論文99-36C-9-1

RISC-based DSP의 효율적인 임베디드 메모리 인터페이스

(Efficient Interface circuits of Embedded Memory for RISC-based DSP Microprocessor)

金 裕 鎭 *, 趙 慶 錄 **, 金 聖 植 **, 鄭 義 錫 *

(You Jin Kim, Kyoung Rok Cho, Sung Sik Kim, and Eui Seok Cheong)

요 약

본 논문에서는 GMS30C2132마이크로프로세서에 DSP연산을 위하여 128K bytes EPROM과 4K bytes SRAM을 내장하고, 이 과정에서 내/외부 메모리 인터페이스 부분이 프로세서와 1싸이클 엑세스가 이루어지도록 버스 제어 인터페이스 구조를 설계하였다. 내장된 128Kbytes EPROM은 메모리 구조 및 데이터 정렬에 따른 동작을 위해 새로운 데이터 확장 인터페이스 구조와 테스트를 위한 인터페이스 구조를 제안 하였으며, 내장된 4K bytes SRAM은 프로세서와 인터페이스를 할 때 DSP 고속 연산에 활용하기 위해 메모리 스택으로써의 이용과 명령어 캐쉬와의 인터페이스, 가변 데이타 크기 제어, 모듈로4Kb의 어드레싱이 가능한 구조를 채택하여 설계하였다. 본 논문의 새로운 구조 적용으로 내장EPROM, SRAM에서 평균 메모리 엑세스 속도가 종전의 40ns에서 20ns로 감소하였고, 가변 데이타 버스 인터페이스 제어로 프로그램 처리 속도가 2배로 개선 되었다.

Abstract

In this paper, we designed an embedded processor with 128Kbytes EPROM and 4Kbytes SRAM based on GMS30C2132 which is RISC processor with DSP functions. And a new architecture of bus sharing to control the embedded memory and external memory unit is proposed aiming at one-cycle access between memories and CPU. For embedded 128Kbytes EPROM, we designed the new expansion interface for data size at data ordering with memory organization and the efficient interface for test. The embedded SRAM supports an extended stack area for high speed DSP operation, instruction cache and variable data-length control which is accessed with 4K modulo addressing schemes. The proposed new architecture and circuits reduced the memory access cycle time from 40ns to 20ns and improved operation speed 2-times for program benchmark test. The chip is occupied 108.68㎟ using 0.6㎛ CMOS technology.

* 正會員, 韓國電子通信研究院 交換電送技術研究所
(Electronics and Telecommunications Research Institute Switching & Transmission Technology Laboratory)
** 正會員, 忠北大學校 情報通信工學科
(Department of Computer and Communications Eng., Chungbuk National University)
接受日字 : 1998年11月30日, 수정완료일 : 1999年8月6日

I. 서 론

멀티미디어 시스템들이 소형화 경량화로 발전함에 따라 저전력화, 고속화가 요구되고 있으며, 이에 따라 시스템의 기능을 온-칩화 하는 시스템IC개발이 이루어지면서 32비트 메모리 내장식 프로세서 등의 사용이 증가되고 있다. 일반적으로 32비트 내장식 프로세서는 RISC CPU에 DSP 기능을 탑재하고 주요 기능을 소프트웨어로 구현하며 수행 속도의 고속화 및 효율적인

실시간 데이타 처리를 할 수 있다. 여기서 RISC프로세서는 실시간의 제어 동작을 처리하며, DSP는 집약적인 데이타의 처리, 임시 데이타 및 어드레스 생성, 동일 데이타의 반복적인 사용, 빠른 곱셈 및 귀한 경로의 수식 연산을 처리하는 장점이 있다. 예로, 유럽의 GSM이동통신 단말기에서 프로토콜 소프트웨어는 RISC가, 음성과 통신 소프트웨어는 DSP가 그 기능을 담당하여 실시간 처리를 하는 프로세서를 사용하고 있다.[1,2,3,4,5] 또한 RISC와 DSP기능을 갖는 프로세서를 내장하는 시스템의 경우 시스템 프로그램 용량이 커지고 있으므로, 이러한 시스템 소프트웨어를 저장할 효율적인 내장 메모리 인터페이스에 대한 연구가 필요하게 되었다.

본 논문에서는 통신 및 멀티미디어에 적합한 32비트 RISC-based DSP[6] 프로세서인 GMS30C2132[7]을 기본 아키텍쳐로 설정하였다. 통신 및 멀티미디어 프로그램은 일반적으로 시스템 제어 프로그램 뿐만 아니라, 수식연산을 처리하는 DSP알고리즘의 수행을 한다. 이런 한 코드 데이터를 내장하고 DSP알고리즘을 효과적으로 수행하기 위해서는 1싸이클의 엑세스 여부가 대단히 중요하다. 본 논문의 제안된 설계로 내장메모리 EPROM, SRAM 모두 1싸이클 엑세스가 가능하다.

설계된 인터페이스 회로는 프로세서에 내장된 EPROM과 범용EPROM라이터 간에 프로그램 시 메모리 구조에 따른 데이터 정렬에서 외부와 확장 데이터 인터페이스로 프로그램을 가능 하도록 설계하였고, 테스트를 위한 인터페이스도 설계하였다. 내장SRAM과 프로세서간의 인터페이스는 내장SRAM을 메모리 스택으로 활용할 수 있도록 하고, 코드 효율을 높일 수 있도록 가변데이타 크기로 제어하였다. 또한 명령어 캐쉬와의 효율적인 인터페이스를 제공하였다. 원형버퍼로써의 명령어 캐쉬는 어드레스의 시작 번지와 끝 번지를 연결하였는데, DSP유닛 사용에 필요한 데이타 어레이를 저장하여 고속 연산에 활용 하도록 제안하고 프로세서와의 1싸이클 엑세스가 이루어 지도록 설계하였다.

본 논문은 제2장에 임베디드 프로세서의 구조, 제3장에는 내장 EPROM인터페이스, 제4장에는 내장 SRAM인터페이스, 제5장에 모의 실험 및 결과에 대하여 기술한다.

Ⅱ. 임베디드 프로세서의 구조

본 논문의 기본 CPU인 GMS30C2132아키텍쳐[7]에 마스크롬, EPROM, 플레쉬메모리등의 비휘발성 메모리와 SRAM, DRAM과 같은 휘발성 메모리를 내장하기 위해서 CPU와 각 메모리 사이의 효율적인 기능의 인터페이스 및 전체 성능에 영향을 주는 내장 메모리와 CPU간의 엑세스 시간의 고속화가 대단히 중요하다.

그림 1은 설계된 MCU의 구조로 사양은 표 1과 같다. 설계된 MCU는 2단 파이프라인을 가지며, 첫번 째 싸이클에서는 명령어를 캐쉬로부터 읽어서 해독하고, 두 번째 싸이클에서는 그 명령어를 실행한다. ALU와 DSP unit은 독립적으로 명령어를 수행하고 고속의 DSP알고리즘 수행을 위해 대부분의 일반 명령어와 DSP명령어를 병렬적으로 1싸이클에 수행할 수 있도록 설계하였다.

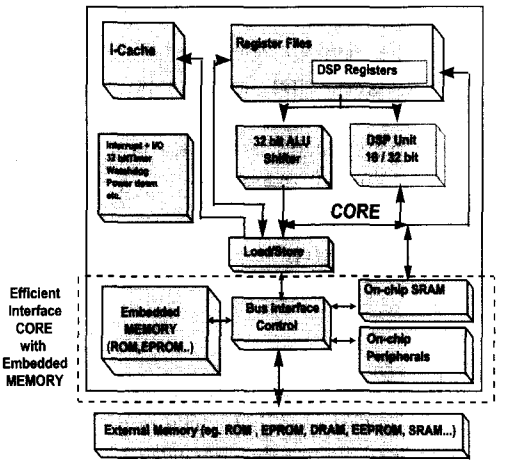

그림 1. 시스템 설계의 인터페이스
Fig. 1. Interface for system design.

표 1. 설계된 MCU의 사양
Table 1. Overview Architecture of designed MCU.

GMS30C2132 CORE Architecture(RISC CPU+DSP)
• Variable Instruction Length(16/32/48-bit) • 2 stage Pipeline(Decode/Execution) • Large Register Files(32 global 64 local Reg) • Combining RISC and DSP functionality • 16/32 Bit Fixed Point DSP Unit • Instruction Cache (128Bytes) • 4G Byte Memory Space
Embedded Memory
• 4Kbytes Embedded SRAM • 128Kbytes Embedded EPROM

이 경우 내장된 EPROM과 프로세서간에는 32비트 데이터 버스 크기로 인터페이스를 가지도록 설계 하였다. DSP알고리즘 수행을 위해서는 메모리에서 버스로 데이터나 명령어를 실어 줄 때, 1싸이클에 엑세스가 가능하도록 하여야 한다. 설계된 메모리 인터페이스 블럭도는 칩 내부에서 SRAM이 CPU와 버스 인터페이스 제어 블럭에 직접 연결되고 EPROM은 버스 인터페이스 제어 블럭을 통하여 CPU와 연결하였다.

설계된 MCU는 명령어가 가변 길이로 데이터의 버스 크기에 따라 하위 비트 데이터를 기준으로 정렬 하므로 바이트별로 사용하는 데이터 버스의 팬인, 팬 아웃의 부하 캐패시턴스의 값이 다르므로 버스 선이 바이트별로 지연이 있을 수 있다. 본 연구에서는 그림 2(a)에서 나타낸 바와 같이 기존의 일반적인 인터페이스에서는 한 개의 데이터 버스가 여러 블럭을 공유하여 버스 바이트 단위 별로 라우팅과 팬인 팬 아웃이 차이가 날수 있다. 이에 대한 개선으로 그림 2(b)와 같이 데이터를 래치하여 바이트별로 버스를 동기화 제어하고 반복셀(Repeat Cell)을 연결하여 지연경로를 최소화 하였으며, 내부 버스를 일관된 모양으로 정리하여 부하 캐패시턴스를 분산 출력 시켜 주는 새로운 BUS인터페이스 제어 블럭을 적용하였다. 이러한 구조 적용시 동기화에 따른 싸이클 손해 latency가 있을 수 있으므로 제어시 그림 2(b)에 나타난 타이밍도 처럼 제어를 하여

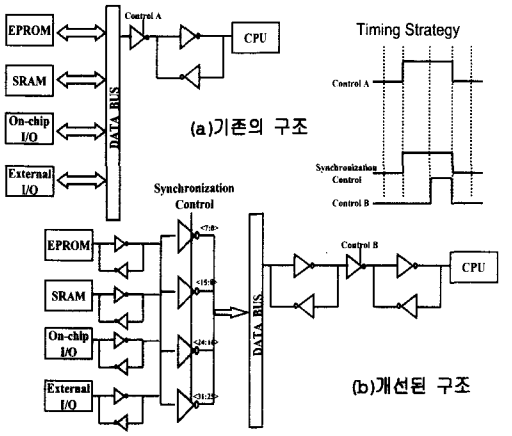

그림 2. 데이터 버스 기생 캐패시턴스 감소를 위한 인터페이스 구조

Fig. 2. Interface architechture for decreasing parasitic capacitance in data bus.
(a) Conventional architecture
(b) Proposed architecuture

latency가 없도록 하였다. 그림 3에서 LPE결과에 의하면 기존의 구조는 비트에 따라 최하위 비트와 최상의 비트가 13pF정도의 차이가 난다. 이것은 엑세스 주기에 제약 조건으로 작용 하였으나, 제안된 설계에서는 해당 바이트, 비트별로 기생캐피시턴스가 차이가 나지 않아 20ns 주기인 1싸이클 엑세스에서 동작이 가능 하였다.

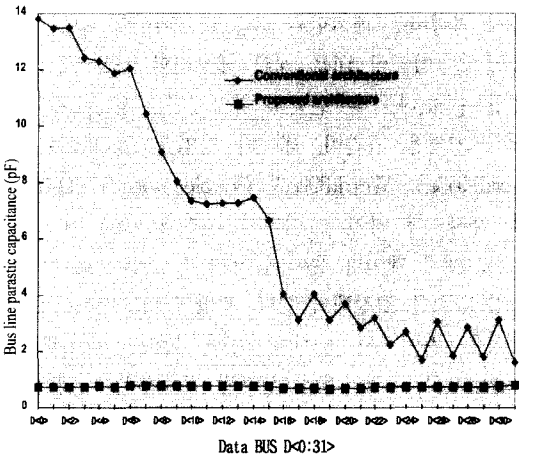

그림 3. 비트 단위별 버스 기생 캐패시턴스 분포

Fig. 3. Distribution of parasitic capacitance per data bit line.

III. 128Kbytes Embedded EPROM 인터페이스 설계

본 논문에서는 CPU인 GMS30C2132아키텍쳐에 128Kbytes EPROM을 내장하기 위해서는 먼저 CPU가 가지는 명령어 형식에 따른 메모리 구조를 정의 한다. 또한 설계된 메모리 구조를 바탕으로 데이타의 정렬 순서를 정의하고, 효율적인 메모리 인터페이스를 위해서 데이타 확장 인터페이스 구조, 내장EPROM테스트 인터페이스 구조 및 로직1의 입력데이타 프로그램 방지 회로 등의 새로운 구조를 제안하고 설계하였다.

1. 메모리 구조 및 데이타 정렬

일반적인 RISC프로세서(non-DSP 프로세서)는 폰 뉴만아키텍쳐 타입으로 명령어와 데이타를 위한 한 개의 데이타 버스를 가진다. 그러나, DSP기능을 가진 프로세서는 프로그램 명령어를 위한 메모리 블럭과 데이타를 위한 메모리 블럭을 독립적으로 가지는 하버드아키텍쳐 타입을 주로 가진다.

설계한 프로세서는 RISC based DSP 타입으로 일반

명령어의 길이는 16비트이다. 그러나, 가변 길이 명령어 체계를 수용한 설계는 DSP 연산을 위한 확장된 명령어와 직접 값 또는 상수의 크기나 어드레스 변위에 따라 명령어의 길이가 32비트, 48비트로 확장되도록 한다. 이것은 기존의 32비트 고정 길이 명령어를 가지는 프로세서 보다는 프로그램 코드의 밀도를 높일 수 있는 특징이 있다. 이는 코드 데이타가 저장된 메모리 블럭에서 명령어를 페치할때 사용되는 싸이클 수를 반으로 줄일 수 있으며 내부 캐쉬 메모리의 사용 효율을 더 높일 수 있다.[8] 본 논문의 프로세서는 4Gbytes의 어드레스 영역을 가지며, 데이타 타입은 Byte(8bit), Half word(16bit), Word(32bit), Double-word(64bit)의 형태를 가지도록 하였다. 따라서 내장EPROM 메모리에는 프로그램과 데이타 등이 동시에 저장 되므로 메모리의 저장 방식을 결정해야 한다. 일반적으로 메모리의 저장 방식은 바이트의 순서에 따라, little-endian과 big-endian으로 나뉘어지나, 설계된 프로세서는 데이타 워드의 마지막 바이트가 메모리의 한 어드레스에서 MSB에 저장되는 big-endian만을 지원하는 메모리 구조로 설계를 하였다. 이는 내장 메모리에 프로그램 시 또는 CPU가 데이타를 읽을 시에 버스에 실리는 데이타의 순서를 데이타의 마지막 바이트가 데이타 버스에 가장 먼저 실리도록 함을 의미한다.[9,10]

그림 4는 big-endian을 지원하는 메모리 구조에서 비트의 크기에 따라 가변 데이타의 엑세스 방법을 나타낸다. 메모리에 byte크기의 데이타 저장 시 입력데이타의 최하위 바이트인 aa가 메모리의 최상위 바이트에 쓰여지고, 메모리에서 데이타 버스로 값을 읽을 때 메모리의 최하위 바이트인 dd가 버스에 먼저 실린다. 또한 Half-word크기의 레지스터에서 메모리로 데이타 저장 시는 입력 데이타의 하위aabb가 메모리의 상위에 쓰여지고, 메모리에서 데이타 버스로 값을 읽을 때 메모리 하위ccdd가 버스에 먼저 실려 CPU의 상위 레지스터로 쓰여진다. 그러나, word크기의 데이타 경우, 메모리의 aabbccdd가 버스에 실려 같은 값이 레지스터에 쓰여진다.

그림 5에서는 설계된 메모리 구조에 따라 범용 EPROM라이터가 지원해주는 8,16,32,64비트 데이타 크기에 대하여 프로그램시에 데이타가 버스에 실려 데이타가 정렬되는 모습을 나타낸다. 그림 5(a)의, 8비트 데이타 크기인 경우는 4싸이클에 거쳐 데이타 값이 실리며, 범용EPROM라이터에서 나오는 데이타 HEX값이 aabbccdd면 각 싸이클의 진행마다 dd,cc,bb,aa의 순서대로 내부 버스에 실린다. 이때, 해당되지 않는 나머지 비트의 값은 1로 채워져 버스에 실린다. 그림 5(b)의 16비트 데이타 크기인 경우, 2싸이클에 거쳐 버스에 실린다. 내장 메모리의 테스트 프로그램 시간을 단축하기 위한 데이터 확장으로는 그림5에 동작을 나타냈다. 그림 5(c)의 32비트 데이타 사이즈의 정렬은 16비트 데이타를 2번 반복하여 버스에 실어 준다. 그림 5(d)의 64비트 데이타 사이즈의 데이타 정렬은 16비트의 데이타 값이 4번 반복되어 범용EPROM라이터의 입력이 aabbccdd이면 첫번째 싸이클에서는 ccdd가 4번 반복되어

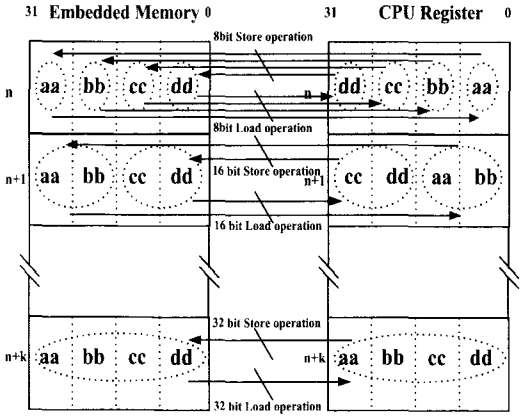

그림 4. 메모리에서 가변 데이터의 엑세스 방법
Fig. 4. Access method of variable data size between embedded memory and CPU register.

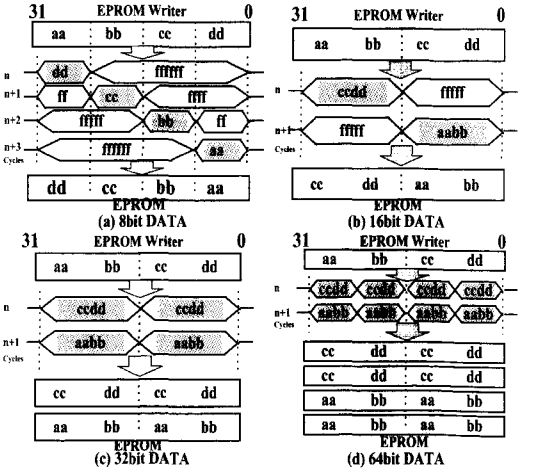

그림 5. 가변 데이터 크기에 따른 EPROM라이터에서 내장 EPROM으로 내부 데이터 정렬
Fig. 5. Internal data bus ordering of variable data size from EPROM writer.

ccddccddccddccdd의 64비트 크기가 EPROM에 쓰여진다. 같은 방법으로, 두번째 싸이클에는 aabb가 4번 반복되어 EPROM메모리에 프로그램 된다.

2. 내장EPROM의 데이타 확장 인터페이스 구조

데이타 확장 인터페이스는 CPU와 내장메모리의 인터페이스, 내장MCU와 외부 시스템과의 인터페이스 등으로 나누어 동작 모드를 설계해야 한다. 동작 모드의 결정은 상위 수준인 시스템 레벨에서의 결정 뿐만 아니라, 내부의 내장 메모리와 CPU간에 인터페이스 설계시 CPU의 동작 요구에 따라 동작 모드의 설계를 달리할 수 있다. 제안된 프로세서에서의 동작모드는 내장EPROM동작모드와 내장EPROM테스트 동작모드로 나누어서 설계하였다.[11~17]

내장EPROM동작모드는 128Kbytes의 EPROM block을 프로그램 하고 검증하기 위한 모드이다. 이 모드에서는 EPROM블럭과 EPROM 제어 블럭을 제외하고 칩 내부의 모든 CPU 및 주변 기능 블럭의 신호는 리셋후의 초기상태로 되어야 한다. 프로그램 및 검증시에는 일반적으로 범용EPROM 라이터를 이용한다. 범용 1Mbits EPROM메모리의 경우는 데이타 버스 크기에 따라 8비트, 16비트로 나눌 수 있으므로 이 두 가지 버스 크기의 프로그램 방식을 모두 지원할 수 있어야 한다. 이것을 위해 입력 어드레스의 LSB를 제어 신호로 사용한다. 가변 데이타 버스 크기로 제어할 수 있는 데이타 확장 인터페이스 구조는 그림 6의 어드레스 쉬프터블럭, 그림 7의 읽기제어블럭, 쓰기제어 블록을 설계하였다. 설계된 칩의 내장EPROM 테스트 동작 모드에서는 제품 양산 테스트시 효율성을 증진 시키기 위하여 설계된 가변 데이타 버스 크기로는 32비트, 64비트 크기의 프로그램이 될 수 있도록 하였다.

그림 6에서 보는 바와 같이 어드레스 쉬프터블럭은 내장 EPROM 어드레스를 A16~A0으로 고정 시키고, 외부와 인터페이스 되는 어드레스를 8비트, 16비트의 데이타 크기에 따라 각각 A16~A0, A15~A0으로 EPROM라이터와 인터페이스 시키게 하였다. 8비트 데이타 크기에서 입력 어드레스가 A16~A0인 경우 내장 EPROM어드레스에 그 대로 맵핑시킨뒤 어드레스의 최하위 A1, A0을 8bit 데이타를 버스에 실어 줄 때 정렬을 하는 제어 신호로써 사용한다. 또한 16비트 데이타 크기에서 입력 어드레스가 A15~A0인 경우 한 비트씩 왼쪽으로 이동 시켜 EPROM라이터와 인터페이스 되는 포트 어드레스A15가 내장 EPROM어드레스 A16으로 맵핑되도록 한다. 이에 따라, 내장 EPROM어드레스 A1을 가지고 16비트 데이타를 버스에 실어줄 때 정렬하는 제이 신호를 만든다. 테스트 모드인 경우는 내장 EPROM어드레스의 32비트 프로그램시에는 A1을, 64비트 프로그램시에는 A3를 제어 신호로써 사용한다.

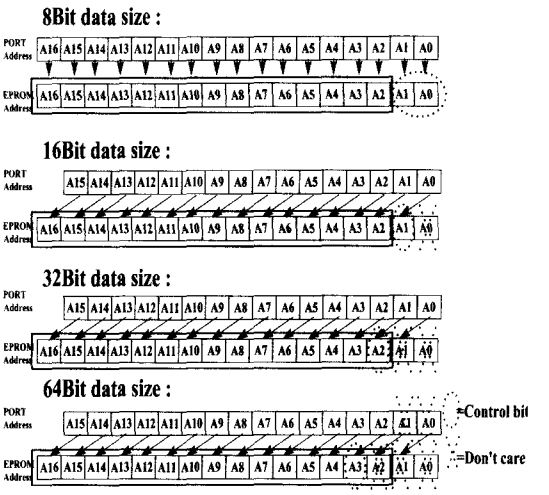

그림 6. 어드레스 쉬프터 블럭
Fig. 6. Address shifter block.

그림 7은 읽기, 쓰기제어 블록이다. 읽기제어 블럭은 내장 EPROM블럭에서 나온 32비트 크기의 데이타가 EPROM라이터로 갈 때 포트에 실리는 값을 8비트 또

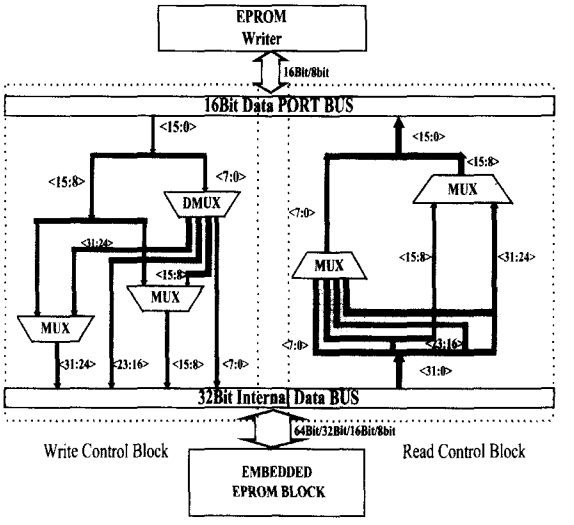

그림 7. 읽기 제어 블록, 쓰기 제어 블럭
Fig. 7. Read control block/Write control block.

는 16비트 크기로 나누어 실리도록 제어하는 블럭이다. 그림 5(a)와 같이 8비트 크기인 경우는 EPROM메모리에 저장된 ddccbbaa가 4싸이클에 걸쳐 차례로 aa,bb, cc,dd로 EPROM라이터로 출력이 되도록 한다. 16비트 크기인 경우는 ccdd,aabb가 EPROM라이터로 출력이 되도록 하는 제어이다. 쓰기 제어 블럭은 EPROM 라이터에서 들어온 8,16비트 데이타를 동작모드에 따라 8,16,32,64비트 데이타 크기로 변환하는 제어 블럭으로 그림 5와 같은 데이타 정렬을 한다.

3. 내장EPROM테스트 인터페이스 구조

본 논문은 웨이퍼 상태에서 내장된 EPROM을 쓰고, 읽고, 지우는 테스트와 패키지 상태에서 쓰고, 검증 하는 테스트로 나누어 설계를 하였으며, 웨이퍼 상태에서의 쓰기는 32비트 프로그램, 64비트 프로그램을 할 수 있도록 하였다. 1Mbits의 EPROM 전체 셀어레이를 프로그램/검증동작 확인시 많은 시간이 필요하게 되는 문점이 있다. 이러한 문제점을 해결하기 위해서는 제조회사의 양산 과정에서 테스트할 때 프로그램 시간을 단축 하기 위한 제안이 필요하다.

설계된 제안에는 해당 어드레스의 LSB를 제어 신호로 사용하여 32비트 프로그램은 16비트 데이타를 동시에 2번 프로그램하며 64비트 프로그램은 16비트 데이타를 동시에 4번 프로그램하여, 128KBytes의 내장 EPROM의 프로그램 시간을 1/2로 단축하기 위한 새로운 구조로 설계하였다.

그림 8은 read dumping mode의 동작을 나타내는 블럭으로 read dumping mode가 아닌 경우는 CPU에서 나온 어드레스를 EPROM어레이 어드레스로 이용하나, read dumping mode가 인에이블 되는 경우 가변 주파수 블럭에서 발생되는 클럭 소스를 바탕으로 어드레스 발생기에서 어드레스를 만들어 EPROM어레이에 보내지게 된다. 이때 모든 CPU의 신호를 리셋시키고 어드레스 발생기에서 어드레스가 발생되면 임베디드 메모리의 내용을 순서대로 출력을 시킨다. 이는 가변 주파수 블럭에서 클럭의 동작 주파수를 제어 하여 내부 행, 열디코더, 센스 앰프의 동작 속도를 측정할 수 있을 뿐 아니라, EPROM 셀 전체 어레이의 프로그램 특성 유무를 알 수 있어 웨이퍼 테스트의 검증용으로 사용할 수 있는 모드이다.

4. 로직1의 입력 데이타 프로그램 방지회로

EPROM라이터에서 나온 8bit또는 16bit값이 데이타 버스에 실려 정렬될때, 내부32비트 데이타 버스는 입력된 해당 데이타 비트를 제외한 나머지 비트에 로직1값으로 쓰여지도록 하여 겹쳐 쓰기를 금지하도록 하는데 EPROM 셀이 프로그램 전에 로직1의 상태이기 때문이다. 그림 9에서와 같이 EPROM 셀을 프로그램하기 위해서는 셀의 게이트에 12.75V, 드레인에 7.7V의 고전압이 인가되어 부유게이트로 hot-electron이 주입된다. 전자 주입에 따라 프로그램 전(로직1)에는 문턱 전압이 1.3V이며, 프로그램 후는 문턱 전압이 0.4V로 낮아져 로직0으로 프로그램 된다. 따라서, 입력 데이타 값이 0인 경우는 셀의 값이0이 되도록 하나, 1이 입력인 경우는 프로그램이 금지되도록 하는 회로가 있어야 한다.[18] 그림9은 쓰기 인에이블 신호가 1이고 입력데이타가 0

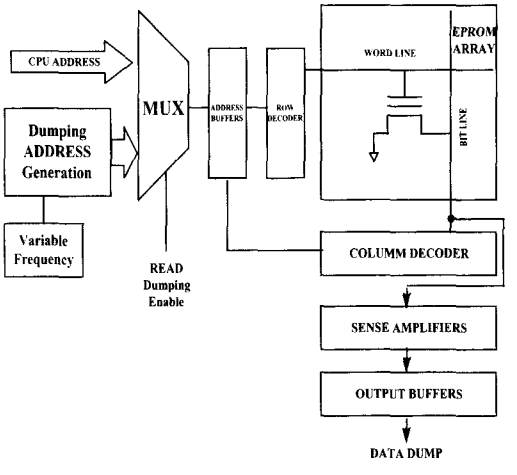

그림 8. 읽기 덤핑 모드
Fig. 8. Read Dumping mode.

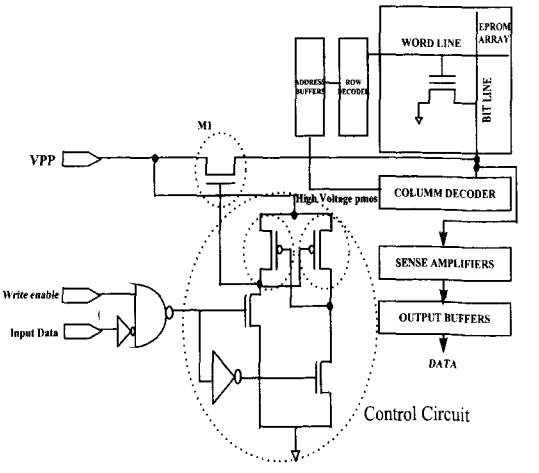

그림 9. 로직 1의 입력 데이터 프로그램 방지 회로
Fig. 9. Prohibition circuits of programming logic 1.

일 때 제어 회로 출력이 1이 되어 고전압 트랜지스터 M1을 on시켜 드레인 출력이 8V정도 되도록 하고 입력 데이터가 1인 경우는 M1을 off하는 회로이다.

Ⅳ. 4Kbytes Embedded SRAM 설계

일반적으로 MCU에 메모리를 내장시 DRAM으로 설계할 수 있으나, 리플레쉬가 필요하여 전력소모의 요인이 된다. 본 논문에서는 DRAM을 대신해서 0.6㎛의 CMOS공정으로 SRAM을 이용하여 4Kbytes의 메모리를 내장 시켰다. SRAM은 리플레쉬가 필요 없고, 휘발성 메모리이고 엑세스 속도를 높일 수 있어 내장된 SRAM설계에서는 메모리 스택으로 사용 가능한 인터페이스, 명령어 캐쉬와의 인터페이스, 가변 데이터 크기 제어 인터페이스, 그리고 modulo 4K wraparound 방식의 동작이 이루어 지도록 내장 SRAM인터페이스를 설계하였다.

1. 메모리 스택 인터페이스

스택은 데이터를 저장할 수 있는 임시 기억 장소로 레지스터 스택과 메모리 스택으로 나누어 설계를 하였다. 레지스터 스택은 2개의 읽기/쓰기 포트를 가진 레지스터 파일이며 64개의 지역 레지스터로써 원형 버퍼로 구성되는데, 시스템 정보나, 임시 기억 장소로 사용된다. 메모리 스택은 내장 SRAM을 이용하여 필요한 스택 영역이 레지스터 파일의 개수를 초과할 때 사용되도록 인터페이스를 설계하였다. 내장 SRAM을 메모리 스택으로 이용하는 장점은 스택의 길이를 제어할 수 있으며, 1개 이상의 스택 설정이 가능해서 한정된 레지스터 스택에 비해 임시 기억 장소를 많이 제공할 수 있어 성능을 향상시킬 수 있다.

그림 10은 레지스터스택과 메모리스택의 PUSH, POP되는 과정이다. 그림 10의 레지스터 스택은 가장 최근의 스택영역을 유지하며, RSP는 현재 스택의 시작점을 알려 주는 레지스터 스택 포인터이고 MSP는 메모리 스택 포인터이다. 레지스터 스택에서는 현재 사용하는 스택이 할당되므로, 레지스터 파일에 새로운 스택을 할당 하고자 할 때, 현재의 스택 영역을 초과하면, 초과된 스택 영역 만큼 겹쳐 써지는 기존의 레지스터 스택은 메모리 스택으로 PUSH된다. 새로운 스택 영역을 해제하면 이전의 스택 영역으로 메모리 스택에 있는 기존의 스택이 레지스터 스택으로 POP되어 새로운 스택 할당 이전의 상태가 된다. 그림10은 원형 버퍼로 구성된 64개의 레지스터 파일에서 RSP를 레지스터 스택 시작 포인트로 하여 새로운 스택A를 할당 하고자 하는데, 스택 영역을 초과하여 기존의 스택B를 겹쳐 쓰게 된다. 이때, 겹쳐 쓰여지는 기존의 스택B는 자동적으로 메모리스택으로 PUSH 되고 MSP를 메모리스택 시작 위치에 놓는다. 새로 할당된 스택A를 해제할 때, 스택A는 비워지고 이때 메모리 스택에서 기존의 스택B가 레지스터스택으로 POP되어 MSP는 원래의 위치로 가고, RSP는 해당 레지스터 스택 위치에 있게 된다.

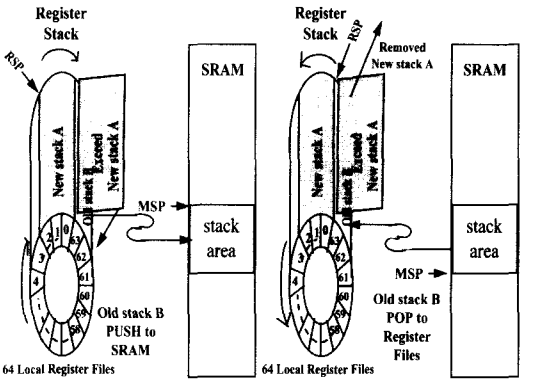

그림 10. 레지스터스택과 메모리스택 동작
Fig. 10. Operation between register stack and memory stack.

2. 명령어 캐쉬 인터페이스

내장SRAM은 명령어 캐쉬와의 효율적인 인터페이스를 이용하여 전체 성능 향상에 기여하도록 설계하였다.[19,20] 내장SRAM에 명령어와 데이터를 저장할 경우 SRAM으로부터 나오는 명령어는 한 사이클 안에 CPU와 인터페이스가 가능하여 자동적으로 명령어 캐쉬로 이동 된다. 이러한 동작은 외부 메모리 엑세스시 서로 방해하지 않도록 분리된 파이프라인을 두어 설계하였다.

RISC에 기반을 둔 DSP 구조를 설계할 때, DSP의 경우 메모리 블럭을 빈번히 이용하고 반복적인 프로그램 루프가 많이 사용되어 명령어 캐쉬대신 명령어 버퍼가 많이 사용된다.

설계된 명령어 캐쉬는 명령어 버퍼 역할을 하며, 루프버퍼(loop buffer)형식의 구조이다. 루프버퍼는 프리페치의 역할을 수행하여, 현재 페치한 명령어 주소로부

터 순차적으로 계속 명령어를 페치한다. 이러한 동작으로 128바이트의 명령어 캐쉬는 적은 용량크기이지만 히트 율을 높일 수 있다.

그림 11은 설계된 명령어캐쉬의 읽기 동작을 나타낸다. 읽기 동작 시에는 프로그램 카운터로부터 읽기 어드레스가 입력되면 halfword단위(16bit)로 3개씩 읽어서 48bit를 출력하게 되는데 32비트 크기 의 데이터가 입력 될 때 OUT1에서는 오피코드(opcode)를 출력하고, OUT2에서는 오퍼랜드(operand)출력을, OUT3는 48비트 명령의 경우 오퍼랜드로 출력을 하게 된다. 즉, 6751c000-0ffcee60와 같은 Hex값의 데이터가 입력되면 48비트 명령어 경우 각각 OUT1에는 6751이 OUT2에는 c000가 OUT3에는 0ffc가 각각 출력되어 명령어 디코더로 보내진다. 쓰기 동작 시에는 레지스터 스택에서 쓰기 어드레스가 입력되어 해당캐쉬 셀을 선택하여 word단위(32bit)로 저장이 된다.

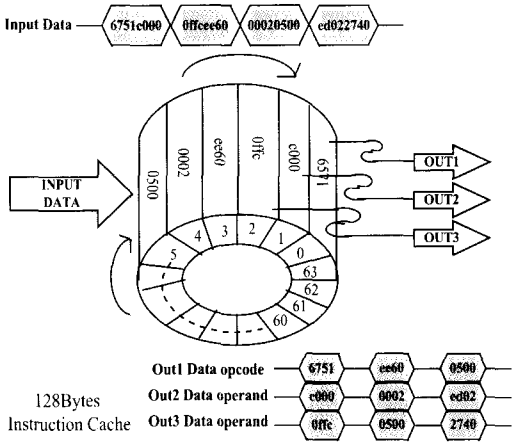

그림 11. 원형 버퍼 구조로 동작하는 명령어 캐쉬
Fig. 11. Instruction cache as circular buffer.

3. 가변 데이터 크기 제어 인터페이스

SRAM을 내장함에 있어서 하드웨어 측면에서 메모리 셀의 크기가 기본적으로 기타 메모리 셀보다는 크므로, 물리적인 용량의 증가에는 한계가 있다. 따라서, 데이터를 저장할 수 있는 코드 밀도를 높여야 하며, 이를 위한 설계가 가변 데이터 크기의 인터페이스 설계이다.

본 논문에서는 명령어의 길이에 따라, byte, half-word, word로 내장 SRAM에 데이터를 저장할 수 있도록 하여 코드 밀도를 높일 수 있도록 설계하였고, 버스의 이용률을 줄여 로드/스토어 명령어 수행 시 엑세스 속도를 효율적으로 높일 수 있다. 그림12는 SRAM셀 2개, 4개의 래치, 3개의 멀티플렉서로 1개의 블럭을 구성하고 이 블럭을 32개 구성한다. 데이터의 저장은 쓰기 제어에서 입력되는 데이터를 각 데이터의 크기 제어 신호에 따라 8,16,32비트의 크기로 값을 저장한다.

읽기 제어에서는 출력되는 데이터를 8,16,32비트 크기로 각 데이터 크기에 따라 변화시켜 주는데, 해당되지 않은 MSB의 바이트 영역은 sign bits에 의해 1또는 0으로 채워진다. 이 구조에서 1개의 블럭 안에 있는 쓰기 멀티플렉서가 내장 SRAM에 입력 데이터를 쓰고 있을 때, 선택 되지 않은 셀에 자신의 값을 다시 쓰여지게 하였다. 즉, 쓰기 신호에 의하여 m1과 m2가 on되고 write latch의 데이터가 S1셀에 저장될 때 S2셀은 Read latch에 저장된 데이터가 MUX B, m2를 통하여 S2에 다시 쓰여지는 동작을 한다. 이는 가변 데이터를 위하여, 쓰기 제어 블럭에서 선택되지 않은 바이트는 버스 충돌을 방지하기 위해 멀티플렉서의 출력을 하이 임피던스 상태로 만들어 준다. 이때 가변 데이터 크기인 경우 선택되지 않은 MSB영역에 자신의 값을 다시 입력하게 함으로써, 정의되지 않은 값에 의한 데이타값 손실을 방지하고, 셀에 지속적으로 값을 유지할 수 있다.

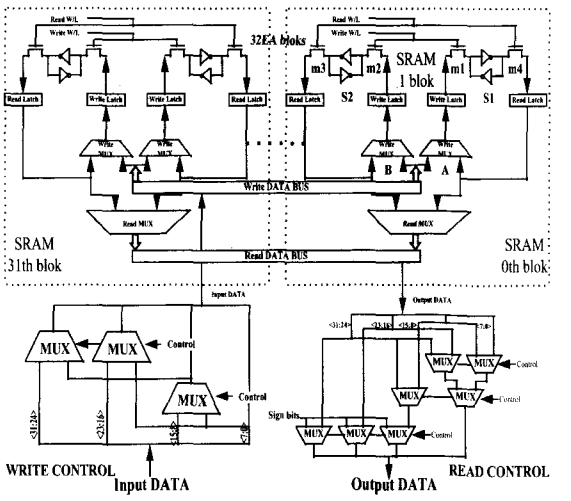

그림 12. 내장 SRAM 가변 데이터 크기의 인터페이스
Fig. 12. Variable data size interface of embedded SRAM.

4. Modulo-4K wraparound 방식의 어드레싱

내장 SRAM의 할당된 메모리 공간은 어드레스가 modulo 4K wraparound 방식으로 동작 되어 DSP 알고

리즘 수행에 적합하도록 하였다. 이것은 DSP알고리즘 특징인 FIR, IIR필터, FFT, DCT, DST변환 알고리즘을 내장 SRAM에서 수행할 경우 외부 메모리 사용보다는 실행 속도를 빠르게 할 수 있다. 내장SRAM은 DSP알고리즘 수행 시 임시적인 데이터 버퍼로써의 메모리 관리가 이루어 진다. 따라서, 동적인 메모리 할당을 해야 하는 실시간 시스템에서는 프로그래머가 일반적으로 주어진 버퍼 메모리 용량에 맞도록 크기를 결정해야 하며, 그 방법에는 크게 두 가지로 나뉘어 진다. 첫째는 선형 어드레스 연산을 하는 FIFO버퍼인데, 이 방법은 읽기 지시자, 쓰기 지시자가 버퍼의 마지막 위치에 도달했는지를 검토해야 하므로 시간에 전체 시스템의 병목현상을 일으킬 수가 있다. 이러한 문제를 제거 하기 위해서 본 설계에서는 그림 13에서 제시한 방법처럼 두번째 방법인 원형 버퍼의 기능을 이용한다. 이 버퍼는 포인터가 버퍼의 끝에 도달 했을 때, 버퍼 어드레스 계산을 동적으로 하거나, 경우에 따라서는 버퍼의 시작 위치에 포인터를 놓기도 한다. 원형 버퍼의 동작인 wraparound 동작의 유효 어드레스는 Base address=B, most-significant buffer size=L, least-significant buffer size=I라고 하면, 어드레스 B로부터 B+L−1까지의 메모리 공간이 유효하다. 프로그램 수행 시 어드레스 연산은 다음과 같은 방법으로 한다.

```
temp = I + offset
if (temp < 0)
  I = temp + L;
else if (temp >= L)
  I = temp - L;
else I = temp
```

offset은 명령어에 따라 결정되며 wraparound동작 시의 offset의 크기는 반드시 버퍼 크기 보다는 작아야 한다. 본 논문의 내장 SRAM설계에서는 내장SRAM의 어드레스 영역을 C000 0000~DFFF FFFF 로 할당하여 wraparound동작이 가능한 modulo 4Kbytes의 동적인 메모리 할당을 하도록 설계하였다. 따라서, Base address가 C000 0000라면 C000 0FFF의 4K의 어드레스 만큼 데이타를 쓰고 읽을 수가 있으며, 이때 C000 0FFF의 어드레스에서 어드레스가 증가하면 Base address는 C000 1000가 되어 C000 1FFF만큼 사용 가능해진다. 이러한 방법으로 DFFF FFFF까지 어드레싱을 할 수가 있다.

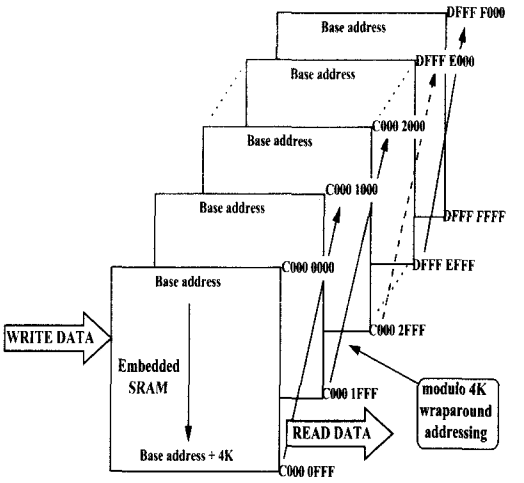

그림 13. modulo-4K wraparound 동작
Fig. 13. modulo-4K wraparound operaton.

V. 모의 실험 및 결과

내장EPROM과 SRAM의 설계는 디지털 블럭과 아날로그 블럭이 같이 설계된 혼합 신호 설계이다. 따라서, 설계된 칩의 전체 기능의 로직 검증 단계에서는 디지털 블럭은 Verilog-XL을 이용하여 검증하였고, 이때 혼합 신호의 경우는 아날로그 블럭을 Verilog-HDL로 모델링을 하여 로직 검증을 하였으며, 아날로그 회로의 검증은 HSPICE로 검증을 하였다. 내장EPROM의 프로그램을 위한 고 전압 회로 설계 및 검증은 고 전압 공정 조건에 따라 혼합 신호로 검증을 완료하였다. 내장 EPROM에서 나온 데이터가 내부 버스에 실리는 읽기 동작에서의 1싸이클 엑세스의 검증을 위해 HSPICE로 시뮬레이션을 한 결과 그림 14와 같이 결과로 동작하였다. 그림14에서 설계된 Tr의 문턱 전압(Vt)의 변화에 따른 Slow(SS3), Typical(TT), Fast(FF3)의 공정 조건으로 검증을 하였으며, 내장EPROM 셀의 프로그램 유무를 레퍼런스 셀과 비교하는 차동 증폭기 센스 앰프에 입력되는 신호 중 opin_ref는 레퍼런스 셀의 전압으로 차동 증폭기의 기준이 되는 신호의 레벨이다. 또 opin_bl은 선택된 셀의 비트라인 신호 레벨이다. 이 두 신호의 전류, 전압의 차이로 프로그램 된 셀(ECELL)과 프로그램 되지 않은 셀(ICELL)이 레퍼런스 레벨과 비교되고 증폭되어 데이터 버스에 실어 주는 신호(odbe)가 액티브 될 때 그 값이 실린다. 그림14에서 EPROM의 데이터 읽기는 20ns안에서 동작함을 알 수가 있다.

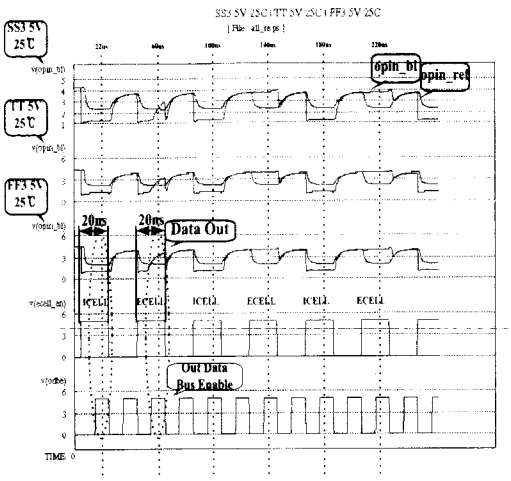

그림 14. 내장 EPROM의 읽기 동작 검증
Fig. 14. Verification of reading from embedded EPROM.

그림 15는 128Kbytes의 EPROM을 내장하였을 경우 외부 입력 클럭 주파수 변화에 따른 데이터 엑세스 싸이클 수를 같은 용량의 외부 EPROM과 비교를 한 것이다. 내장EPROM의 테스트는 read dumping mode에서 하였다. 결과에 따라 내부메모리는 50MHz까지는 1 싸이클 엑세스가 가능하였으나, 외부 메모리인 경우 일반적인 범용메모리는 20MHz에서 1 싸이클 엑세스가 가능하였다.

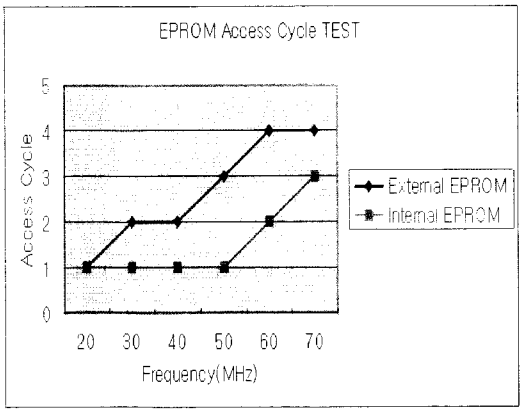

그림 15. 내장 EPROM과 외장 EPROM의 엑세스 사이클 비교
Fig. 15. Access cycle test between internal and extermal EPROM with CPU.

내장SRAM의 설계의 검증은 기존의 프로세서에서 DRAM을 사용하여 리플레쉬가 가능할 때 전체 칩의 전류(Idd)가 5V@50MHz에서 최대 190mA였으나, SRAM 사용하고 제안된 인터페이스로 회로 설계 결과는 5V@50MHz에서 최대 50mA정도로 기존의 칩보다 70%이상 감소하였다. 그림 16는 내장SRAM에 데이터를 프로그램하고 그 값을 포트에 출력하는 테스트 프로그램을 실행하면서 동작 주파수와 전압의 변화를 테스트 한 결과로써, 5V의 33MHz까지 데이터가 포트에 출력되는 결과를 얻었다.

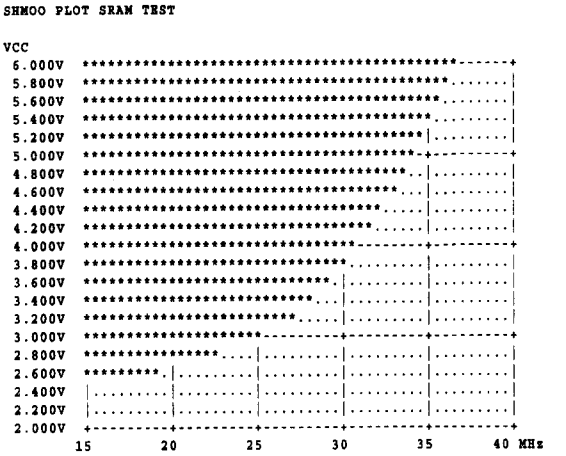

그림 16. 내장 SRAM의 동작 전압과 주파수의 테스트 결과
Fig. 16. Shmoo plot of embedded SRAM.

VI. 결 론

본 논문에서 설계된 모든 내장 메모리 블럭은 full custom 설계 방식으로 게이트 레벨의 로직 설계 및 레이아웃을 하였다. 그림 17은 설계된 MCU로써 EPROM block은 0.6㎛ 3-metal 2-poly CMOS공정으로 설계하였으며, SRAM block과 기타 로직 블럭은 0.6㎛ 3-metal 1-ploy CMOS공정으로 설계하였다. 전체 칩 크기는 10236㎛ 10618㎛이며 총 게이트수는 내장 메모리 셀을 제외하고 250,000개이고 50MHz에서 동작하였다.

본 논문에서는 RISC-based DSP 아키텍처에 적합한 내장 메모리 인터페이스의 효율적인 제어를 위한 새로운 구조를 제안한 결과 128Kbytes의 내장EPROM에서 평균 메모리 엑세스 속도를 종전의 40ns이상에서 20ns 이하로 2배 이상의 속도가 증가됨을 얻었다. 테스트 모드에서는 32/64bit크기의 프로그램이 가능해 종전의 최

대16비트 프로그램보다는 2배 이상의 빠른 프로그램 속도를 높일 수 있었으며, 리플레쉬가 필요 없는 SRAM을 내장하여 기존의 DRAM내장보다는 전력소모를 70%이상 줄였다.

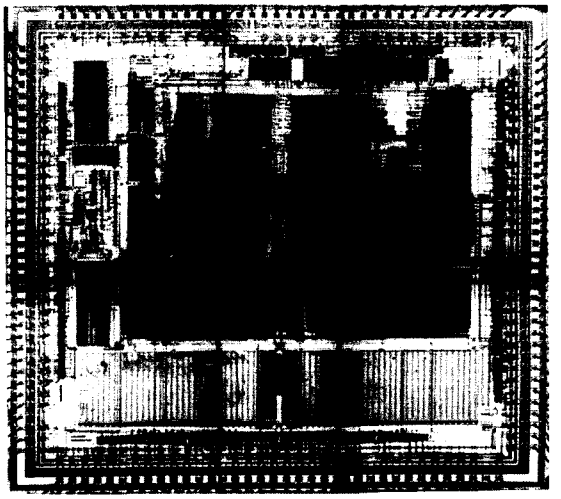

그림 17. 설계된 내장 128KB EPROM과 4KB SRAM 사진

Fig. 17. Photograph of designed chip with embedded 128Kbytes EPROM and 4Kbytes SRAM.

참 고 문 헌

[1] G. Kane and J. Heinrich, MIPS RISC Architecture. Prentice Hall, Englewood Cliffs, NJ. 1991.

[2] K. Narasimhan and K.D. Nilsen, Portable Execution Time Analysis for RISC Processors. Unpublished Technical Report. 1994.

[3] Ron Cates, "Processor Architecture Considerations for Embedded Controller Applications," in IEEE MICRO, 1988, pp. 28-38.

[4] E.P. HARRIS et al, "Technology Directions for Portable Computers," Proceeding. IEEE, vol. 83, pp. 636-658, April, 1995.

[5] M.K. Lee, B. Y. Choi, and S. I. Son, "VLSI Design of High Performance RISC," in Proceedings of ICEIC 91, Yanbian China, Aug 1991.

[6] Michael Dolle, Solid State Circuits VOL. 32, NO7, 1997, IEEEA 32b RISC/DSP Microprocessor with Reduced Complexity

[7] 32Bit-Microprocessor GMS30C2132, GMS30C2116, GMS37C26128 USERS Manual *by MCU Design LG Semicon Co. Ltd.*

[8] DSP Processor Fundamentals Architecture and Features *by BERKELEY DESIGN TECHNOLOGY p. 81~92.*

[9] Miquel Huquet and Tomas Lang, "A Reduced Register File for RISC Architecture," in Computer Architecture News, pp. 22-31, Sept 1985.

[10] David G. Bradlee, Susan J. Eggers and Robert R. Henny "The Effect on RISC Performance of Register Set Size and Structure Versus Code Generation Strategy," in ACM, 1991.

[11] Silicon Processing for the VLSI Era Volume2 *by LATTICE PRESS* p. 623~638.

[12] F. Kazerounian, et al, "A 5 volt high density poly-poly erase flash EEPROM cell," in IEDM Tech. Dig.. DEC 1988. pp. 436-439.

[13] R. Gastaldi, et al, "A 1-Mbit CMOS EPROM with enhanced verification." IEEE J. Solid-State Circuits. vol. 23, no. 5. pp. 150-115 Oct. 1988.

[14] Y. Miyawaki, et. al. "A New Erasing and Row Decoding Scheme for Low supply voltage Operation 16Mb/64Mb Flash EEPROMs," Symp. on VLSI Circuit, Digest of Technical Papers, pp. 85-86, May, 1991.

[15] Landers, G. "5V-Only EEPROM mimics Static-RAM Timing Electronics," Vol. 55, pp. 127, 1882.

[16] Yoshikawa. K, et. al. "0.6㎛ EPROM Cell Design based on a New Scaling Scenario," IEDM. Tech. Dig, pp. 587, 1989.

[17] Atsumi. S,et. al. "A 12ns 4Mb CMOS EPROM, ISSCC Tech. Dig, pp. 74, 1987.

[18] J. Dickson, "On-Chip High-Voltage Generation in NMOS Integrated Circuits Using an Improved Voltage Multiplier Technique," IEEE J. Solid-State Circuits, vol. SC-11, pp. 374-378, June, 1976.

[19] Jeffrey D. Gee, "Cache Performance of the SPEC 92 Benchmark Suite," IEEE MICRO, Vol.13, No5. Aug, pp. 17-27, 1993.

[20] C. H. Perleberg and A. J. Smith. "Branch

Target Buffer Design and Optimization," IEEE Trans. on Computers. Vol. 42, No. 4.pp. 396-412. Apr. 1993.

저 자 소 개

金 裕 鎭(正會員)

1971년 12월 30일생. 1996년 2월 원광대학교 전자공학과 공학학사 1999년 2월 충북대학교 정보통신공학과 공학석사 1995년 12월~1999년 5월 LG종합기술원, LG반도체 MCU 설계실 연구원(GSM단말기개발, 8,32bit 마이콤회로설계) 1999년 6월~현재 한국전자통신연구원(ETRI) 무선 ATMLAN팀 연구원 주관심분야는 Embedded Memory, ASSP, ASIC설계, Algorithm (Firmware) Programming

金 聖 植(正會員)

1965년 12월 28일생. 1988년 2월 경북대학교 전자공학과 학사 1999년 8월 충북대학교 정보통신공학과 공학석사 주관심분야는 EPROM, Flash내장 Embedded MCU회로설계

趙 慶 錄(正會員) 第 36卷 C編 第 1號 參照

1955년 6월 22일생. 1977년 2월 경북대학교 전자공학과 공학사 1989년 3월 일본 동경대학교 전자공학과 공학석사 1992년 3월 일본 동경대학교 전자공학과 공학박사 1979년~1986년 (주)금성사 TV연구소선임연구원. 1992년~현재 충북대학교 정보통신공학과 부교수 주관심분야는 VLSI시스템설계, 통신시스템의 LSI개발, 고속마이크로프로세서설계

鄭 義 錫(正會員)

1969년 10월 18일생. 1991년 2월 한국항공대학 항공전자공학 학사 1994년 2월 한국항공대학 항공전자공학 석사 1994년 3월~현재 한국전자통신연구원 연구원 주관심분야는 ASIC설계, Testability