論文97-34C-6-1

# 루프의 중첩을 이용한 저전력 상위 수준 합성

# (Power-Conscious High Level Synthesis Using Loop Folding)

金 大 弘 *, 崔 起 榮 *

(Daehong Kim and Kiyoung Choi)

요 약

상위 수준에서 저전력을 고려하는 경우, 보다 넓은 관점에서 다양한 변환 기법들이 시스템 설계에 적용될 수 있으며, 그리하여 적은 비용과 노력으로 하위 수준의 설계에 비해 훨씬 더 큰 전력 감소 효과를 얻을 수 있다. 본 논문에서는, 저전력 시스템의 상위 수준 합성을 위해 전력 측면에서 루프 중첩과 같은 변환 기법을 제안하며, 데이터 경로 위주의 회로에서 스위칭 활동도의 감소를 통 해 실행 유닛에서 소모되는 전력을 감소시키는데 초점을 맞춘다. 변환 알고리즘은 구현 및 실험을 위해 미 버클리대에서 만들어 진 HYPER라는 상위 수준 합성 프로그램에 통합되었다. 실험 결과 데이터 경로 위주의 회로에 대해 최대 50%까지의 전력 감소가 이루어짐을 보인다.

**Abstract**

By considering low power design at higher levels of abstraction rather than at lower levels of abstraction, we can apply various transformation techniques to a system design with wider view and obtain much more effective power reduction with less cost and effort. In this paper, a transformation technique, called power-conscious loop folding is proposed for high level synthesis of a low power system. Our work is focused on reducing the power consumed by functional units in a data path dominated circuit through the decrease of switching activity. The transformation algorithm has been implemented and integrated into HYPER, a high level synthesis system for experiments. In our experiments, we could achieve a power reduction of up to 50% for data path dominated circuits.

## I. 서 론

1980년대까지 시스템의 특성을 결정 짓는 가장 중요한 요소들 중의 하나는 속도였다. 그리하여, 최소한의 비용으로 속도를 증가시키기 위한 많은 노력이 있어 왔고, 이러한 노력은 구현되는 실리콘 면적을 제한 조건으로 속도를 최대화하기 위해 제안된 여러 기법들로 나타났다. 하지만, 최근 들어, 노트북, 휴대폰과 같은 휴대용 응용 제품과 시스템을 위한 시장이 급격하게 성장하고, 또한 높은 클럭 주파수로 동작하는 시스템에 있어 고전력 소모로 인한 신뢰도 문제와 그에 따른 팩키징 비용의 증가가 발생함에 따라, 저전력을 고려한 설계 기술은 VLSI 설계의 모든 측면에서 점점 더 중요해지고 있으며, 동시에 시스템 설계에 있어 주요 관심 사 중의 하나가 되고 있다.

저전력을 고려한 하위 추상화 수준에서의 설계, 즉 회로와 논리 수준의 설계를 위해 지금까지 많은 연구가 있어 왔고, 많은 효과적인 방법들이 제안되어 왔다.[1] [2] [3] 하지만, 보다 상위 추상화 수준에서 저전력 설계가 고려된다면, 훨씬 더 효과적인 전력 감소를 얻을 수 있다. 상위 수준에서는 시스템 설계의 초기 단계에서부터 더 넓은 안목을 가지고 많은 다양한 변환 기

* 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1月31日, 수정완료일:1997年6月2日

법을 적용할 수 있고, 그로 인해 보다 적은 비용과 노력으 로 훨씬 더 큰 전력 감소를 얻을 수 있다. 그런 이유로 이 논문에서는 저전력 시스템의 상위 수준 합성을 위한 변환 기법에 초점을 둔다.

저전력 상위 수준 합성을 위해 상당히 많은 연구가 있어 왔다. Chandrakasan은 다양한 변환 기법을 이용하여 데이터 경로(data path) 위주의 주문형 회로에서 전력을 감소시키기를 시도하였다. [4] [5] 그의 접근 방식은 회로에 가급적 많은 동시성(concurrency) 을 도입하여 회로의 속도를 도입 전에 비해 상대적으로 높인 후, 원래의 속도 제한 조건을 위반하지 않는 범위에서 전압을 최대로 낮춤으로써, 저전력을 유도하는 것이다. 궁극적으로 공급 전압을 낮추기 위해서, 그는 루프의 반복적 펼침(loop unrolling), 리타이밍(retiming) 그리고 파이프라이닝과 같은 변환 기법을 이용하였다. 루프의 반복적 펼침 기법은 동시성을 증가시키기 위해서 이용되었고, 리타이밍과 파이프라이닝은 임계 경로 (critical path)의 길이를 줄이기 위해 이용되었다. 병렬성(parallelism)의 증가로 인해 정전용량이 선형으로 증가함에도 불구하고, 전력 소모가 공급 전압의 제곱에 비례하는 관계로 낮아진 공급 전압이 정전용량의 증가를 보상하기 때문에, 전체 전력 소모는 감소하게 된다. 이용되는 변환 기법의 대부분은 기본적으로 전통적인 상위 수준 합성에서 이용되는 변환 기법과 동일하다. 하지만, 그러한 변환 기법을 통해 얻어진 결과를 평가하기 위해서 이용되는 비용 함수는 서로 다르다.

[6] 에서, 저전력을 고려한 데이터 경로 설계를 위해 스케줄링(scheduling)과 자원 지정 (resource binding) 알고리즘들이 제안되었다. 그들은 실행 유닛(덧셈기, 뺄셈기 등)과 레지스터의 입력으로 들어가는 신호에 대해 천이의 수를 최소화시키는 방법을 제안하였는데, 이는 유효 정전 용량(switched capacitance)을 효과적으로 감소시킨다. 이러한 결과는 후보 노드(candidate node)들을 가능한 한 가까운 컨트롤 스텝에 스케줄링하고, 그 노드들을 같은 자원에 지정함으로써 성취되어진다. 이때, 후보 노드들은 같은 실행 유닛의 연속적인 동작 사이에 입력으로 들어오는 연산자들의 값에 있어서 가능한 한 변화가 없도록 선택된다. 더하여, [7] 에서는 실행 유닛의 활동도(activity)를 감소시키기 위해 루프 상호 교환(loop interchange), 피연산자 재배열(operand reordering)과 같은 변환 기법들이 제안되고 있다.

기본적인 컨트롤 플로우 소자로서, 루프는 단위 시간당 처리량(throughput)과 자원 이용도(resource utilization)의 최적화를 위해 시도되는 다양한 변환 기법들의 주요한 대상들 중 하나였다. [8] 와 [9] 에서는 병렬 하드웨어의 이용도와 단위 시간당 처리량에 있어 괄목할 만한 향상을 얻기 위해 루프 중첩(loop folding)이라는 변환 기법이 제안되었다.

본 논문에서는, 전통적인 루프 중첩의 개념을 저전력 설계 관점으로 전환시켜 스위칭으로 인해 발생되는 전력 소모를 감소시키는데 초점을 맞춘다. 우리는 저전력을 고려한 루프 중첩 변환 기법을 이용하여 실행 유닛 상에서 입력 연산자들의 천이의 수를 최소화함으로써 전력 소모를 감소시킨다. 변환 기법의 알고리즘은 상위 수준 합성 프로그램인 HYPER [10] 속으로 통합되어 Silage 표현으로부터 합성된 저전력 하드웨어의 네트리스트를 얻게 된다. 알고리즘의 평가를 위해, 전력 분석 프로그램인 SPA [11] 에 필요한 약간의 수정을 가한 후, 그것을 이용하여 전력 감소를 측정한다.

본 논문은 다음과 같이 구성된다. 2장에는 피연산자 공유, 전력 측면에서의 루프 중첩 *(power-conscious loop folding)*에 대한 기본 개념, 제시된 루프 중첩 기법이 전력 소모에 미치는 영향이 설명된다. 3장에는 변환 알고리즘이 설명되어 있으며, 4장에는 실험 결과가 제시된다. 마지막으로 5장에는 결론과 앞으로의 연구 계획이 제시된다.

## II. 기본 개념

이 장에서는 피연산자 공유(operand sharing)가 전력 소모에 미치는 영향이 조사되며, 그리고 나서, 저전력을 고려한 루프 중첩이 피연산자 공유를 통하여 전력 소모를 어떻게 감소시키는지가 설명된다.

### 1. 피연산자 공유

DSP(Digital Signal Processor)와 같은 데이터 경로 위주의 주문형 회로에서 전체 전력 소모의 대부분을 차지하는 것은 당연히 데이터 경로에서 소모되는 전력이다. 이런 이유 때문에 입력 피연산자에 대한 변화를 최소화함으로써 실행 유닛의 스위칭 활동도(switching activity)를 감소시키려는 다양한 변환 기법들이 제안되었다. [4] [7] 여기에서 설명되는 피연산

자 공유도 그러한 기법들 중의 하나이다. 일반적으로 전력(switching power) 소모는 입력 피연산자들의 상관 관계(correlation)에 의존한다. 따라서 같은 실행 유닛상에서 두 개의 연속적인 입력 피연산자들 사이에 높은 상관 관계가 성립한다면, 전력 소모는 감소한다. 하지만, 연속적인 입력 피연산자들 사이에 있어 상관 관계의 정확한 계산은 시뮬레이션을 바탕으로 하기 때문에, 매우 많은 시간을 필요로 한다.

전형적으로 상위 수준 합성의 입력은 CDFG(Control Data Flow Graph)로 변환되며, 상위 수준 합성 시스템은 그러한 CDFG를 중간 형태의 포맷(intermediate form)으로 받아들인다. CDFG상에서 노드(node)는 후에 실행 유닛이나 레지스터로 지정되는 연산을 나타내며, 에지(edge)는 데이터 흐름이나 제어 흐름을 나타낸다. 합성 중에 여러 개의 연산 노드들이 하드웨어 공유를 통하여 하나의 실행 유닛으로 구현될 수 있다. 이런 경우, 두 개의 연속적인 연산 노드의 실행 사이에 각 연산 노드로의 입력 피연산자들이 연속으로 변하기 때문에 할당된 하나의 실행 유닛의 스위칭 활동도가 증가하는 상황이 대부분 발생할 수 있다. 피연산자 공유 기법은 적어도 하나의 공통된 입력 피연산자를 가지는 두 개 이상의 연산 노드들을 하나의 동일한 실행 유닛에 지정한다. 그리하여, 실행 유닛의 입력 신호들 사이의 시간적 상관 관계(temporal correlation)를 최대로 함으로써 스위칭으로 인한 전력 소모의 감소를 얻을 수 있다. 입력 피연산자가 두 개인 실행 유닛에서 단지 하나의 입력 피연산자만이 변할 때의 평균 전력 소모를 $P_1$, 두 개의 입력 피연산자가 동시에 변할 때의 평균 전력 소모를 $P_2$, 그리고 $\alpha$를 그 둘 사이의 비(= $P_1/P_2$)라고 가정하자. 하나의 실행 유닛이 하나의 공통된 입력 피연산자를 가지는 n개의 연산 노드들에 의해 공유된다면 그 때의 전력 감소는 다음 식에 의해 주어진다.

$$\begin{aligned} \text{전력 감소율} &= 1 - \frac{\text{피연산자 공유 후에 소모되는 전력}}{\text{피연산자 공유 전에 소모되는 전력}} \\ &= 1 - \frac{P_2 + (n-1)\cdot P_1}{nP_2} \\ &= \frac{(n-1)\cdot(1-\alpha)}{n} \end{aligned}$$

$\alpha$의 전형적인 값이 12비트 곱셈기에 대해 약 0.65이고 12비트 덧셈기에 대해 약 0.75 라고 한다면,[7] n = 4에 대해 각각 26%와 18%의 전력 감소가 얻어진다.

Musoll에 의해 제안된 스케줄링 알고리즘은 이러한 피연산자 공유 기법을 이용하여 전통적인 스케줄링 알고리즘에 비해 5%에서 8%까지의 전력 감소를 얻었다. 위의 기법의 한계는 DSP 응용 회로를 포함해서 대부분의 설계에 대해 공통된 입력 피연산자를 찾기가 그다지 쉽지 않다는 것이다. 본 논문은 전력 측면에서의 루프 중첩 *(power-conscious loop folding)*이라는 참신한 루프 변환 기법을 제시하여, 루프 내에 숨겨져 보이지 않는 공통된 입력 피연산자들을 찾는다. 이 변환 기법은 필터와 같은 DSP 응용 회로에 대해 매우 큰 전력 감소 효과를 가져다 준다.

## 2. 루프 중첩 기법

여기서 제시된 루프 중첩 변환 기법은 루프 반복(iteration)들을 중첩시키는 기본적 절차가 전통적인 기법과 같긴 하지만 둘 사이의 목적의 차이로 인해 전체적으로는 다소 다른 방식이라 할 수 있다.

### 1) 전통적인 루프 중첩

루프 중첩은 원래 연속적인 루프 반복들의 실행 시간들 사이에 부분적인 중첩(partial overlap)을 도입함으로써 루프 실행 시간의 단축을 얻거나 자원 이용도의 향상을 도모하는 변환 기법이다. 그림 1은 루프의 실행 시간을 단축시키는 루프 중첩 변환 기법의 한 예이다. 하나의 덧셈기와 하나의 곱셈기만이 할당된다고 가정하자. 이때, 타임 스텝 4에 있는 덧셈 연산 노드가 다음 루프 반복으로 옮겨진다고 하면 루프 본체의 총 실행 시간은 4에서 3으로 감소된다.

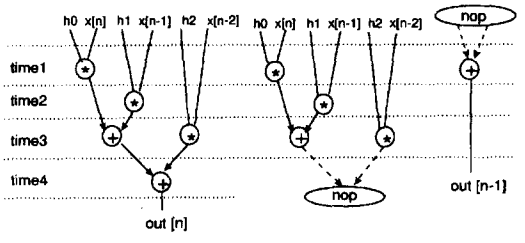


그림 1. 전통적인 루프 중첩 기법을 가하기 전과 후의 CDFG

Fig. 1. CDFG before and after the conventional loop folding.

일반적으로 루프가 n번의 반복을 수행하고, 루프 중첩 전의 루프 본체의 실행 시간을 $\lambda$, 루프 중첩 후의 루프 본체의 실행 시간을 $\delta$, 그리고 루프 중첩 후 루프의 전체 실행 시간을 Tfold 라고 한다면 루프의 시

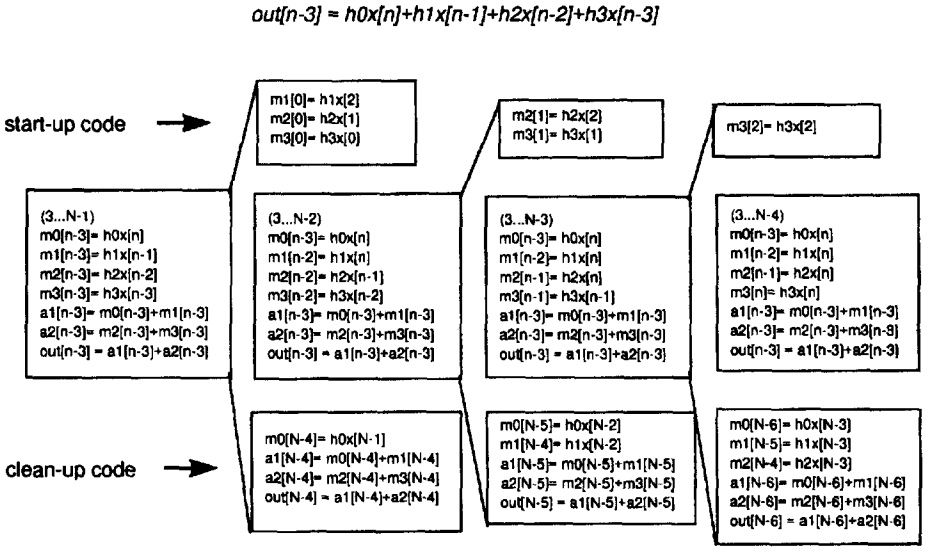


그림 2. 저전력을 고려한 단계적인 loop folding
Fig. 2. Steps for power-conscious loop folding.

작과 끝에서의 오버헤드를 고려한 전체 루프의 실행 시간 Tfold는 다음과 같다.[12] 주어진 식에서 (n-1)* $\delta$는 루프 중첩 후 루프 본체의 실행 시간이며, $[\lambda/\delta]*\delta$는 중첩의 오버헤드인 시작 코드와 마무리 코드의 실행시간을 일컫는 것이다.

$$T_{fold} = (n+\lceil \lambda / \delta \rceil -1)\cdot \delta$$

2) 전력 측면에서의 루프 중첩 기법

전통적인 루프 중첩 변환 기법이 루프의 실행 시간의 단축이나 자원의 높은 이용률에 목적을 둔 반면, 제안되는 루프 중첩 기법은 실행 유닛의 입력 피연산자에 있어서의 변화를 최소화함으로써 실행 유닛의 스위칭 활동도를 감소시키고 나아가 전력 소모를 줄이는 데 그 목적이 있다. 제안되는 기법은 필터와 같은 DSP 응용 회로에 대해 대단히 효과적이다. 그 이유는 대부분의 DSP 응용 회로들에 대해, 그것들의 행위 수준 표현에 다음과 같은 형태의 방정식이 빈번히 이용되기 때문이다.

$$y[n] = \sum_i h_i \cdot x[n-i]$$

위의 방정식에서 보이듯이, 출력 값은 상수와 지연 신호 값을 곱한 결과들의 합에 의해 결정된다. 방정식은 내재적으로 루프를 나타낸다. 하나의 곱셈기가 방정식에서 필요로 하는 곱셈 연산들에 의해 공유된다고 가정하자. 곱셈기의 스위칭 활동도는 각 곱셈 연산 쌍의 연속적인 실행 사이에 발생하는 두 개의 입력 피연산자 값의 변화에 의해 결정된다. 어떠한 변환 기법도 적용하지 않은 경우, 양쪽의 입력 피연산자들은 매 반복마다 그들의 값을 변화시킨다. 하지만, 공유된 하나의 곱셈기에서, $y[n]$에 대해서 $h_i \cdot x[n-i]$, 그리고 $y[n+1]$에 대해서 $h_{(i+1)} \cdot x[(n+1)-(i+1)]$이 연속적으로 계산되어 지도록 두 개의 연속적인 반복 사이에 중첩 기법이 적용된다면 곱셈기의 한쪽 입력 값이 변하지 않게 되므로 전력 소모가 감소할 수 있다. 이 경우에 물론 루프의 시작과 끝에 부가적인 코드들(시작 코드와 마무리 코드)이 필요하게 된다. 하지만, 많은 수의 루프 반복이 행해진다고 가정한다면 그 부가적인 코드들이 전체 에너지 소모나 실행 시간에 미치는 영향은 무시할만하다.

그림 2에 있는 예는 위에서 설명된 전력 측면에서의 루프 중첩 기법의 단계적인 절차를 설명해 준다. 이 예는 4차 FIR(Finite Impulse Response, 유한 임펄스 응답) 필터의 동작을 나타낸다. 그림에서 보이듯이, 루프 반복의 수는 중첩 스텝이 진행됨에 따라 하나씩 감소하며, 그에 상응하여 시작 코드(start-up code)와 마무리 코드(clean-up code) 가 수정된다.

지연 신호를 나타내는 항의 수가 n(이번 예에서는 4)이라고 한다면 중첩 스텝의 최대 수는 n-1(이번 예에

서는 3)이 된다. 매 스텝마다, 곱셈의 결과들이 저장될 레지스터들이 생성되며, 부가적인 시작 코드와 마무리 코드들이 삽입된다. 필요로 하는 레지스터의 총 수는 새로 생성된 레지스터들의 라이프 타임(life time)이 상대적으로 길기 때문에 증가된다. 그것들이 다수의 변수들에 의해 공유되는 상황은 거의 발생할 수 없다. 이번 예에서는 약 6개의 레지스터 오버헤드가 필요하다. 하지만, 전형적인 데이터 경로에서, 곱셈기와 같은 상대적으로 큰 실행 유닛이 전체의 면적과 전력 소모를 좌우 하기 때문에, 면적과 전력 소모 측면에서 레지스터 오버헤드가 미치는 효과는 작다. 보다 자세한 실험 결과는 4장의 표2에서 제공하는 면적 측정치에 있다. 본 논문에서는 단지 곱셈기에서의 전력 감소에 초점을 둔다.

단지 하나의 입력 피연산자만이 변할 때 곱셈기에서 소모되는 평균 전력을 $P_{mul1}$, 두 개의 입력 피연산자가 동시에 변할 때의 평균 전력 소모를 $P_{mul2}$, 그리고 $\alpha$를 그 둘 사 이의 비(= $P_{mul1}/P_{mul2}$)라고 가정하자. 위의 예에서, 단지 하나의 곱셈기가 4개의 곱셈 연산에 의해 공유된다면, 그때 루프 중첩 전의 전력 소모는 4 $P_{mul2}$이고, 중첩 후의 전력 소모는 ($P_{mul2}$ + 3 $P_{mul1}$)이 되어, 그 결과 2.1 장에서 설명되었듯이, $\frac{3}{4}(1-\alpha)$의 전력 감소를 얻게 된다. 이 경우 $\alpha$의 전형적인 값(12 비트 곱셈기에 대해 0.65)을 대입해 보면 약 26%의 전력 감소가 얻어짐을 알 수 있다. 그런데 곱셈기에 의해 소모되는 전력은 전체 전력 소모의 상당 부분을 차지하기 때문에, 전체적으로도 상당한 전력 감소 효과를 얻을 수 있다.

일반적으로 방정식상에 지연 신호를 나타내는 n개의 항이 있고, 그리하여 (n-1)번의 중첩 스텝이 시행된다면, 이때의 전력 감소 RP와 레지스터 오버헤드 OR는 다음과 같다.

$$R_P = \frac{(n-1)\cdot(1-\alpha)}{n} * 100\ (\%)$$

$$O_R = \frac{n\cdot(n-1)}{2}$$

## Ⅲ. 저전력을 고려한 루프 중첩 알고리즘

알고리즘을 설명하기 전에, 두개의 용어, 즉 기본 지수(base index)와 기본 피연산자 (base operand)를 정의하자. 루프에 있어서의 기본 지수란 루프 본체에서 입력 피연산자들을 위해 사용되는 지수들 중 가장 큰 것을 일컬으며, 기본 피연산자는 기본 지수에 상응하는 입력 피연산자를 가리킨다.

전력 측면에서의 루프 중첩 변환 기법은 다음의 세 가지 단계로 구성되어 있다.

i) 기본 피연산자를 제외한 모든 입력 피연산자들에 대해 지수를 1만큼 증가시킨다(결과적으로 입력 피연산자의 지연 시간은 1만큼 감소된다.).

ii) 스텝 1에서 1만큼 지수가 증가된 피연산자를 입력으로 받아들이는 각 곱셈 연산에 대해 그 결과가 저장되는 배열의 지수를 1만큼 증가시킨다. 이 과정은 이어지는 루프 반복들에서 이용되어질 새로운 변수(레지스터)를 발생시킨다.

iii) 시작 코드와 마무리 코드를 발생시킨다.

위의 단계들은 모든 입력 피연산자들의 지수들이 기본 지수와 같아질 때까지 반복된다.

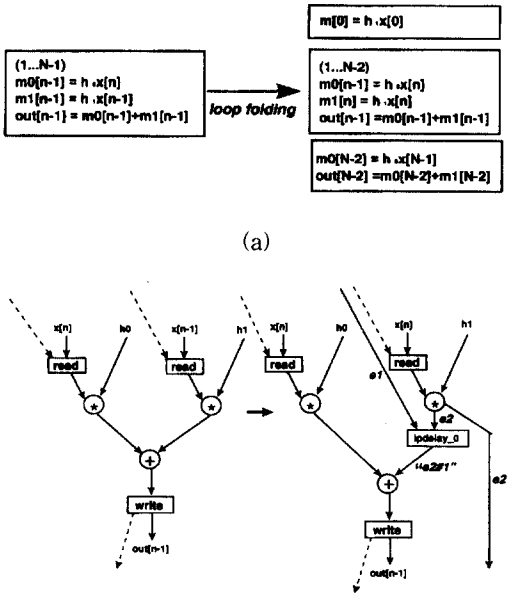


그림 3. 저전력을 고려한 loop folding 전후의 CDFG

Fig. 3. CDFG before and after power-conscious loop folding.

그림 3은 보다 단순한 예를 보이고 있다.

그림 3(a)는 루프 중첩 전후의 코드의 변화를 보이고 있으며, 그림 3(b)는 그에 상응하는 2개의 CDFG를 보이고 있다. 왼쪽에 있는 것은 저전력을 고려한 루프

중첩을 가하기 전의 CDFG이며, 오른쪽의 것은 중첩 후의 CDFG이다. 노드들은 기능 연산(functional operation), 읽기 연산, 쓰기 연산, 그리고 지연을 나타내며, 에지들은 노드들 사이의 종속성(dependency)을 나타낸다. 이때, 점선은 제어 종속성(control dependency)을, 실선은 데이터 종속성(data dependency)을 각각 나타낸다. CDFG는 계층적이고 재귀적인 특성을 가지고 있기 때문에, 루프와 같은 제어 구문은 가장 상위 CDFG상에서는 하나의 노드로서 표현된다. 따라서 루프 본체를 나타내는 그림의 두 개의 CDFG들도 루프 그 자체를 포함하는 상위 CDFG에서는 각각 하나의 노드로서 표현되나, 여기에서는 설명을 위해 루프 노드 자체에 대한 CDFG를 그대로 보이고 있다.

그림 3(a)에서 알 수 있듯이, 예가 간단하기 때문에 위의 3가지 단계를 한번만 수행하고 나면 루프 중첩 변환이 완성된다. 3가지 단계의 수행 후에는, 그림 3(b)의 우측에 있는 CDFG에서 보이듯이, 노드 lpdelay_0와 에지 e1, e2, 그리고 "e2#1"이 삽입된다. 에지 e1은 시작 코드를 위한 CDFG로부터 루프 본체를 나타내는 CDFG로의 데이터 종속성을 가리키며, 에지 e2는 지연 노드 lpdelay_0로의 데이터 종속성을 나타내는 동시에, 마무리 코드를 위한 CDFG로의 데이터 종속성을 나타낸다. 노드 lpdelay_0는 다음 반복들에서 이용될 곱셈 결과들을 저장하는 일종의 큐(queue)를 나타낸다. 에지 "e2#1"은 곱셈 연산 노드의 출력과 한 반복 후의 덧셈 연산 노드의 입력 사이의 데이터 종속성을 나타낸다. 에지 "e#n"은 n번의 반복을 넘어 존재하는 데이터 종속성을 나타낸다.

위에서 설명한 3가지 단계를 CDFG상에서 적용시킬 경우, 각각 다음의 단계에 상응한다.

i) 기본 피연산자를 제외한 모든 입력 피연산자에 대해 x[n-k]를 x[n-k+1]로 대체시킨다.

ii) 스텝 1에서 수정된 입력 피연산자를 가진 곱셈 연산 노드와 덧셈 연산 노드 사이에 지연 노드가 존재하지 않는다면 지연 노드와 에지를 삽입한다. 지연 노드의 출력 에지의 이름을 "e#1"라 명명한다. 만약 이미 "e#n"이라고 명명된 출력 에지를 가진 지연 노드가 존재한다면 단지 에지의 이름만 "e#(n+1)"로 대체시킨다.

iii) 2개의 부가적인 CDFG를 발생시킨다. 하나는 시작 코드를 위한 것이고 다른 하나는 마무리 코드를 위한 것이다. 또한 루프 본체를 나타내는 CDFG를 포함하여 3개의 CDFG 사이의 종속성을 나타내기 위한 에지를 삽입한다. 물론 보다 상위 계층의 CDFG 상에서는 루프 본체가 하나의 노드로 나타내어지는 것처럼 시작 코드와 마무리 코드를 나타내는 CDFG도 하나의 노드로서 나타내어진다.

여기서 묘사되는 CDFG는 계층적 그래프이다. 이러한 계층성은 앞에서도 언급했듯이, 루프나 조건 분기(conditional branch) 등과 같은 제어 구문을 나타내기에 적절하다.

```
PCLP() {
  Search graph nodes for loop;
  For each loop {
    Search for input nodes to multiplication;
    Find base index and base operands;
    Repeat {
      Call Folding() for one step of folding;
    } until (all the input nodes have the same
      index as the base operands)
  }
}

Folding() {
  For all the non-base input nodes {
    Increase index by one;
    If (any delay node between multiplication
      node and adder node) {
      Increase the value in the name of the
        output edge of the delay node by one;
    } else {
      Create the delay node and edges;
    }
    Create nodes and edges for start-up and
    clean-up codes;
  }
}
```

그림 4. 변환 알고리즘을 위한 의사 코드
Fig. 4. Pseudo-code for transformation algorithm.

그림 4는 변환 알고리즘에 대한 의사 코드를 보이고 있다.

함수 PCLP()는 최상위 수준의 CDFG로부터 루프 노드를 찾은 후, 각 루프를 나타내는 CDFG상에서 곱셈 연산 노드로의 입력을 나타내는 노드들을 찾는다. 그들로부터 기본 지수와 기본 피연산자를 결정하고 난 다음 Folding() 함수를 부름으로써 중첩 단계들을 반복하기 시작한다. 함수 Folding()은 입력 피연산자들의 지수를 증가시키고, 지연 노드와 에지들을 삽입하며, 나아가 시작 코드와 마무리 코드들을 위한 그래프 노드들과 에지들을 창출해 나가면서 주어진 CDFG를 수정한다.

## Ⅳ. 실험 결과

변환 알고리즘은 버클리 대학에서 개발된 상위 수준 합성 프로그램인 HYPER속으로 통합되었다. HYPER는 입력으로서 Silage 표현을 받아들인 후 그것을 플로우그래프 데이터베이스 포맷(.afl 파일)으로 바꾼다. 우리는 .afl 파일에 제안된 변환 알고리즘을 적용한 후 HYPER에서 제공하는 스케줄링과 하드웨어 지정(hardware mapping) 루틴을 이용하여 SDL 파일들을 얻었다. 이 장에서는 회로상의 실행 유닛에 의해 소모되는 전력과 전체 회로에 의해 소모되는 총 전력에 대한 측정치를 보인다. 또한, 루프 중첩 변환 기법을

적용하지 않고 합성된 회로와 적용하여 합성된 회로에 대한 측정 결과를 서로 비교한다. 전력의 측정을 위해 전력 분석 프로그램인 SPA가 이용되었다. 보통 RT 수준의 여러 전력 측정기가 50-100% 이상의 오차를 가지는 데 반해, 입력 데이터의 상관 관계를 상위 비트와 하위 비트로 나눈, DBT(Dual Bit Type) 모델을 이용하여 라이브 러리를 특성화한 SPA는 스위치 수준 시뮬레이터인 IRSIM에 비해 10-15% 정도의 오차 만을 가지고 있다. 본 실험에서는 SPA의 입력 netlist에 약간의 변화를 가하여 한번의 반복에 대해 루프의 전력 소모를 측정하였다. 이 실험에서는 모든 회로 합성에 있어서 단지 하나의 곱셈기만을 할당시켰다.

표 1. 전력 소모의 비교
Table 1. Comparison of power consumption.

| 회로 | Folding 여부 | 유효 정전 용량 (pF) | | 전력 (mW) | | $전력_{실행유닛}$ | 전력 감소량 (%) | |
|---|---|---|---|---|---|---|---|---|
| | | 실행 유닛 | 전체 회로 | 실행 유닛 | 전체 회로 | $전력_{전체회로}$ | 실행 유닛 | 전체 회로 |
| FIR | 부 | 2119 | 3147 | 44.1 | 65.6 | 67.3 | 58.7 | 51.2 |
| | 가 | 800 | 1407 | 18.2 | 32.0 | 56.9 | | |
| wavelet | 부 | 683 | 1632 | 11.4 | 27.2 | 41.8 | 20.2 | 11.4 |
| | 가 | 544 | 1446 | 9.1 | 24.1 | 37.6 | | |
| noise canceller | 부 | 1675 | 3767 | 10.7 | 24.1 | 44.5 | - 7.4 | - 14.9 |
| | 가 | 1523 | 3651 | 11.5 | 27.7 | 41.7 | | |
| volterra | 부 | 19 | 327 | 0.3 | 4.5 | 5.7 | 0 | - 30 |
| | 가 | 16 | 312 | 0.3 | 5.2 | 5.2 | | |

표 1은 11차 FIR 필터, wavelet 필터, 소음 제거 회로(noise canceller), volterra 필터 등 4개의 회로에 대해 측정된 결과의 비교를 보이고 있다. 3번째 열은 실행 유닛과 전체 회로에 대해 각각 측정된 유효 정전 용량을 비교하고 있다. 마찬가지로 4번째 열은 루프의 한 반복에 대해 실행 유닛과 전체 회로에서 소모된 전력을 나타낸다. 5번째 열에는 실행 유닛에서 소모된 전력과 전체 회로에서 소모된 전력 사이의 비가 제시되어 있다. 이 비는 전체 회로의 전력 소모 중에서 실행 유닛에 의해 소모되는 전력이 차지하는 비율을 나타내며, 실행 유닛이 전력 측면에서 회로의 나머지 부분을 얼마나 압도 하는지를 판단하는 근거로서 사용될 수 있다. 표의 마지막 열은 루프 중첩 변환 기법을 적용한 후 실행 유닛과 전체 회로에서 각각 얻을 수 있는 전력 감소량을 나타낸다. 마지막 열의 두 측정치 사이의 비교를 통해 실행 유닛에서의 전력 감소 효과가 루프 중첩 기법에 의해 새로이 추가된 멀티플렉서와 레지스터에서의 전력 소모에 의해 다소 감소됨을 알 수 있다. 또한, 실행 유닛에서의 전력 감소는 제어 유닛에 의해 소모되는 전력에 의해서도 다소 그 효과가 감소된다. 이러한 사실들이 전체 회로의 전력 감소량이 실행 유닛에서의 전력 감소량보다 더 작은 이유이다.

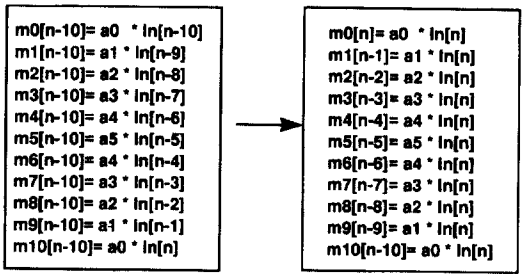


그림 5. 상수들의 대칭성
Fig. 5. Symmetry of constants.

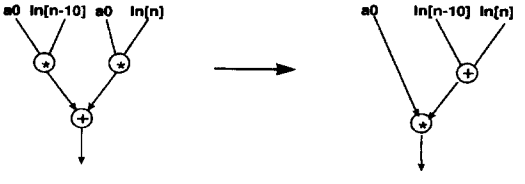


그림 6. 분배 법칙을 통한 곱셈 연산의 감소 (a) 2개의 승산 연산 (b) 1개의 승산 연산
Fig. 6. Reductions of the number of multiplication operations using distributivity. (a) 2 multiplications (b) 1 multiplication

11차 FIR 필터에 대해서는 예상했던 것보다 훨씬 더 큰 전력 감소량을 얻을 수 있었다. 그러한 큰 전력 감소는 곱셈 연산 노드의 두 입력 피연산자 중의 하나인 상수들의 대칭성과 관련이 있다. 필터에 대한 원래의 표현에는 루프 본체의 한 반복에 대해 11개의 곱셈 연산이 존재하는데 반해, 그림 5에서 보이듯이, 변환된 표현에는 단지 6개의 유효한 곱셈 연산만이 존재한다. 이런 이유로 인해, 루프 중첩 변환이 가해진 후, 곱셈기는 상수의 대칭성이 없는 일반적인 경우에 비해 약 절반의 전력만을 소모한다. 이와 같이, 상수의 대칭 구조는 전력 측면에서의 루프 중첩(*power- conscious*

*loop folding)* 변환 기법의 효과를 증폭시킨다. 물론, 본래의 표현이 마지막 출력 결과 값을 얻기 위해 먼저 (In [n-10] +In [n]), (In [n-9] +In [n-1]) 등과 같은 방식으로 덧셈을 하고 그 결과 값과 a0, a1, ....., a5와의 곱셈을 행한다면, 루프 중첩을 행하지 않더라도 대칭성의 효과는 그대로 유지된다. 실제로 상수들의 대칭성이 존재하는 경우, 이러한 분배 법칙을 이용한 대수 변환 기법(algebraic transformation technique)도 덧셈 연산 수와 임계 경로의 길이를 그대로 유지하면서 동시에 곱셈 연산의 수를 줄이기 위해 이용될 수 있다. 그림 6은 상수들의 대칭 구조하에서 CDFG에 적용된 분배 법칙을 보이고 있다. 어쨌든 FIR 필터에 있어서의 큰 전력 감소는 상수들의 대칭성과 피연산자 공유가 공동으로 작용하여 발생된 결과라고 할 수 있다.

LMS(Least Mean Square) 알고리즘을 이용한 소음 제거 회로와 volterra 필터에 대해서는 실행 유닛과 전체 회로에서 소모되는 전력이 표 1의 세 번째 열에서 보이는 유효 정전 용량의 감소에도 불구하고 오히려 증가한다는 것이 관찰된다. 그것은 표 2에서 보 이듯이, 루프 중첩에 의하여 루프의 지연 시간(latency)이 감소하기 때문이다. 지연 시간의 감소는 루프 중첩 변환 기법의 또 하나의 장점이다. 공정한 비교를 위하여, 실행 유닛과 전체 회로에서의 에너지 감소량이 측정되었다. 표 2는 둘 사이의 비교를 보이고 있다. 이 표에서 보면 전력의 증가에도 불구하고, 소음 제거 회로와 volterra 필터에서 지연 시간의 감소로 인하여 에너지는 감소한다는 것이 관찰된다. 하지만, 위의 두 회로에 대한 에너지의 감소량은 여전히 다른 회로에 비해서 아주 작다. 그에 대한 이유는 각각 다음과 같다. 소음 제거 회로에서는, 상수와 지연 신호간의 곱셈의 수가 CDFG상의 전체 곱셈 수에 비해 작다는 사실을 들 수 있고, volterra 필터에서는, 실행 유닛에 의해 소모되는 전력이 회로 전체의 전력을 좌우하지 못한다는 사실(표 1의 다섯 번째 열에서 알 수 있듯이, 실행 유닛에서의 전력 소모와 전체 회로에서의 전력 소모의 비가 루프 중첩 전후에 겨우 5%를 넘고 있다.)을 들 수 있다. 특히, volterra 필터에서는 위의 이유로 인해 실행 유닛에서는 20%나 에너지 감소가 일어났으나, 전체 회로에서는 겨우 4.9%의 에너지 감소가 일어났음을 알 수 있다. 따라서 전력 측면에서의 루프 중첩 변환 기법은 이런 종류의 회로에 대해서는 그다지 효과적이지 않다고 할 수 있다. 하지만, 표 2에서 관찰되듯이, 전체 전력이 실행 유닛의 전력에 의해 좌우되는 회로에 대해서는 50%까지 전력 감소를 얻는 것이 가능하다.

표 2. 면적, latency, 전력 그리고 에너지의 비교

Table 2. Comparison of area, latency, power, and energy.

| 회로 | Folding 여부 | 면적 ($mm^2$) | latency (클럭 사이클 수) | 전력 (mW) | | 에너지 (nJ) | | 에너지 감소량 (%) | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | 실행 유닛 | 전체 회로 | 실행 유닛 | 전체 회로 | 실행 유닛 | 전체 회로 |
| FIR | 부 | 8.98 | 12 | 44.1 | 65.6 | 53.0 | 78.7 | 62.2 | 55.2 |
| | 가 | 10.71 | 11 | 18.2 | 32.0 | 20.0 | 35.2 | | |
| wavelet | 부 | 3.45 | 15 | 11. | 27.2 | 17.1 | 40.8 | 20.5 | 11.3 |
| | 가 | 5.25 | 15 | 9.1 | 24.1 | 13.6 | 36.2 | | |
| noise canceller | 부 | 3.53 | 39 | 10.7 | 24.1 | 41.9 | 94. | 9.1 | 3.1 |
| | 가 | 4.18 | 33 | 11.5 | 27.7 | 38.1 | 91.3 | | |
| volterra | 부 | 1.00 | 18 | 0.3 | 4.5 | 0.5 | 8.2 | 20 | 4.9 |
| | 가 | 1.06 | 15 | 0.3 | 5.2 | 0.4 | 7.8 | | |

표 2의 세 번째 열에서 보여지는, 중첩 후 면적의 측정치는 한 번의 반복을 수행하는 데 필요한 회로의 면적에 레지스터 오버헤드를 더한 값이다. 따라서, 면적에 있어서의 증가는 주로 레지스터에 기인한 오버헤드를 나타낸다. 오버헤드의 양은 중첩 스텝의 수에 비례한다. 여러 번의 루프 반복을 행한다 하더라도, 증가된 레지스터 오버헤드 중 이미 사용된 것은 루프 반복시 재사용 가능하기 때문에, 시작 코드에 대한 레지스터 오버 헤드이외에 더 이상의 오버헤드는 없다.

## V. 결론 및 향후 과제

본 논문에서 우리는 저전력 시스템의 상위 수준 합성을 위해 전력 측면에서의 루프 중첩 변환 기법에 대해 토의하였다. 이 변환 기법으로 인해 우리는 필터와 같은 DSP 응용 회로에 대해 매우 큰 전력 감소량을 얻을 수 있었다. 그러한 감소는 실행 유닛상의 입력 피연산자의 변화를 최소화함으로써 실행 유닛의 스위칭 활동도를 감소시키는데 기반을 두고 있다. 전력 측면에서의 루프 중첩 변환 기법을 통해, 비록 부가적인 코드

오버헤드(시작 코드와 마무리 코드)와 레지스터 오버헤드가 발생된다 하더라도, 전력 소모가 실행 유닛의 전력 소모에 의해 좌우되는 그러한 회로에 대해서는 큰 전력 감소를 얻을 수 있었다.

변환 기법은 몇몇 DSP 응용 회로에 대해 테스트되었다. 결과는 FIR 필터와 같은 회로에 대해 50%까지의 전력 감소를 얻는 것이 가능하다는 것을 보인다. 하지만, 이 기법은 전체 전력 소모가 제어 유닛의 전력 소모에 의해 좌우되는 회로에 대해서는 그다지 효과적이지 못하다.

본 논문에서는 1개의 곱셈기를 할당한 경우를 다루었다. 곱셈기를 2개 또는 그 이상 할당한 경우에는 단순히 루프 중첩이라는 변환 기법에 한정된 문제가 아니라, 스케줄링과 자원 지정(resource binding) 단계에서 곱셈기에서 가급적 적은 전력을 소모하도록 그 입력단에서의 상관 관계를 고려해야 하므로, 그 자체도 하나도 연구 대상이라고 할 수 있다. 또한 본 논문에서는, 전력 측면에서 루프 중첩을 행할 때, 곱셈기의 입력으로 들어오는 상수들 사이의 상관 관계의 영향을 고려하지 않았다. 루프 중첩으로 인해 한쪽 입력 피연산자가 변화를 일으키지 않을때, 다른 쪽 상수 입력의 상관 관계가 고려되어 스케줄링이 이루어진다면, 그 두 개의 기법이 상승 효과(synergic effect)를 일으켜 더욱 큰 전력 감소가 이루어질 수 있다. 현재 2개 이상의 곱셈기를 할당하는 경우에 대한 연구와 상수 피연 산자들 사이의 상관 관계를 고려한 스케줄링 방법에 대한 연구가 진행되고 있다. 또한 시스템 수준의 전력 감소를 얻기 위해 DSP 프로세서 상에서 수행되는 DSP 응용 소프트웨어의 생성에 제안된 기법을 적용하려는 계획을 가지고 있다.

## 참 고 문 헌


[1] V. Tiwari, P. Ashar, and A. Malik, "Technology mapping for low power," in Proc. Design Automation Conference., pp. 74-79, 1993.

[2] J. Monteiro, S. Devades, and A. Ghosh, "Retiming sequential circuits for low power," in Proc. International Confernece on Computer-Aided Design, pp. 398-402, 1993.

[3] Tsui, M. Pedram, and A. Despain, "Technology decomposition and mapping targeting low power dissipation," in Proc. Design Automation Conference, pp. 68-73, 1993.

[4] A. P. Chandrakasan, M. Potkonjak, R. Mehra, J. M. Rabaey, and R. W. Brodersen, "Optimizing power using transformations," IEEE Trans. on Computer-Aided Design, vol. 14, pp. 12-30, Jan. 1995.

[5] A.P. Chandrakasan, M. Potkonjak, J.M. Rabaey, and R. W. Brodersen, "HYPER-LP: A system for power minimization using architectural transformations," in Proc. International Conference on Computer-Aided Design, pp. 300-303, Nov. 1992.

[6] E. Musoll and J. Cortadella, "Scheduling and resource binding for low power," in Proc. International Symposium on System Synthesis, pp. 104-109, 1995.

[7] E. Musoll and J. Cortadella, "High-level synthesis technique for reducing the activity of functional units," in Proc. International Symposium on Low Power Design, pp. 99-104, 1995.

[8] C. T. Hwang, Y. C. Hsu, and Y. L. Lin, "Scheduling for functional pipelining and loop winding," in Proc. Design Automation Conference, pp. 764-769, 1991.

[9] G. Goossens, J. Vandewalle, and H.D. Man, "Loop optimization in register-transfer scheduling for DSP-systems," in Proc. Design Automation Confernece, pp. 826-831, 1989.

[10] C. Chu, et al., "HYPER:An interactive synthesis environment for high performance real time applications," in Proc. International Conference on Computer Design, Nov. 1989.

[11] P. E. Landman and J. M. Rabaey, "Architectural power anlysis: the dual bit type method," IEEE Trans. on VLSI Systems, vol. 3, pp. 173-187, June. 1995.

[12] Giovanni De Micheli, Synthesis and Optimization of Digital Circuits, pp. 224-225, McGraw-Hill, 1994.

225, McGraw-Hill, 1994.

[13] A. P. Chandrakasan and R. W. Brodersen, "Minimizing power consumption in digital CMOS circuits," in Proc. of the IEEE, vol. 83, pp. 498-523, Apr. 1995.

[14] A. Raghunathan and N. K. Jha, "An iterative improvement algorithm for low power data path synthesis," in Proc. International Conference on Computer-Aided Design, pp. 597-602, 1995.

[15] P. N. Hilfinger, Silage reference manual, 1993.

## 저 자 소 개

**金 大 弘(正會員)**

1971년 11월 15일생. 1995년 서울대학교 전자공학과 졸업(공학사). 1997년 서울대학교 대학원 전자공학과 졸업(공학석사). 1997년 ~ 현재 서울대학교 전기공학부 박사과정 재학중. 주 관심분야는 Low Power Design, High Level Synthesis, VLSI 설계 등임

**崔 起 榮(正會員)** 第 34卷 C編 第 3號 參照

현재 서울대학교 전기공학부 부교수