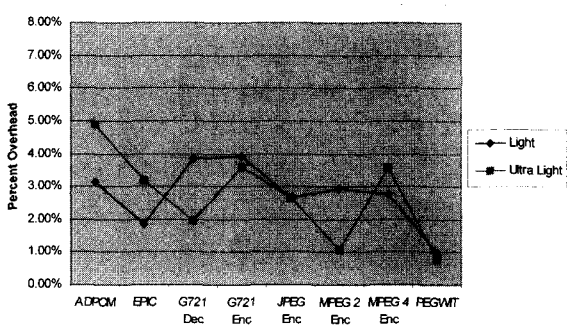


그림 11. 수행 사이클 오버헤드 비율
Fig. 11. A overhead rate of execution cycle.

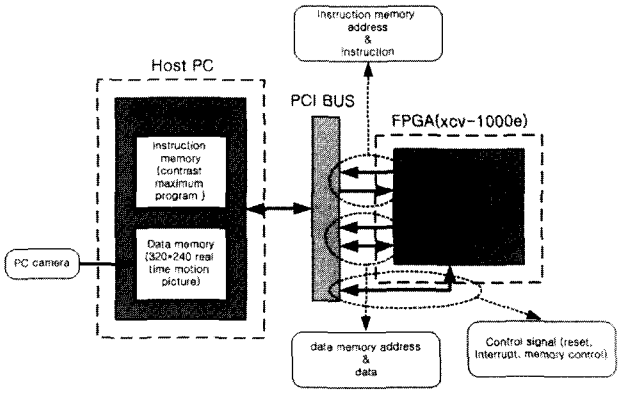


그림 12. X32V 마이크로프로세서 검증 환경
Fig. 12. A environment of X32V microprocessor verification.

100만 게이트 FPGA에 구현된 X32V 마이크로프로세서는 2069개의 slice(16%), 1079개의 register(4%), 3682개의 LUT(14%)로 구성되어있다. 그림 12는 X32V 마이크로프로세서의 검증 환경으로 X32V 마이크로프로세서는 PCI 버스를 통해 호스트 PC의 가상 명령어 메모리로부터 채도 극대화 프로그램의 명령어들을 전달 받으며, 호스트 PC의 가상 데이터 메모리로부터 실시간 영상 데이터를 전달받거나, 채도 극대화 처리된 영상 데이터를 메모리로 전달한다.

PC 카메라로부터 전달되는 실시간 영상은 가로 320 픽셀, 세로 240 픽셀로 이루어져 있으며, 각 픽셀을 R, G, B가 각각 8-bit으로 구성되어있다. 그림 13은 X32V 검증 환경의 GUI(Graphic User Interface)로 왼쪽 위 부분이 PC 카메라로부터 받은 영상이며, 오른쪽 위부분이 X32V 마이크로프로세서가 채도 극대화 알고리듬을 수행하여 출력하는 영상이다. 아래쪽은 X32V 마이크로프로세서가 수행하는 채도 극대화 알고리듬의 X32V 명령어들이다. X32V 마이크로프로세서는 320*240 실시간 영상의 채도 극대화 연산을 초당 10~13 프레임 속도로 처리함을 확인하였다.

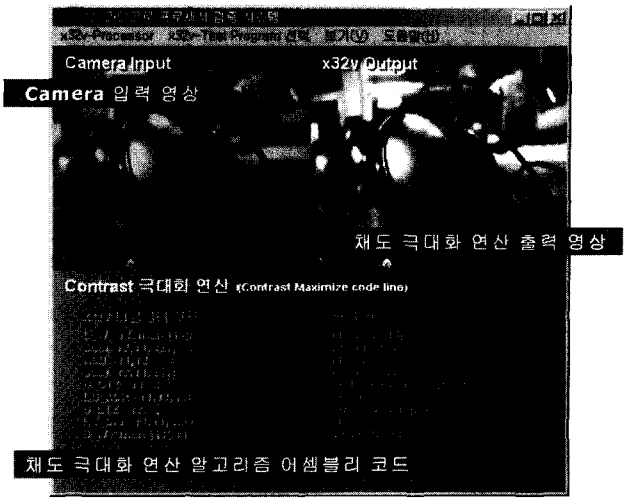


그림 13. X32V 마이크로프로세서 검증 시스템의 GUI
Fig. 13. X32V microprocessor verification system GUI.

Ⅳ. 결 론

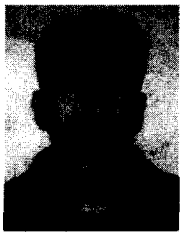
본 논문은 프로그램 코드 사이즈 감소를 위해 32-bit, 24-bit, 16-bit 명령어 포맷을 혼합 사용하여 3가지 명령어 모드를 갖는 X32V ISA를 제안하였으며, 이를 기반으로 32-bit 5-stage pipeline RISC 마이크로프로세서를 설계하였다. X32V ISA 검증을 위해 벤치마크 프로그램의 명령어 메모리 코드 크기를 3가지 모드 별로 산출하여 비교하였으며 그 결과 Light 모드, Ultra light 모드는 Default 모드에 비해 각각 최소 8%, 27% 코드 크기가 감소하는 것을 확인하였다. X32V ISA를 기반으로 설계된 X32V 마이크로프로세서는 Xilinx FPGA를 이용하여 33Mhz 동작환경에서 실시간 영상 처리 수행을 검증하였다.

참 고 문 헌

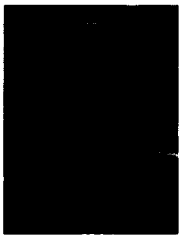
[1] David A. Patterson and John L. *Hennessy, Computer Organization & Design*, Morgan Kauf mann Publishers, pp. 449-509, 1998.
[2] Steve Furber, *ARM system on chip architecture*, Addison Wesley, pp. 74-100, 1998.
[3] Arvind Krishnawamy and Rainv Gupta, "Mixed width instruction sets,"*Communication of ACM (CAMC)*, Vol. 48, Num. 8, pp. 47-52, Aug. 2003.
[4] Jeremy Lau, Stefan Schoenmackers, Timothy Sherwood and Brad Calder, "Reducing Code Size With Echo Instructions," CASES, pp. 1-11, San Jose, USA, Nov. 2003.
[5] Simon segars, Keith calrke and Liam Coudge, "Embedded Control Problems, Thumb and the

ARM7TDMI," *IEEE Micro*, pp. 23-30, Oct. 2003.
[6] Israel Koren, *Computer Arithmetic Algorithms*, John Wiley & Sons, pp. 51-56, 1978.
[7] Jan M. Rabaey and Massoud Pedram, *LOW POWER DESIGN METHODOLOGIES*, Kluwer Academic Publishers, pp. 170-186, 1996.
[8] B. Parhami, *Computer Arithmetic Algorithms and Hardware Design*, Oxford University Press, pp. 143-166, 2000.
[9] Jan M. Rabaey and Massoud Pedram, *LOW POWER DESIGN METHODOLOGIES*, Kluwer Academic Publishers, pp. 170-186, 1996.

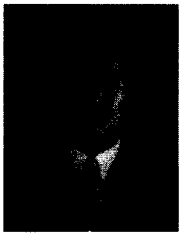
저 자 소 개



박 기 현(정회원)
2003년 서경대학교 컴퓨터공학과 학사 졸업.
2003년~현재 서경대학교 컴퓨터공학과 석사 과정.
<주관심분야: 내장형 마이크로프로세서, 암호화 프로세서>

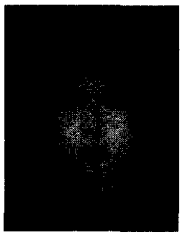


이 광 엽(정회원)
1985년 서강대학교 전자공학과 학사.
1987년 연세대학교 전자공학과 석사.
1994년 연세대학교 전자공학과 박사.
1989년~1995년 현대전자 선임연구원,
1995년~현재 서경대학교 컴퓨터공학과 조교수.
<주관심분야: 마이크로프로세서, 암호프로세서>

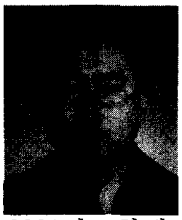


김 영 수(정회원)
1995년 인하대학교 전자공학과 학사 졸업.
1997년 인하대학교 대학원 전자공학과 석사 졸업
1997년~2001년 삼성전자 반도체 SYSTEM LSI 본부 근무
2001년~현재 한국전자통신연구원 시스템IC 설계팀 근무
<주관심분야: 설계자동화, 멀티미디어 설계>

조 한 진(정회원)
1982년 한양대학교 전자공학과 졸업.
1987년 New Jersey Institute of Technology 전기공학과 공학석사.
1992년 University of Florida 전기공학과 공학박사.
1992년~현재 한국전자통신연구원 시스템IC 설계팀 팀장
<주관심분야: 통신, 컴퓨터, 신호처리, 반도체>



오 민 석(정회원)
2003년 서경대학교 컴퓨터공학과 학사 졸업
2003년~현재 서경대학교 컴퓨터공학과 석사 과정.
<주관심분야: 저전력 마이크로프로세서, Computer arithmetic>



한 진 호(정회원)
1998년 한국과학기술원 전기 및 전자공학과 학사 졸업
2001년 한국과학기술원 전자전산학과 전기 및 전자공학 석사 졸업
2001년~현재 한국전자통신연구원 기반기술연구소 고성능SoC연구부 시스템IC 설계팀 근무
<주관심분야: SoC Platform, Configurable Processor Design, Embedded Processor Design, Low Power Circuit>

배 영 환(정회원)
1985년 한양대학교 전자공학과 학사 졸업
1987년 한양대학원 전자공학과 석사 졸업
1987년~현재 한국전자통신연구원 반도체원천기술연구소 책임연구원
<주관심분야: VLSI CAD, 하드웨어/소프트웨어 통합설계,Embedded System설계>