論文97-34C-5-1

데이타 상관 증가에 의한 저전력 상위 수준 합성

(Low Power High Level Synthesis By Increasing Data Correlation)

申 東 宛 *, 崔 起 榮 **

(Dongwan Shin and Kiyoung Choi)

요 약

최근 들어, VLSI의 성능과 집적도가 증가하고, PDA와 같은 휴대용 장비의 수요 증가에 따라 전력 소비 문제가 전자 시스템의 설계에서 중요한 관심사가 되었다. 저전력 설계는 모든 설계 수준에서 연구되어 왔는데, 특히 상위 수준에서 저전력을 고려 하는 것이 아주 효과적이다. 본 논문에서는 데이터 경로 위주의 회로에서 실행 유닛의 유효 정전 용량을 감소시키기 위하여, 실행 유닛의 입력 데이터 상관을 증가시키는 스케쥴링과 바인딩을 동시에 수행하는 방법을 제시한다. 본 논문에서 제시한 방법은 HYPER 합성 시스템의 스케쥴링과 자원 지정 과정에 구현되었다. 본 논문에서 제시한 방법은 다양한 실험 예제에 대하여, 원래의 HYPER 합성 시스템과 비교하여 실행 유닛의 경우 23.0%, 전체 회로의 경우 14.2%의 소비 전력 감소를 얻었다.

Abstract

With the increasing performance and density of VLSI circuits as well as the popularity of portable devices such as personal digital assistance, power consumption has emerged as an important issue in the design of electronic systems. Low power design techniques have been pursued at all design levels. However, it is more effective to attempt to reduce power dissipation at higher levels of abstraction which allow wider view. In this paper, we propose a simultaneous scheduling and binding scheme which increases the correlation between consecutive inputs to an operation so that the switched capacitance of execution units is reduced in datapath-dominated circuits. The proposed method is implemented and integrated into the scheduling and assignment part of HYPER synthesis environment. Compared with original HYPER synthesis system, average power saving of 23.0% in execution units and 14.2% in the whole circuits, are obtained for a set of benchmark examples.

I. 서 론

이전의 상위 수준 합성에 관련된 대부분의 연구는 VLSI의 면적과 성능을 최적화하는 방법에 대하여 집중적으로 연구 되어왔다[1] [2]. 최근 들어, 디지털 시스템의 성능과 복잡도의 증가에 따른 과도한 열 방출과, PDA와 개인 통신 장비와 같은 휴대용 장비의 수요 급증으로 말미암아 저전력 설계가 중요한 관심사로 떠올랐다. 저전력 설계 자동화에 대한 연구의 대부분은 논리 수준, 회로 수준, 레이아웃 수준의 하위 수준에 집중되어 왔다. 상대적으로 상위 수준 저전력 설계 자동화에 대한 연구는 거의 이루어지지 않았다.

이전에 제안된 방법으로는, 레지스터 전송 수준에서 데이터 경로의 병렬화나, 파이프라이닝과 같은 구조 변환(architectural transformation)을 이용하여 공급 전압을 감소시키거나[3] [4], 데이터 사이의 상관을 보

* 正會員, LG半導體 DT연구소
(LG Semicon. Co., DT LAB.)
** 正會員, 서울大學校 電氣工學部
(School of Electrical Engineering, Seoul National University)
接受日字:1997年2月24日, 수정완료일:1997年4月17日

존하는 모듈 선택[5], 자원 할당(allocation)과 자원 지정(assignment)[6][7]과 같은 방법들이 제시되었다. 그리고 [8] [9]에서는 피연산자를 공유하는 시블링 연산(sibling operation)을 같은 실행 유닛으로 할당하는 스케쥴링과 바인딩 방법을 제안하였다. 이들은 데이터 경로 위주의 회로에서 7-10%의 실행 유닛에서의 소비 전력 감소를 얻을 수 있었다.

본 논문에서는 데이터 경로 위주의 회로에서 실행 유닛의 입력 데이터 상관을 증가시켜, 유효 정전 용량을 감소시키는 스케쥴링과 바인딩을 동시에 실행하는 방법을 제시한다. 우리의 방법은 데이터 경로 위주의 회로에 매우 적당하다. 본 논문의 방법은 고전적인 리스트 스케쥴링 방법을 이용하면서, 우선 순위 함수로서 연산사이의 유효 정전 용량을 이용한다. 연산의 유효 정전 용량은 Landman이 제시한 Dual-Bit Type 모델을 이용하여 구한다.

본 논문의 2절에서는 저전력 상위 수준 합성에 대한 이론적 배경에 대하여 다룬다. 3절에서는 소비 전력을 고려한 스케쥴링과 바인딩 방법에 대하여 설명한다. 4절에서는 실험 결과를 보이고, 5절에서는 연구에 대한 결론을 맺는다.

Ⅱ. 상위 수준 합성에서 소비 전력 분석

디지털 시스템의 전력 소비의 원인은 동적 소비 전력, 단락 회로 소비 전력, 누설 소비 전력으로 나눌 수 있다. CMOS로 구현된 디지털 시스템에서 단락 회로 소비 전력과 누설 소비 전력은 전체 시스템의 소비 전력의 15%이하이고, 회로의 설계 형태에 많은 영향을 받기 때문에, 상위 수준 합성에서는 동적 소비 전력을 최소화한다.

동적 소비 전력은 디지털 시스템의 스위칭에 의해 발생하며, 식 (1)과 같다.

$$Power = C_{eff}(V_{sw} \cdot V_{DD})f \qquad (1)$$

여기서 f는 주파수이고, V_{sw}는 스위치된 전압(switched voltage)이고, V_{DD}는 공급 전압이고, C_{eff}는 유효 정전 용량(effective capacitance)이다. C_{eff}는 활동도 인자(activity factor) α와 물리적인 정전 용량 C_L의 곱으로 표현할 수 있다.

소비 전력은 전압의 제곱에 비례하기 때문에, 전압을 감소시키면 많은 전력을 감소시킬 수 있지만, 회로의 지연 시간이 증가하게 된다. 따라서 소비 전력을 감소시키는 첫 번째 방법은 시스템의 성능을 증가시킨 다음, 공급 전압을 낮추어 처리량(throughput)을 유지하는 것이다. 이것을 위해서 제안된 방법으로 병렬화, 파이프라이닝 등을 들 수 있다[3][4].

소비 전력을 감소시키는 두 번째 방법은 소비 전력이 주파수와 직접 비례하기 때문에 속도를 감소시키는 것이다. 하지만 고정된 타이밍 제한 조건을 가지는 실시간 응용 분야에는 주파수를 감소시킬 수 없다. 단 외부 데이터 율(external data rate)은 고정되지만, 하드웨어가 시간을 최대로 활용할 수 있도록 내부 클럭을 선택할 수 있다. 클럭 선택을 위한 설계 공간 탐색 방법은 [10] [11]에 잘 나타나 있다.

소비 전력을 감소시키는 세 번째 방법으로 유효 정전 용량을 감소시키는 것이다. 유효 정전 용량을 감소시키는 방법은 전역 계산 자원에 대한 접근을 줄이기 위하여 분산 계산(distributed computing)을 하거나 참조의 국부성(locality of reference)[12]을 이용하거나, 사용하지 않고 있는 모듈을 전력 관리를 통하여 powerdown을 하거나[13], 자원 공유를 최소화하여 데이터 상관을 보존하여 스위칭 활동도를 감소 시키거나[4], programmability를 필요로 하는 응용 분야에 범용 프로세서 유닛보다 작은 전력을 소비하는 특정한 모듈을 사용하는 방법[14] 등이 있다.

Ⅲ. 소비 전력을 고려한 스케쥴링과 바인딩

일반적으로 데이터 경로 위주의 회로에서는 데이터 경로의 소비 전력이 가장 많은 부분을 차지한다. 그 중에서도 실행 유닛이 소비 전력의 대부분을 차지한다. 실행 유닛의 소비 전력은 실행 유닛의 물리적인 정전 용량과 피연산자의 스위칭에 의존한다.

일반적으로 피연산자의 스위칭은 상관(correlation)에 의존한다. 따라서 같은 실행 유닛상에 두 개의 연속적인 피연산자 사이에 높은 상관 관계가 성립한다면 소비 전력은 감소한다. 연속적인 입력 피연산자의 상관의 정확한 계산은 시뮬레이션을 필요로 하며, 많은 시간이 소요된다.

CDFG(Control/Data Flow Graph) 상에 있는 여러 개의 연산은 자원 공유를 통하여 같은 실행 유닛으로 구현될 수 있다. 이런 경우, 두 개의 연속적 연산의

실행사이에 각 연산의 입력 피연산자들이 연속적으로 변화하기 때문에 실행 유닛의 스위칭 활동도가 변하게 된다. 이 때 스케쥴링과 바인딩 과정에서 실행 유닛의 입력 신호들 사이의 시간 상관(temporal correlation)을 증가시킴으로써 스위칭에 의한 소비 전력을 감소시킬 수 있다.

본 논문에서는 상관의 계산을 위하여 입력 시뮬레이션 벡터는 사용자가 알고리즘 검증 과정에서 준다고 가정하고, 상관에 의해서 유효 정전 용량을 계산하는 Dual-Bit Type 방법[15]을 이용한다.

1. 목적 아키텍쳐

본 논문은 그림 1과 같은 목적 아키텍쳐(target architecture)를 가정한다. 각 실행 유닛은 single-ported 레지스터 파일을 가지고 있다. 변수들은 연산이 끝난 후에 버퍼를 통하여 레지스터 파일에 기록된다. 연결선(interconnection)은 tri-state buffer를 가지지 않고, 레지스터에 기록을 할 때는 멀티플렉서를 이용한다. 각 실행 유닛은 여러 개의 레지스터를 구동할 수 있는 전용 출력 버퍼(dedicated output buffer)를 가지고 있다.

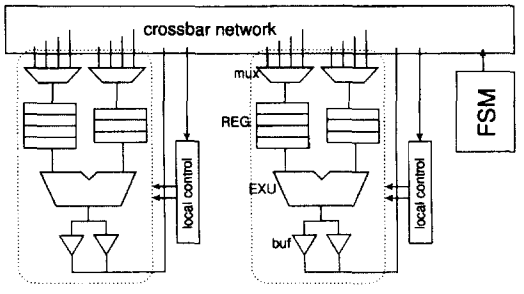

그림 1. 목적 아키텍쳐
Fig. 1. Target architecture.

2. 소비 전력을 고려한 스케쥴링과 바인딩

본 논문은 소비 전력을 고려한 스케쥴링과 바인딩을 동시에 행한다. 기본적인 스케쥴링 알고리즘은 리스트 스케쥴링 알고리즘을 이용한다. 제안하는 알고리즘은 실행 유닛의 개수는 자원 할당 과정에서 결정되었다고 가정한다. CDFG $G(V,E)$와 실행 유닛의 type의 개수 $\boldsymbol{a}$와 latency 제한 조건 lat가 주어지면, 제안하는 알고리즘은 그림 2와 같다.

이 알고리즘에서 모든 자원에 대하여 처음으로 스케쥴할 연산은 ALAP 시간이 가장 작은 것을 선택한다. 왜냐하면 처음에 스케쥴될 연산의 유효 정전 용량은 스케쥴링이 완전히 끝난 후에 가장 마지막에 스케쥴된 같은 instance의 연산과의 유효 정전 용량을 구해야 하기 때문에, 스케쥴링의 초기에는 이 값을 알 수 없다. 그래서 처음 스케쥴할 연산은 단순히 스케쥴의 latency를 감소시키기 위하여 ALAP 시간이 가장 작은 것을 선택한다. 그리고 나서 각 instance k에 대하여 가장 나중에 스케쥴된 연산들을 L_k에 기록한다.

LIST_LP ($G(V,E)$, $\boldsymbol{a}$, lat) {
  Compute the last possible start time t^L by As Late As Possible scheduling;
  Schedule operations that have the earliest t^L at control step 0;
  Put the most lately scheduled operations into L'_k;
  Determine candidate operations U'_k of which all predecessors have been scheduled;
  $upper = 1.0$;
  $lower = 0.0$;
  if (**SOLVE**(*upper*) is TRUE) **return** TRUE;
  if (**SOLVE**(*lower*) is FALSE) **return** FALSE;
  repeat
    if (**SOLVE**((*upper-lower*)/2) is TRUE) *lower* = (*upper-lower*)/2;
    else *upper* = (upper-lower)/2;
  } **until** (*upper-lower* < △));
  return TRUE;
}
SOLVE(α) {
  $U_k = U'_k$;
  $L_k = L'_k$;
  repeat {
    Select type k with the lowest power cost, $k = 1, 2, \ldots, n_{res}$, U_k is not empty;
    Compute the switched capacitance C_k between $l_{k,i}$ and $u_{k,j}$ $\forall l_{k,i} in L_k$, $\forall u_{k,j} in U_k$;
    Select $u_{k,j}$ operation that minimizes $\alpha \cdot c_{k,ij} C_{k,\text{avg}} + (1-\alpha) \cdot t_j^{h/L} / t_{k,abg}^L \forall c_{k,ij} \in C_k, \forall t_j^L \in t^L$
    Schedule $u_{k,j}$ at control step, {finish time of predecessors of $u_{k,j}$, finish time of $l_{k,i}$} + 1;
    Assign(Bind) $u_{k,j}$ to the same instance as $l_{k,i}$;
    Update the set of most lately scheduled operations L_k;
    Update the set of candidate operations U_k;
  } **until** ($\forall k$, U_k is empty);
  if (lat is met) **return** TRUE;
  else return FALSE;
}

그림 2. 저전력을 위한 스케쥴링과 바인딩 알고리즘
Fig. 2. a simultaneous scheduling and binding algorithm for low power.

이 알고리즘은 여러 가지 type 중에서 전력을 적게 소비하는 type 부터 스케쥴링과 바인딩을 한다. 이것은 전력을 적게 소비하는 type의 연산을 먼저 스케쥴함으로써, 전력을 많이 소비하는 type의 후보 연산의 갯수를 증가시키기 위한 것이다. 후보 연산의 개수가 많아지면, 더 많은 가능성이 존재하여, 더 많은 소비 전력 감소 효과를 얻을 수 있다. 예를들면 덧셈기가 곱셈기보다 적은 전력을 소비하므로 덧셈기를 먼저 스케쥴링한다.

C_k는 가장 나중에 스케쥴된 연산들의 집합 L_k와 후보 연산의 집합 U_k사이의 유효 정전 용량의 집합이다. 연산간의 유효 정전 용량을 계산하는 방법은 DBT 방법을 이용하고 있다. C_k는 유효 정전 용량 행렬(switched capacitance matrix)로 나타날 수 있으며, 행렬의 항의 개수는 모듈 instance의 개수와 후보 연산의 개수의 곱에 비례한다. 그리고 피연산자의 교환법칙(commutative law)이 성립하는 type의 연산(덧셈기, 곱셈기)은 피연산자를 서로 바꾸어 계산해 보아서 유효 정전 용량이 더 작은 것을 선택한다. 왜냐하면 피연산자의 순서에 따라 피연산자의 상관이 변하고, 소비 전력에도 영향을 미치기 때문이다.

우선 순위 리스트는 $\alpha \cdot c_{k,ij} / C_{k,\,avg} + (1-\alpha) \cdot t_j^L / t_{k,\,avg}^L$에 의하여 결정된다. 여기서 $C_{k,\,avg}$는 C_k의 평균이며, $t_{k,\,avg}^L$는 후보 연산들의 ALAP 시간의 평균이다. α와 $1-\alpha$는 각각 유효 정전 용량과 ALAP 시간의 가중치(weight)이다. α가 커지게 되면, 유효 정전 용량을 ALAP 시간보다 더 많이 고려하게 되고, 스케쥴시 유효 정전 용량이 작은 것을 먼저 선택하게 된다. 반면에 α가 작아지면 ALAP 시간이 작은 것을 우선적으로 선택하게 되어, slack을 우선 순위 함수(priority function)로 이용하는 리스트 스케쥴링과 같게 된다. 즉 α가 작아지면, 유효 정전 용량을 우선 순위 함수로 이용할 때 발생하는 스케쥴의 latency 증가를 막을 수 있다. 일단 후보 연산 중에서 스케쥴될 연산이 선택되면, 즉 $l_{k,i}$와 $u_{k,j}$사이의 유효 정전 용량이 가장 작다고 하면, 선택된 연산 $u_{k,j}$의 instance는 가장 최근에 스케쥴된 연산 $l_{k,i}$의 instance와 똑같게 되고, control step은 최근에 스케쥴된 연산 $l_{k,i}$와 predecessor 연산들의 control step을 보고 결정한다.

우선 순위 함수로 처음에는 유효 정전 용량을 이용하기 때문에, 주어진 latency 제한 조건을 만족하지 않을 수 있다. 그러한 경우에는 우선 순위 함수에서 유효 정전 용량의 가중치를 감소시켜야 하는데, 이것은 α를 감소시키는 것과 같다. 따라서 latency 제한 조건을 만족하지 못할 때에는, 이진 탐색을 수행하여 latency 제한 조건을 만족시킨다.

위의 저전력 스케쥴링과 바인딩 알고리즘을 7차 FIR 필터의 예를 들어 설명하겠다. 일반적으로 FIR 필터는 다음과 같은 식으로 표현할 수 있다.

$$\sum_{i=0}^{p-1} x_i c_i \tag{2}$$

여기서 p는 차수이고, x_i는 입력 신호이며, c_i는 계수이다. 만일 속도가 중요한 문제가 아니면, 면적을 감소시키기 위해서 곱셈 연산을 여러 번의 shift-add 연산을 반복하여 하드웨어의 복잡도를 줄일 수 있다. 하지만 속도가 중요할 때에는 곱셈 연산을 위하여 곱셈기를 이용하여야 한다. 우리는 후자의 경우에 관심을 두고 연산의 수행 순서에 따라 소비 전력이 어떻게 변하는 지를 알아보자. 한 예로서 $p=7$이고 계수 c_0, c_1, c_2, c_3의 값이 각각 -1870, 1867, -740, -1804이고 곱셈기의 비트 폭은 15라고 하자. 그림 3은 7차 FIR 필터의 Silage code와 CDFG를 나타내고 있다. 그림 3에서 실선은 데이터의 흐름(data flow)을, 점선은 제어 신호의 흐름(control flow)을 나타낸다. 그리고 D 는 지연 요소(delay element) 처리를 위한 전송(transfer) 모듈을, IO 는 입/출력을 위한 IO 모듈을 나타낸다.

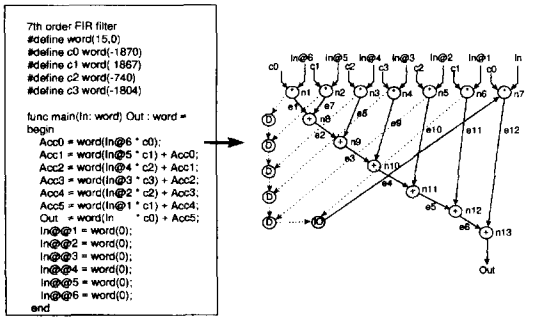

그림 3. 7차 FIR 필터의 Silage 코드와 CDFG
Fig. 3. Silage code and CDFG of the 7th order FIR filter.

표 1. 7차 FIR 필터의 스케줄링과 바인딩
Table 1. Scheduling and binding of 7th order FIR filter.

mult1	mult2	add1	last scheduled nodes	candidates
n1(cs 0)	n2(cs 0)		n1, n2	n3, n4, n5, n6, n8
		n8(cs 1)	n1, n2, n8	n3, n4, n5, n6
	n6(cs 1)		n1, n6, n8	n3, n4, n5
n5(cs 1)			n5, n6, n8	n3, n4
n3(cs 2)			n3, n6, n8	n4, n9
		n9(cs 3)	n3, n6, n9	n4
n4(cs 3)			n4, n6, n9	n10
...	...	...	...	...

Latency 제한 조건은 8이고, 자원 제한 조건은 곱셈기 2개, 덧셈기 1개, transfer unit 1개, ioUnit을 1개로 가정하자. 처음에 스케줄될 수 있는 후보 연산은 $n1$, $n2$, ..., $n6$의 6개의 곱셈 연산이다. 그 중에서 ALAP 시간이 가장 작은 것은 $n1$과 $n2$이다. 따라서 $n1$과 $n2$는 control step 0에 스케줄되고 mult1과 mult2의 서로 다른 instance로 할당된다(표 1참고). $n1$과 $n2$가 스케줄됨에 따라, 후보 연산에서 $n1$과 $n2$는 제거되고, 그들의 후보 연산인 $n8$의 덧셈기가 추가된다. 후보 연산 중에서 곱셈이 덧셈보다 소비 전력측면에서 비용(cost)이 낮으므로, 덧셈을 먼저 스케줄하게 된다. 따라서 $n8$을 control step 1에 스케줄한다. 다음 과정으로 가장 최근에 스케줄된 연산 $n1$, $n2$와 같은 type을 가진 후보 연산들 사이의 유효 정전용량 행렬(switched capacitance matrix)을 만든다. 직관적으로 CDFG에서 보면, $n2$와 $n6$의 피연산자중 한개가 $c1$으로 서로 같다. 즉 $n2$와 $n6$사이의 상관이 높아서 작은 값의 유효 정전 용량을 가질 것이다. 따라서 이들을 같은 instance mult2에 할당한다. $n6$가 스케줄 되었으므로 후보 연산에서 $n6$를 제거한다. 그리고 mult2의 가장 최근에 실행된 연산은 $n6$가 된다. 다음으로 $n1$과 $n5$는 모든 입력 변수가 서로 다르지만, 상수 피연산자의 상관이 다른 후보 연산 보다 크기 때문에, 후보 연산 중에서 가장 작은 유효 정전 용량을 가지게 되므로 같은 instance mult1에 할당된다. 마찬가지로 하면, $n5$와 $n3$은 피연산자 $c2$가 서로 같아서 같은 instance mult1에 할당된다. 이런 과정을 계산해 나가면, 표 1과 같은 결과를 얻을 수 있다. 괄호 안은 각 연산이 스케줄된 control step을 나타낸다.

지금 까지는 우선 순위 함수로 유효 정전 용량만을 이용했기 때문에 latency 제한 조건이 만족하지 않았다면 α를 감소시켜서 유효 정전 용량의 가중치를 감소시키고, ALAP 시간의 가중치를 증가 시켜야 한다. 그러면 control step 1에 스케줄될 연산은 $n5$가 아니라 $n3$가 된다. 그리고 스케줄링과 바인딩을 수행할 때 상관을 증가시키기 위하여 피연산자가 서로 바뀌는 경우가 생기게 되는데, 레지스터 바인딩(register binding)을 위하여 이 정보를 기록해야 한다. 위와 같은 과정을 반복하면, 그림 4와 같은 7차 FIR 필터의 스케줄링과 바인딩한 결과를 얻을 수 있다. 그림에서 회색으로 나타난 node는 mult2 instance를 의미한다.

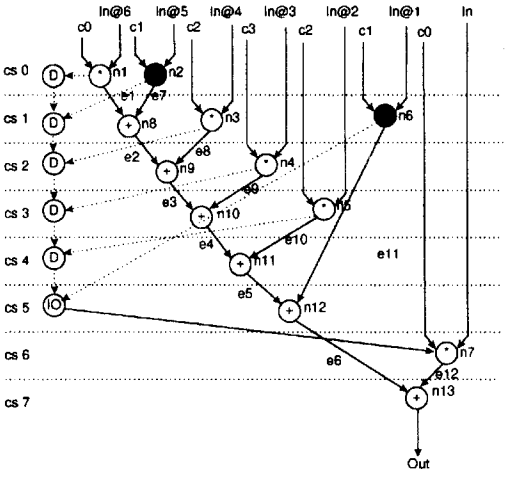

그림 4. FIR 필터의 스케줄된 CDFG
Fig. 4. Scheduled CDFG of the 7th order FIR filter.

Ⅳ. 실험 결과

본 논문에서 제안된 알고리즘은 HYPER 합성 환경(HYPER synthesis environment)[16]에서 모듈의 자원 지정 과정, 스케줄링 과정과 레지스터의 자원 지정 과정에 구현되어 삽입되었다. HYPER 합성 시스템의 target architecture는 제 3.1절에 설명하였다. HYPER합성 시스템은 Silage로 표현된 행위 수준 표현과 처리량 제한 조건(throughput constraint)이 주어지면, 면적을 최소화하는 구조를 합성한다. HYPER 합성 시스템의 자원 지정 과정은 무작위 초기 자원 지

정(random initial assignment)을 한 후 반복적으로 개선(iterative improvement)시킨다. 스케쥴링 과정은 자원 이용률(resource utilization)을 극대화하는 개선된 리스트 스케쥴링 알고리즘을 사용한다. 자원 지정 과정과 스케쥴링 과정을 여러 번 반복하여 자원 이용률을 최대화한다. 자원 할당 과정은 스케쥴링과 자원 지정의 결과에 기초하여 자원의 수를 조정한다. 레지스터의 자원 지정은 graph coloring을 이용한다.

표 2. 7차 FIR 필터의 HYPER와 제안된 방법의 비교(I)

Table 2. Comparison between HYPER and proposed method of 7th order FIR filter(I).

module	HYPER			proposed			energy reduct. (%)	energy reduct. (%)
	switched cap.(pF)	energy (nJ)	power (mW)	switched cap.(pF)	energy (nJ)	power (mW)		
EXU	389	9.7	12.2	290	7.3	9.1	25.4	25.4
MUX	15	0.4	0.5	20	0.5	0.6	-33.3	-33.3
BUS	137	3.4	4.3	118	2.9	3.6	13.9	13.9
CONT	46	1.2	1.4	46	1.2	1.4	0.0	0.0
CLK	24	0.6	0.8	25	0.6	0.8	-4.2	-4.2
BUFF	17	0.4	0.5	18	0.4	0.6	-5.9	-5.9
REG	80	2.0	2.5	77	1.9	2.4	3.8	3.8
total	709	17.7	22.2	594	14.9	18.6	16.2	16.2

표 3. 7차 FIR 필터의 HYPER와 제안된 방법의 비교(II).

Table 3. Comparison between HYPER and proposed method of 7th order FIR filter(II).

module	HYPER			proposed			energy reduct. (%)	energy reduct. (%)
	switched cap.(pF)	energy (nJ)	power (mW)	switched cap.(pF)	energy (nJ)	power (mW)		
EXU	386	9.6	10.7	291	7.3	9.1	24.6	15.0
MUX	21	0.5	0.6	18	0.4	0.6	14.3	0.0
BUS	152	3.8	4.2	135	3.4	4.2	10.5	0.0
CONT	46	1.2	1.3	46	1.2	1.4	0.0	-7.7
CLK	36	0.9	1.0	34	0.9	1.1	5.6	-10.0
BUFF	17	0.4	0.5	18	0.5	0.6	-5.9	-20.0
REG	84	2.1	2.3	81	2.0	2.5	3.6	-8.7
total	742	18.5	20.6	624	15.6	19.5	15.9	5.3

본 논문에서 레지스터 전송 수준의 소비 전력 예측을 위하여 HYPER 합성 시스템에 구현된 SPA(Stochastic Power/Area Analysis)를 이용하였다. SPA는 데이터 경로의 경우 DBT 모델을 제어 경로의 경우에는 ABC 모델을 이용하고 있다. 예측의 오차는 데이터 경로의 경우, 스위치 수준 시뮬레이터인 IRSIM[11,17]과 비교하여 평균 15%의 오차를 가지고 있으며, 제어 경로의 경우는 평균 9%의 오차와 10%의 표준 편차를 가지고 있다.

1. 7차 FIR 필터

제 3.2절에서 설명한 그림 1의 FIR 필터를 HYPER 합성 시스템을 이용하여 합성하였다. 스케쥴링과 바인딩의 결과는 그림 4와 같다. 표 2는 HYPER의 합성 결과와 본 논문에서 제시한 알고리즘을 이용하여 합성한 결과를 비교하고 있다. 여기서 latency 제한 조건은 8이고, 곱셈기는 2개이고, 덧셈기, transfer Unit, ioUnit은 각각 1개씩 사용하였다. 제안한 알고리즘으로 우선 순위 함수를 유효 정전 용량만 고려하였을 때는, latency는 11이었다. 따라서 latency 제한 조건을 만족하지 않기 때문에 α를 감소 시켜서 latency 제한 조건을 만족하도록 하였다. 전체 소비 전력에 대하여 실행 유닛(EXU)의 소비 전력의 비는 49% 정도이고 제어기(CONT)의 소비 전력 비는 6%이다. 표 2에서 제안된 방법으로 얻은 실행 모듈의 에너지 감소는 25.4%이고 전체의 에너지 감소는 16.2%이다. 위의 경우에서 latency가 같기 때문에 소비 전력 감소는 에너지 감소와 같다. 표 3은 덧셈기를 2개로 증가시켰을 때의 결과이다. HYPER 합성 시스템으로 합성하였을 때 latency는 9이고, 제안된 방법으로 하였을 때는 latency가 8이었다. 이처럼 HYPER 합성 시스템의 latency가 증가한 것은 리스트 스케쥴링의 우선 순위 함수가 자원 이용률(resource utilization)이기 때문이다[1]. 표 3에서 실행 유닛의 에너지 감소는 24.6%이고 전체의 에너지 감소는 15.9%이다. 그리고 실행 모듈의 소비 전력 감소는 15.0%이고, 전체의 소비 전력 감소는 5.3%이다. 이 예의 경우에는 latency가 감소하여 에너지의 감소율이 소비 전력의 감소율보다 실행 유닛에서는 9.6%, 전체는 10.6% 정도 크다.

2. 다른 합성 예제

표 4는 8개의 상위 수준 합성 예제의 특징을 설명하고 있다. 여기서 fir11은 11차 FIR 필터이고, cascade,

1) 자원 이용율면에서 HYPER 합성 시스템이 유리하기 때문에 제안된 방법의 경우 자원을 더 필요로 하는 경우가 생길 수 있으나, 우리가 실험한 예제에 대해서는 그러한 경우가 발생하지 않았다.

gm, wf는 Avenhaus 필터의 서로 다른 구현 예이다. 그리고 dct는 discrete cosine transformation이고, iir7은 7차 IIR 필터이고, lattice는 lattice 필터이고, 마지막으로 nc는 LMS(least mean squares) 알고리즘을 이용하는 잡음 제거기(noise canceller)이다. Resource allocation 열은 합성시의 자원 제한 조건을 나타낸다. 표 5는 표 4의 다양한 합성 예제에 대하여 HYPER 합성 시스템으로 합성한 결과와 본 논문에서 제안한 방법으로 합성한 결과의 latency, 실행 유닛과 전체 칩의 유효 정전 용량을 비교하고 있다. 이 표에서 유효 정전 용량에 공급 전압의 제곱을 곱하면 에너지와 같다. 제안된 방법은 HYPER 합성 시스템과 비교하여 실행 유닛의 유효 정전 용량 감소는 평균 23.0%이고 전체 칩의 유효 정전 용량 감소는 평균 14.2%였다.

표 4. 벤치마크 예제의 특징
Table 4. Characteristics of benchmark examples.

benchmark	Number of operations			critical	resource
	mult(*)	add(+)	sub(-)	path	allocation
fr11	11	10		11	+(2), *(2)
cascade	17	16		10	+(2), *(2)
gm	11	24	8	34	+(3), *(2), -(1)
wf	12	20	14	22	+(3), *(2), -(3)
dct	16	13	13	7	+(2), *(2),-(2)
iir7	20	10	4	11	+(2), *(2),-(1)
lattice	9	6	3	10	+(1), *(2),-(1)
nc	32	25	1	12	+(3), *(2),-(1)

표 5. latency와 유효 정전 용량의 비교
Table 5. Comparison of latency and switched capacitance.

bench-mark	HYPER			proposed			swit. cap. reduct.	
	lat-ency	EXU (pF)	total (pF)	lat-ency	EXU (pF)	total (pF)	EXU (%)	total (%)
fir11	12	2164	4774	12	1487	3750	31.3	21.4
cascade	12	507	1279	11	427	1147	15.7	10.3
gm	34	792	3084	34	635	2819	19.8	8.9
wf	23	299	2887	23	277	2594	7.4	10.1
dct	14	170	1069	14	85	852	50.0	20.3
iir7	15	2144	5418	15	1363	3851	35.5	28.9
lattice	10	933	2305	10	798	2136	14.5	7.3
nc	24	1680	4878	23	1521	4587	9.4	6.0

V. 결 론

본 논문은 데이터 경로 위주의 응용 분야에서 연산간의 데이터 상관을 증가시켜 실행 유닛의 유효 정전 용량을 줄이는 스케줄링과 바인딩 방법을 제안하였다. 제안된 알고리즘은 리스트 스케줄링 방법에 우선 순위 함수를 실행 유닛 사이의 유효 정전 용량으로하여, 소비 전력이 최소가 되도록 스케줄링과 바인딩을 한다. 만일 latency 제한 조건을 만족하지 않을 때에는 우선 순위 함수에서 ALAP 시간의 가중치를 증가시켜서 latency 제한 조건을 만족시킨다. 본 논문에서 제안한 알고리즘은 HYPER 합성 시스템 환경의 스케줄링과 자원 지정 과정에 구현되어 삽입되었으며, 여러 가지 합성 예제에 대하여 실험을 하였다. HYPER 합성 시스템의 합성 결과와 비교했을때, 실행 유닛의 소비 전력 감소는 평균 23.0%이고, 전체 회로의 평균 소비 전력 감소는 14.2%였다.

참 고 문 헌

[1] D. Gajski and N. Dutt, *High-level Synthesis: Introduction to Chip and System Design.* Kluwer Academic Publishers, 1992.

[2] G.D. Micheli, *Synthesis and Optimization of Digital Circuits.* New York: McGraw Hill, Inc., 1994.

[3] A.P. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low-power CMOS digitial design," *IEEE J. of Solid-State Circuits,* pp. 473-484, 1992.

[4] A.P. Chandrakasan, M. Potkonjak, R. Mehra, J. Rabaey, and R.W. Brodersen, "Optimizing power using transformation," *IEEE Tr. on CAD/ICAS,* pp. 12-31, Jan. 1995.

[5] L. Goodby, A. Orailoglue, and P.M. Chau, "Microarchitectural synthesis of performance-constrained, low-power VLSI designs," in *Proc. of Int'l Conf. on Computer Design,* pp. 323-326, Oct. 1994.

[6] A. Raghunathan and N.K. Jha, "Behavioral synthesis for power," in *Proc. of Int'l Conf. on Computer Design,* pp. 318-322, Oct.

1994.

[7] A. Raghunathan and N.K. Jha, "An ILP formulation for low power based on minimizing switched capacitance during datapath allocation," in *Proc. of Int'l Symp. on Circuits & Systems,* pp. 1069-1073, May 1995.

[8] E. Musoll and J. Cortadella, "Scheduling and resource binding for low power," in *Proc. of Int'l Symp. on System Synthesis,* pp. 104-109, Apr. 1995.

[9] Y. Fang and A. Albicki, "Joint scheduling and allocation for low power," in *Proc. of Int'l Symp. on Circuits & Systems,* pp. 556-559, May 1996.

[10] R. Mehra and J. Rabaey, "Behavioral level power estimation and exploration," in *Proc. of Int'l Symp. on Low Power Design,* pp. 197-202, Apr. 1994.

[11] A. Raghunathan and N.K. Jha, "An iterative improvement algorithm for low power data path synthesis," in *Proc. of Int'l Conf. on Computer-Aided Design,* pp. 597-602, Nov. 1995.

[12] R. Mehra, L.M. Guerra, and J. Rabaey, "Low power architectural synthesis and the impact of exploiting locality," *Journal of VLSI Signal Processing,* 1996.

[13] M.B. Srivastava, A.P. Chandrakasan, and R.W. Brodersen, "Predictive system shutdown and other architectural techniques for energy efficient programmable computation," *IEEE Tr. on VLSI Systems,* pp. 42-55, Mar. 1996.

[14] A. Abnous and J. M. Rabaey, "Ultra-low-power domain-specific multimedia processors," in *Proc. of IEEE VLSI Signal Processing Workshop,* Oct. 1996.

[15] P. Landman and J. Rabaey, "Architectural power analysis: the dual bit type method," *IEEE Tr. on VLSI Systems,* pp. 173-187, June 1995.

[16] J. Rabaey, C. Chu, P. Hoang, and M. Potkonjak, "Fast prototyping of datapath-intensive architectures," *IEEE Design and Test of Computers,* pp. 40-51, 1991.

[17] A.Salz and M. Horowitz, "IRSIM: an incremental MOS switch-level simulator," in *Proc. of Design Automation Conference,* pp. 173-178, June 1989.

저 자 소 개

申 東 宛(正會員)

1971년 11월 9일생. 1995년 한양대학교 전자공학과(공학사). 1997년 서울대학교 전자공학과(공학석사). 1997년 ~ 현재 LG 반도체 DT 연구소 재직중. 주관심 분야는 디지털 시스템의 설계 자동화 및 VLSI 설계

崔 起 榮(正會員) 第 34卷 C編 第 3號 參照

현재 서울대학교 전기공학부 부교수