정보학연구 제3권 제1호 2000년 3월(pp.117-124)
The Studies of Information Technology. Vol.3. No.1

설계 자동화를 위한 저전력 하드웨어 할당 알고리듬

최지영*/인치호**

요 약

본 논문은 설계 자동화를 위한 저전력 하드웨어 할당 알고리듬을 제안한다.
제안한 알고리듬은 스케줄링의 결과를 입력으로 받아들이고, 각 기능 연산자에 연결된 레지스터 및 연결 구조가 최대한 공유하도록 제어스텝마다 연산과 기억 소자의 상호 연결 관계를 고려하여 기능 연산자, 연결 구조 및 레지스터 할당한다. 또한 자원의 수를 최대로 줄임으로써 할당 과정동안 저전력의 요소인 캐패시턴스를 동시에 줄인다. 제안된 알고리듬은 비교 실험을 통하여 기존의 저전력을 고려하지 않은 방식들과 비교함으로서 본 논문의 효율성을 보인다.

Ⅰ. 서론

특수 목적 및 소량 생산이 요구되는 ASIC (Application Specific Integrated Circuit)이 산업 전반에 걸쳐 제품의 소형화와 고급화의 필수적인 요건이 되고 있기 때문에 직접회로의 설계에서 제작에 이르는 회송(turn-around)시간의 단축과 비용의 감소를 위해 CAD 기술은 필수적이다.[1] 또한 설계할 IC칩의 동작 기술로부터 칩 제조를 위한 마스크(mask) 도면을 자동으로 생산하는 설계자동화(Design Automation) 는 CAD기술의 최종 목표이다.[2-4] 현재 논리 설계나 레이아웃 설계 자동화의 많은 연구가 진행되어 상용화되고 있으나, VLSI의 집적도와 복잡도의 증가에 따라 짧은 설계 시간, 설계 초기 단계에서 칩의 성능 평가에 따른 다양한 설계, 자동화에 의한 디버깅 시간 단축 및 적은 오류, 설계 과정에 대한 다큐멘테이션등의 특징을 가진 상위 레벨 합성에 대한 연구가 활발히 진행되고 있다. 상위 레벨 합성은 설계하고자하는 시스템의 동작 기술로부터 주어진 제한 조건과 목적 함수를 만족하는 레지스터 전송(register-transfer)레벨의 구조를 생성하는 단계로서 스케줄링, 할당, 바인딩으로 구성된다. 스케줄링은 동작기술에서 연산(operation)들을 특정한 제어스텝에 할당하는 과정이다.[5] 할당은 구현되는 하드웨어 면적이 최소가 되도록 연산을 기능 연산자(functional unit)에, 변수(variable)를 레지스터에 지정하고 레지스터와 연산자 사이의 연결구조(interconnection)로 버스(bus)나 멀티플렉서(multiplexer)를 할당하는 과정이다.[6] 그러나 지난 수 년간 상업적 VLSI를 위한 실제적인 중요성의 설계 범위는 면적과 최대한의 성능이었고, 저 전력을 고려한 것은 거의 고려되지 않았다. 설계자는 전력 소모를 단지 후에 고려하였고 저 전력을 위한 최적화보다 packaging을 결정하기 위하여 결정하였다. 그러나 최근의 전력

* 청주대학교 제어 및 컴퓨터 전공
** 세명대학교 컴퓨터과학과

소모는 설계의 전체적인 성능에서 점점 더 중요하게 되었다.

CMOS 회로에서 전력 소모의 주된 원인은 스위칭전력(switching), 단락전류(short-circuits currents), 누설전류(leakage currents)인데 주로 스위칭전력이 주로 차지하고 있다. CMOS 게이트에서 소모되는 평균 전력은 $\frac{1}{2} \cdot \frac{C_L \cdot V_{DD} \cdot N}{T}$ 이다. [7] C_L는 게이트 출력 캐패시턴스, V_{DD}는 공급전압, N은 연산의 주기 T 동안 게이트 출력 트랜지션(gate output transition)의 수이다. 위 식에서 저전력 소모 방안을 3가지 방법을 들 수 있다. 첫째, 캐패시턴스(C_L)를 줄인다. 둘째, 공급전압(V_{DD})를 줄이거나 마지막 transition activity(N/T)를 줄이는 것이다. 공급 전압을 줄이는 것은 전력을 소비하는데 큰 역할을 하는데 공급 전압이 줄어들면 전력은 제곱근으로 줄어들기 때문이다. 본 논문에서는 캐패시턴스를 줄이는데 초점을 둔다. 회로에서 캐패시턴스는 레지스터, 기능연산자, 바인딩 즉, 멀티플렉서 버스 등에 의해 결정된다. 그러므로 최소의 자원을 할당하게되면 캐패시턴스의 전력 소모도 줄어들게 된다.

따라서 본 논문에서는 설계 자동화를 위한 새로운 저전력 하드웨어 할당과 바인딩을 동시에 수행하는 알고리듬을 제안하였다. 본 알고리듬은 저전력을 고려하여 각 기능 연산자에 연결된 레지스터(register) 및 연결 구조(interconnection)가 최대한 공유하도록 제어 스텝마다 연산과 기억 소자(storage element)의 상호 연결 관계를 고려하여 할당 바인딩함으로써 자원의 수를 최대한 줄이도록 함으로써 할당 동안에 캐패시턴스를 줄이도록 한다. 그러므로 캐패시턴스와 면적의 최소화는 일치하는 공통적 목적이다. 기존의 문제점을 해결함과 동시에 전체 비용을 줄인다.

본 논문의 구성은 1장에서의 서론에 이어 2장에서는 본 논문에서 제안한 저전력 하드웨어 할당 알고리듬을 다루고, 3장은 비교 실험을 통하여 본 알고리듬의 효율성을 보이고, 마지막 4장은 본 논문의 결론으로 구성하였다.

Ⅱ. 저 전력 하드웨어 할당 알고리듬

본 논문에서 제안한 저전력 할당 및 바인딩에 대한 알고리듬은 그림 1과 같다. 입력은 스케쥴링 결과를 받으며 전 처리 과정으로 기능 연산자가 할당 및 바인딩 될 모든 연산의 이동도를 계산한다. 연산의 이동도는 데이터의 의존도를 조사하여 구한다. 전 처리 과정에서 각 연산의 이동도를 계산한 후 첫 번째 제어 스텝부터 단계별로 레지스터, 기능 연산자, 연결 구조를 할당 바인딩 한다. 이때 한 제어 스텝에 대해서 기능 연산자의 할당 및 바인딩이 끝나면 다음 제어 스텝들에서 존재하는 연산의 이동도들을 수정한다. 전체 제어 스텝에 대해서 할당 및 바인딩을 수행한 후, 연결 구조 병합을 행한다.

```
Input_mobility( ); /* 모든 연산의 최대 구간 */
while(control step) {
  Register_allocate( ); /* 레지스터 할당 바인
                           딩 */
  Function_allocate( ); /* 기능 연산자 할당
                           바인딩 */
  Mobility_modify( ); /* 이동도 수정 */
  Inter_allocate_merge( );
        /* 연결 구조 할당 및 바인딩 및 병
        합 */
}
```

그림 1. 저전력 하드웨어 할당 알고리듬

2.1 레지스터의 할당과 바인딩

저전력을 고려한 레지스터의 할당 및 바인딩은 단계별 제어 스텝마다 다음과 같이 수행한다. 첫 번째 레지스터가 할당될 형을 분류한다. 즉, 변수 또는 상수 이전의 제어 스텝에서 행해진 기능연산자의 출력인지 분류한다. 두 번째 분류한 형에 따라 레지스터에 할당한다. 변수인 경우 첫 번째 제어 스텝이면 새로운 레지스터에 할당하고, 그 다음 제어 스텝이면 이전 제어 스텝에서 사용한 레지스터를 그대로 할당한다. 즉, 자원을 재사용함으로써 캐패시턴스의 전력 소모가 줄어든다. 다시 말해 저전력을 고려한 경우이다. 또한 상수인 경우 레지스터 할당에서 제외된다. 이전 제어 스텝에서 행해진 기능 연산자의 출력인 경우, 먼저 다른 연산의 입력인지 조사한다. 다른 연산의 입력인 경우, 기능 연산자의 형과 입력을 받는 연산의 형을 고려하여 레지스터에 할당한다. 다른 연산의 입력인 경우가 아니면 기능 연산자의 형만을 고려하여 레지스터에 할당한다. 레지스터의 할당과 바인딩 알고리듬은 그림 2와 같다.

```
Register_allocate( )
{
  Input_type_search( ); /* 입력 값의 형 분류 */
  if(control step >1) then Reg_chart_search( );
  /* 이전 제어 스텝에서 할당된 레지스터 파악 */
  Variable_allocate( );
  In_fun_allocate( );
  /* 기능 연산자의 출력에 레지스터 할당 */
     Register_chart( );
}
```

그림 2. 저전력 레지스터 할당 알고리듬

<표 1>은 제어 스텝 1과 제어 스텝 2에 해당되는 자원을 재사용하는 즉, 저전력을 고려한 레지스터 할당이다.

〈표 1〉 Register_chart

CS	레 지 스 터				
1	R1	R2	R3		
2	R1	R2	R3	R4	R5

루프가 존재하는 경우 한 루프의 처음과 끝에 사용되는 레지스터를 동일한 것으로 그림 3과 같이 할당한다.

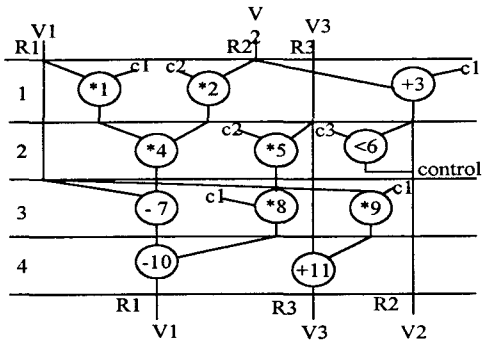


그림 3. 루프가 포함된 레지스터 할당 및 바인딩

2.2 기능 연산자의 할당 및 바인딩

레지스터의 할당과 바인딩을 수행한 후, 현재 제어 스텝에 존재하는 각 연산에 대해서 기능 연산자를 할당 및 바인딩을 수행한다. 즉 각 연산의 계산된 수행 시간을 만족하면서 최대한 작은 면적을 지닌 기능 연산자를 셀 라이브러리에서 선택하기 위해서 기능 연산자를 할당 및 바인딩할 연산의 다섯 변수 즉, 자체 분포수, 상대 분포수, 자체 고정수, 상대 고정수 및 자체 이동도를 조사한다. 먼저 자체 분포수 (self

distributed number)는 기능 연산자를 할당하고자 하는 연산과 같은 제어 스텝에 있고 같은 형을 지닌 연산의 개수를 나타내고, 상대 분포수 (relative distributed number)는 기능 연산자를 할당하고자 하는 연산과 다른 제어 스텝에 있고 같은 형을 지닌 연산의 최대 개수를 나타낸다. 또한 자체 고정수 (self fixed number)는 기능 연산자를 할당 하고자 하는 연산의 이동도와 같은 제어 스텝에 존재하고 같은 형을 지닌 연산의 이동도를 조사했을 때 이동도가 영인 개수이고, 상대 고정수 (relative fixed number)는 기능 연산자를 할당 하고자 하는 연산과 다른 제어 스텝에 있고 같은 형을 지닌 연산의 이동도를 조사했을 때 한 제어스텝 당 이동도가 영인 최대의 개수이다. 자체 이동도 (self mobility)는 기능연산자를 할당하고자 하는 연산의 이동도이다.

기능 연산자를 할당 바인딩하는 예로, 그림 3의 비교기(comparator) 에 기능 연산자를 할당하는 과정을 살펴본다. 비교기의 초기 입력은 두 번째 제어 스텝에 스케쥴링이 되었다. 그러나 비교기의 변수들을 조사한 결과 비교기에 두 번째 제어 스텝과 세 번째 제어 스텝 즉 두 개의 제어 스텝을 차지하는 기능 연산자를 할당하는 것이 가능하다. 다시 말하면, 하나의 연산이 복수개의 제어 스텝에 걸쳐 있는 멀티사이클링 (multi-cycling)이다. 그러므로 셀 라이브러리 상에서 지연시간이 교정시간과 같고 가능한 작은 면적을 지닌 기능 연산자를 선택하여 비교기에 할당한다. 교정시간이란 두 개의 제어스텝의 시간에서 연결구조 지연시간과 레지스터 지연시간을 제외한 시간을 말한다.

기능 연산자의 할당과 바인딩 알고리듬은 그림 4와 같다.

```
Function_allocate( )
{
  if (control step ==1) then 네 변수 조사( );
/* 자체 분포수, 자체 고정수, 상대 분포수, 상
   대 고정수 조사 */
  if(same operation && same mobility)
    Existed_function_allocate( );
    /* 기존에 존재한 기능 연산자를 연산에
       할당 */
    Function_unit_allocate( ); /*기능 연산자
                                 할당 */
    Function_register_chart( );
    /* 기능 연산자와 레지스터 정보를 지닌
       표 작성 */
}
```

그림 4. 기능 연산자의 할당 및 바인딩 알고리듬

전체적인 기능 연산자와 레지스터의 테이블 (Function_register_chart)은 <표 2>와 같다.

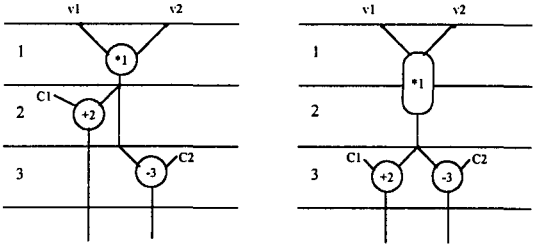
〈표 2〉 Function_register_chart

	FU1(*)			FU2 (*)			FU3 (+)			FU4 (<)			FU5 (-)		
	op	a	b	op	a	b	op	a	b	op	a	b	op	a	b
S1	1	R1	c1	2	c2	R2	3	R3	c1						
S2	4	R4	R5	5	c2	R3				6	R2	c2			
S3	8	c1	R5	9	R1	c1							7	R1	R4
S4							11	R3	R5				10	R1	R4

첫 번째 제어 스텝인 경우는 자체 분포수, 자체 고정수, 자체 이동도, 상대 분포수를 고려하여 기능 연산자를 할당한다. 그 다음 제어 스텝부터는 먼저 존재한 연산에 기능 연산자를 조사한다. 만약, 기능 연산자를 할당할 연산에 적합한 기능 연산자가 존재하면 그대로 할당하고, 존재하지 않으면 위의 다섯 변수를 조사하여 기능 연산자를 할당한다.

2.3 이동도 수정 단계

연산에 기능 연산자를 할당 시 복수개의 제어 스텝을 이용하는 멀티사이클링(multi-cycling)을 이용한 경우, 그 연산에 매달린 모든 연산에 대해서 이동도를 조정한다. 복수개의 제어스텝은 라이브러리 상에서 딜레이 교정시간과 같다. 예로, 그림 5 (a)에서 *1은 자체 분포수가 1, 자체 이동도가 2이고 자체 고정수, 상대 분포수, 상대 고정수가 모두 0이므로 복수개의 제어 스텝을 이용하여 기능 연산자를 할당 및 바인딩한다. 따라서 *1의 출력을 받는 +2는 이동도가 1에서 0으로 바뀐다. 그림 5의 (b)는 할당 및 바인딩된 결과이다.



(a) 이동도 수정 전 (b) 이동도 수정 후

그림 5. 이동도 수정의 예

2.4 연결 구조의 할당 및 바인딩

기능 연산자의 할당 및 바인딩을 수행한 후, 연결 구조의 할당 및 바인딩을 수행한다. 첫 번째, 제어 스텝인 경우 각 연산에 대해 새로운 멀티플렉서를 할당한다. 두 번째, 제어 스텝부터는 기능 연산자의 형과 입력 형을 조사하여 가능한 같은 형을 찾아 멀티플렉서를 할당한다. 세 번째, 멀티플렉서의 입력 수를 조사한다. 한 개인 경우 멀티플렉서를 생략하고, 한 개가 아닌 경우 각 멀티플렉서의 제어스텝을 조사한다. 제어스텝이 서로 중복되지 않거나, 중복이 되어도 입력 값이 동일하면 병합한다. 이때 병합한 멀티플렉서는 버스로 대칭한다.

그림 6은 저전력을 고려한 전체적인 레지스터, 기능 연산자, 연결구조의 할당 바인딩의 결과이다.

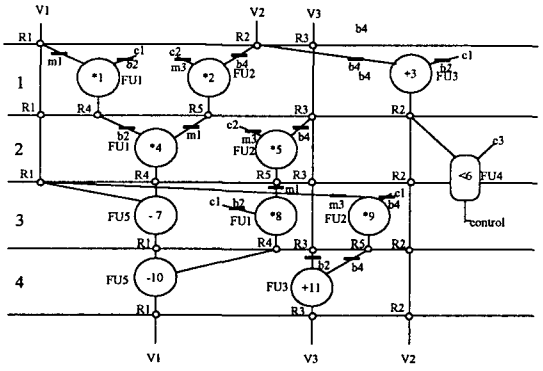


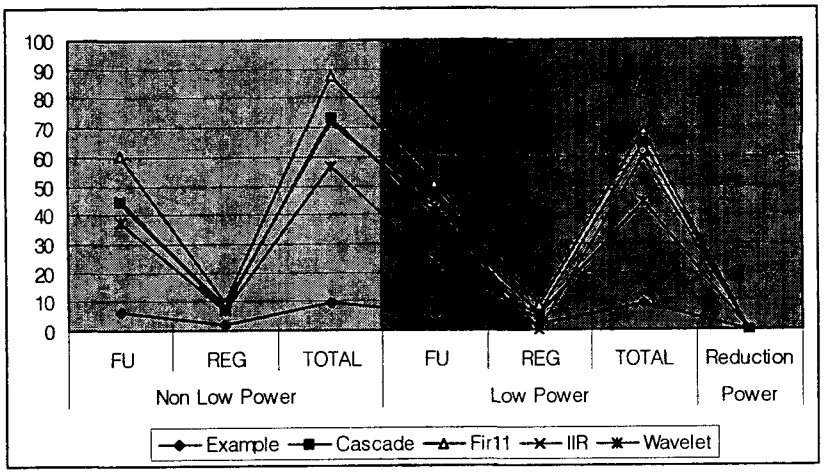
그림 6. 저전력을 고려한 전체적인 할당 바인딩의 결과

III. 실험 결과

본 논문에서 제한된 알고리즘을 구현하기 위해 AVIION SYS에서 C로 구현하였다. 각 기능 연산자 레지스터 바인딩의 자원은 면적을 최소화하기 위해 최소의 자원을 사용하여 저 전력을 위한 할당 알고리듬 과정을 나타내었다. 벤치마크 회로는 DSP 필터를 사용하였다.

〈표 3〉 구현된 저전력 회로의 전력 평가 비교

benchmark	Non Low Power			Low Power			Power Reduction
	FU	REG	TOTAL	FU	REG	TOTAL	
Example	6.3	2.3	9.8	5.9	2.0	9.1	8%
Cascade	44.2	7.5	73	40.8	5.7	64.9	11%
Fir11	60.6	9.6	87.5	50.6	8.5	68.8	17%
IIR	44.9	8.4	71.8	42.7	7.5	60.0	16%
Wavelet	37.3	6.9	56.5	23.0	5.3	44.6	21%



<표 3>은 합성된 benchmark회로와 저 전력을 구현하지 않은 회로와의 전력 비교를 하였다. 실험을 통해 제안된 알고리즘을 사용함으로써 평균 15%의 전력 감소를 얻었다. 특히, Wavelet Filter의 경우에는 저전력 스케줄링과 리소스 할당이 적용되어 가장 큰 감소율 21%의 효율성을 보였다.

Ⅳ. 결론

본 논문에서는 설계 자동화를 위한 새로운 저전력 하드웨어 할당 알고리듬을 제안한다.

본 알고리듬은 AVIION SYS에서 C로 구현하였으며 다음과 같이 수행한다. 첫 번째 제어 스텝부터 단계별로 레지스터, 기능연산자, 연결구조를 할당 바인딩을 수행한 후 연결 구조 병합을 행한다. 종속 관계에 있는 할당을 최대로 하여 자원을 수를 최소화하는 즉, 저전력을 고려한 할당 알고리듬을 제안하였다. 비교 실험결과에서 볼 수 있듯이 기존의 저전력을 고려하지 않은 방식보다 작은 전체 칩 면적비용을 얻을 수가 있었다. 또한 궁극적으로 비용 감소와 전력 감소를 가져 왔다

향후 연구 과제로는 설계 자동화를 위한 저

전력 하드웨어 할당 알고리듬 바탕으로 합성 결과에 대한 예측과 평가에 대한 연구와 다양한 벤치마크 실험이 선행되어야 하겠다.

참고문헌

[1] M. A. Breuer, Digital System Design Automation, Computer Science Press, Inc., 1975.

[2] S. M Rubin, Computer Aides for VLSI Design, Addison_Wesley.

[3] S. G. Shiva, "Automatic Hardware Synthesis", Proceedings of the IEEE, Vol.71, No1, pp.76-87, Jan.1983.

[4] Daniel D. Gajski, Silicon Compilation, Addison-Wesley, 1988.

[5] R. Camposano, "From Behavior to Structure: High-Level Synthesis", IEEE Design & Test of Computer, pp.8-19, Oct.1990.

[6] James R. Armstrong, F. Gall Gray." Structured Logic Design With VHDL", 1993

[7] Anantha P. Chandrakasan, Robert W. Brodeersen, "Low Power Digital CMOS Design", Kluwer Academic Publishers, pp219-256, 1995

A Low Power Hardware Allocation Algorithm for Design Automation

Ji-young, Choi*/Chi-ho, Lin**

This paper proposes a new heuristic algorithm of a low power hardware allocation for Design Automation.

The proposed algorithm works on scheduled input graph and allocates functional units, interconnections and registers by considering interdependency between operations and storage elements in each control step, in order to share registers and interconnections connected to functional units, as much as possible. The low power factor of the capacitance is reduced during the allocation. As the resource number reduce maximal .

This paper shows the effectiveness of the algorithm by comparing experiments of existing system of the non low power.

* The Major of control and computer, Chongju University
** Dept. of Computer Science, Semyung University