論文99-36C-10-7

멀티미디어 데이타 처리를 위한 그래픽 프로세서 설계

(Design of a Graphic Processor for Multimedia Data Processing)

高 翊 相 *, 韓 宇 宗 **, 鮮 于 明 勳 *

(Eck-Sang Ko, Woo-Jong Hhan, and Myung-Hoon Sunwoo)

요 약

본 논문은 그래픽 프로세서(Graphic Coprocessor: GCP)의 설계 및 구현에 대해 기술한다. 설계된 그래픽 프로세서는 멀티미디어 서버용 프로세서의 그래픽 전용 프로세서로 사용 가능하다. GCP 명령어 집합은 멀티미디어 데이타의 병렬성을 이용하기 쉬운 SIMD 및 Superscalar 등의 병렬 아키텍쳐 개념을 적용하여 설계하였다. 설계된 GCP는 4개의 주 프로세서에 의해 공유되는 형태이며 공유에 따른 명령어의 병목현상을 해결하기 위한 스케줄러와 연산을 위한 4개의 기능 유니트를 내장하고 있다. 최대 4개 명령어의 동시 수행이 가능한 GCP는 Verilog HDL로 모델링하고 논리 합성하였다. 약 56,000개의 게이트로 구성되는 GCP는 SOG 라이브러리의 제약으로 인하여 30 MHz로 동작하며 CIF 영상 규격에 대해 초당 63 프레임의 DCT 연산 및 초당 21 프레임의 FBMA 연산을 수행 할 수 있다.

Abstract

This paper presents an architecture and its instruction set for a graphic coprocessor(GCP) which can be used for a multimedia server. The proposed instruction set employs parallel architecture concepts, such as SIMD and Superscalar. GCP consists of a scheduler and four functional units. The scheduler solves an instruction bottleneck problem causing by sharing with four general processors(GPs). GCP can execute up to 4 instructions in parallel. It consists of about 56,000 gates and operates at 30 MHz clock frequency due to speed limitation of SOG technology. GCP meets the real-time DCT algorithm requirement of the CIF image format and can process up to 63 frames/sec for the DCT Algorithm and 21 frames/sec for the Full Block matching Algorithm of the CIF image format .

* 正會員, 亞州大學校 電氣電子工學部

(School of Electronical and Electronic Eng., Ajou Univ.)

** 正會員, 韓國電子通信研究院

(Electronics and Telecommunications Research Institute)

※ 본 연구는 한국전자통신연구원(ETRI)의 연구비에 의해 연구되었으며, 반도체설계교육센터(IDEC) 사업의 부분적인 지원을 받아 수행되었습니다.

接受日字:1999年7月5日, 수정완료일:1999年9月30日

Ⅰ. 서 론

멀티미디어 서비스에 대한 요구가 늘어나면서 비디오, 오디오 등 다양한 멀티미디어 데이타의 처리에 적합한 멀티미디어 프로세서가 개발되어 사용되고 있는데, 특히 데이타 양이 많은 비디오 신호를 처리하는 그래픽 프로세서는 멀티미디어 프로세서의 고속 고성능을 위해 필수적인 보조 프로세서이다. 외국에서 개발된 많은 프로세서들은 그래픽 보조 프로세서가 내장되어 있으며 차세대 멀티미디어 서비스는 그 용도가 다양해

지고 데이타 양이 방대해 지기 때문에 이러한 서비스를 고속으로 처리하기 위한 고성능의 멀티미디어용 서버는 필수이며 이에 따른 고속의 보조 프로세서 개발도 필수적이다. 특히 데이타 양이 방대한 비디오 신호를 처리하기 위한 그래픽 보조 프로세서의 성능 향상은 전반적 시스템 성능 향상에 매우 중요하다.

현재 개발되어 사용중인 멀티미디어 프로세서들은 MicroUnity사의 MediaProcessor[1], Chromatic사의 Mpact[2] 및 Mpact2[3,4], Philips사의 TriMedia[5], Texas Instrument사의 TMS320C6x[6] 그리고 Analog Device사의 ADSP-2106x[7] 등이 있다. 또한 고성능 범용 마이크로 프로세서에 멀티미디어 데이타 처리를 위한 명령어를 추가한 것으로는 Sun Microsystems사의 UltraSPARC[8], Intel사의 ix86[9], HP사의 PA-RISC[10] 및 SGI사의 MIPS 아키텍쳐가 있으며 이들은 각각 VIS(Visual Instruction Set)[11], MMX(Multi-Media Extension)[12], MAX(Multimedia Acceleration Extensions), MAX-2[13] 및 MDMX(Mips Digital Media Extension)[14] 명령어 집합을 추가하였다.

이와 같은 멀티미디어 프로세서들의 공통적인 특징은 SIMD(Single Instruction stream on Multiple Data streams), 슈퍼스칼라(Superscalar), VLIW(Very Long Instruction Word), 멀티스레드(Multithread), 벡터프로세서 등의 다양한 병렬처리 기술을 이용하였으며 처리 데이타의 크기를 조절할 수 있게 설계되어 데이타의 동적 범위가 작은 영상이나 음성 데이타를 처리할 때 여러 개의 처리기 구조로 분할되어 동시에 여러 개의 데이타를 처리하여 성능을 개선시킨다는 것이다. 그러나 이들은 다수의 기능 유니트를 내장한 고가의 칩으로서 die 면적을 많이 차지하므로 적은 수의 기능 유니트로서 멀티미디어 데이타를 효율적으로 처리하는 그래픽 프로세서의 개발이 필요하다.

본 논문에서는 멀티미디어 전용 프로세서의 그래픽 전용 프로세서인 그래픽 프로세서(Graphic Coprocessor : GCP)의 설계 및 구현에 대해 기술한다. GCP 명령어 집합은 멀티미디어 데이타의 병렬성을 이용하기 쉬운 SIMD 및 슈퍼스칼라 등의 병렬 아키텍쳐 개념을 적용시켰다. 멀티미디어 데이타 처리에 적합한 형태로 설계된 GCP 명령어 집합은 멀티미디어 알고리즘 구현 시 자주 사용되는 핵심 명령어 및 영상 신호처리에 효과적인 명령어들을 포함한다. 설계된 GCP는 4개의 주 프로세서(General Processor: GP)에 의하여 공유되는 형태이다. 공유에 따른 명령어 병목 현상을 해결하기 위해 스케줄러를 내장하고 있고 4개의 기능 유니트(GALU, GMUL, GBMU, GSAD)를 내장하고 있다. 기능 유니트는 한 개의 데이터 경로가 목적에 따라 여러 개로 분할될 때 8, 16, 32, 64 비트 등의 다양한 데이터 형태의 연산을 지원한다.

본 논문은 다음과 같이 구성된다. 2장에서 GCP 명령어 집합에 대해 기술하며 3장에서는 GCP 아키텍쳐에 대해 논한다. 4장에서는 구현 및 성능 분석에 대해서 논하며 마지막으로 5장에서 결론을 맺는다.

Ⅱ. GCP 명령어 집합

이 장에서는 GCP 명령어 집합 설계에 대해 기술한다. GCP 명령어 집합은 멀티미디어 프로세서의 아키텍쳐[1-10] 및 기존의 멀티미디어 알고리즘과 멀티미디어 확장 명령어 집합[11-14]을 분석하고 개선하여 새로운 명령어 집합 설계에 반영하였으며 특히 멀티미디어, 영상처리, 3-D 그래픽스 처리에 적합하도록 설계되었다. 방대한 양의 멀티미디어 데이타 처리를 위하여 SIMD 및 슈퍼스칼라 등의 병렬 아키텍쳐 개념을 적용시켰다. 설계된 명령어 집합은 SIMD 스타일 명령어, 포화(saturation) 연산 지원, Bit-manipulation 연산 지원 등의 특징을 지닌다.

GCP 명령어 집합은 멀티미디어 알고리즘을 효율적으로 처리하고 SIMD 형태 아키텍쳐의 병렬성을 이용하기 위해 데이타 형태를 그림 1과 같은 4가지 형태 - 나눔 바이트(Packed byte), 나눔 워드(Packed word), 나눔 더블워드(Packed double-word), 나눔 쿼드워드(Packed quad-word) - 로 구분하였다. 하나의 나눔 바이트는 8개의 바이트 데이타가 1개의 64 비트 워드에 저장된다. 이 데이타 형태는 각 바이트가 객체 소자로 구성된다는 점에서 일반적인 64 비트 워드와는 구분된다. 따라서 이와 같은 데이터 형태를 이용하면 하나의 나눔 바이트에 2개 화소의 칼라 영상 정보 - 적색, 녹색, 청색 및 alpha 값 각 2개씩 - 를 저장할 수 있어 유리하다. 일반적으로 멀티미디어 데이타는 8, 16, 32, 64 비트 단위로 이루어지며 멀티미디어 프로그램은 덧셈 및 곱셈과 같은 단위 연산에 바탕을 두므로 본 논문에서 정의한 데이타 형태를 이용할 경우에 병렬성을 이용하여 처리가 용이한 장점을 갖는다.

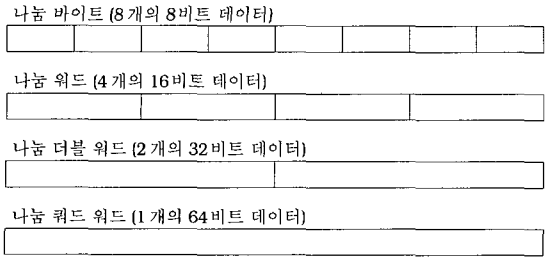


그림 1. 나눔 데이타 형태
Fig. 1. Packed Data Types.

GCP 명령어 집합의 또다른 특징은 포화 연산에 있다. 포화 연산은 연산의 결과가 표현 가능한 수의 범위를 넘는 경우 연산 결과를 최대값 또는 최소값으로 채워 주는 것이다. 예를 들어 나눔 워드에 대한 가산에서 연산의 오버플로우(overflow)가 발생하는 경우에만 연산 결과를 최대값으로 채워준다. 마찬가지로 연산의 언더플로우(underflow)가 발생하는 경우에는 최소값으로 연산 결과를 채워준다. 이런 포화 연산은 영상 합성, 음향 신호 혼합 등의 알고리즘에서 잦은 오버플로우나 언더플로우의 Exception 처리부담을 줄여 하드웨어 제어를 간단하게 한다. 또한 GCP 명령어 집합은 움직임 추정 알고리즘에서 많이 사용되는 SAD(Sum of Absolute Difference) 값 추출을 위한 전용 명령어인 GSAD 명령어를 지원한다.

GCP 명령어 집합은 기능 유니트에 따라서 GALU 명령어 집합, GMUL 명령어 집합, GBMU 명령어 집합, GSAD명령어 집합의 4개 명령어 집합으로 구분되며 표 1에 나타나 있다. GALU 명령어 집합은 18개의 명령어로 구성되며 산술 및 논리 연산을 수행한다. 가산 및 감산 명령어는 포화 연산과 랩어라운드(wraparound) 연산 두 가지 모드를 지원하며 포화 연산 모드에서 signed 포화 연산과 unsigned 포화 연산을 모두 지원한다. 그림 2는 나눔 워드 데이터에 대한 GSADDU 명령어의 동작 예를 나타낸다. GSADDU 명령어는 unsigned 수에 대한 가산 명령어이다. 그림 2에서 상위 2개의 워드의 연산 결과는 오버플로우가 발생하지 않았으므로 결과값이 그대로 데스티네이션에 쓰였지만 하위 두 개 워드의 가산 결과는 오버플로우가 발생하였으므로 unsigned 16 비트로 표현 가능한 최대값인 16'hFF(16 진수)로 채워진다.

표 1. GCP 명령어 집합
Table 1. The GCP Instruction Set.

	명령어	동 작 설 명
GALU 명령어집합	GADD	Packed Addition with Wraparound
	GSADDU	Unsigned Packed Addition with Saturation
	GSADDS	Signed Packed Addition with Saturation
	GSUB	Packed Subtraction with Wraparound
	GSSUBU	Unsigned Packed Subtraction with Saturation
	GSSUBS	Signed Packed Subtraction with Saturation
	GCMPE	Packed Compare for Equal
	GCMPGU	Unsigned Packed Compare for Greater Than
	GCMPGS	Signed Packed Compare for Greater Than
	GABS	Packed Absolute of Subtraction
	GMINU	Unsigned Packed Minimum
	GMINS	Signed Packed Minimum
	GAND	Packed Logical AND
	GANDN	Packed Logical AND NOT
	GOR	Packed Logical OR
	GORN	Packed Logical OR NOT
	GXOR	Packed Logical XOR
	GNOT	Packed Logical NOT
GMUL 명령어집합	GMULH	Signed Packed Multiply High
	GMULL	Signed Packed Multiply Low
GBMU 명령어집합	GLSHL	Packed Logical Shift Left
	GLSHR	Packed Logical Shift Right
	GASHL	Packed Arithmetic Shift Left
	GASHR	Packed Arithmetic Shift Right
	GSHFH	Pixel Shuffle High
	GSHFL	Pixel Shuffle low
	GCOPY	Pixel Group Copy
	GWCHG	Pixel Word Change
	GPACKU	Pixel Truncate, insert and unsigned Pack
	GPACKS	Pixel Truncate, insert and Signed Pack
	GUNPKH	Pixel Unpack High
	GUNPKL	Pixel Unpack Low
GSAD 명령어집합	GSAD	Pixel Sum of Absolute Differnece

GMUL 명령어 집합은 GMULH와 GMULL로 구성된다. GMULH 명령어는 나눔 워드 단위의 곱셈 연산 후 결과값인 더블 워드의 상위 워드만을 데스티네이션에 패킹(packing)하여 저장하며 GMULL 명령어는 결과값이 하위 워드만을 저장한다. 그림 3에 GMULH의 동작 예가 나타나 있다.

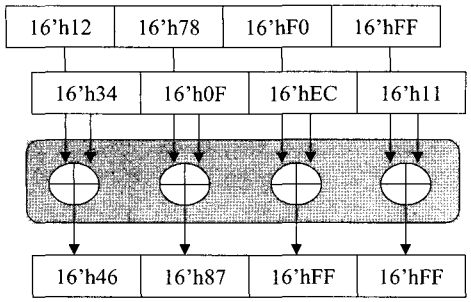


그림 2. GSADDU 명령어의 동작 예
Fig. 2. The GSADDU Instruction Example.

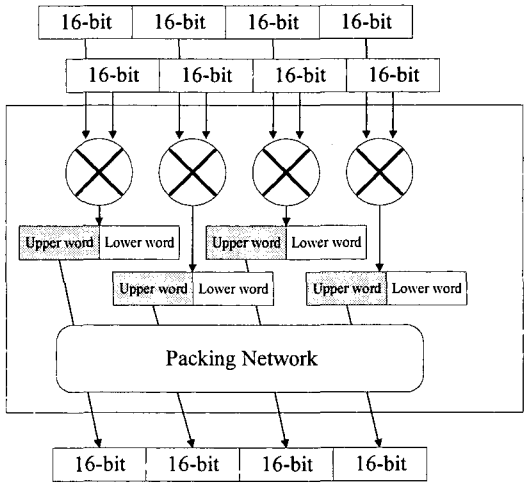
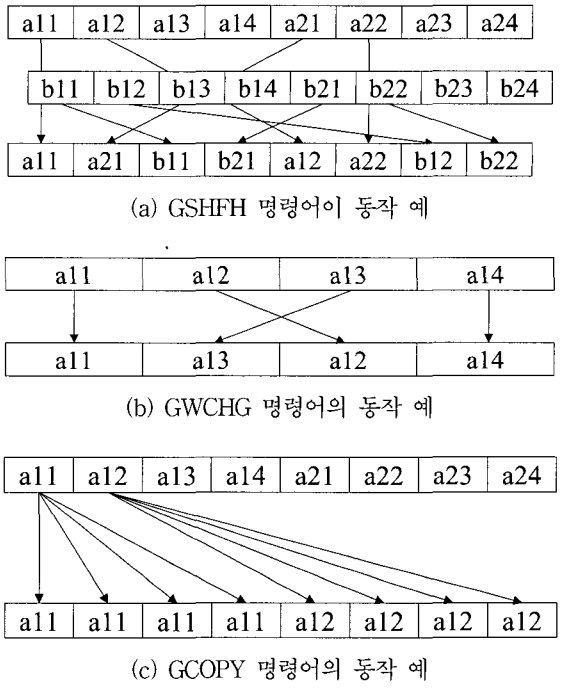


그림 3. GMULH 명령어 동작 예
Fig. 3. The GMULH Instruction Example.

GBMU 명령어 집합은 GLSHL, GLSHR, GASHL, GASHR 등과 같은 논리 및 산술 쉬프트 명령어와 GSHF, GCOPY, GWCHG, GPACK, GUNPK 등과 같이 데이터를 원하는 형태로 효율적으로 변환하는 명령어로 구성된다. 논리 및 산술 쉬프트 명령어는 GBMU 유니트에 내장된 배럴쉬프터(Barrel Shifter)를 이용하여 나눔 데이타에 대해 어떤 비트의 쉬프트도 가능하다. 이들 GBMU 명령어는 MPEG, 3-D 그래픽스에서의 geometric transform에 유리하며 밴드 시퀀셜 데이타와 밴드 인터리브드 데이타간의 상호 변환에 유리하다. 특히, 행렬 연산에서 전치행렬을 구하는 등 행렬 데이타를 원하는 형태로 변환하는 경우에 효율적으로 사용할 수 있다. 그림 4에 GBMU 명령어의 동작 예가 나타나 있다.



(a) GSHFH 명령어이 동작 예

(b) GWCHG 명령어의 동작 예

(c) GCOPY 명령어의 동작 예

그림 4. GBMU 명령어의 동작 예
Fig. 4. The GBMU Instruction Example.

GSAD 명령어는 GSAD유니트에서 수행되는 움직임 추정 알고리즘 전용 명령어로서 두개 오퍼랜드의 나눔 바이트 단위 데이타간의 절대값 차의 합을 구하는 명령어이다. 그림 5에 GSAD명령어 집합과 동작 예가 나타나 있다.

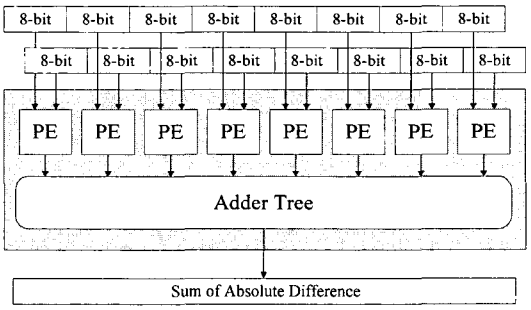


그림 5. GSAD 명령어 동작 예
Fig. 5. The GSAD Instruction Example.

Ⅲ. GCP 아키텍쳐

이 장에서는 설계된 GCP 아키텍쳐에 대하여 기술한다. 멀티미디어 데이타는 그 양이 방대하고 명령어 수

준 병렬성이 높은 점을 이용하여 SIMD 및 수퍼스칼라 등의 병렬 처리 기법을 도입하여 설계하였다. GCP는 크게 스케줄러와 4개의 기능 유니트로 구성된다. 스케줄러는 GCP가 4개의 주 프로세서(General Processor: GP)에 공유되어 발생하는 명령어 병목 현상을 제거한다. 스케줄러는 기능 유니트별 라운드로빈 방식 스케줄링 기법을 사용하는 동시에 각 GP에 대한 기능 유니트 사용 히스토리를 관리하여 한 GP가 특정 기능유니트를 독점하여 사용하는 것을 방지한다. 기능 유니트로는 GALU(Graphic ALU), GMUL(Graphic Multiplier), GBMU(Graphic Bit Manipulation Unit), GSAD(Graphic Sum of Absolute Difference) 등을 내장하고 있다. 제안된 GCP 아키텍처의 특징은 다음과 같으며 그림 6에 설계된 GCP 아키텍처가 나타나 있다.

- 화소 단위의 데이타 처리를 위한 SIMD형태의 연산 구조
- 4개의 GP에 의해서 공유, 스케줄러를 사용하여 서로 다른 4개의 그래픽 명령어를 병렬 수행
- 나눔 데이타 연산 지원
- RISC스타일에 적합한 아키텍처
- Bit-manipulation 명령어 지원
  - MPEG, 3-D 그래픽스 등에서 효율적인 geometric transform 지원
  - 데이타 변환 명령어 지원
- 수퍼스칼라 + SIMD 방식의 연산구조

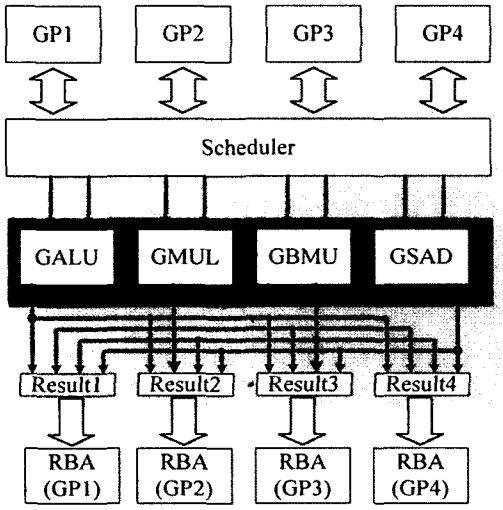


그림 6. GCP 아키텍처
Fig. 6. The GCP Architecture.

GALU는 산술 및 논리 연산을 수행하는 유니트이다. GALU는 GMUL이나 GSAD로부터 연산된 결과값을 합성하거나 최대값 및 최소값의 추출에 유용하게 사용된다. GALU는 포화 연산과 랩어라운드 연산을 지원하며 포화 연산 모드에서는 unsigned 포화 연산과 signed 포화 연산을 모두 지원한다. 8, 16, 32, 64비트로 구성 가능한 나눔 데이터 연산을 지원하는 GALU는 18개의 명령어로 구성된 GALU 명령어 집합을 가지고 있다. GALU는 8개의 8 비트 ALU가 연결(concatenation)된 형태로 구성되며 그림 7에 GALU의 아키텍처가 나타나 있다.

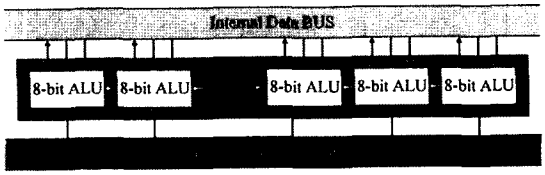


그림 7. GALU 아키텍처
Fig. 7. The GALU Architecture.

GMUL은 4개의 16×16 곱셈기와 패킹 네트워크(Packing Network)로 구성되어 있으며 그래픽스나 영상 분야 데이타의 곱셈(unsigned×unsigned, unsigned ×signed, signed×unsigned, signed×signed)을 지원한다. 그래픽스나 영상 분야의 데이터는 일반적으로 픽셀의 기본 단위인 8 비트로 구성되어 있는 경우가 많으므로 GMUL은 4개의 8 비트×8 비트, 혹은 16 비트× 16 비트 연산을 하나의 명령어로 수행하는 SIMD 스타일의 명령어를 지원한다. GMUL은 그림 8에서 보듯 4개의 16 비트 x 16비트 연산의 결과를 패킹 네트워크를 통하여 상위 또는 하위16 비트로 패킹하여 데스티네이션 레지스터에 저장한다.

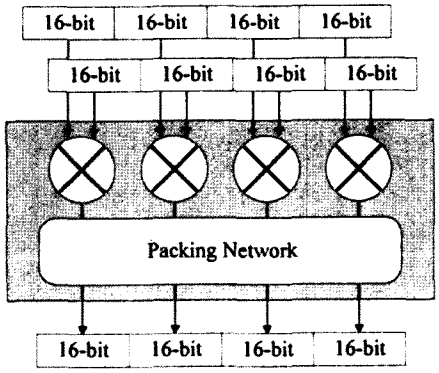


그림 8. GMUL 아키텍처
Fig. 8. The GMUL Architecture.

GBMU는 GSHFH, GSHFL, GCOPY, GWCHG 등의 명령어를 사용하여 데이타를 원하는 형태로 변환시킨다. 또한 내장된 배럴 쉬프터를 이용하여 모든 나눔 데이타에 대한 산술 및 논리 쉬프트 연산을 지원한다. 따라서 연산 결과값을 바이트, 워드 단위로 쉬프트 시켜 저장할 수 있으므로 GP에 내장되어 있는 그래픽 레지스터의 효율적 사용을 가능하게 한다.

GSAD는 움직임 추정 알고리즘 전용 유니트로서 움직임 벡터 추출에 많이 사용되는 SAD(Sum of Absolute Difference) 값을 구하는데 사용된다. GSAD는 SIMD 구조를 갖는 8개의 프로세싱 엘리먼트(Processing Element: PE)와 트리형 가산기로 구성된다. 내장된 8개의 프로세싱 엘리먼트는 AD(Absolute Difference) 값을 구하며 트리형 가산기는 8개의 AD값들을 더하여 SAD 값을 얻는다. 이렇게 구해진 SAD값을 토대로 움직임 벡터를 추출할 수 있다. 또한, GSAD는 움직임 추정 알고리즘에 따라 가변적인 기준 블록의 크기의 변화에도 능동적으로 대처할 수 있다.

본 논문에서 설계된 스케줄러는 기능 유니트별 라운드로빈 스케줄링 기법을 사용하여 병렬로 수행 가능한 명령어를 결정하며 한 GP가 계속하여 동일한 기능 유니트를 독점 사용하는 것을 방지한다. 스케줄러는 4개의 GP로부터 전달된 명령어들의 태그 필드(GALU: 00, GMUL: 01, GBMU: 10, GSAD: 11)를 비교하여 동시에 수행 가능한 명령어를 결정한다. 이때 GP에서 전달된 명령어의 유효성 여부를 판단한 후 스케줄링하게 되며 명령어의 유효 여부는 GP에서 전달되는 제어신호인 GCP_dispatch 신호로서 판별한다. GP는 유효한 명령어를 GCP로 이슈하는 동시에 GCP_dispatch 신호를 인가하게 된다. GP로부터 전달된 유효한 명령어 중 동시에 수행이 가능한 명령어만이 기능 유니트로 디스패치되며 스케줄러는 디스패치된 명령어를 이슈한 GP에 GCP_ready 신호를 보내어 해당 GP가 이슈한 명령어가 수행됨을 통보한다. 기능 유니트로 디스패치된 명령어의 수행이 완료되면 그 명령어를 이슈한 GP에 GCP_wreq 신호를 전달하여 명령어의 수행이 완료됨을 통보하고 동시에 GP의 RBA(Result Bus Arbiter)의 사용권을 요청한다.

GCP와 GP간의 명령어 이슈 타이밍 도가 그림 9에 나타나 있다. 그림 9의 T0 시간 슬롯에서는 GP가 이슈한 명령어가 유효하고 다른 GP에서 이슈한 명령어와 충돌이 없기 때문에 기능유니트로 디스패치되어 GCP_ready 신호가 인가된다. 그러나 T1 시간 슬롯에서 GP가 이슈한 명령어는 유효한 명령어가 아니므로 다른 GP명령어와 충돌에 상관없이 기능유니트로 디스패치되지 않는다. 그림 10에 GP의 RBA와의 타이밍 도가 나타나있다. 그림 10의 T2 시간 슬롯에서 GCP는 스케줄러는 RBA의 사용권을 요청하지만 GP가 요청을 허가하지 않았으므로 연산 결과값인 d2가 GP의 ROB(Reorder Buffer)에 저장되지 못한다. 그림 10의 T0, T1, T3 시간 슬롯에서는 연산 결과값이 ROB에 저장된다. 스케줄러는 FSM(Finite State Machine)으로 구현되었으며 데이타 중복성(redundancy) 체크와 데이타 충돌(conflict) 방지는 GP와 컴파일러에서 담당한다.

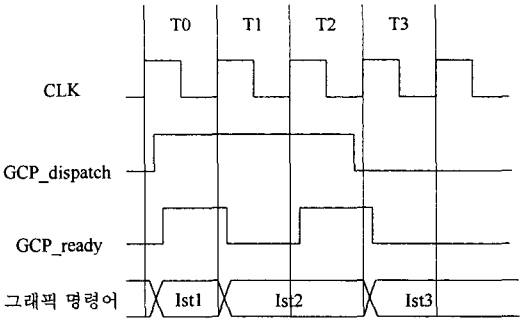


그림 9. GCP와 GP간의 명령어 이슈 타이밍 도
Fig. 9. Timing Diagram between GCP and GP.

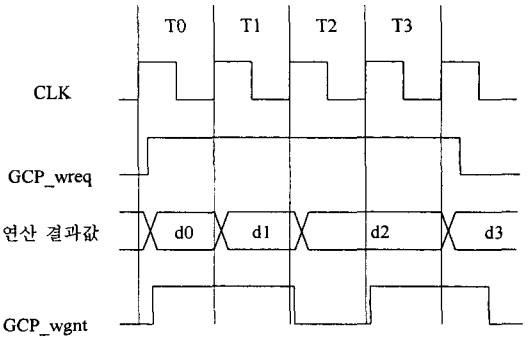


그림 10. GCP와 RBA 간의 연산 결과 저장 타이밍 도
Fig. 10. Timing Diagram between GCP and RBA.

2개 이상의 GP가 동시에 같은 기능 유니트를 사용하는 명령어를 이슈하게 되면 기능 유니트에 대한 자원 충돌(resource conflict)이 일어나게 된다. 이를 해결하기 위해 각 GP에 우선 순위(priority)를 지정하여 2개 이상의 GP가 동시에 같은 기능 유니트를 사용하는 명령어를 이슈하는 경우에 GP의 우선 순위에 따라 명령어를 스케줄링할 수 있다. 그러나 이 방식의 문제점은

우선 순위가 높은 GP가 동일한 기능 유니트를 사용하는 명령어를 계속하여 이슈하는 경우에 낮은 우선 순위의 GP도 동일한 기능 유니트를 사용하여야 하는 명령어를 이슈한다면 우선 순위가 낮은 GP가 이슈한 명령어는 우선 순위가 높은 GP의 명령어가 모두 디스패치될 때까지 스톨되는 것이다. 이는 특정 GP가 기능유니트를 독점 사용하게 되므로 불합리한 스케줄링 방식이다.

이러한 문제점을 해결하기 위하여 본 연구에서 설계된 GCP의 스케줄러는 GP에 우선 순위를 주는 동시에 4개의 GP에 대한 명령어 이슈 히스토리를 관리하는 기능 유니트별의 라운드로빈 방식을 채택하였다. 이때 각 GP의 명령어 이슈 히스토리는 이전에 이슈되어 기능 유니트로 디스패치된 명령어의 태그 필드로서 각 GP가 어떤 기능 유니트를 사용했었는지에 대한 기록이다. 이 방식은 높은 우선 순위의 GP가 낮은 우선 순위의 GP와 자원 충돌이 발생할 때 특정 기능 유니트를 독점 사용하는 것을 방지한다. 즉, 2개 이상의 GP가 같은 기능 유니트를 사용하는 명령어를 이슈한 경우에 높은 우선 순위를 갖는 GP가 그 이전 순간에 같은 기능 유니트를 사용하지 않았다면 디스패치하고 연속하여 같은 기능 유니트를 사용하려 하는 것이라면 스톨 시키고 우선 순위가 낮은 GP의 명령어를 디스패치한다. 예를 들어 2개의 GP가 동일한 기능 유니트를 사용하려고 하는 경우의 스케줄러의 상태 천이도가 그림 11에 나타나 있다.

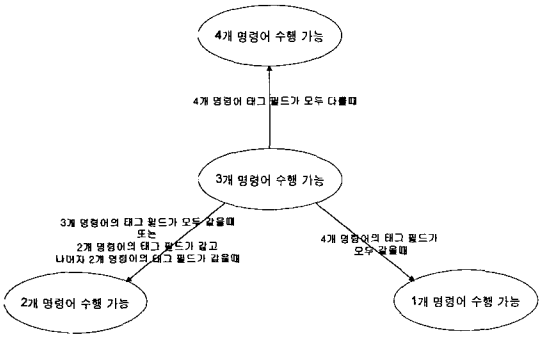
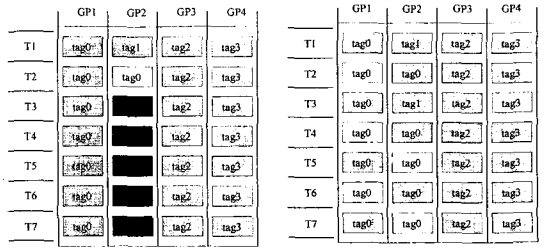


그림 11. 3개 명령어만 수행 가능한 경우의 상태 천이도

Fig. 11. The State Diagram of Only 3 Instruction Executable Case.

그림 12(a)에 GP의 우선 순위만을 고려한 스케줄링 방식에 대한 예가 나타나 있다. GP들의 우선 순위는 GP1이 가장 높고 GP4가 가장 낮다고 가정하였다. 그림에서 T는 시간 슬롯을 의미하며 회색 사각형은 그 시간 슬롯에서 기능 유니트로 디스패치 가능한 명령어이며, 흰색 사각형은 그 시간 슬롯에서 이슈 되었으나 우선 순위가 낮아 스톨되는 명령어를 나타낸다. 그리고 빗금친 사각형은 그 시간 슬롯 이전에 이슈되었으나 계속하여 스톨되는 명령어를 나타낸다. 사각형 안의 태그 필드는 그 명령어가 사용할 기능 유니트의 태그 필드로서 tag0, tag1, tag2, tag3은 각각 GALU 명령어 집합, GMUL 명령어 집합, GBMU 명령어 집합, GSAD 명령어 집합을 나타낸다.

예를 들면 그림 12(a)의 시간 슬롯 T2에서 GP1의 tag0는 그 시간 슬롯에서 디스패치되는 명령어이고 GP2의 tag0는 그 시간 슬롯에 이슈되었으나 낮은 우선 순위 때문에 스톨되는 명령어이다. 그리고 시간 슬롯 T3에서 GP2의 tag0는 T2에 이슈되었으나 우선 순위가 높은 GP1이 같은 기능 유니트를 사용하는 명령어를 이슈하여 스톨되는 명령어이다. 우선 순위만 고려된 방식에서는 우선 순위가 높은 GP가 같은 기능 유니트를 사용하는 명령어를 계속하여 이슈하면 우선 순위가 낮은 GP가 그와 동일한 기능 유니트를 사용하는 명령어를 이슈할 경우에 계속하여 스톨 된다(그림 12(a) GP2의 T2 - T7 구간).



(a) 우선 순위만을 고려한 동작 예 (b) 라운드로빈 방식 스케줄링 동작 예

그림 12. 스케줄링 방식에 따른 동작 예

Fig. 12. Operation Examples according to Scheduling Schemes.

그림 12(b)에 본 연구의 스케줄러에서 채택한 라운드로빈 방식의 스케줄링 예가 나타나 있다. 시간 슬롯 T2에서 GP1과 GP2가 모두 GALU를 사용하는 명령어를 이슈하고 있다. 이때 비록 GP1이 우선 순위가 높지만 스케줄러가 T2 이전 슬롯인 T1에서 GALU를 이미 사

용하였다는 히스토리를 저장하고 있어서 GP1의 명령어는 스톨시키고 GP2의 명령어를 디스패치시킨다. 예를 들어 그림 12에서 GP1은 GALU 유니트를 계속하여 사용하려 하고 GP2는 MAC(Multiply and Accumulation) 연산을 수행하기 위해 GMUL과 GALU를 반복적으로 사용하려 한다고 가정해보자. 그림 12(a)와 같은 방식에서는 우선 순위가 낮은 GP2는 GP1이 GALU 명령어를 계속하여 이슈하는 한 GP2의 GALU 명령어는 계속하여 스톨된다. 이것은 우선 순위가 높은 GP가 특정 기능 유니트를 독점 사용하므로 우선 순위가 낮은 GP에겐 불리한 스케줄링 방식이다. 그러나 본 연구에서 채택한 그림 12(b)와 같은 기능 유니트별 라운드로빈 방식은 한 GP가 특정 기능 유니트를 독점하지 못하게 함으로서 히스토리 체크가 없이 우선 순위만을 고려한 방식보다 더욱 효율적인 스케줄링이 가능하다.

Ⅳ. 구현 및 성능 평가

결정된 설계사양에 따라서 Verilog HDL로 모델링을 수행하였고 CadenceTM 캐드 툴로 시뮬레이션하여 기능을 검증하였다. 그림 13에 스케줄러에 대한 시뮬레이션 파형을 보였다. 스케줄러 시뮬레이션에서는 4개의 GP들이 명령어를 연속적으로 이슈하고 연산 결과값이 바로 GP로 보내진다고 가정하였다. GCP를 공유하는 4개의 GP를 각각 GP1, GP2, GP3, GP4라 할 때 각각의 GP에서 이슈된 명령어를 I1, I2, I3, I4라 하고 각 명령어의 태그 필드를 tag1, tag2, tag3, tag4라 하였다. Current_State는 현재 동시에 수행 가능한 명령어의 개수를 의미하며 Next_State는 태그 필드를 비교하여 다음 사이클에 동시 수행 가능한 명령어의 개수를 세고

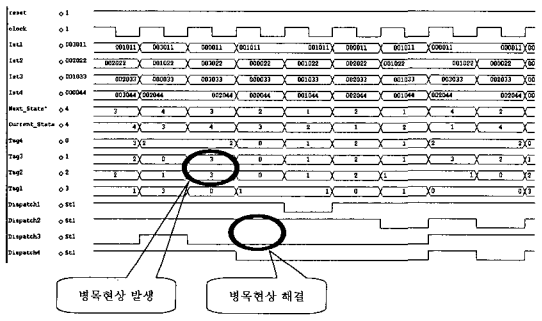


그림 13. 스케줄러 시뮬레이션 결과 파형
Fig. 13. Simulation Waveform of Scheduler.

이에 따른 스케줄러의 다음 상태를 표시한다. tag1~tag4가 모두 다를 때 Next_state는 4가 되며, 두개가 같은 경우 3, 세개가 같은 경우 2, 네개가 모두 같은 경우 1로 상태를 천이한다. 그림 13에 명령어 병목 현상의 발생에 따라 스케줄러가 병목 현상을 해결하여 동시에 수행 가능한 명령어를 각 해당 기능 유니트로 디스패치하는 것이 나타나있다.

시뮬레이션을 통하여 기능 검증을 완료한 모델은 레지스터와 게이트 레벨로 작성하여 공정에 상관없이 논리 합성이 용이하게 모델링하였으며 0.6㎛ SOG 라이브러리(KG75000)를 이용하여 SYNOPSYS™ 캐드 툴로 논리 합성하였다. GCP의 전체 게이트 수는 56,148개이며 worst case 동작 주파수는 SOG 라이브러리의 제약으로 인하여 30 MHz 이다. GCP의 게이트 수는 약 50,000개로서 기존의 멀티미디어 프로세서[1-10]의 게이트수가 수 백만개인 것과 비교하여 매우 적은 수준이다.

설계된 GCP의 성능을 평가하기 위해서 멀티미디어 알고리즘에 대하여 HDL모델을 통해 시뮬레이션을 수행하였다. 성능 평가를 위해 사용된 동작 주파수는 30 MHz이며 JPEG, MPEG 디코딩을 위한 DCT 연산과 MPEG 인코딩 과정의 움직임 추정 알고리즘을 벤치마킹 하였다. 움직임 추정 알고리즘은 FBMA[15]를 사용하였으며 기준 블럭의 영상크기는 16 × 16, 탐색 영역의 크기는 8로 가정하였고 DCT 연산은 Chen 알고리즘[16]을 사용하였으며 기준 블럭의 크기는 8 × 8로 가정하였다. 설계된 GCP는 CIF 영상 규격에 대하여 초당 21 프레임의 FBMA 연산과 63 프레임의 DCT 연산의 수행이 가능하며 QCIF 영상 규격에 대하여는 초당 83 프레임의 FBMA 연산과 256 프레임의 DCT 연산의 수행이 가능하다. 이는 CIF와 QCIF 영상 규격의 실시간 기준을 만족하는 성능이며 표 2에 측정된 성능이 나타나있다. GCP의 성능이 기존의 멀티미디어 프로세서에

표 2. 성능 평가
Table 2. Performance Evaluation.

알고리즘	성능
DCT (352 x 288)	63 frames/sec
DCT (176 x 144)	256 frames/sec
DCT (720 x 480)	18 frames/sec
FBMA (352 x 288)	21 frames/sec
FBMA (176 x 144)	83 frames/sec

비해 우수하지는 않으나 기존의 멀티미디어 프로세서는 수 백만개의 게이트로 구성되는데 반하여 GCP의 게이트수가 약 50,000개에 불과한 것을 감안하면 만족할 만한 성능이다. 또한 논리 합성 시 0.6㎛ SOG 라이브러리가 아닌 스탠다드 셀 라이브러리를 사용하고 최신의 공정을 사용하면 동작 속도를 증가시킬 수 있어 더욱 향상된 성능을 가질 수 있을 것이다.

V. 결 론

본 논문에서는 멀티미디어 전용 프로세서의 그래픽 전용 프로세서인 그래픽 프로세서(Graphic Coprocessor: GCP)를 설계하였다. 명령어 집합은 멀티미디어 데이타의 병렬성을 이용하기 쉬운 SIMD 및 슈퍼스칼라 등의 병렬 아키텍쳐 개념을 적용시켰으며 멀티미디어 데이타 처리에 적합한 형태로 설계하였으며 멀티미디어 알고리즘 구현 시 자주 사용되는 핵심 명령어 및 영상 신호처리에 효과적인 명령어들을 포함한다. GCP는 4개의 주 프로세서에 의하여 공유되는 형태로 설계되었으며 공유에 따른 명령어 병목 현상을 해결하기 위해 스케줄러를 내장하고 있다. 스케줄러는 기능 유니트별 라운드로빈 방식의 스케줄링 방식을 채택하였으며 이와 동시에 각 GP의 기능 유니트 사용 히스토리를 관리하여 한 GP가 특정 기능 유니트를 독점 사용하는 것을 방지한다. GCP는 기능 유니트인 GALU, GMUL, GBMU 및 GSAD를 내장하고 있으며 한 개의 데이터 경로가 목적에 따라 여러 개로 분할될 때 8, 16, 32, 64 비트 등의 다양한 데이터 형태의 연산 및 포화 연산을 지원한다. 설계된 GCP는 Verilog HDL을 이용하여 모델링하였으며 0.6㎛ SOG 라이브러리(KG75000)를 이용하여 SYNOPSYS™ 캐드 툴로 논리 합성하였고 Cadence™ 캐드 툴로 시뮬레이션 하였다. 전체 게이트 수는 56,148개이며 30 MHz로 동작하는 GCP는 CIF 영상 규격에 대하여 초당 63 프레임의 DCT 연산 및 21 프레임의 움직임 추정 알고리즘을 수행 할 수 있는 성능을 가진다.

참 고 문 헌

[1] Curtis Abbott et al., "Broadband Algorithms with the MicroUnity Mediaprocessor," in proc. *COMPCON*, Feb., 1996.

[2] Chromatic Research, Inc., "Mpact Data Sheet," 1995.

[3] Microprocessor Report, "Chromatic's Mpact2 Boosts 3D," pp. 5-10., Nov. 1996.

[4] Robert E. Owen, S. Purcell, "An Enhanced DSP Architecture for the Seven Multimedia Functions: The Mpact2 Media Processor," *IEEE Workshop on SiGNAL -Processing Systems*, pp. 76-85, Nov. 1997.

[5] Philips Semiconductors, "VLIW Computer Architecture," July 1996.

[6] Texas Instruments Inc., "TMS320C62xx User's Manual," 1997.

[7] Analog Devices Inc., "ADSP 2106x SHARC User's Manual," 1996.

[8] Lavi A Lev., Andy Charnas. et al., "A 64-bit Microprocessor with Multimedia Support," *IEEE Journal of Solid-State Circuits*, vol. 30 no. 11, pp. 1227-1238, Nov. 1995.

[9] A. Peleg and U. Weiser, "MMX Technology Extension to the Intel Architecture," *IEEE Micro*, vol. 16 no. 4, pp. 42-50, Aug. 1996.

[10] Lee R., "Real-time MPEG Video via Software Decompression on a PA-RISC Processor," *Proceeding of IEEE COMPCON*, pp. 186-192. Mar. 1995.

[11] M. Trembley, M. O'Connor, V. Narayannan, L. He, "VIS Speeds New Media Processing," *IEEE Micro*, vol. 16 no. 4, pp. 10-20, Aug. 1996.

[12] Intel co., "Intel Architecture MMX™ Technology," Mar. 1996.

[13] R. Lee, "Subword Parallelism with MAX2," *IEEE Micro*, vol. 16 no. 4, pp. 51-59, Aug. 1996.

[14] Silicon Graphics Inc., "MIPS Extension for Digital Media with 3D," Dec. 1996.

[15] J. S. Baek, S. H. Nam, and M. K. Lee, "A Fast Array Architecture for Block Matching Algorithm," in Proc. *IEEE ISCAS*, London,

U.K., May 1994, pp. 211-214.

[16] Chen W. H., Harrison S. C., and Fralic S. C., "A Fast Computational Algorithms for the Discrete Cosine Transform," *IEEE Trans. on Comm.*, vol. COM-25, no. 9, Sep., 1977.

저 자 소 개

高 翊 相(正會員)

1997. 2 : 아주대학교 전자공학 학사.
1999. 8 : 아주대학교 전자공학 석사.
1999. 8~현재 : (주) 삼성전자. ※ 관심분야 : DSP 칩, 영상 및 신호처리용 ASIC 설계

鮮于 明勳(正會員) 第 34卷 C編 第 8號 參照

韓 宇 宗(正會員) 第 34卷 B編 第 5號 參照