논문 2008-45CI-2-6

비트열 처리를 위한 저비용 명령어 세트

(A Low Cost Instruction Set for Bit Stream Process)

함 동 현*, 이 형 표*, 이 용 석**

(Donghyeon Ham, Hyoung Pyo Lee, and Yong Surk Lee)

요 약

대부분의 미디어 압축 코덱에는 가변 길이 부호 기법이 적용된다. 본 논문에서는 이러한 가변 길이 부호의 복호 과정을 가속하기 위해 비트열 처리 전용 레지스터와 이를 이용하는 비트열 처리 전용 명령어 세트를 추가하는 방법을 제안한다. 본 논문에서 제안하는 명령어 세트는 프로세서에 기본적으로 존재하는 데이터 패스를 최대한 활용하고 비트열 정보를 비트열 입력 포트 대신 메모리에서 읽어온다. 따라서 제안하는 명령어 세트는 프로세서의 변형을 최소화하고 추가적인 입력 제어기와 버퍼 없이 범용 프로세서에 적용하여 가변 길이 부호의 복호과정을 가속할 수 있다. 제안하는 명령어 세트의 데이터 패스를 TSMC 0.25㎛ 라이브러리를 이용하여 합성한 결과, 65 비트의 메모리와 344 게이트가 필요하였으며 0.19 ns의 추가적인 지연 시간이 있었다. 제안하는 명령어 세트는 H.264/AVC의 가변 길이 부호의 복호 수행 시간을 약 55 % 감소시켰다.

Abstract

Most of media compression CODECs adopts the variable length coding method. This paper proposes special registers and instruction set for bit stream process in order to accelerate the decoding process of the variable length code. The instruction set shares the conventional data path to minimize additional costs. And bit stream is read from the memory instead of the special port. Therefore the instruction set minimizes the change of the processor, and is adopted without any additional input controller and buffer, and accelerate decoding process of variable length code. The data path of the instruction set needs additional 65 bits memory and 344 equivalent gates, 0.19 ns delay under TSMC 0.25㎛ technology. The instruction set reduced the execution time of the variable length code decoding process in H.264/AVC by about 55%.

Keywords: 비트열 프로세서, 명령어 세트, ASIP, 가변 길이 부호

Ⅰ. 서 론

정보기술의 발달로 인해서 정보의 양은 매우 커졌으며 이와 같은 정보를 이용하는 멀티미디어 기기에 있어서 압축 기술은 꼭 필요한 기술이 되었다. 정보를 압축하기 위한 기술은 오랜 기간 동안 연구되어 왔다. 그 중 가변 길이 부호화는 바이트 단위의 고정된 비트 길이로 정보를 나누어서 저장하던 것을 각각의 정보를 다른 길이의 비트열로 표현함으로써 전체 비트열의 길이를 줄인다. 가변 길이 부호화는 정지화상과 동영상, 음향, 일반 데이터의 압축 등, 대부분의 압축 기술에 적용 된다. 최근 사용되고 있는 동영상 압축 코덱 중 가장 좋은 효율을 보이고 있는 H.264/AVC 에서도 문맥 기반 적응적 가변 길이 부호화(CAVLC, Context-Adaptive Variable Length Code)라는 좀 더 발전된 가변 길이 부호화 방식을 적용하고 있다[1].

가변 길이 부호화를 통한 압축은 복호 과정에서 부호 길이의 다양성 때문에 기존의 바이트 단위의 데이터 처리를 위한 프로세서에서 데이터를 처리할 경우 많은 연산을 요구한다. 이러한 문제를 해결하기 위해서 가변 길이 부호를 복호하는 하드웨어를 통한 가속이 연구되었다[2~5]. 하드웨어를 통한 가속은 하드웨어 추가에 따

* 학생회원, ** 평생회원, 연세대학교 전기전자공학과
(Department of Electrical and Electronic Engineering, Yonsei University)

※ 본 연구는 한국과학기술재단 특정기초연구(No. R01-2006-000-10156-0) 지원으로 수행되었으며, IDEC (IC Design Education Center)에 의해 지원되는 EDA 툴이 사용되었습니다.
접수일자: 2008년1월11일, 수정완료일: 2008년2월29일

른 비용 부담을 가지며 사용자의 요구에 따라 다양하게 프로그램 할 수 없기 때문에 여러 형태의 가변 길이 부호를 가속할 수 없는 한계를 갖는다. 이러한 단점들을 보완하기 위해 비트열 처리에 특화된 비트열 프로세서나 ASIP (Application Specific Instruction-Set Processor)이 연구 되었다[6~8]. 비트열 처리에 특화된 프로세서들은 비트열 처리 전용 명령어 세트를 가지고 있으며 비트열 정보를 입력 받는 외부 포트를 추가로 가지고 있다.

기존의 비트열 처리 전용 명령어 세트들은 하드웨어를 통한 비트열 처리 부분을 그대로 프로세서의 데이터 패스에 추가하여 적용하였다. 비트열 정보는 비트열 정보를 위한 외부 포트를 통해 읽어오기 때문에 추가적인 입력 제어 장치와 입력 버퍼가 필요하며 또한 입력 포트도 추가되어야 한다. 따라서 기존의 비트열 처리 전용 명령어는 적용하는데 추가적인 비용이 크게 발생하며 다른 프로세서에 적용하는데 어려움이 있다. 본 논문에서는 새로운 비트열 처리 전용 명령어 세트를 제안하고 제안하는 명령어 세트를 통해 비트열 처리 가속을 위해 추가적으로 발생하는 비용을 최소화한다. 제안하는 명령어는 프로세서에 기본적으로 존재하는 데이터 패스를 활용하였으며 구현에 필요한 하드웨어 오버헤드가 작기 때문에 다른 프로세서에 쉽게 적용할 수 있다.

본 논문의 나머지 부분은 다음과 같이 구성되어 있다. 본론 1 절에서는 기존의 하드웨어와 비트열 프로세서에서의 비트열 처리 부분을 분석하고 본론 2 절과 3 절에서는 새로운 비트열 처리 전용 명령어 세트를 제안한다. 실험에서는 제안하는 명령어의 데이터 패스 부분의 합성을 통해 추가적인 비용을 살펴보고 MIPS 기반의 간결한 RISC에 적용하여 H.264/AVC CAVLC의 복호 과정을 시뮬레이션하고 명령어의 적용 전과 후의 성능을 비교하고 분석한다. 그리고 마지막으로 결론을 맺는다.

II. 본 론

1. 기존 비트열 처리 방식

기존의 MPEG4를 위한 비트열 프로세서나 H.264/AVC를 위한 ASIP, 하드웨어를 통한 비트열 처리 부분은 비트열 처리를 위해 그림 1과 같은 모듈을 공통적으로 갖는다[2~8]. LSB(Least Significant Bit)는 최하위 비트를 나타내고 MSB(Most Significant Bit)는 최상위 비트를 나타낸다. 버퍼 레지스터 R1, R2에 현재

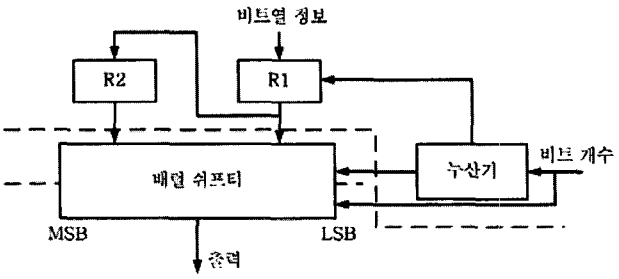


그림 1. 비트열 처리 모듈의 블록 다이어그램
Fig. 1. Block diagram of the bit stream process module.

위치의 비트열을 가지고 있으며 누산기는 버퍼 레지스터 안에서의 비트열의 현재 위치를 저장하고 있다. 배럴 쉬프터에서는 버퍼 레지스터의 값을 누산기에 저장되어 있는 위치 값만큼 최상위 비트 쪽으로 쉬프트하거나 누산기에 저장되어 있는 위치에서부터 원하는 비트 수만큼 최하위 비트 쪽으로 정렬한다. 일반적인 쉬프트 연산만을 목적으로 한 기존의 배럴 쉬프터를 통해서는 버퍼 레지스터의 값을 누산기에 저장되어 있는 위치 값만큼 최상위 비트 쪽으로 쉬프트 하는 것만 가능하다. 누산기에 저장되어 있는 위치 값으로부터 원하는 비트 수를 최하위 비트로 정렬(alignment)하기 위해서는 추가적인 하드웨어가 필요하다. 그림 2는 앞에서 설명한 비트열 정렬의 예를 보여준다. 비트열 처리 과정을 모두 하드웨어로 구현할 경우에는 비트열의 현재 위치를 최상위 비트 쪽으로 정렬하는 과정만으로 비트열 처리 과정을 진행할 수 있다. 그러나 비트열 프로세서나 H.264/AVC를 위한 ASIP의 경우 비트열 정보를 비교하고 계산하는 과정을 위해서 비트열의 현재 위치에서부터 일정 비트만큼의 비트를 최하위 비트로 정렬한 값이 필요하다. 따라서 기존의 비트열 프로세서나 H.264/AVC를 위한 ASIP은 특수 배럴 쉬프터를 적용해왔다. 기존의 전용 비트열 처리 명령어 세트들은 그림 1의 모듈을 데이터 패스에 그대로 적용하였기 때문에 비트열

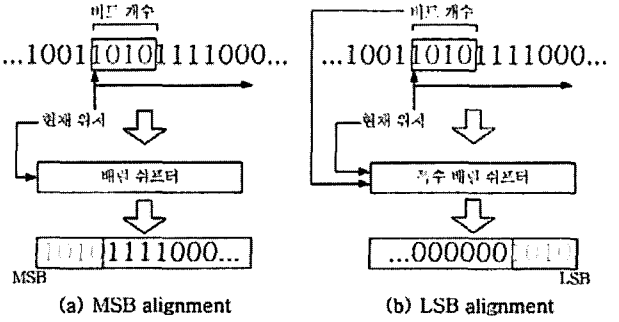


그림 2. 비트열 정렬의 예
Fig. 2. Example of the bit stream alignments.

을 입력 받는 추가적인 포트가 필요하며 비트열 입력을 위한 입력 제어기와 버퍼가 필요하다.

2. 제안하는 전용 명령어 세트

가. 비트열 전용 레지스터

본 논문에서 제안하는 명령어 세트는 비트열 처리 가속을 위해 비트 버퍼 레지스터와 비트 위치 레지스터를 사용한다. 비트 버퍼 레지스터는 현재 비트열 위치가 가리키는 곳의 32 비트의 비트열 정보를 저장하고 있다. 비트 버퍼 레지스터로 인하여 비트열 정보를 읽을 때마다 비트열 정보를 매번 로드하는 과정을 생략할 수 있다. 비트 위치 레지스터는 총 35비트의 비트열 위치 정보를 저장한다. 하위 4 비트는 비트 버퍼 레지스터의 상위 16 비트 사이의 현재 비트열 위치를 저장하고 상위 31 비트는 현재 비트열 위치보다 16 비트 뒤의 비트열의 메모리 위치를 저장한다.

나. 비트열 처리 전용 명령어

프로세서 내에 존재하는 배럴 쉬프터를 최대한 활용하여 추가적인 비용을 최소화 하기위해서 기존의 연구에서 한 번에 처리되던 비트열 처리 과정을 두 단계로 나누었다. 첫 번째 단계에서 비트 버퍼 레지스터에 저장되어 있는 비트열 정보를 비트열의 현재 위치가 최상위 비트가 되도록 정렬한다. 두 번째 단계에서는 정렬된 비트열 정보를 이용하여 원하는 비트 수 만큼의 상위 몇 비트를 최하위 비트 쪽으로 정렬하거나 최상위 비트에서부터 연속적으로 나오는 0의 개수를 산출한다. 그림 3은 앞에서 설명한 두 단계의 과정을 나타낸다.

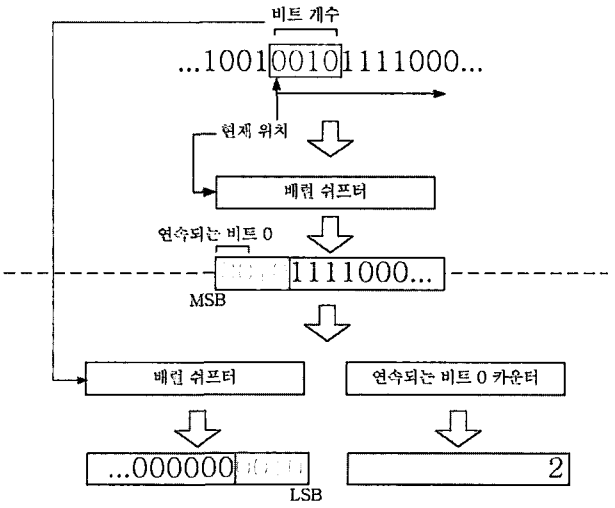


그림 3. 비트열 처리 과정의 예
Fig. 3. Example of the bit stream process.

첫 번째 단계는 비트 버퍼 레지스터의 값을 비트 위치 레지스터의 하위 4 비트의 값만큼 왼쪽 쉬프트해서 구현할 수 있다.

두 번째 단계에서 원하는 비트 수 만큼의 상위 몇 비트를 최하위 비트 쪽으로 정렬하는 과정은 마스크와 배럴 쉬프트의 왼쪽 회전(rotate) 과정만으로 구현할 수 있다. 첫 번째 과정에서 정렬된 비트열 정보의 하위 16 비트를 0으로 마스크하고 원하는 비트 수만큼 왼쪽 회전을 하고 그 결과의 상위 16비트를 0으로 마스크하면 원하는 비트만큼의 상위 비트들이 최하위 비트 쪽으로 정렬한 결과를 얻을 수 있다. 그림 4는 배럴 쉬프터를 사용하여 두 단계를 거쳐 비트열의 현재 위치에서부터 원하는 양의 비트 정보를 최하위 비트로 정렬하는 과정의 데이터 흐름을 보여준다.

비트열의 현재 위치에서부터 연속되는 비트 0의 개수를 산출하는 과정은 첫 번째 단계를 공유함으로써 추가 비용을 줄인다.

결과적으로 기본적인 배럴 쉬프터의 입력과 출력 부분에 마스크해 주는 부분, 최상위 비트에서부터 연속적으로 나오는 0의 개수를 산출하는 모듈이 추가된다.

비트열 처리를 위한 전용 명령어는 각 단계의 동작에 맞추어서 첫 번째 단계를 나타내는 명령어 1 개와 두 번째 단계의 두 가지 동작을 나타내는 명령어 2 개가 추가되어 총 3 개의 명령어가 필요하다. 여기에 오퍼랜드를 얻는 방식에 따라 명령어가 더 추가 될 수 있다.

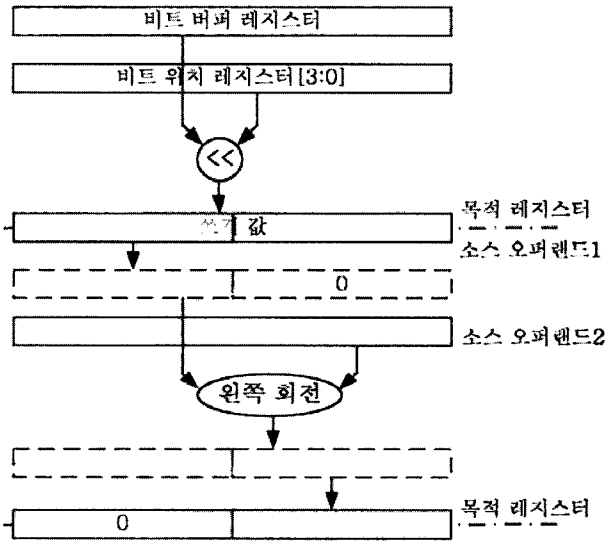


그림 4. 배럴 쉬프터를 사용하는 비트열 처리 과정의 데이터 흐름도
Fig. 4. Data flow of the bit stream process using barrel shifter.

다. 비트열 읽기 전용 명령어

본 논문에서는 비트열 정보는 메모리로부터 읽는 것으로 가정하고 비트열 읽기 과정을 위해 프로세서의 메모리 읽기 기능을 이용한다. 이로 인해 추가적인 비트열 입력 포트와 외부 입력 제어 장치가 필요하지 않기 때문에 범용 프로세서에 명령어를 적용하기가 쉬워진다.

메모리로부터 비트열 정보를 읽어오기 위해서는 비트열 정보를 저장하고 있는 메모리 공간의 주소를 알아야 하며 비트열의 현재위치도 알아야 한다. 비트열 정보를 저장하고 있는 메모리 공간의 주소와 비트열의 현재 위치는 메모리에 저장하고 비트열 처리를 할 때마다 불러와서 현재 비트열 위치가 가리키는 비트열 정보를 메모리로부터 읽는데 사용할 수 있다. 그러나 비트열 처리를 할 때마다 위와 같은 동작을 하게 되면 비트열 정보를 메모리로부터 읽어 들이는데 매우 많은 시간을 사용하게 된다. 또한 부호의 길이가 짧을 경우 같은 위치의 비트열 정보를 불필요하게 여러 번 읽게 되는 경우도 생긴다. 따라서 제안하는 비트열 읽기 방법은 전용 비트열 위치 레지스터와 비트열 버퍼 레지스터를 이용하여 반복적인 메모리 참조를 줄인다.

비트 버퍼 레지스터는 비트열의 현재 위치가 상위 16 비트 내에 있도록 유지함으로써 항상 현재 비트 위치부터 최소 16 비트의 비트열을 비트 버퍼 레지스터에서 바로 읽어 올 수 있도록 한다. 따라서 현재 비트열의 위치가 비트 버퍼 레지스터의 상위 16 비트를 넘어서게 되면 다음 비트열 정보를 읽어야 한다. 본 논문에서는 비트열 위치 레지스터와 비트열 버퍼레지스터를 초기화하는 명령어와 비트열 위치를 갱신하는 명령어를 통해서 비트열 위치 레지스터와 비트열 버퍼 레지스터를 관리하도록 한다.

비트 위치 레지스터 갱신 명령어는 프로세서의 메모리 로드 관련 명령을 이용하여 구현한다. 메모리 로드 관련 명령에서 주소 생성에 이용되는 덧셈기를 이용하여 비트열 위치 레지스터의 값과 갱신할 비트열의 위치를 더하고 더한 결과를 비트열 위치 레지스터에 쓴다. 비트열 위치 레지스터의 하위 5번째 비트 값이 변하게 될 경우는 비트열의 현재 위치가 비트 버퍼의 하위 16 비트를 가리키게 된다. 비트열 갱신 명령어는 현재 비트열의 위치가 비트 버퍼 레지스터의 상위 16비트 사이를 가리키도록 해야 하므로 새로운 비트열 정보 16 비트를 메모리를 통해서 읽는다. 메모리 주소는 비트 위치 레지스터의 상위 31 비트에 비트 0을 붙인 32비트의 값이 된다. 메모리를 통해 읽은 16 비트의 비트열 정보는 비트 버퍼 레지스터의 하위 16 비트에 써지고 비트 버퍼 레지스터의 하위 16 비트 값은 상위 16 비트에 써진다. 그림 5는 비트열 위치 갱신 명령어의 데이터 흐름을 보여준다.

비트열 전용 레지스터를 초기화 하는 명령어는 비트열 버퍼의 메모리 주소와 비트열의 현재 비트 위치를 더하고 비트 위치 레지스터에 저장한다. 비트열 위치 갱신 명령어와 같은 방법으로 메모리로부터 비트열 정보를 읽어온다.

표 1. 데이터 패스의 합성 결과 (시간 : ps)
Table 1. Synthesis result of the data path (time : ps).

		명령어 적용 전	명령어 적용 후
쉬프터	게이트	990	1111(121 증가)
	시간	1890	2006(116 증가)
덧셈기	게이트	906	1095(189 증가)
	시간	1733	1918(185 증가)
연속적인 비트 0 카운터	게이트	-	34
	시간	-	1297
합계	게이트	1896	2240(344 증가)
	메모리	-	67(32+35) 비트

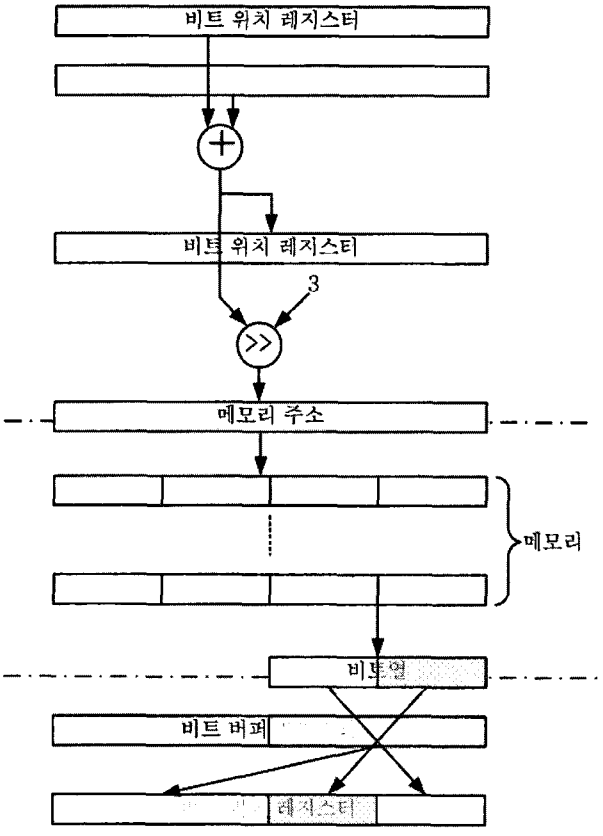


그림 9. 비트열 위치 갱신 과정의 데이터 흐름도
Fig. 5. Data flow of the bit stream position updating process.

비트열 위치 갱신 명령어가 메모리를 통해서 읽는 비트열 정보는 현재 위치의 비트열 정보가 아니라 현재 위치 보다 16 비트 뒤의 비트열 정보이기 때문에 비트 위치 레지스터는 비트열의 현재 위치보다 16 비트 뒤의 위치를 저장해야 한다. 따라서 비트열 전용 레지스터의 초기화 과정은 비트열 전용 레지스터 초기화 명령어 뒤에 비트열 위치 갱신 명령어로 비트열 위치 레지스터의 값을 16 증가시켜야 한다.

비트열 읽기 전용 명령어는 35 비트의 비트열 위치 레지스터의 덧셈 연산을 위해 3 비트의 덧셈기가 추가적으로 필요하다.

III. 실 험

1. 데이터 패스의 설계 및 합성

제안한 명령어 세트를 위한 데이터 패스를 설계하고 합성하였으며 비교를 위해 기존의 비트열 처리 모듈 또한 설계하고 합성하였다. 그림 6은 제안하는 명령어 세트를 위한 데이터 패스의 블록 다이어그램이다. 덧셈기와 쉬프터, 연속적인 비트 0 카운터에 멀티플렉서와 비트 전용 레지스터를 추가하여 비트열 전용 명령어들을 수행할 수 있도록 설계하였다. 그림 6에서 회색이 아닌 덧셈기와 쉬프터는 프로세서의 덧셈기와 쉬프터를 사용할 수 있다. 덧셈기는 CLA(Carry Lookahoed Adder)와 CSA(Carry Select Adder)를 혼합하여 설계하였다. 연속적인 비트 0 카운터는 기존 연구를 참조하였다[2]. VerilogHDL 언어로 설계하였으며 TSMC 0.25 ㎛ 공정 라이브러리를 사용하여 Synopsys사의 Design Compiler로 합성하였다

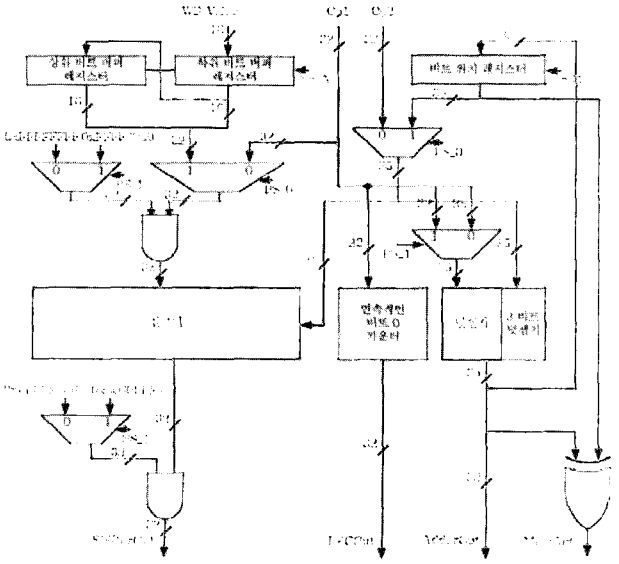


그림 6. 데이터 패스의 블록 다이어그램
Fig. 6. Block diagram of the data path.

합성 결과 기존의 연구에서 적용되었던 비트열 처리 모듈은 820 게이트와 35 비트의 메모리 면적이 필요하며 최대 지연시간은 2104 ps가 나왔다. 제안하는 명령어 세트를 위한 데이터 패스의 합성 결과는 각각의 데이터 패스에 대해 표 1에 나타내었다. 게이트 수는 2 입력 NAND 게이트 기준으로 나타내었으며 시간은 ps 단위이다. 제안하는 명령어 세트의 적용을 위해 총 344 게이트와 67 비트의 메모리 면적이 필요하며 최대 0.19 ns 의 지연시간이 추가 되었다. 제안하는 명령어 세트를 위한 데이터 패스는 기존의 비트열 처리 모듈에 비해 게이트 수가 476 게이트 적게 필요하며 지연시간에 주는 영향이 작다는 것을 확인할 수 있다. 필요로 하는 메모리 비트 수는 많지만 외부에 입력 버퍼가 필요하지 않으므로 무시할 수 있다.

2. 명령어 세트의 성능 시뮬레이션

제안하는 명령어 세트를 프로세서에 적용하였을 경우의 성능 향상을 보기 위해서 LISA(Langauge for Instruction-Set Architecture) 언어를 이용하여 MIPS 기반의 간결한 RISC 프로세서를 설계하고 제안하는 명령어 세트를 적용하였다. 적용된 프로세서는 5 단 파이프라인 구조를 갖는다. CoWare 사의 Processor Designer 를 이용하여 설계한 프로세서의 시뮬레이터와 C 컴파일러, 어셈블러 등의 개발환경을 생성하였다. 가변 길이 부호화 복호 과정의 성능 향상을 보기 위해 H.264/AVC CAVLC의 복호 과정 부분을 최근 논문에서 제안된 산술연산[9] 알고리즘을 적용하여 C 언어로

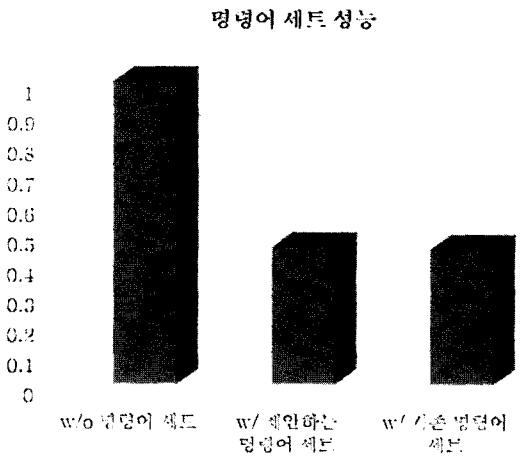


그림 7. 수행 시간 비교
Fig. 7. Comparison of the execution time.

작성하여 컴파일 하고 시뮬레이션 하였다. 제안한 명령어 세트를 적용하였을 경우와 적용하지 않았을 경우를 비교하였으며 비트열 정보를 메모리에서 읽어올 때의 지연과 과정을 두 단계로 나누었기 때문에 생기는 오버헤드를 측정하여 기존의 방법과도 비교하였다. 그림 7은 제안하는 명령어의 적용 전과 적용 후, 기존 명령어 적용 후의 수행 시간을 나타낸 그래프이다. 명령어 적용 전의 수행 시간 기준으로 값을 표기했다. 제안하는 명령어는 H.264/AVC CAVLC의 수행시간을 약 55% 감소시켰으며 기존 명령어를 기준으로 약 2%의 오버헤드만을 갖는다.

Ⅳ. 결 론

본 논문에서는 비트열 처리 과정의 가속을 위해 전용 비트열 레지스터와 전용 비트열 처리 명령어를 제안하였다. 제안한 비트열 처리 명령어 세트는 프로세서의 기본적인 데이터 패스의 공유를 통해 새로운 데이터 패스의 추가를 최소화 하였다. 비트열 정보를 메모리로부터 읽어 오기 때문에 추가적인 입력 포트가 필요하지 않고 프로세서의 구조 변형 또한 필요하지 않으며 외부에 별도의 입력 제어 장치나 버퍼가 필요하지 않다. 명령어 적용을 위해 344 게이트와 65 비트 메모리의 추가 면적이 필요하며 지연시간이 최대 0.19 ns 증가한다. H.264/AVC CAVLC 복호 과정의 수행 시간을 약 55% 감소시켰으며 기존의 방법과의 성능 차이가 크지 않았다. 따라서 본 논문에서 제안하는 명령어 세트는 낮은 추가 비용으로 비트열 처리 과정을 가속할 수 있으며 여러 프로세서에 적용하기 쉽다.

참 고 문 헌

[1] JVT, "Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC)," May 2003.

[2] W. Di, G. Wen, H. Mingzeng, and J. Zhenzhou, "A VLSI Architecture Design of CAVLC Decoder," *5th IEEE Int. Conf. on SIC*, vol.2, pp.962-965, Oct. 2003.

[3] H. C. Chang, C. C. Lin, and J.I. Guo, "A Novel Low-Cost High-Performance VLSI Architecture for MPEG-4 AVC/H.264 CAVLC Decoding," *Proc. IEEE ISCAS*, May 2005, pp.6110-6112.

[4] Heng-Yao Lin; Ying-Hong Lu; Bin-Da Liu; Jar-Ferr Yang, "Low power design of H.264 CAVLC decoder," *Circuits and Systems, 2006. ISCAS 2006. Proc. 2006 IEEE Int. Sym. on*, vol., no., pp. 4 pp.-, 21-24 May 2006.

[5] Guo-Shiuan Yu; Tian-Sheuan Chang, "A zero-skipping multi-symbol CAVLC decoder for MPEG-4 AVC/H.264," *Circuits and Systems, 2006. ISCAS 2006. Proc. 2006 IEEE Int. Sym. on* , vol., no., pp. 4 pp.-, 21-24 May 2006.

[6] D. Wu, T. Hu, and D. Liu, "A Single Scalar DSP based Programmable H.264 Decoder," *Proc of the SSoCC*, Tammsvik, Sweden, Apr. 2005.

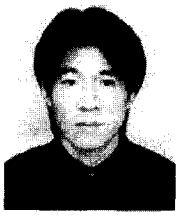
[7] M. Berekovic, H.J. Stolberg, M.B. Kulaczewski, and P. Pirsch, "Instruction Set Extension for MPEG-4 Video," *The Journal of VLSI Signal Processing-Systems for Signal, Image, and Video Technology*, vol. 23, no. 1, pp. 27-49, Oct. 1999.

[8] Yung-Chi Chang, Chao-Chih Huang, Wei-Min Chao, and Liang-Gee Chen, "An Efficient Embedded Bitstream Parsing Processor for MPEG-4 Video Decoding System," *The Journal of VLSI Signal Processing-Systems for Signal, Image, and Video Technology*, vol. 41, no. 2, pp. 183-191, 2005.

[9] Y.-H. Kim, Y.-J. Yoo, J. Shin, B. Choi and J. Paik, "Memory-Efficient H.264/AVC CAVLC for Fast Decoding," *IEEE Trans. on Consumer Electronics*, Vol.52, Iss.3, Aug. 2006.

저 자 소 개

함 동 현(학생회원)
2005년 성균관대학교 정보통신 공학부 학사 졸업.
2006년~현재 연세대학교 전기 전자 공학과 석사과정.
<주관심분야 : 영상신호처리, SoC 설계>



이 형 표(학생회원)
2001년 건국대학교 전자공학과 학사 졸업.
2001년~현재 삼성전자 DM총괄 선임연구원
2006년~현재 연세대학교 전기전자공학과 석사과정.
<주관심분야 : 영상신호처리, SoC 설계>

이 용 석(평생회원)
1973년 연세대학교 전자공학과 학사 졸업.
1977년 University of Michigan Electrical Engineering 석사 졸업.
1981년 University of Michigan Electrical Engineering 박사 졸업.
1993년~현재 연세대학교 전기전자공학과 교수.
<주관심분야 : 마이크로프로세서 설계, VLSI 설계, DSP 프로세서 설계, 고성능 연산기 설계>