정확도가 향상된 ASIC 메모리 라이브러리의 소비전력추출에 관한 연구
A Study on an Advanced Methodology of ASIC Memory Library Power Characterization with High Accuracy

정 승민 (정보통신과)
Seung-Min Jung (Dept. of Information & Communication)

Key Words : Power consumption(소비전력), Memory library(메모리 라이브러리), ASIC memory, Memory compiler(메모리발생기), Embeded memory(내장형메모리), Critical path

ABSTRACTS : In this paper, An advanced methodology of ASIC memory power characterization is suggested. Currently, It has many interpolation and extrapolation errors. So, less accurate in memory library power characterization. An advanced method of ASIC memory power characterization solves these problems. it is impossible to characterize power consumption for all configuration of memory compiler which can be generated. So, we need to characterize many configurations as possible for more accurate power consumption of memory compiler library. This paper suggests the new method for more accurate and speedy results of memory compiler library power characterization. That includes advanced critical path circuit for power and methodology of memory compiler library power characterization.

1. 서론

주문형반도체칩을 의미하는 ASIC(Application Specific Integrated Circuits)은 이제 일반화된 용어가 되어있다. 사람의 몸이 다양한 종류의 세포(Cell)로 구성되어있듯이 ASIC 칩도 셀(Cell)이라고 하는 로직 회로 단위로 구성되어있다. 다양한 동작기능을 갖는 여러 개의 셀이 모여 하나의 라이브러리(Library)를 구성하며 고성능 셀이 많은 라이브러리일수록 고성능, 고집적 ASIC 칩을 만들 수 있는 것이다. 라이브러리에는 단순동작기능위주의 셀이 모인 primitive 셀 라이브러리가 있으며 복잡한 기능의 ASIC chip을 제작하기 위해서는 primitive 셀 이외에, 데이터를 저장하고 읽을 수 있는 메모리(Memory) 라이브러리를 갖추고 있어야 한다. 다양한 사이즈의 메모리를 칩 설계시 적용할 수 있다는 것은 ASIC 칩 설계자로서 가져야 할 당연한 설계환경이며 이에 ASIC 메모리 콤파일러를 개발하는 라이브러리 개발자는 이를 충족시킬 수 있는 고성능 메모리 콤파일러를 구축해야한다.[1][2]

메모리 콤파일러의 특성추출(Characterization)은 일반 단순로직 라이브러리와는 달리 정확성을 위한 특성추출 과정이 매우 복잡하고 시간 또한 많이 소요된다. 특히 소비전력(Power consumption)을 추출하는데는 메모리 회로의 복잡성, 다양성으로 인해 HSPICE를 이용하는데 한계가 있으므로 HSPICE에 비하여 시뮬레이션 정확도가 다소 떨어지지만 속도 면에서 약 1000 배 빠른 PowerMill, TimeMill과 같은 Event-Driven 방식의 상용툴을 적용하여 왔다. 최근에 HSPICE와 근접한 정확도와 switched 방식의 시뮬레이션 툴의 속도를 갖는 툴이 등장하기는 했으나 생성(Generation)할 수 있는 모든 메모리

configuration에 대한 파워특성에 대해 database를 구축한다는 것은 여전히 불가능하다.[3][4] 따라서 전체 라이브러리 파워소모 특성추출의 정확성을 위해서는 가능한 많은 configuration에 대한 파워소모 특성추출을 실시하는 것이 중요하다. 메모리 파워특성의 정확성은 실제 시뮬레이션 된 파워결과들로부터 발생하는 interpolation 에러와 extrapolation 에러를 최소화 하는데 있기 때문이다.[5][6][7]

본 논문에서는 메모리 콤파일러의 개발단계중 소비전력 추출과정에 있어서 좀 더 진보된 방법을 제안하고자 한다. 전통적인 메모리 콤파일러 소비전력 추출과정을 소개하고 문제점을 제시하며, 개선된 소비전력 추출 방법을 제안하며, 2-Port 메모리 compiler의 특정 사이즈에 대하여 제시된 방법에 의해 소비전력을 추출해보고 HSPICE에 의한 소비전력결과와 비교하여 정확성을 검증해본다. 아울러 제시된 소비전력 특성추출법에 의해 전체 라이브러리의 소비전력의 정확성 향상 정도를 산술적으로 계산하여 제시하였다.

2. ASIC Compiled 메모리의 소비전력 특성추출

2. 1 Compiled 메모리의 정의

라이브러리와 메모리의 종류에 따라 다소 차이는 있으나 작게는 16 Bits에서 최대 512K Bits 용량 크기에 이르는 다양한 메모리의 layout 과 schematic , netlist 를 자동으로 생성해주며 compiled 메모리의 데이터 저장용량(Capacity)은 보통 다음의 식으로 정의된다.

$$Memory\ Capacity = Rows\ X\ Cols \quad (1)$$
$$Rows = Number\ of\ Words\ /\ Ymux \quad (2)$$
$$Cols = Number\ of\ Bits\ X\ Ymux \quad (3)$$

여기서 Ymux는 메모리의 aspect ratio를 결정해준다.

2. 2 전형적인 메모리의 소비전력 특성추출법

라이브러리와 메모리의 종류에 따라 다소 차이는 있으나 자동생성될 수 있는 메모리의 경우의 수는 대략적인 산술 계산만으로도 수만에서 수십만에 이르고 메모리 종류가 다양한 경우 전체 메모리의 소비전력 특성추출을 구한다는 것은 현실적으로 불가능한 일이다. 따라서 이산(descrete) 방법을 사용하는 것이 보통이다. 우선 영역 내 생성될 수 있는 메모리 중에서 라이브러리를 검증하기에 적합한 대표적인 사이즈를 선정한다. 선정된 메모리에 대하여 실제 메모리를 생성한 후 회로 시뮬레이션 툴을 이용하여 소비전력을 추출한다. 시뮬레이션 되지 않은 나머지의 메모리 사이즈에 대해서는 시뮬레이션 된 메모리의 소비전력 결과로부터 interpolation 혹은 extrapolation을 통하여 간접적으로 얻어진다.

2. 3 기존 소비전력 특성추출법의 문제점

시뮬레이션 되지 않은 나머지의 메모리 사이즈에 대해서는 시뮬레이션 된 메모리의 소비전력 결과로부터 interpolation 혹은 extrapolation을 통하여 간접적으로 얻어지므로 〔그림 2〕에서 보이는 것처럼 실제 시뮬레이션 되는 메모리 사이즈의 경우의 수를 최대화하는 것이 전

체 메모리의 소비전력 특성추출의 정확도를 높일 수 있는 것이다. 하지만 메모리의 사이즈가 커져 최대에 가까워질수록 시뮬레이션에 의한 추출은 불가능해진다. 결국 실제 시뮬레이션 되는 메모리 사이즈의 경우의 수는 수십에서 수백 개 정도에 국한 될 수밖에 없고 나머지에 대해서는 interpolation 혹은 extrapolation을 통하여 간접적으로 얻어질 수밖에 없다. 따라서 기존의 방법에 의해서는 메모리의 소비전력 특성추출의 정확성에 한계가 있는 것이다.

3. 개선된 ASIC Compiled 메모리의 소비전력 특성 추출법

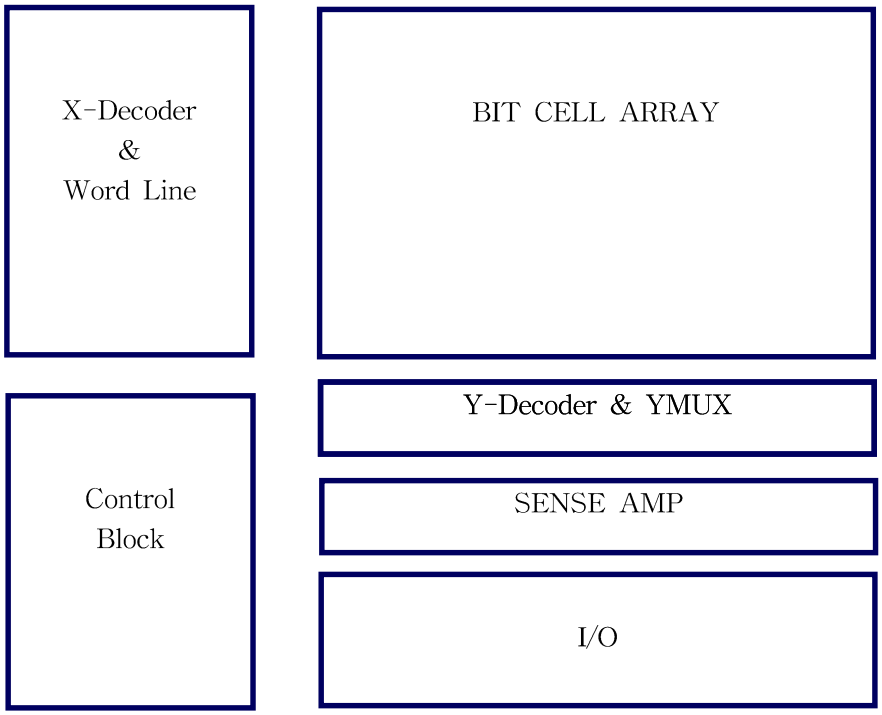
본 논문에서 제안하고자 하는 새로운 compiled 메모리의 소비전력 특성추출법의 근거는 메모리의 회로가 leaf-cell이라고 하는 반복적인 회로들의 규칙적인 배열에 의해 구성된 점을 이용한 것이다. 〔그림 1〕과 같이 메모리회로의 대부분은 기본 Bit를 저장하는 단위 셀인 Core-Cell 이 반복적으로 tiling 된 Bit Cell Array로 구성되 있고, 주변을 X-Address Decoder, Y-Address Decoder 와 YMUX, Word-Line Driver, Sense AMP, I/O Block, Control Block 등으로 구성되어 있다. 이중 Control Block을 제외하고 Bit Cell Array를 포함 나머지 회로들은 해당 블록의 leaf-cell 들이 메모리의 사이즈에 따라 자동으로 그 수를 늘이거나 줄이는 방법에 의해 Compiler 가 S/W 적으로 자동으로 배열하도록 하는 것이다. 따라서 메모리 소비전력을 계산하는 데 있어서 전체회로에 대하여 시뮬레이션 하는 대신 각각의 메모리 회로 블록에 있어서 leaf-cell에 대한 소비전력만을 미리 구한 뒤 메모리의 사이즈와 동작 모드에 따라 실제 동작하는 leaf-cell의 수 만큼을 계산하여 전체 메모리의 소비전력을 계산해 주기만 하면 되는 것이다. Control Block 의 경우 반복되는 leaf-cell이 별도로 없으므로 Control Block 내에 있는 회로를 구분하여 메모리의 사이즈와 동작 모드에 따라 별도로 소비전력을 시뮬레이션 해야 한다. 전형적인 동기(Synchronous)신호인 CLOCK 신호를 갖는 메모리에 대하여 파워추출과정은 다음과 같다.

3. 1 Power 추출을 위한 critical path 회로의 구성

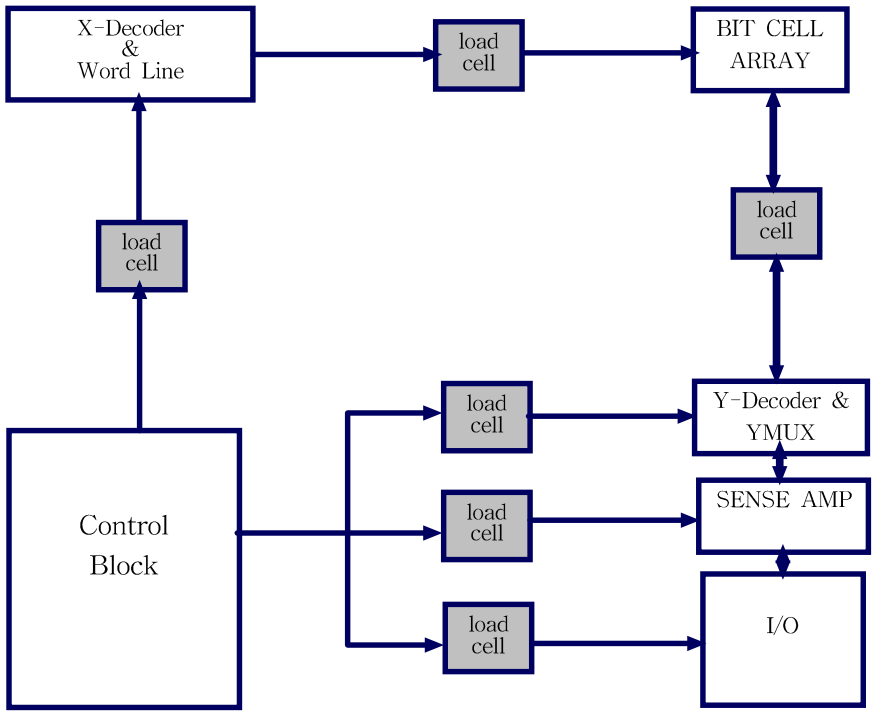
〔그림 2〕와 같이 〔그림 1〕의 메모리회로를 각 블록의 leaf-cell과 load-cell로 단순화 시킨 회로가 critical path 회로이다. critical path 회로는 메모리의 모든 동작모드를 표현하는 가장 단순화된 모델링 회로이다. 따라서 critical path 회로를 통하여 각 블록만의 소비전력을 쉽게 구할 수 있게 되며 전체 메모리의 소비전력을 계산하기 위한 계산식도 추출할 수 있게 된다. Critical path 회로 구성시 주의 할 점은 X 축 방향 load cell의 경우 AC timing 특성추출에서 사용된 load value 값을 I/O 수로 나누어 주어야한다.

3. 2 Arrayed-Leaf Cell Block 의 Power 시뮬레이션

〔그림 2〕에서 Control Block을 제외하고 Bit Cell Array를 포함 나머지 회로들은 해당 블록의 leaf-cell들이 메모리의 사이즈에 따라 자동으로 그 수를 늘이거나 줄이는 방법에 의해 Compiler가 S/W적으로 자동으로 배열되는 반복 구조인 것이다. 라이브러리와 메모리의 종류에 따라 차이는 있겠으나 대략적으로 다음에 따라 메모리 소비전력을 추출하고 계산해 낼 수 있다.



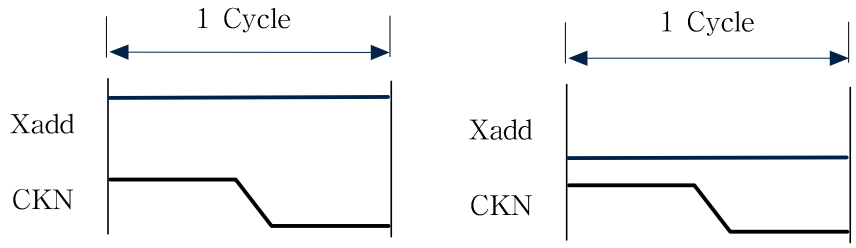
[그림 1] 메모리기본 구조



[그림 2] 메모리 Critical Path 회로

3. 2. 1 X-Address Decoder와 Word-Line Driver

두 회로는 보통 쌍으로 묶이게 되며 다음에 따라 블록파워를 구한다. 동작 pin은 CKN 이고 2 cycle 동안 rise 1번 fall 1번의 동작을 한다. 이유는 CKN의 event에 있어 Xadd 가 High와 Low의 확률이 1/2 이라는 가정하에서 2 cycel 동안의 average current인 P(xadd) 를 구한다.



[그림 3] X-Address Decoder 와 Word-Line Driver

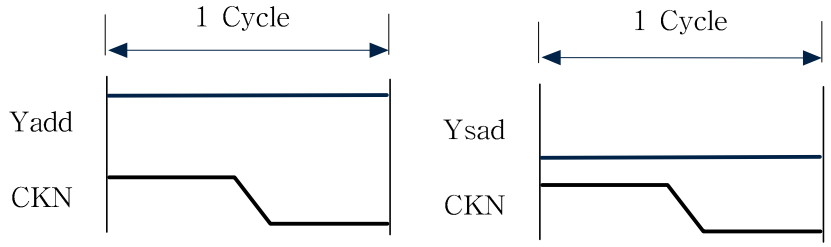
address 1-bit 의 결과이므로 X-Address 의 갯수만큼을 구해진 power 에 곱해 주면 X-Address Decoder 와 Word-Line Driver 블록의 전체 power를 추출하게 된다.

$$Pxaddt = P(xadd) \times (number\ of\ X\text{-}address) \quad (4)$$

만일, Control Block의 Power가 포함되어 있다면 Control Block만의 소비전력을 별도로 구해서 P(xadd)에서 빼 주어야 한다.

3. 2. 2 Y-Address Decoder 와 YMUX 블록

Y-Address Decoder(혹은 Column Address Decoder) 와 YMUX 블록의 power를 계산하기 위해서 구성된 circuit이다. 2 cycle 동안 Yadd High, Low 에서 CKN 이 1번씩의 transition이 있고 이것의 AVG current를 구하게 된다. 회로 시뮬레이터를 통해서 Y-Address Decoder와 YMUX 블록의 power current Pyadd 를 구하고 Y-address 수를 곱해주면 Y-Address Decoder 와 YMUX 블록의 실제 power를 계산해 낼 수 있다.



[그림 4] Y-Address Decoder 와 YMUX

$$Pyaddt = P(yadd) \times (number\ of\ Y\text{-}address) \quad (5)$$

3. 2. 3 Read Mode 에서의 Sense AMP와 I/O 블록

Sense AMP는 Read Mode 동작시에만 동작하며 I/O의 개수와 보통은 동일하다. Sense AMP와 I/O에 대한 leaf-cell에 대하여 CKN이 enable 될 때 Control Block에 의한 Sense AMP enable에 의하여 동작하기까지의 소비전력이므로 Control Block과 Sense AMP와 I/O 블록을 합친 결과 Pcsaio에서 Control Block만의 소비전력 Pc를 빼준 다음 I/O 수만큼 곱해주면 된다. Sense AMP와 I/O 블록의 소비전력 Psaio는 다음과 같다.

$$Psaio = (Pcsaio - Pc) \times (number\ of\ I/O) \quad (6)$$

3. 2. 4 Write Mode에서의 IO 블록

Write Mode 에서는 Sense AMP는 동작하지 않으며 역시 Control Block과 연동되므로 Control Block과 I/O 블록을 합친 결과 Pcio에서 Control Block만의 소비전력 Pc를 빼준 다음 I/O 수만큼 곱해주면 된다. Write Mode에서의 I/O 블록의 소비전력 Psaio는 다음과 같다.

$$Pio = (Pcio - Pc) \times (number\ of\ I/O) \quad (7)$$

3. 2. 5 Bit Cell Array Block와 Pre-Charge Block

Critical Path 회로에는 1개의 Bit Cell 에 대해서 모델링이 되어 있으므로 Cols 배 해주어야한다. 또한 회로의 동작에는 지금까지의 모든 블록들이 있어야 하므로 구한 소비전력 Pa에서 빼주어야한다. Word-Line에 대해서 모든 Cols의 Bit Cell들과 Pre-Charge 회로가 동작하므로 Cols를 곱해주어야 한다.

$$Pcore = (Pa - (Pxadd + Pyadd + Pcsaio)) \times Cols \quad (8)$$

3. 3 Non Arrayed-Leaf Cell Block의 Power 시뮬레이션

〔그림 1〕에서 Control Block이 Non Arrayed-Leaf Cell Block이므로 CKN enable 시 Control Block에 대한 파워 Pc 만을 추출하면 된다.

3. 4 메모리 전체의 Power Calculation

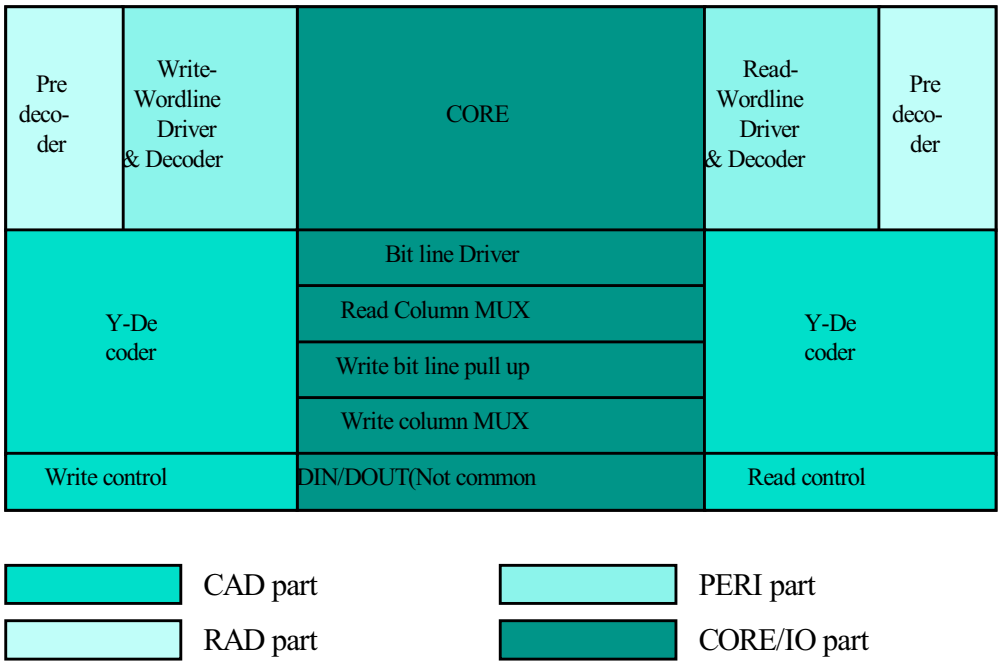
메모리 전체의 소비전력은 Read-Mode와 Write-Mode에 대하여 별도 식으로 계산된다.

$$Pread = (Pxaddt + Pyaddt + Psaio + Pcore + Pc) \quad (9)$$
$$Pwrite = (Pxaddt + Pyaddt + Pio + Pcore + Pc) \quad (10)$$

4. 2-Port 메모리를 이용한 결과 비교

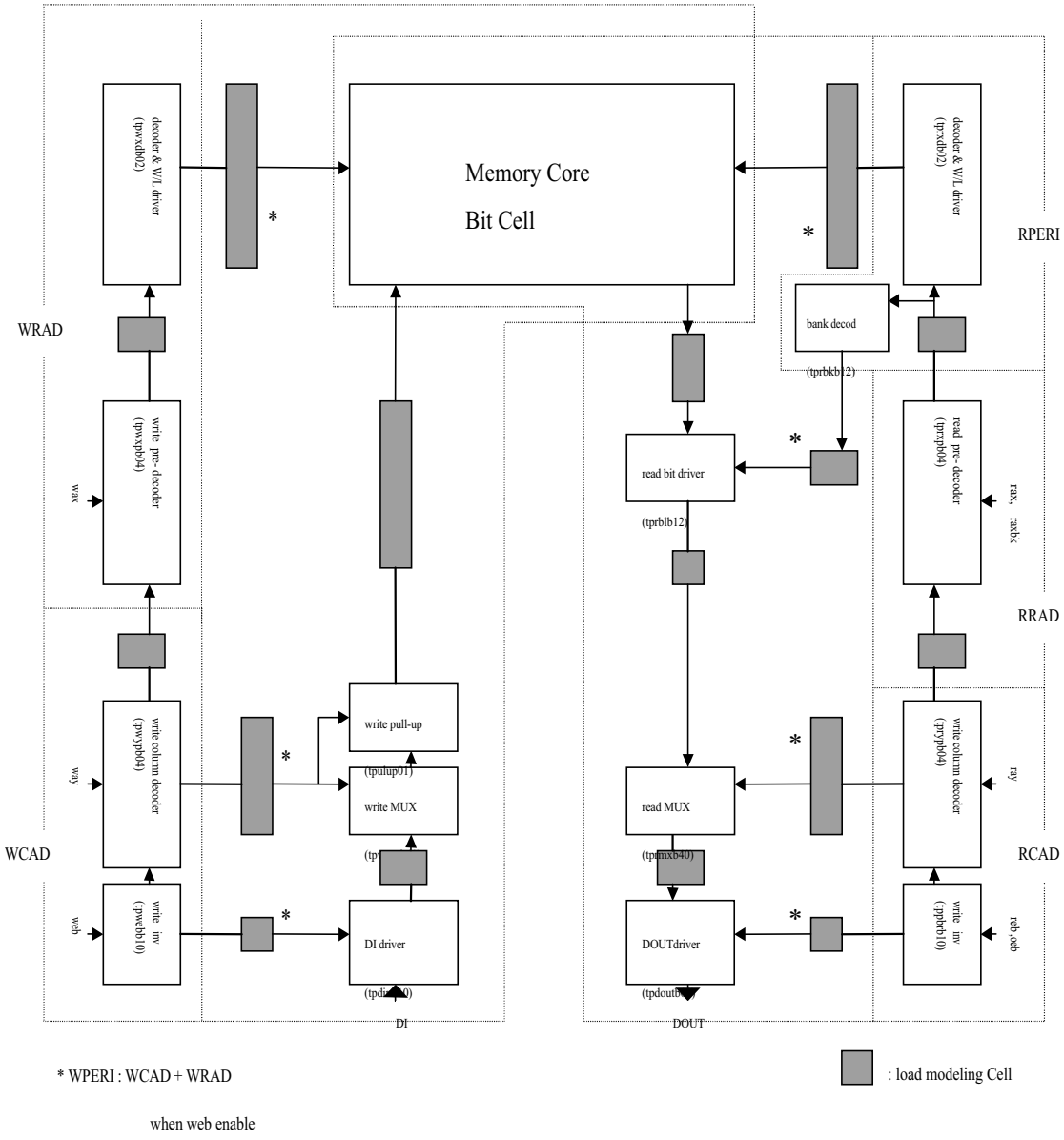
본 논문에서 제안된 방법에 의한 소비전력결과를 검증하기 위해 Write-Mode 동작과 Read-Mode 동작포트가 별도로 구성되고 파워계산이 좀 더 복잡한 0.35um 메모리

compiler 라이브러리인 2-Port Asynchronous RAM에 대하여 제안된 방법을 적용하여 Critical Path를 구성하고 회로블록별로 파워를 추출하였으며 구해진 모듈별 파워를 이용하여 계산된 결과와 메모리 전체회로를 회로시뮬레이션한 결과를 비교하였다. 〔그림 5〕는 2-port RAM의 block diagram 이다.



[그림 5] 2-port memory 구조

2-port RAM은, 한 port는 read 전용, 다른 port는 write전용으로 만들어져 있다. 〔그림 6〕은 메모리 critical path 회로도이다. Power 시뮬레이션을 위해서는 〔그림 5〕에서와 같이 critical path를 power 시뮬레이션을 위해서 새롭게 구성 할 필요가 있다. 특히 * 가 있는 load cell 의 경우 모델링된 전체 load value 값을 I/O 수로 나누어 주어야한다. Table 1에 block 별로 추출한 0.35um 2-Port RAM 1024x4 의 소비전력으로 기준 data와 8% 오차를 보였다. 실제 메모리 동작에 있어서 OEB(output enable), CSN(chip enable), REB(read enable), WEB(write enable) 등과 같은 signal 에 대해서는 transition 의 예측이 불가능하다. 따라서 새로운 추출방법에 의한 결과가 기준 Data 보다는 다소 작게 나왔다. 그러나 몇 개의 추가적인 시뮬레이션 비교를 통하여 Derating Factor를 추출하여 메모리 라이브러리 전체에 적용한다면 오차를 최소화 할 수 있을 것이다.



[그림 6] Power Simulation을 위한 회로의 재구성

Table 1. 0.35um 2-Port RAM 1024x4 소비전력 추출 결과비교분석

(Temp 25 , Typical process , cycle time 30N, input slope : 0.1ns,output load :0pF)

power 성분	Suggested Method Result (mA)	Reference Data (mA)
Prcad(Read Column Address Block)	0.095	메모리전체 회로에 대한 HSPICE 시뮬레이션 Results
Prrad(Read Row Address Block)	0.145	
Prperi(Read Peri Block)	0.117	
Prcore(Read Core Block)	0.095	
Pwcad(Write Column Address Block)	0	
Pwrad(Write Row Address Block)	0.075	
Pwperi(Write Peri Block)	0.383	
Pwcore(Write Core Block)	0.645	
Pread(Read Mode Power)	0.715	
Pwrite(Write Mode Power)	0.959	
Pt(Total AVG Power)	1.674	1.812

5. 결론

메모리 콤파일러의 특성추출은 일반 단순로직 라이브러리와는 달리 정확성을 위한 특성추출 flow가 매우 복잡하고 시간 또한 많이 소요된다. 특히 소비전력을 추출하는데는 메모리 회로의 복잡성, 다양성으로 인해 기존의 회로시뮬레이터에 전적으로 의존하는데 한계가 있다. 또한 compiled 메모리의 특성상 생성할 수 있는 모든 메모리 configuration에 대한 파워특성에 대해 database를 구축한다는 것은 불가능하다. 따라서 전체 라이브러리 소비전력 특성추출의 정확성을 위해서는 가능한 많은 configuration에 대한 소비전력 특성추출을 실시하는 것이 중요하다. 메모리 파워특성의 정확성은 실제 시뮬레이션 된 파워결과들로부터 발생하는 interpolation 에러와 extrapolation 에러를 최소화하는데 있기 때문이다.

본 논문에서는 leaf-cell의 반복구조인 메모리회로의 특성을 이용하여 각 블록별 소비전력을 구해놓고 전체 파워계산 식에 이들의 값을 적용하여 원하는 메모리 사이즈의 최종 파워를 계산함으로 해서 기존의 파워추출의 단점이었던 시뮬레이션 한계와 이에 따른 interpolation 에러와 extrapolation 에러의 증가문제를 해결할 수 있는 개선된 기법을 제안하여 메모리 라이브러리 전체적으로 소비전력 값의 정확도를 향상시킬 수 있게 되었다. 이 방법은 메모리의 회로 구조에 따라 블록별 시뮬레이션 추출을 해야하며 전체 소비전력 계산식을 결정 해야 한다.

샘플 메모리 instance인 2-port RAM에 대하여 제안된 방법에 의한 결과, 기준 자료와 8% 오차를 보였으며 이는 더욱 정확한 회로모델링과 derating factor의 적용을 통하여 최소화 할

수 있다. 다만, 좀더 다양한 크기의 메모리에 대하여 기준 자료와 제안된 방법의 소비전력 결과 비교를 통하여 메모리 크기에 따른 오차의 경향을 분석하고 이를 반영할 필요가 있다.

제안된 메모리 파워추출기법을 이용하면 시뮬레이션 결과의 수를 시간제한 없이 무한정 늘릴 수 있으므로 소비전력 추출의 오차를 극소화 할 수 있을 것으로 기대된다.

참고문헌

〔1〕 정승민, 1999, "ASIC 메모리 콤파일러 라이브러리 특성추출 기법", *용인송담대학논문집*, 제2집, pp53-65.

〔2〕 Betty Prince, 1990, "Semiconductor Memories", Texas Instruments, USA.

〔3〕 Martin Margala, 1999, " Low-Power SRAM Circuit Design", *Proceedings of the 1999 IEEE International Workshop of Memory Technology, Design and Testing*, pp115-122.

〔4〕 Joerg Vollrath, 1999, "Tutorial: Characterizing SDRAMs", *Proceedings of the 1999 IEEE International Workshop of Memory Technology, Design and Testing*, pp62-71.

〔5〕Alessandro De Gloria, Paolo Faraboschi, and Mauro Olivieri , Jun.1994, "Design and Characterization of a Standard Cell Set for Delay Insensitive VLSI Design", *IEEE Transactions on Circuits and Systems*, vol.41, No.6, pp.410-415

〔6〕 K.Anami, M.Yoshimoto, H.Shinohara, Y.hirata, and T.Nakano, Aug.1983 ,"Design considerations of a static memory cell", *IEEE J. Solid-state Circuits*, vol.SC-18,no.4, pp.414-418

〔7〕 R.C. Jaeger and R. M.Fox, June 1985 , "Phase plane analysis of the upset characteristics of CMOS RAM cells" ,*Proc.Univ./Govt./Industry Microelectron.Symp.*, pp.183-187