단일 코어 셀 확장을 이용한 다중포트 메모리 컴파일러

(Multiport Memory Compiler using Single Core Cell Expansion)

김 선 권 † 이 용 진 † 권 성 훈 †† 김 원 종 ††† 신 현 철 ††††
(Sunkwon Kim) (Yongjin Lee) (Sunghoon Kwon) (Wonjong Kim) (Hyunchul Shin)

요 약 본 논문에서는 빠른 시간 내에 설계자의 요구사양을 만족하는 메모리를 자동으로 합성해주는 새로운 멀티포트 메모리 컴파일러를 제안하였다.

제안한 컴파일러의 장점은 하나의 메모리 코어 셀을 규칙적으로 배치, 확장하여 메모리를 합성하고, 동시에 합성된 메모리내의 임계경로만을 추출하여 빠르게 검증할 수 있다는 것이다. 또한, 레이아웃 상에서의 전원선 폭 조절, 전원선 공유 등의 기법으로 메모리의 성능을 향상시킬 수 있도록 하였다.

본 컴파일러를 사용하여 0.25 μm CMOS 1-poly, 2-metal 공정에서 최대 8개의 쓰기 포트, 16개의 읽기 포트, 64워드, 256비트 사이의 여러 가지 멀티포트 메모리를 자동 합성하였다. 합성 결과 메모리의 성능 및 면적 면에서 만족할 만한 결과를 얻었으며, 레이아웃 전체에서의 시뮬레이션 시간보다 10배정도 빠른 시간 내에 동작을 정확히 검증하였다.

Abstract In this paper, we propose a new memory compiler which can automatically generate multi-port memories in a short design time while satisfying the user constraints.

Advantages of the compiler are to synthesize memories by placing and expanding a core cell, according to the number of ports and word length, and to verify the performance by extracting and analyzing critical paths. The performance of the memory is optimized by adjusting wire-width and by sharing power lines.

Using the memory compiler, several multi-port memories including a memory of 8 write ports, 16 read ports, 64 words, and 256 bits have been synthesized for a 0.25 μm CMOS 1-poly, 2-metal process. Experimental results show that the performance and area of the generated memory is satisfactory, and the simulation time of the extracted layout for critical paths is reduced significantly by eliminating the non-critical paths during circuit extraction.

1. 서 론

화상과 음성 데이타 등을 처리하는 시스템에서는 메모리가 전체 회로의 많은 부분을 차지하고 있다. 메모리는 크게 각각의 포트를 가지는 레지스터들 그룹으로 구현하거나 멀티포트 메모리로 구현 가능하다. 멀티포트 메모리는 각각의 포트들을 공유하는 레지스터들의 그룹이다. 멀티포트 메모리는 레지스터들의 포트들을 공유함으로써, interconnection 하드웨어들(bus, MUX, tri-state buffer)의 수를 감소시킬 수 있다[1]. 또한 멀티포트 메모리 기반의 설계는 단일 포트 메모리 기반의 설계보다 더욱 구조적이고 모듈화 되어 있으며 작은 칩 면적에 구현할 수 있다.

이와 같은 멀티포트 메모리의 장점 때문에, 시스템의 성능을 향상시키기 위하여 고성능의 멀티포트 메모리의

† 비 회 원 : 삼성전자 반도체사업부 연구원
phin77@nownuri.net
yjlee@dslab.hanyang.ac.kr
†† 비 회 원 : 한양대학교 전자공학과
wtim@dslab.hanyang.ac.kr
††† 비 회 원 : 한국전자통신연구원 연구원
wjkim@etri.re.kr
†††† 종신회원 : 한양대학교 전자공학과 교수
shin@mail.hanyang.ac.kr
논문접수 : 2000년 4월 24일
심사완료 : 2001년 1월 26일

필요성이 지속적으로 증가하고 있다. 특히, 시스템의 라이프사이클을 만족시키기 위해서, 시스템 설계자는 설계 사양에 맞는 다양한 구조 및 성능의 멀티포트 메모리를 단기간 내에 레이아웃으로 생성할 수 있는 메모리 컴파일러를 요구한다. 따라서, 짧은 설계기간 내에 고성능의 멀티포트 메모리를 자동 생성하는 메모리 컴파일러는 실제 설계에 매우 유용하게 사용될 수 있다. 메모리는 규칙적, 반복적인 구조를 갖고 있기 때문에, 먼저 메모리 구조를 결정한 후에 기본적인 셀 들을 반복, 배치함으로서, 원하는 메모리를 자동적으로 생성 가능하다.

지금까지 멀티포트 메모리 컴파일러에 대해서는 많은 연구가 진행되어 왔다. [2]의 메모리 생성기는 행 인에이블 게이트 신호(column enable gate signal)를 사용하였다. 메모리의 워드 수가 증가할 경우 계층적인 설계를 위하여 행 인에이블 게이트를 첨가하여 메모리의 가로대 세로 비를 조정 가능하게 하였다. [3]에서는 LSI Logic 사의 MEMCOMP 라이브러리를 사용하여 메모리를 자동으로 생성한다. 이 메모리 합성기는 최대 64워드 72비트까지의 3포트(1W2R) 또는 6 포트(3W3R, 2W4R) 메모리를 생성 가능하다. [4]에서는 제약조건을 고려한 디자인 흐름(constraint driven design flow)을 통하여 최적화된 SRAM을 생성하는 AURORA라는 합성기를 개발하였다. [5]에서는 최대 64~4096워드, 4~256비트사이의 단일 포트 ROM과 RAM을 생성 가능하도록 하였다. [6]에서는 최대 1024워드 32비트까지 RAMGEN이라는 dual-port 합성기를 제안하였고, [7]에서는 최대 이중포트 8K까지 합성가능 한 BiCMOS SRAM 합성기를 개발하였다. [8,9]에서는 고속동작의 멀티포트 메모리를 제안하였다. [10]에서는 저 전력 SRAM 설계를 위하여 계층적 비트라인 분할 방법을 사용함으로써 비트 라인의 커패시턴스를 줄여 전력 소모와 지연시간을 감소시켰다. [11]에서는 저 전압 SRAM설계를 위하여 sub-block 구조를 제안하였으며 1.6V 전원전압에서 최대 40MHz까지 동작할 수 있도록 하였다.

기존의 메모리 설계방식의 단점은 두 가지로 요약될 수 있다. 대부분의 기존에 사용되었던 셀 참조 방식은 사용자가 원하는 사양의 기본 셀 (leaf cell)들이 미리 설계되어 있어야 한다는 제약 조건이 있다. 멀티포트 메모리의 경우에는 읽기 포트 수, 쓰기 포트 수의 조합이 많기 때문에, 그 조합에 따른 모든 경우의 코어 셀을 미리 만들어 사용하기가 어렵다.

대부분 설계된 회로의 정확한 시뮬레이션을 위하여, 메모리 레이아웃 전체에 대한 파라미터를 추출하여 성능을 검증하였다. 그러나, 대용량 메모리의 경우에는 시뮬레이션 시간이 오래 걸리는 단점이 존재한다.

본 논문에서 제안하는 멀티포트 메모리 합성기는 다음과 같은 장점을 갖는다. 먼저, 하나의 코어 셀로부터 사용자가 원하는 읽기 및 쓰기 포트수의 조합으로 메모리 코어 셀을 24 포트(16읽기, 8쓰기 포트)까지 확장 가능하다. 합성에 필요한 셀의 수는 소수만을 라이브러리에 저장하고 사양에 따라 변형하여 사용하도록 체계적인 방법을 개발하였다. 둘째, 메모리내의 임계 경로상의 레이아웃에 대해서만 파라미터를 추출하여 시뮬레이션 함으로써 빠르게 성능을 검증할 수 있도록 하였다. 또한, 메모리내의 전원선 폭을 자유롭게 조절 가능하며, 면적을 줄이기 위하여 VDD/GND를 공유하도록 하였다.

본 논문의 구성은 다음과 같다. 2장에서는 멀티포트 메모리 컴파일러 시스템에 대하여 설명한다. 3장에서는 멀티포트 메모리의 전체 스케메틱과 전체배치도에 대하여 설명한다. 4장에서는 멀티포트 메모리 컴파일러에서 사용하는 코어 셀 확장 방법, 배치, 배선 등의 세부적인 알고리듬을 기술한다. 5장에서는 메모리 생성 및 검증 실험 결과를 보이고, 마지막으로 6장에서는 결론을 맺는다.

2. 멀티포트 메모리 컴파일러 시스템

본 멀티포트 메모리 컴파일러는 설계자가 원하는 입력 사양을 만족하는 메모리 레이아웃을 빠른 시간 내에 자동으로 생성한다.

본 논문에서 제안하는 멀티포트 메모리 컴파일러의 전체 블록 다이어그램은 그림 1과 같다.

2.1 멀티포트 메모리 컴파일러의 입력

멀티포트 메모리 합성기에 사용되는 입력은 두 가지, 즉 회로 생성에 필요한 매개변수(input parameters)와 단위 셀 라이브러리(leaf cell library)이다. 매개변수들은 합성하고자 하는 메모리의 읽기 포트 수, 쓰기 포트 수, 워드 수, 비트 수, 코어 셀 내부의 파워라인 폭 등이다. 단위 셀 라이브러리는 단위 셀 레이아웃과 단위 셀 크기, 단위 셀에 필요한 입출력, 전원선 위치 등에 대한 정보를 포함한다. 메모리 코어 셀은 하나만 저장하여 라이브러리를 간단히 하였으며, 입·출력포트 수에 따라 자동적으로 변형하여 합성에 사용한다.

2.2 멀티포트 메모리 컴파일러

멀티포트 메모리 컴파일러 알고리듬은 크게 코어 셀 생성 및 확장, 셀 어셈블, 블록배선, 임계경로상의 레이아웃 추출, 레이아웃 출력으로 이루어진다. 1 비트 메모리 코어 셀 라이브러리로부터, 입력으로 주어진 포트 수에 따라 메모리를 확장하여 합성하고, 메모리의 주변회로들, 즉, 센스앰프, I/O 버퍼, 디코더의 라이브러리를 사

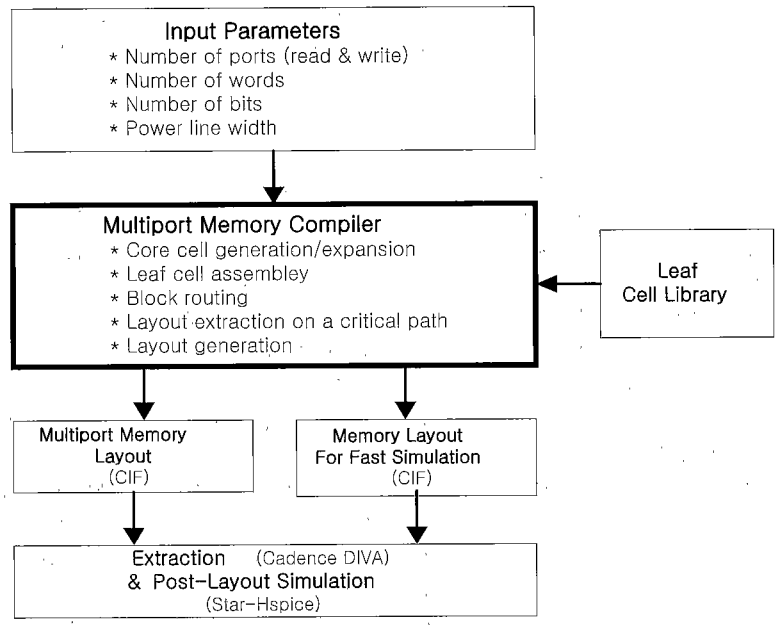


그림 1 멀티포트 메모리 컴파일러 블록 다이어그램

용하여 메모리 전체를 어셈블 한다. 또한 각 셀들간에 필요한 배선을 수행함으로써, 메모리 레이아웃을 완성한다.

2.3 멀티포트 메모리 컴파일러의 출력

본 논문에서 개발한 멀티포트 메모리 컴파일러는 멀티포트 메모리 레이아웃과 빠른 시뮬레이션을 위한 레이아웃을 출력한다. 합성된 결과는 CIF(Caltech Intermediate Format)로 출력하여 상용 레이아웃 툴과 데이타를 공유할 수 있도록 하였다.

3. 멀티포트 메모리 스케메틱과 전체 배치구조

3.1 스케메틱 회로도

본 장에서는 본 논문에서 사용한 멀티포트 메모리의 스케메틱과 메모리 레이아웃 배치구조에 대하여 설명한다.

본 연구에서 사용한 멀티포트 메모리의 스케메틱은 그림 2와 같다[5]. 읽기 동작에서는 읽기 전용 디코더가 인에이블 되면, 1 비트 저장 셀내의 데이타를 읽기 비트라인에 실어 센스앰프를 거쳐 증폭하여 출력한다. 데이타의 입력 부분에는 메모리에 쓰여질 신호를 안정적으로 전달하기 위하여 버퍼 체인을 사용하였다.

3.2 멀티포트 메모리 배치구조

멀티포트 메모리는 서로 다른 어드레스 메모리 셀에 동시에 데이타를 쓰고 읽을 수 있으며, 같은 어드레스

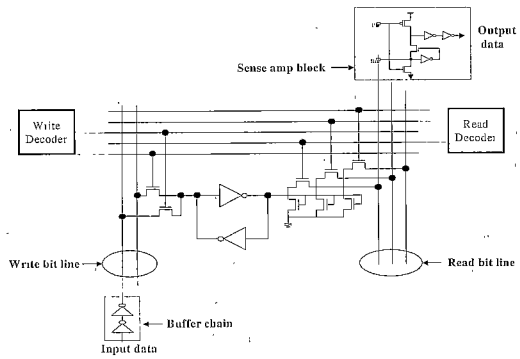


그림 2 멀티포트 메모리 스케메틱

메모리 셀의 데이타를 메모리 셀에 연결된 모든 읽기 전용 포트를 통해 동시에 읽을 수 있다.

그림 3은 멀티포트 메모리의 전체 배치구조를 보여준다. 메모리 코어 셀 블록을 기준으로 위에는 센스앰프와 데이타 출력 단자가 놓이며, 아래에는 입력단자가 배치된다. 포트 수에 따른 확장이 용이하도록 그림 3에서와 같이 읽기 전용 포트의 디코더는 메모리 코어 셀 블록의 오른쪽에 위치하고 쓰기 전용 포트의 디코더는 메모리 코어 셀 블록의 왼쪽에 위치한다. 서로 독립적으로

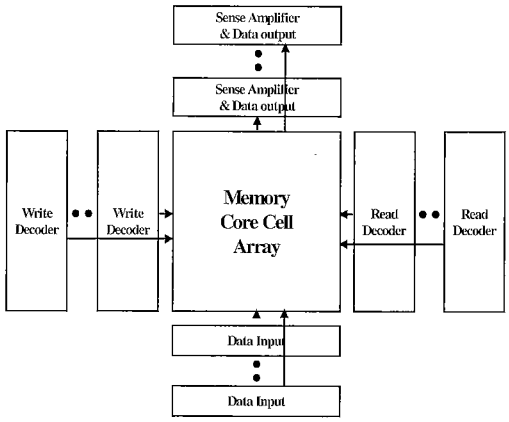


그림 3 멀티포트 메모리의 전체 배치구조

사용하기 위해서 각각의 포트들은 서로 비동기적으로 동작한다. 또한, 각 포트가 독립적으로 동작하기 위해서 각 포트의 디코더와 어드레스 라인 및 데이타의 입·출력 블록이 별도로 존재한다.

4. 멀티포트 메모리 컴파일러의 알고리듬

본 장에서는 컴파일러에 사용한 타일 기반 방법(tile-based method)과 메모리 생성 알고리듬의 주요 부분에 대하여 자세히 설명한다. 본 연구에서 개발한 멀티포트 메모리 컴파일러의 합성 알고리듬은 크게 메모리 코어 셀 생성, 단위 셀 어셈블, 블록 배선, 임계경로 추출, 레이아웃 출력 부분으로 나누어진다.

4.1 타일 기반 방법

회로설계에 사용되는 단위 셀(leaf cell)들을 미리 준비하고 이 셀들을 배열함으로써 원하는 레이아웃을 얻는 기법을 타일 기반 방법이라 한다. 타일은 단위 셀(leaf cell) 또는 단위 셀로 구성된 셀들을 의미한다. 타일은 레이아웃의 절대좌표 및 상대좌표에 따라 배열되며, 타일들의 접속방법에 따라 속성이 정의된다.

타일의 속성으로는 배치(placement), 접속(abutting), 공유(overlapping), 반사(mirroring), 회전(rotating) 등이 있다.

4.2 코어 셀 생성(Core cell generation/expansion)

본 연구에서 개발한 컴파일러에 필요한 메모리 코어 셀은, 미리 레이아웃으로 구현된 라이브러리를 참조하여 사용하는 기존의 방식에서 벗어나, 1비트 저장 셀로부터 사용자가 원하는 사양의 읽기/쓰기 포트 수만큼 자동적으로 확장이 가능하도록 하였다. 그림 4는 24포트(16 읽기 포트, 8 쓰기 포트)까지 확장된 하나의 메모리 코어 셀의 내부 레이아웃 구조 및 포트 배치 순서를 보여준다. 읽기 포트가 1개 이상 일 때에는 읽기 포트 확장 영역 내에서 읽기 워드 인에이블 트랜지스터(read word enable tr.)를 규칙적으로 배치하여 확장한다. 같은 방법으로 쓰기 포트도 외부에서 주어진 출력 포트 수에 맞게 자동으로 배치하여 확장한다. 그림 4의 읽기/쓰기 포트 확장 영역 내부에 표기된 숫자 1, 2, 3, 4는 포트가 확장될 때 각 워드 인에이블 트랜지스터의 배치와 확장 순서를 나타낸 것이다. 포트를 확장할 때에는 메모리 성능을 고려하여, 수평방향의 읽기/쓰기 워드라인을 코어 셀의 상·하로 나누어 배치하였다. 1 비트 코어 셀의 최상단/최하단에 수평방향으로 존재하는 파워 라인 폭은 사용자가 외부에서 조절할 수 있게 하여, 메모리 성능을 최적화 할 수 있도록 하였다. 코어 셀 내부의 수평방향 워드라인은 metal-1, 수직방향의 비트라인들은 metal-2를 사용하였다.

4.3 단위 셀 어셈블 (Leaf cell assemble)

단위 셀 어셈블 과정에서는 확장된 메모리 코어 셀과 라이브러리들을 본 논문에서 결정한 메모리 구조에 맞게 배치하여 합성한다. 특히 메모리 블록들을 배치할 때에는 접속(connection by abutment) 기법을 사용하여 블록들 사이의 연결 신호선을 최소화하였다. 접속 방법을 이용한 연결은 각각의 라이브러리 셀들이 이웃하여 놓일 때 라이브러리 셀들 사이의 단자가 중첩되어 연결되도록 하는 방법이다. 접속 방법을 이용한 연결은 어셈블 과정시 각각의 블록들 내에서 복잡한 연결선들이 필요하지 않기 때문에 전체 회로 면적이 줄어들고 회로의 성능도 향상되는 장점이 있다.

메모리 코어 셀을 배치하여 메모리 블록을 합성할 때, 인접한 위·아래 셀들을 상·하 대칭구조로 배치하여 코어 셀의 VDD와 GND를 공유하였다. 이와 같은 플립(flip)방식으로 인접한 블록들 (메모리 코어 셀, 센스 앰프블록, 데이타 입·출력 블록) 간에 파워라인을 공유하여 면적을 줄였다.

4.4 블록 배선 (Block routing)

블록 배선은 단위 셀 어셈블 과정을 마친 후, 신호선의 연결이 완료되지 않은 블록들 사이의 신호선을 연결하는 과정이다. 단위 셀 어셈블 과정에서 메모리 내의 블록들 간의 신호선 연결을 최소화하였기 때문에 메모리 코어블록과 디코더블록의 입출력 단자들의 일부분을 배선하면 된다. 메모리블록과 센스앰프, 메모리블록과 입력 버퍼 사이의 연결은 피치 매칭(pitch matching)방식으로 구현하였다[12].

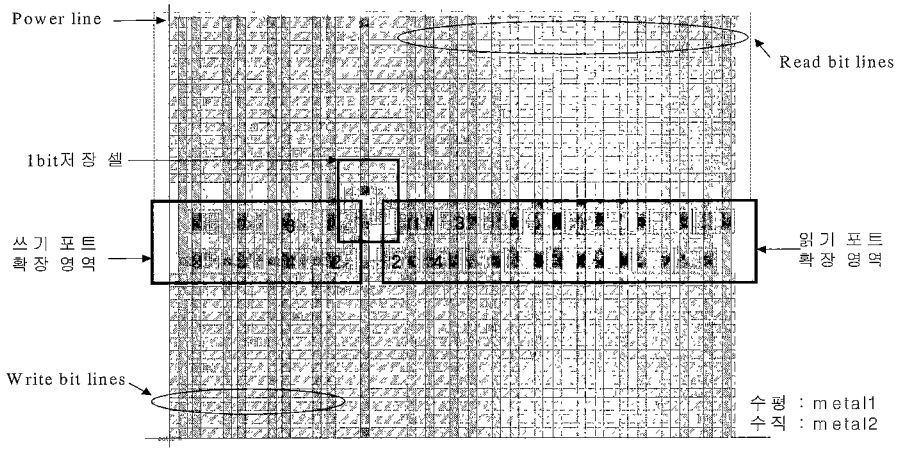


그림 4 24포트(16 읽기 포트, 8 쓰기 포트) 메모리 코어 셀

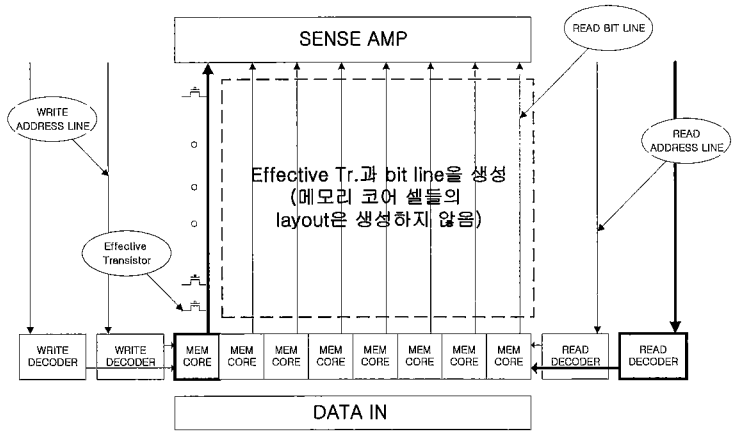


그림 5 메모리의 어드레스 억세스 시간 임계경로

4.5 임계경로 상의 레이아웃 추출(Layout extraction for critical paths)

임계경로상의 레이아웃 추출 과정은, 합성된 메모리 레이아웃의 검증시간 단축을 위하여, 회로의 임계경로상에 있는 레이아웃을 추출하는 과정이다. 메모리의 읽기 동작은 다음과 같다. 읽기 주소(read address)가 인가되면 지정된 주소의 읽기 디코더가 워드라인을 인에이블시켜서 저장된 메모리의 값이 센스앰프를 통하여 출력된다. 그러므로, 어드레스 억세스 시간(address access time)의 임계경로는 RC 지연시간이 가장 큰 읽기 주소 라인에서부터 메모리 코어를 거쳐, 데이타를 출력하는 읽기 비트라인으로 구성된 경로이다. 메모리의 어드레스 억세스 시간 임계경로는 그림 5에서 보는바와 같이 굵은 선으로 표시된다. 그림 5의 점선 영역에서는 비트 라인과 지연시간에 영향을 주는 유효 트랜지스터만을 레이아웃으로 생성함으로써 복잡도를 크게 줄여서 효율적인 검증이 가능하도록 하였다.

그림 6은 각 비트라인에 추가한 유효 트랜지스터를 나타낸 것이다. 유효 트랜지스터는 워드라인이 인에이블되면 메모리 코어 셀 내부의 값을 비트라인에 실어 주는 역할을 하는 트랜지스터를 의미한다. 시뮬레이션 정확도를 향상시키기 위하여 메모리 블록내의 각 비트라

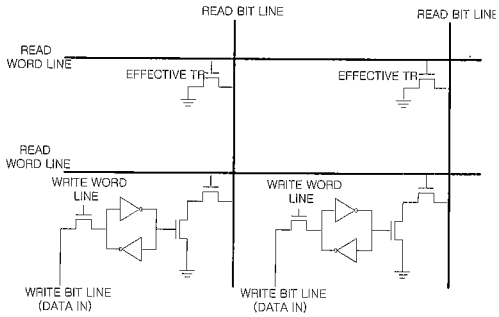


그림 6 각 비트 라인에 추가한 effective transistor

인의 지연시간에 영향을 주는 유효 트랜지스터를 각 비트라인에 연결하여 사용하였다.

5. 실험 결과

본 연구에서 제안한 컴파일러를 이용하여 다양한 크기와 포트 수에 대하여 멀티포트 메모리를 자동 합성하였다. 각 단위 셀 라이브러리는 1 poly, 2 metal의 0.25 μm CMOS공정으로 설계하였다.

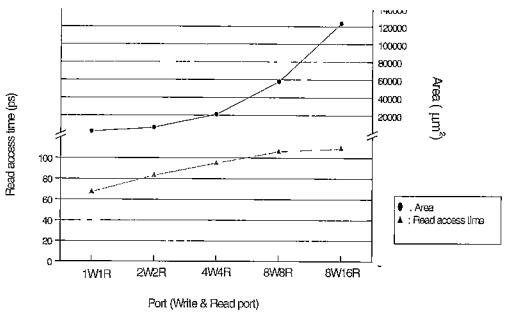


그림 7 포트 수에 따른 지연시간과 면적에 대한 그래프

그림 7의 그래프에는 여러 가지 포트 수에 따른 멀티포트 메모리의 read access time과 레이아웃 면적 사이의 관계를 보였다. 2 포트 메모리에 비하여 24포트 메모리의 면적은 약 7배 증가하였고 read access time은 약 63% 증가하였다. 본 연구에서는 성능과 면적을 고려하여 메모리 코어의 확장을 24포트(16 읽기, 8쓰기)까지 가능하도록 하였다.

본 연구에서 제안한 멀티포트 메모리 컴파일러를 사용하여 합성한 레이아웃으로부터 파라미터를 추출하여 UltraSPARC-II workstation (CPU: 296MHz, RAM: 512MB, swap: 2GB) 에서 시뮬레이션 하였다. 시뮬레이션은 정상상태 (Vdd = 2.5V, Temp = 25 °C) 에서 실행하였다.

표 1 자동 합성된 멀티포트 메모리의 시뮬레이션 결과

멀티포트메모리(64word x 8bits)	검증 요소	full layout	Layout on critical path with effective tr.
2W2R (718*172 $\mu m2$)	transistor 수	9680 (100 %)	1364 (14.98 %)
	simulation time (CPU time)	1636.82 s (100 %)	207.08 s (12.65 %)
	read access time	12.333 ns (100 %)	12.246 ns (99.29 %)
4W4R (980*325 $\mu m2$)	transistor 수	17132 (100 %)	2696 (15.74 %)
	simulation time (CPU time)	2919.77 s (100 %)	382.59 s (13.10 %)
	read access time	14.212 ns (100 %)	14.046 ns (98.83 %)
8W8R (1460*625 $\mu m2$)	transistor 수	32576 (100 %)	5360 (16.45 %)
	simulation time (CPU time)	14342.13 s (100 %)	594.96 s (4.15 %)
	read access time	17.718 ns (100 %)	17.688 ns (99.83 %)
8W16R (1960*910 $\mu m2$)	transistor 수	49536 (100 %)	10224 (20.64 %)
	simulation time (CPU time)	*	1148.73 s
	read access time	*	20.930 ns

* : 주어진 하드웨어 자원으로 시뮬레이션 할 수 없음

표 1은 다양한 포트에 대한 메모리 면적과 전체 레이아웃에서와 임계경로상의 레이아웃에 대하여 시뮬레이션한 결과를 보여준다. 멀티포트 메모리 면적은 0.25 μm 공정으로 설계하여 발표된 다른 결과가 없기 때문에 크기 비교가 어렵지만, 비교적 만족할 만한 결과를 얻었다. 임계경로상의 레이아웃을 추출하여 시뮬레이션 한 결과, 메모리 전체의 레이아웃에 대한 시뮬레이션에 비하여 98.8% 이상의 정확도를 유지하면서 평균적으로 10배정도 빠른 검증이 가능하였다. 여기서 어드레스 억세스 시간이란 메모리의 읽기 동작시의 동작속도를 측정하는 파라미터로서, 어드레스가 지정되어 있고 read enable신호가 인가될 때 출력단자에서 유효한 데이타가 인출되기까지의 시간을 의미한다. 표 1에서의 8W16R의 전체 회로에 대한 시뮬레이션은 메모리 부족으로 시뮬레이션이 가능하지 않았다.

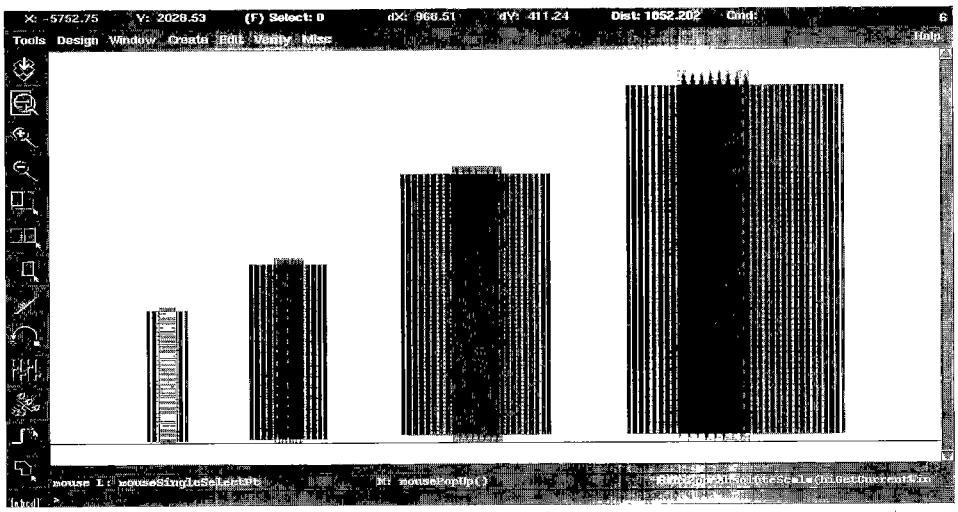
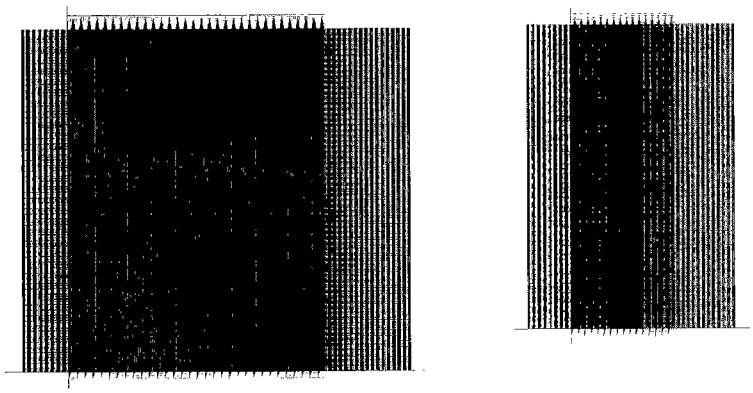


그림 8 다중포트 메모리 컴파일러에서 생성된 다양한 메모리
(2W2R, 4W4R, 8W8R, 8W16R 64words*8bits)



(a) 24-Port(8W16W) : 64words*32bits (b) 20-Port (8W12R) : 64words*16bits
그림 9 자동 합성된 메모리 레이아웃

그림 8은 8bit의 메모리를 다양한 크기로 자동 합성한 레이아웃이다. 그림 9는 자동합성된 24포트(8W16R) 32bit와 20포트(8W12R) 16bit 메모리의 레이아웃이다.

6. 결 론

본 논문에서는 빠른 시간 내에 설계사양을 만족하는 메모리를 자동으로 합성해주는 새로운 멀티포트 메모리 컴파일러를 개발하였다. 제안한 컴파일러는 단일 메모리 코어 셀을 확장하고 규칙적으로 배치하여 최대 24포트까지의 메모리를 합성할 수 있도록 하였으며, 동시에 합성된 메모리내의 임계경로만을 추출하여 빠르게 검증할 수 있도록 하였다. 또한, 레이아웃 상에서의 전원선 폭 조절, 전원선 공유 등의 기법으로 메모리의 성능을 향상시킬 수 있도록 하였다. 본 컴파일러는 최대 8개의 쓰기 포트, 16개의 읽기 포트, 64워드, 256비트 사이의 여러 가지 멀티포트 메모리를 자동 합성할 수 있다. 합성 결과 메모리의 성능 및 면적 면에서 만족할 만한 결과를 얻었으며, 레이아웃 전체에서의 시뮬레이션 시간보다 10배정도 빠른 시간 내에 동작을 정확하게 검증할 수 있음을 확인하였다. 향후 연구과제로는 저전력 소모와 고

속 동작을 위한 셀 및 메모리 구조에 대한 연구가 있다.

참 고 문 헌

[1] Dongha Park and Hyunchul Shin, "Partitioning for Minimal Memory in Hardware-Software Codesign," IEEE Proc. of the ISCAS, vol. 4, pp.648-651, May. 1996.

[2] Hirofumi Shinohara, Noriaki Matsumoto, Kumiko Fujimori and Shuichi Kato, "A Flexible Multi-Port Compiler for Datapath," IEEE Custom Integrated Circuits Conference, 16.5.1-16.5.4 1990.

[3] Kuang-Pin Tsao, Nick Zhu and Tung Pham, "A High performance Memory compiler for Multi- Port RAMs," Proceedings of IEEE ASIC Conference and exhibit, p3-6.1~p3-6.4 1990.

[4] Ajay Chandna, C.David Kibler, Richard B.Brown, Mark Roberts, Karem A.Sakallah, "The Aurora RAM Compiler," 32 Design Automation Conference, pp.261-266, 1995.

[5] 김정범, 권오형, 홍성제 "ASIC 용 메모리 컴파일러 설계," 대한 전자공학회, vol.35 No.8, August 1998.

[6] Cristina Silvano, Giancarlo Sada, Laura Populin, "Ramgen: A Dual Port Static RAM Embedded SRAM Compiler," IEEE Journal of Solid-State Circuits, vol. 27, No. 3, March 1992

[7] Tim Dao and Frank J.Svejda, "A Dual-Port SRAM Compiler for 0.8μm 100K BiCMOS Gate Arrays," IEEE Customer Integrated Circuits Conference, pp. 22.4.1-22.4.3, 1991.

[8] Creigton Asato, Robert Montoye, John Gmuender, E. Wade Simmons, Atsushilke, John Zasio, "A 14-port 3.8ns 116-Word 64b Read-Renaming Register File," IEEE International Solid-State Circuits Conference, pp. 104-105, 1995

[9] Wei Hwang, Rajiv V. Joshi and Walter H. Henkels "A 500-MHz, 32-Word x 64-Bit, Eight-Port Self-Resetting CMOS Register File," IEEE J. Solid State Circuits, vol. 34, No. 1, January 1999.

[10] Ashish Karandikar and Keshab K.Parhi "Low Power SRAM Design using Hierarchical Divided Bit-Line Approach," International Conference on Computer Design, October 1998.

[11] James S. Caravella "A Low Voltage SRAM For Embedded Applications," IEEE J. Solid State Circuits, vol. 32, NO. 3, March 1997.

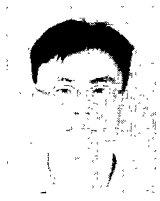
[12] Bryan T. Preas, et al. Physical Design Automation of VLSI Systems. The Benjamin/Cummings Publishing Company, Inc., California, 1988.

김 선 권

1999년 한양대학교 전자공학과 학사. 2001년 한양대학교 전자통신전파공학과 석사. 현재 삼성전자 반도체사업부. 관심분야는 VLSI CAD(Codesign, floor-planning, memory compiler)

이 용 진

1999년 한양대학교 전자공학과 학사. 2001년 한양대학교 전자통신전파공학과 석사. 현재 삼성전자 반도체사업부. 관심분야는 VLSI CAD, SDR



권 성 훈

1993년 한양대학교 전자공학과 석사. 1999년 한양대학교 전자공학과 박사. 관심분야는 VLSI CAD, 디지털 신호 처리

김 원 종

1989년 전남대학교 전자공학과 학사. 1992년 한양대학교 전자공학과 석사. 1999년 한양대학교 전자공학과 박사. 현재 ETRI 연구원. 관심분야는 VLSI CAD, Digital System 설계

신 현 철

1974년 ~ 1978년 서울대학교 전자공학 학사. 1978년 ~ 1980년 한국과학기술원 전기 및 전자공학 석사. 983년 1987년 U.C Berkeley 전기 및 전자공학 박사. 1980년 ~ 1983년 금오공과대학 조교수. 1985년 ~ 1987년 U.C Berkeley ERL 연구원. 1987년 ~ 1989년 AT&T Bell Lab Murray Hill, N.J. 연구원. 1989년 ~ 현재 한양대학교 교수. 1997년 ~ 현재 IDEC 한양대학교 지역센터 센터장