다중 전압과 모듈 배정을 활용한 온도 고려의 Datapath 합성

박신조[1], 김태환[2]

[1]한국과학영재학교, [2]서울대학교 전기컴퓨터공학부

(Temperature-Aware Datapath Synthesis Utilizing Multiple Voltage and Module Binding)

Park Shinjo[1], Kim Taewhan[2]

[1]Korea Science Academy, [2]Seoul National Univ. Dept. of EE/CS

요 약

칩의 온도 상승에 대한 우려는 최근 점점 가시화되고 있다. 즉, 설계 집적도의 증가에 따른 전력 소모 밀도의 증가는 바로 칩 온도 상승으로 이어지고 있다. 이러한 칩 온도 상승은 성능 저하와 패키징 비용 증가 뿐 만 아니라, 칩의 신뢰성, 칩 수명에서도 나쁜 악영향을 초래한다. 본 연구는 칩 온도 상승을 억제하기 위한 상위 단계 합성을 제안하고 있다. 구체적으로 본 연구의 핵심은 다중 전압 할당과 연산에 대한 모듈 바인딩(배정)을 동시에 고려한 새로운 저온도 설계 기법을 시도한다. 과거의 이중 threshold 전압 할당과 모듈 바인딩은 각각 누설 전류와 동적 전류를 줄이기 위해 적용된 반면 본 연구는 온도 최소화 측면에서 연구를 시도한 점에서 다른 설계 가능성을 보여 준다고 하겠다.

1. 서 론

SOC (System-on-chip) 설계에서 연산(operation)들을 수행하는 HW 단위를 모듈(module) 이라 부른다. (예: 가산기, 감산기 등) 상위 단계 합성에서 이러한 모듈들이 할당되고 연산들은 할당된 모듈에 배정이 되어 수행이 된다. 칩 온도 측면에서 특정 모듈이 상대적으로 많은 연산들을 수행하도록 배정되었다면 그 모듈은 많은 전력 소모를 가질 것이고 따라서 열 발산이 많아지며 그 모듈과 주위에 온도 상승을 유발하게 된다. 반대로, 모듈들이 골고루 연산들을 배정 받았다면 열 발산이 분산되어 최고 칩 온도도 상대적으로 낮아지는 효과를 낼 것이다. 본 연구는 이러한 모듈 배정이 최고 온도 상승을 최소로 만드는 방향으로 되도록 하는 방법을 찾는 것이다. 추가적으로 기존 연구와의 다른 점은 모듈에 공급되는 전압의 크기를 다르게 배정함으로써 이것이 열 발산에 미치는 영향을 반영하는 기법을 제안한 것이다. 이제 본 연구 내용에 대한 자세한 설명을 한다.

만약 저발열 (low thermal) 설계가 제대로 되어 있지 않은 경우에는 성능에 비해 많은 열이 나서 상용화할 수 없을 것이다. 이 열을 어떻게 줄일 것인가? 서로 다른 연산을 하나의 모듈에서 수행하는 데는 연산을 전환하는 데 드는 비용이 있다. 같은 연산을 하나의 모듈에 몰아주거나, 전환 비용이 적은 연산끼리 몰아주면 비용을 줄일 수 있다. 이것까지의 연구 결과는 기존의 연구 논문에서도 찾아 볼 수 있다. 그런데 모듈의 공급 전압을 고려하면 어떻게 최적화할 수 있는가? 전압 배정과 모듈 배정을 모두 동시에 고려할 경우 얼마나 효과가 있을까? 모 연구는 이에 대한 해답을 찾고자 한다.

2. 배경 지식

"전환 비용"(switching activity)은 모듈에서 서로 다른 연산 사이를 전환할 때 발생하는 비용이며, 연산의 특성이다. "스위칭 테이블"은 각각 경우의 전환 비용을 나타내는 표이다. 가로에서 세로로 읽어 가며, 0은 같은 연산을 수행하여 스위칭 비용이 없다는 것이다.

앞으로 언급할 방법은, 오른쪽의 스위칭 테이블의 전환 비용을 줄여서 전환 비용의 최고치를 줄이거나, 전압을 줄이는 것이다. 그러나 전압을 줄이면 연산을 실행하는 속도가 감소하므로 부작용을 일으킬 수 있다. 초저전압 프로세서들이 비슷한 가격의 보통 전압 프로세서에 비해서 성능이 안 좋은 것이 증거이다.

표 2 스위칭 테이블의 예시

	0	1	2
0	0	3.6	3.4
1	3.5	0	3.9
2	3.0	3.3	0

회로에서 나는 열을 측정하려면, 회로를 작동 시켜 측정해야 한다. 실제로 열전도 실험을 하려면 칩 제조부터 전압 설계 등을 직접 해야 하므로 시간이 오래 걸린다. 그러나, 설계 단계 동안 실험을 할 때는 이것을 자동화해 주는 소프트웨어를 사용하게 될 것이다. 이를 이용한 시뮬레이션 방법 및 결과 분석은 결과 분석 장에서 자세히 다루겠다.

3. 과거 연구 정리

열 발산을 억제하기 위한 상위 단계 합성 (high-level synthesis) 연구에는 [1], [2]가 있다. [1]은 [2]의 알고리즘을 향상시킨 것으로 네트워크에 기반한 최적의 알고리즘을 사용하였다. 기존의 연구들 ([1], [2])은 최고 전환 비용만 고려하거나, 최고 전환 및 전체 전환 비용 둘 다 고려하는 방식으로 동작한다. 연구 [3]은 전력 소모를 줄이기 위한 다중 전압을 할당하는 알고리즘을 제안하였다.

연구 [1]과 [2]에서는 전압에 관한 고려가 없으며, 연구 [3]의 저전압 할당은 공급전압에 따른 열 발산 최소화를 고려한 것과는 다른 목적으로 연구되었다. 과거 연구를 보아도 공급 전압 할당과 연산 모듈의 바인딩을 동시에 고려하여 칩의 peak 온도를 줄이는 연구는 아직 없으며 여기에서 효용성을 입증하고자 한다.

4. 알고리즘 제안

이 연구에서 개발한 알고리즘은 전압에 의한 잉여 에너지 손실을 방지하기 위해서 연산마다 다른 전압을 사용한다. 다음은 알고리즘의 동작 방법이다.

입력: 스케줄링이 되어 있는 DFG, 스위칭 테이블, 사용할 전압(들)

출력: 바인딩이 된 DFG

이 값들을 받은 다음 이와 같이 동작한다.

단계 1: 맨 처음에 실행되어야 할 작업들을 바인딩 한다. 이들 작업들은 가장 빠르게 실행되어야 하는 작업이며, 다음 작업이 있는지를 고려해서 전압을 설정해야 한다.

단계 2: 그 다음으로 실행되어야 하는 연산을 바인딩하기 위해서 바인딩할 수 있는 모듈과 전압을 모두 따져 본다.

단계 3: 그 작업의 데드라인을 만족하는 범위 안에서 가장 전력을 적게 소모하는 바인딩을 찾는다. 이 때 여러 가지 방법을 사용할 수 있다.

단계 4: 위에서 찾은 바인딩은 실제로 바인딩하고 다음 연산으로 넘어간다.

단계 5: 단계 2-4를 모든 연산에 대해서 수행한다.

이 알고리즘은 가능한 모든 노드에서 데드라인을 어기지 않는 바인딩 중의 최적의 바인딩을 찾아낸다. 핵심 부분 단계 2와 3에서 정확하고 빠르게 노드들을 걸러내느냐가 중요하다. 짧은 예제들을 보면서 바인딩이 어떻게 이루어지는지 알아보자.

4.1 전압 바인딩

전압을 줄이면 연산 속도는 줄어든다. $P \propto V^2$이므로 줄어든 전압의 제곱에 비례해서 전력이 줄어든다. 저전압 바인딩을 사용할 수 없는 경우도 있다. 데드라인에 맞추어 실행할 수 없으면 불가능하다. 구체적인 사례를 들어서 저전압 바인딩의 동작을 보자.

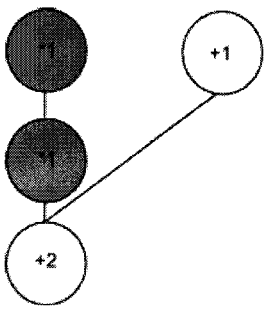

그림 1 바인딩 예제

연산 *1의 데드라인은 각각 1,2, +2의 데드라인은 3, +1의 데드라인은 2라고 하자. +1 연산을 느리게 실행시키면 2 단위 시간에 실행되고, 이 연산의 결과 값을 필요로 하는 +2 연산은 시간 3에 실행되어서 문제가 없다.

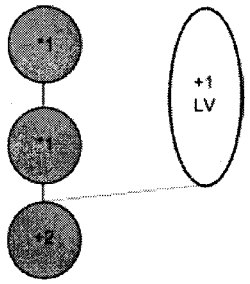

그림 2 저전압 바인딩

저전압으로 +1 연산을 바인딩하면 전압이 절반으로 감소하기 때문에 $P = \sum_{i=1}^{n} V_i^2 \times C_{sw_i}$ 식에 의해서 전력은 1/4배로 감소한다는 것을 알 수 있다. 저전압 바인딩이 많이 들어가면 전력을 많이 줄일 수 있다.

저전압 바인딩에서 충돌이 생기는 상황을 보자. 연산 +1, +2의 데드라인은 모두 1이고, *1의 데드라인은 2라고 하자. 바인딩 알고리즘은 이 경우 +1, +2를 모두 고전압 모드로 바인딩한다.

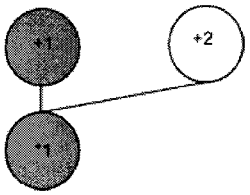

그림 3 충돌이 생기는 노드

만약 이렇게 하지 않으면 데드라인 조건을 지킬 수 없고 프로그램 실행은 실패한다. 이런 문제를 해결하려면 데드라인을 넉넉하게 주어야 한다.

4.2 모듈 바인딩

하나 이상의 모듈에 바인딩하면 한 모듈에만 바인딩 하는 것에 비해서 어떤 장점이 있는가? 하나의 모듈에 모두 바인딩하면 한 번에 하나의 작업만 실행되므로 시간이 많이 소요된다. 여러 모듈에 나누어서 바인딩하면 한 번에 여러 작업을 실행시켜서 속도도 빨라진다.

또 다른 장점은 Switching capacity를 줄일 수 있다는 것이다. 한 모듈에서 많은 연산을 처리하면, 연산마다의 전환 비용을 모두 계산해야 한다. 서로 다른 모듈로 연산을 나누면 한 모듈에 같은 연산을 몰아주어서 처리 비용을 줄일 수 있다. 비단 같은 연산을 몰아주지 않아도 전환 비용이 적은 연산끼리 몰아줄 수 있다.

전환 비용을 줄이기 위해서라면 동작 A_i가 모듈에 바인딩한

다음에 결정할 수 있다. 동작 A_{i+1}을 바인딩하기 전에, 다음 동작과 같은 시간대에서 실행될 수 있는 연산이 얼마만큼의 시간을 소모하느냐 등을 비교해 보면 전체 전환 비용을 줄일 수 있다.

만약 충돌이 일어나는 경우, 충돌이 나는 경우는 제외한다. 이 방법의 단점은 모든 것을 다 돌아보기 때문에 시간이 그만큼 많이 걸린다는 것이며, 이것을 보완하기 위한 Branch-and-bound 알고리즘을 사용하는 연구가 진행되고 있다.

최종적으로는 두 가지의 최적화 옵션을 다 같이 사용하므로, 이 알고리즘이 최종적으로 최적화시키는 값은 다음과 같다.

$$C = w_1 \times \sum C_{sw} + w_2 \times V^2 C$$

여기에서 이루어진 모든 실험 결과들은 이 값을 절약하는 방법으로 진행된다.

5. 결과 및 정리

실험 및 프로그램을 단순화하기 위해서 2가지 종류의 전압을 사용하였으며, 모듈 개수를 2개로 제한하였다. 기존 알고리즘과의 비교 항목은 전체 전력이다. 이 논문의 바인딩 알고리즘의 특성 상 전압이 낮아지고 높아지는 부분이 생기기 때문에, 이를 반영하려면 전력을 측정해야만 한다. 수작업으로 얻은 바인딩에서 계산할 수 있는 결과는 총 스위칭 용량이다. 이와 전압을 통해서 전체 전력을 계산할 수 있다.

● EW Filter에 대한 실험 결과

여기서는 이미 만들어져 있는 몇 개의 데이터 흐름 그래프를 바인딩해 볼 것이다. 처음으로, 참고자료 [4]에 있는 EW Filter 그래프를 바인딩해 볼 것이다. 이 데이터 액세스 그래프는 +, * 연산 2개를 사용하며, 총 모듈은 적어도 3개 필요하다. 이제 기존의 방법과 새로운 방법을 비교하도록 하자.

연산이 10개이므로 스위칭 테이블은 다음을 사용한다.

표 3 스위칭 테이블

	1	2	3	4	5	6	7	8	9	10
op1	0	3	4	4	6	2	2	6	2	4
op2	6	0	6	1	3	1	2	2	7	1
op3	8	6	0	2	5	1	5	7	5	2
op4	3	2	4	0	8	6	1	1	4	8
op5	2	6	5	4	0	7	5	4	1	1
op6	1	8	8	2	8	0	2	4	3	6
op7	3	5	2	2	3	2	0	3	3	6
op8	4	8	5	4	7	8	7	0	6	5
op9	6	4	2	7	8	2	1	2	0	6
op10	8	5	2	7	6	8	1	8	7	0

$$C = w_1 \times \sum C_{sw} + w_2 \times V^2 C$$

여기서도 저전압은 1.5V, 고전압은 3V를 사용한다. 식에서 w_1, w_2 값은 모두 1이고 C값은 모두 스위칭 테이블에서 온다. 전압을 고려하지 않은 기존 방법을 사용했을 때 그래프는 다음과 같다.

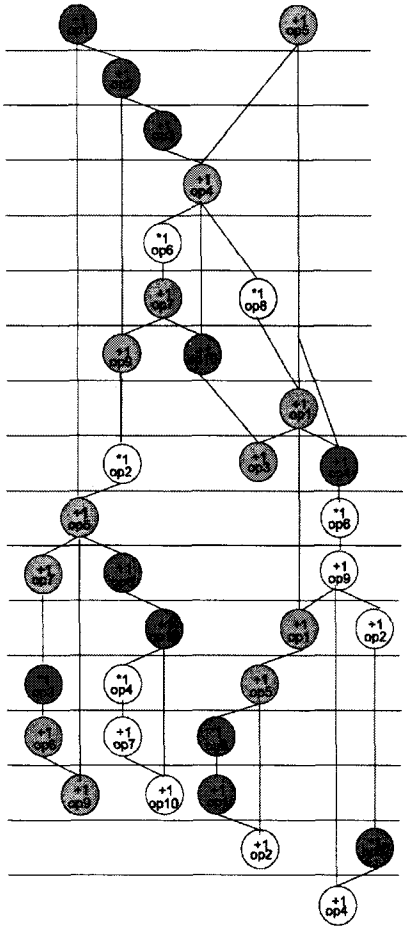

그림 4 EW Filter: with older binding method

그래프에서 진한 회색은 모듈 1, 연한 회색은 모듈 2, 흰색은 모듈 3이다. 이제 각각 모듈에 대해서 전환 비용과 전력을 구해보자.

모듈 1	41
모듈 2	42
모듈 3	34
C_1	369
C_2	378
C_3	306

모듈 2는 항상 같은 연산을 처리했기 때문에 전환 비용에 의한 전력이 0이다. 이번에는 저전압 모드를 사용한다.

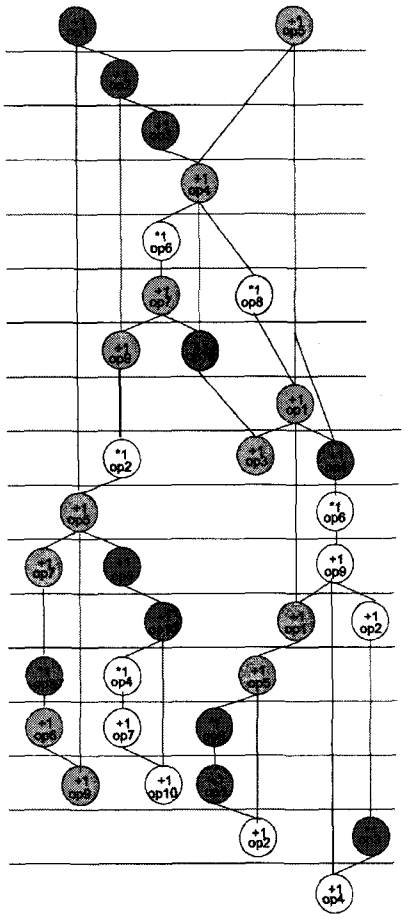

그림 5 EW Filter: 제안한 기법에 의한 결과

정리해 보면 다음과 같다.

모듈 1	36
모듈 2	48
모듈 3	32
C_1	310.5
C_2	364.5
C_3	261

전환 비용에 의해서 발생하는 전력량은 똑같지만, 연산 중 일부는 저전압 연산이다. 이 연산들은 1/2 전압에서 수행되기 때문에 전력을 1/4만 사용한다. 이러한 복잡한 그래프에서는 저전압 연산을 자주 사용할 수 없기 때문에 이 정도만 되더라도 전력을 줄일 수 있다.

- AR Filter에 대한 실험 결과

참고자료 [4]에 있는 또 다른 그래프의 예를 들어 보자. 이제 이 필터에 대해서도 위와 같은 비교 과정을 거칠 것이다. 그런데 여기서는 *1, *2, +1 연산으로 구성되어 있으므로 아까 와는 또 다른 스위칭 테이블을 사용한다.

	1	2	3	4	5	6	7	8	9	10
op1	0	3	2	4	4	7	2	4	5	2
op2	8	0	1	7	2	5	3	4	4	6
op3	5	8	0	1	2	6	8	1	6	6
op4	1	4	1	0	5	1	5	6	8	6
op5	6	6	4	4	0	7	3	4	5	7
op6	1	7	8	5	3	0	4	6	8	5
op7	6	4	3	2	6	3	0	8	7	2
op8	6	2	6	4	5	3	1	0	5	8
op9	3	4	6	5	1	4	4	5	0	4
op10	4	1	6	4	7	8	2	8	1	0

표 6 스위칭 테이블

여기서는 총 사용하는 모듈이 3개이다. 이제 구형의 바인딩부터 보자.

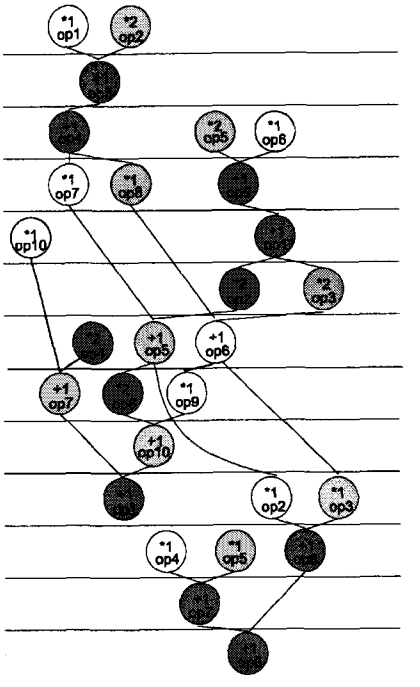

그림 6 AR Filter: 기존 결과

여기서는 저전압 모드가 사용되지 않았다. 이 경우에서 모듈별로 사용하는 전환 전력을 알아보자.

모듈 1	56
모듈 2	24
모듈 3	40
C_1	504
C_2	216
C_3	460

전환 비용에 의해서 이 정도의 전력이 얻어진다는 것을 확인했다. 이제 새로운 바인딩 방법으로 시도해 보자.

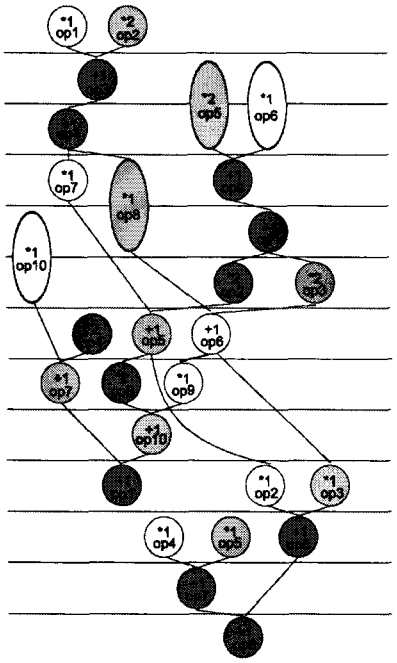

그림 7 AR Filter: 본 제안한 기법에 의한 결과

이제 여기에서의 전력을 비교하겠다.

모듈 1	56
모듈 2	24
모듈 3	40
C_1	504
C_2	155.25
C_3	315.5

역시 전환 비용 면에 있어서 큰 감소는 없었지만, 전환 전력이 많이 감소하였다.

- 실험 결과 정리

그 동안 나왔던 모든 결과를 비교해 보자. 먼저 맨 처음 나왔던 예제 그래프이다.

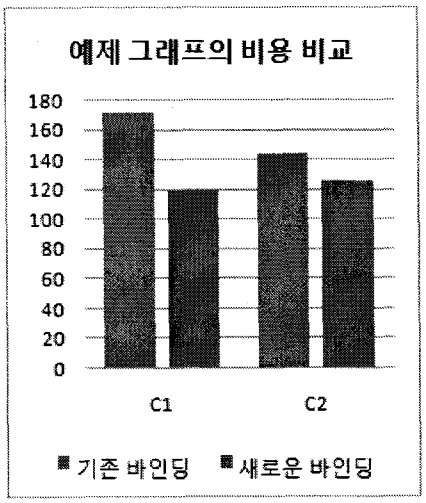

그림 8 예제 그래프의 비용 비교

그래프에서 볼 수 있듯이, 모든 모듈에서 전력에 의한 비용이 최소화되었다. 다음은 두 번째로 든 EW Filter의 비용이다.

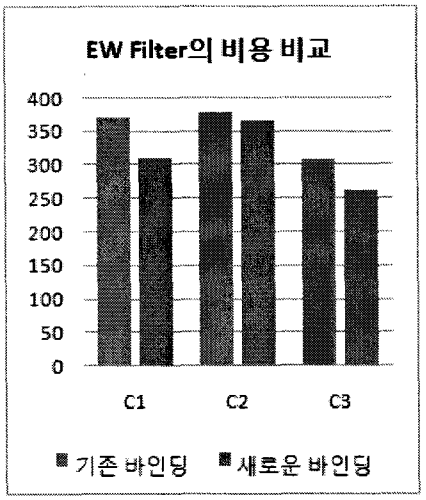

그림 9 EW Filter의 비용 비교

세 모듈 다 전력에 의한 비용이 줄어드는 것을 볼 수 있었다. 마지막으로 AR Filter의 비용 비교이다.

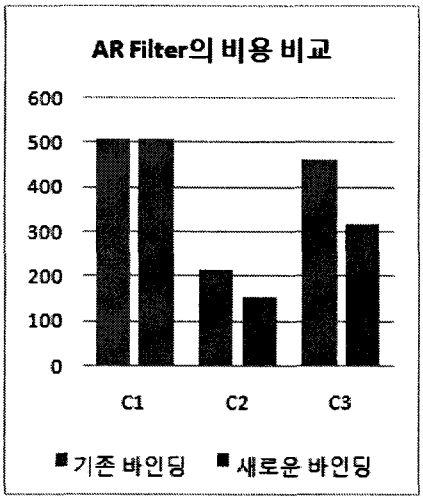

그림 10 AR Filter의 비용 비교

모듈 1개에서는 비용이 줄지 않았지만, 전체적으로 보았을 때는 감소하였다. 표로 정리해 보면 다음과 같다.

예제 그래프			
	기존	새로운	감소폭
C_1	171	119.25	**69%**
C_2	144	126	88%
EW Filter			
	기존	새로운	감소폭
C_1	369	310.5	84%
C_2	378	364.5	96%
C_3	306	261	85%
AR Filter			
	기존	새로운	감소폭
C_1	504	504	100%
C_2	216	155.25	71%
C_3	460	315.5	**68%**

표 9 총정리

감소폭이 없었던 일부 경우를 제외하더라고, 최저 90%에서 최고 70%까지 소모 전력을 감소시킬 수 있었다. 이 정도 효율이라면 저전압에 의한 연산 수행 시간 저하나 바인딩 및 스케줄링 과정의 복잡함을 감수하고도 남을 수준의 최적화이다.

6. 결론

제안된 알고리즘이 지향하는 방향은 전체 Peak 전력 소모 최소화를 통한 저온도 설계이다. 기존의 알고리즘과는 달리 저전압 모드를 효율적으로 사용하므로 실질적인 열은 그다지 높지 않을 것이다. 기존의 바인딩 방법에서는 전체 전압이 일정하지만 이 방법은 다중 전압을 사용하기 때문에 비교를 하기 힘들 것이다. 만약 비교를 하려면 실행 시간의 손해와 훨씬 적은 전력도 고려해 주어야 할 것이다.

7. 참고 문헌

[1] P. Lim and T. Kim, "Thermal-Aware High-level Synthesis Based on Network Flow Method," ACM International Conference on HW/SW Co-design, 2006.

[2] R. Mukherjee and M. Seda, Peak "Temperature Control and Leakage Reduction During Binding in High Level Synthesis," ACM/IEEE Design Automation Conference, 2005.

[3] K. Shin and T. Kim, "Leakage Power Minimization for the Synthesis of Parallel Multiplier Circuits," ACM Grate Lakes Symposium on VLSI and CAD, 2004.

[4] T. Kim and C. L. Liu, "An Intregrated Algorithm for Incremental Data Path Synthesis", The Journal of VLSI Signal Processing, 1995.