論文99-36C-4-2

스위칭 동작 최소화를 통한 저전력 데이터 경로 최적화

(A Low Power-Driven Data Path Optimization based on Minimizing Switching Activity)

林 世 鎭 *, 趙 浚 東 **

(Se Jin Lim and Jun-Dong Cho)

요 약

본 논문은 데이터 의존적인 CMOS 회로(예: DSP) 의 전력량을 감축하기 위한 상위 수준 합성 기법에 대한 연구이다. 상위수준 합성은 스케줄링, 자원 및 레지스터 할당의 세 가지로 나누어서 수행한다. 스케줄링 시의 저전력 설계의 목적은 자원 할당 시 입력을 재 사용할 수 있는 가능성을 증가시키는 것이다. 스케줄링 후에 자원 및 레지스터 할당 문제는 가중치가 부가된 양립 그래프로 표현하여 최소비용흐름 알고리즘을 수행함으로써 스위칭 동작횟수가 적은 해를 얻는다. 제안된 알고리즘은 저전력 레지스터 및 자원 할당 문제에 대하여 $O(n^3)$ (n 은 그래프의 노드수) 시간에 최적해를 제공한다. 벤치마크 회로에 대한 실험 결과는 15% 의 전력 감축 효과를 나타낸다

Abstract

This paper presents a high level synthesis method targeting low power consumption for data-dominated CMOS circuits (e.g., DSP). The high level synthesis is divided into three basic tasks: scheduling, resource and register allocation. For lower power scheduling, we increase the possibility of reusing an input operand of functional units. For a scheduled data flow graph, a compatibility graph for register and resource allocation is formed, and then a special weighted network is then constructed from the compatibility graph and the minimum cost flow algorithm is performed on the network to obtain the minimum power consumption data path assignment. The formulated problem is then solved optimally in polynomial time. This method reduces both the switching activity and the capacitance in synthesized data path. Experimental results show 15% power reduction in benchmark circuits.

I. 서 론

최근 들어 설계방법은 상위 수준의 설계부터 하위 수준의 설계로 진행되는 top-down 설계구조로 변하고 있다. 상위 수준의 데이터 경로 합성은 회로의 동작적 기술로부터 RTL(register transfer level)로의 합성을 의미한다[1] [2] [16].

회로의 합성은 주어진 조건 즉, 수행시간이나 면적 또는 전력 소비의 제한 요소를 만족시키면서 목적함수를 최소화하는 것이다. 예를 들면 수행시간은 조건이 되며 면적의 최소화는 목적함수가 된다. 상위 수준 합

* 正會員, 三星電子(株) 情報通信總括
(Samsung Electronics Co., LTD. Telecommunication R&D Center)

** 正會員, 成均館大學敎 電氣電子컴퓨터工學部
(School of Electrical and Computer Engineering, Sungkyunkwan University)

※ 본 논문은 교육부 반도체 분야 소규모과제 번호 ISRC 96-E-2020에 의하여 수행되었습니다.
接受日字:1998年2月9日, 수정완료일:1999年3月20日

성의 첫 번째 단계는 회로의 동작기술을 그래프(graph)적 표현으로 나타내며 데이터 흐름과 제어 흐름으로 구성된다. 다음 단계는 이러한 동작의 그래프 표현을 구조적으로 변환하는 것이다. 이는 스케줄링(scheduling)단계와 할당(allocation) 과정 단계로 구성 될 수 있는데 스케줄링은 연산자를 제어 구간(control step)에 할당하는 것이다. 할당 과정은 연산자들과 변수 등을 FU (functional unit), 레지스터, 멀티플렉서나 버스와 같은 연결 요소 등에 할당하는 것이다. 스케줄링의 목적은 사용 가능한 제한된 하드웨어를 이용하여 수행 시간을 최소화하는 것이다.

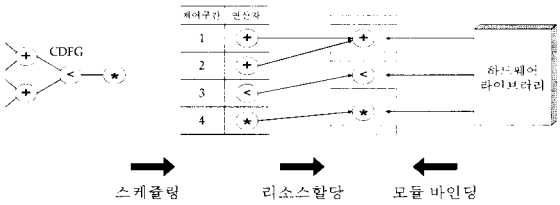


그림 1. 상위 수준 합성 과정
Fig. 1. High Level Synthesis Flow.

하드웨어 할당은 사용되는 FU의 개수를 줄이는 것이다. 다른 제어 구간에서 수행되는 연산자들은 같은 FU를 사용할 수 있으며 하드웨어 할당은 그러한 연산자들을 그룹화 하여 그룹의 수를 줄이는 것을 목적으로 한다. 레지스터 할당에서는 한 제어 구간에서 생성되고 다른 제어 구간에서 사용되는 변수들은 기억 장소에 저장이 되어야 한다. 생존 구간이 겹치지 않는 변수들은 같은 레지스터에 할당될 수 있다. 레지스터 할당 단계에서는 레지스터의 수를 최소화하는 것을 고려 할뿐만 아니라 연결구조도 단순화 시켜야 한다. 연결 구조는 FU와 레지스터를 연결하기 위해 버스나 멀티플렉서 등을 사용한다. 레지스터 할당 과정이 연결 구조에 영향을 미치는 것과 같이 상위 수준 합성과정의 여러 작업들은 서로 의존적이며 따라서 한 단계의 결과가 다른 단계의 결과에 영향을 미친다. 대부분의 상위 수준 합성과정은 NP (None-Deterministic Polynomial)-hard 문제로 알려져 있다. 이는 최적의 결과를 찾기 위해 수행되는 시간은 데이터 크기의 증가에 따라 지수적으로 증가하거나 최적의 결과를 찾지 못하는 경우가 발생할 수가 있다는 것이다. 그러므로 빠른 시간 안에 최적의 결과와 근사적인 결과를 얻는 알고리즘의 개발이 필수적이다.

저전력 회로구현은 여러 설계 수준의 범위를 포함해야 하나 특히 회로 설계 초기 단계의 결정은 다음 단계에 큰 영향을 미치므로 상위 수준에서의 최적화는 매우 중요하다. 이전에 제안된 방법으로는 데이터 경로의 병렬화 또는 파이프라이닝과 같은 구조 변화(architectural transformation)를 이용하여 공급전압을 감소시키거나 [3] [4], 피연산자를 공유하는 시블링 연산(sibling operation)을 같은 FU(functional unit)에 할당하는 스케줄링과 바인딩 방법을 제시하였다. [5] [6] [7] 에서는 maxcost flow를 이용한 레지스터 할당문제를 제안하였고 [8] 에서는 유전자 알고리즘을 사용하여 면적, 평균전력, peak전력, 속도의 최적화를 시도하였다. 또한 [9] 에서는 회로의 규칙성을 고려하여 연산을 FU에 할당함으로써 연결 구조상에서 발생하는 전력을 최소화하였다. 본 논문에서는 저전력 설계 시 성능을 그대로 유지하면서 전력량을 감축 할 수 있는 효과적인 방법의 제안을 목표로 하였다. 즉, RTL (register transfer level) 수준에서 스케줄링 성능을 유지하면서 입력 변수를 리소스 및 레지스터에 재 할당하여 스위칭 동작 횟수를 최소화하는 알고리즘의 개발을 목표로 하였다. 또한, 하드웨어를 바인딩하면서 연결구조를 단순화하여 전력소모를 최소화하는 방법을 제시한다.

본 논문의 결과는 독창적인 것이며 [7] 의 논문과 유사하나 구별되는 점은 상위수준 데이터 경로 합성을 위한 저전력 리소스 및 레지스터 할당에서 Multiplexor를 고려하는 등 레지스터와 리소스의 할당을 동시에 최적화 하는 문제를 최소비용 흐름(mincost flow)알고리즘으로 정형화([7] 은 max-cost flow를 이용하였으며 본 논문에서 제안된 방식이 더욱 간결함)하여 해를 구하였다. 본 논문의 구성은 다음과 같다. 2장에서는 전력소모의 원인 및 모델을 살펴본다. 3장에서는 제안된 알고리즘인 최소 비용의 흐름 알고리즘 [10] 과 이를 적용한 저전력 구현 방법에 대해 알아본다. 4장에서는 실험 및 결과를 보이고 5장에서는 결론을 맺는다.

Ⅱ. 전력소모 원인 및 기존의 저전력 설계 방법

CMOS 회로에서 전력 소모의 원인은 다음과 같이 나타낼 수 있다.

$$P = P_{switching} + P_{short-circuit} + P_{leakage}$$
$$= \frac{1}{2} \cdot \frac{C \cdot V_{dd} \cdot N}{T} + I_{sc} \cdot V_{dd} + I_{leakage} \cdot V_{dd} \quad (식\ 2.1)$$

여기서 P는 소모되는 전체 전력을 나타내며 V_{dd}는 공급전압, T는 클락 주기이다. $P_{switching}$은 CMOS 회로의 입력의 스위칭에 따라 공급전압으로부터 충전되거나 접지로 방전될 때 발생되는 스위칭 또는 dynamic 전력소모를 나타낸다. C는 load 캐패시턴스를 나타내며 N은 각 클락 주기마다 게이트 출력에서의 천이수 즉, 스위칭 동작 수를 나타낸다. $P_{short-circuit}$은 NMOS와 PMOS가 동시에 동작할 경우 공급전원에서 접지로 직접 흐르는 short-circuits전류에 의해 소모되는 것을 나타낸다. $P_{leakage}$는 벌크(bulk)지역에서 역 방향 다이오드의 누설 전류에 의하여 발생하는 것인데 주로 공정 기술에 의해 결정이 된다. 이밖에도 subthreshold 트랜지스터 전류도 전력을 소모한다. 즉, CMOS 회로에서 소모되는 전력은 스위칭 전력, short-circuit 전력, 누설 전력이다. 이중에서도 VLSI 회로에서 스위칭 전력이 전체 전력 소모량의 대부분(90%)을 차지한다 [3].

스위칭 전력을 최소화하는 저전력 방법 [18] [19] [20] 은 $P_{switching}$의 식에서 알 수 있다. 즉, 공급 전압 V_{dd}을 줄이거나 캐패시턴스를 줄이거나 스위칭 동작수($\frac{N}{T}$)를 줄이는 것이다. 소모되는 전력은 공급전압의 자승에 비례하므로 공급전압을 줄이는 것은 매우 효과적이다. 하지만, 공급 전압이 감소함에 따라 회로의 지연시간이 늘어나게 된다. 따라서 병렬화, 파이프라이닝 등을 사용하여 회로의 성능을 증가시킨 다음 공급 전압을 낮추어 처리량(throughput)을 유지하는 것이다. 이를 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 활용 할 수 있도록 내부 클럭을 선택할 수 있다. 클락 선택을 위한 설계 공간 탐색 방법은 [11] [12] 에 잘 나타나 있다. 소비 전력을 감소시키는 세 번째 방법으로 유효 정전 용량을 감소시키는 것이다. 유효 정전 용량을 감소시키는 방법은 분산계산(distributed computing)을 하거나 참조의 국부성(locality of reference) [13] 을 이용하거나, 사용하지 않고 있는 모듈을 전력 관리를 통하여 power-down을 하거나 [14], 자원 공유를 최소화하여 데이터 상관을 보존하여 스위칭 활동도를 감소시키거나 [4], programmability를 필요로 하는 응용 분야에 범용 프로세서 유닛보다 작은 전력을 소비하는 특정한 모듈을 사용하는 방법 [15] 등이 있다.

전력 소모 식에서의 스위치 캐패시턴스(switched capacitance)는 부하 캐패시턴스와 구동기의 스위칭 동작의 곱으로 나타난다. 레지스터의 전력 소모는 입출력의 스위치 캐패시턴스에 비례한다. 만약 레지스터가 3개의 데이터 i, j, k를 공유할 수 있다고 하자(그림 2.1 참조). 입력 멀티플렉서(Mux)는 데이터를 레지스터에 저장하기 위해 여러 데이터 중 회로에 알맞은 데이터를 선택하고 디멀티플렉서(Demux)는 레지스터에 저장된 데이터를 회로 수행에 적당한 장소로 데이터를 보낸다. Power(Register) ∝ switching(x) × ($C_{out,Mux}$+$C_{in,Register}$)+switching(y) × ($C_{out,Register}$+$C_{in,DeMux}$)이다. Glitch에 의한 switching이 없다고 가정하면 switching(x)=switching(y)이므로 Power (Register)=switching(y) × C_{total}이다. C_{total}은 모듈 라이브러리(module library)에 의해서 주어진 고정된 값이다.

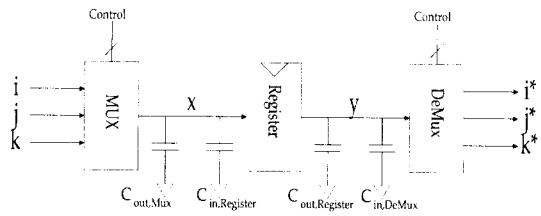


그림 2.1. 멀티플렉서를 이용한 레지스터 공유 모델
Fig. 2.1. Register Model with Multiplexor.

그러므로 어느 경우에나 레지스터의 출력 단에서 스위칭 동작을 줄이는 것은 레지스터의 전력소모를 최소화하는 것이다. 레지스터 할당 관점에서 공유 가능한 데이터들이 같은 레지스터에 할당이 될 때 레지스터에서의 스위칭은 저장되어 있는 한 데이터 변수가 다른 데이터로 대치될 때 일어난다. 예를 들어 X, Y, Z, W가 공유 가능한 데이터이고 레지스터를 공유한다고 가

정하자. 이들의 레지스터 안의 데이터 천이는 X→Y→Z→W의 체인 형태로 된다. 입력 변수 값이 알려져 있다면 정확한 스위칭 동작은 constant + H(X, Y) + H(Y, Z) + H(Z, W)이다. 여기서 H(i , j)는 두 변수 i, j 간의 해밍 거리(Hamming distance)이고 constant=H(레지스터의 초기 저장값, X)이다. 해밍거리는 각 변수간의 상이한 비트의 수를 의미한다.

Ⅲ. 제안된 저전력 데이터 경로 합성 알고리즘

1 스위칭 동작율 계산

전체 회로 중 데이터 경로에서의 전력 소모는 큰 부분을 차지한다. 이중에서도 FU의 전력소모가 큰 비중을 차지한다. FU의 전력 소모는 입력의 변화량에 의존한다. 그림 2.2는 8×8 radix-4 Booth multiplier [3] 의 전력 소모를 나타낸다. 한 입력을 고정시키고 한 입력을 임의적으로 주어졌을 때의 에너지를 nanoJ/operation으로 나타낸다. (2)는 (3)의 평균을 나타낸 것이고 (1)는 두 입력을 모두 임의적으로 주어졌을 때의 평균 에너지 소모를 나타낸다.

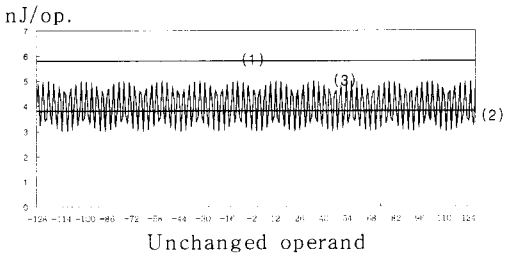


그림 2.2.Radix Booth Multiplier의 에너지 소모량
Fig. 2.2.Energy Consumption of Radix Booth Multiplier.

(1)과 (2)를 비교하였을 때 곱셈기의 평균 전력 소모는 한 입력이 고정되었을 때 35%가 적게 소모되었다. 입력 변수의 평균 해밍 거리(Average Hamming distance, AHD)는

$$H(x) = AHD(x) = \lim_{n\to\infty} \frac{\sum_{i=1}^{n} H(x_i, x_{i-1})}{n} \quad (식\ 2.1)$$

이며 여기서 $H(x_i, x_{i-1})$는 x_i와 x_{i-1} 사이의 해밍거리(Hamming distance)이며 x_i는 제어구간(control step) i에서의 입력 x의 값이다. 해밍거리는 두변수의 2진수를 XOR를 취하여 구해진다. 즉, 해밍거리는 두 개의 2진수사이의 bit의 차이 수를 말한다. 입력 y에 대한 평균 해밍거리 H(y)도 유사하게 표현된다. 연산기의 두 입력의 변화가 적을수록 전력 소모량은 감소한다는 것을 알 수 있다. 또한 H(x) 및 H(y)는 스케줄링 및 리소스 바인딩에 따라 변하게 된다. 그러므로 식 2.1을 최소화하는 스케줄링 및 리소스 바인딩 알고리즘이 제안된다.

연산자의 입력 변화에 대한 스위칭 동작율 계산은 CDFG의 반복(iteration)적인 시뮬레이션을 통해 이루어진다. 초기 입력(primary input)이 논리적으로 1이 되는 확률은 0.5이고 스위칭 동작을 결정하는 단계에서 변수의 값은 CDFG의 시뮬레이션을 통해 얻어진다 [13]. CDFG의 시뮬레이션은 새로운 초기 입력값에 대하여 반복적으로 수행되며 입력값은 bit-width에 의하여 정규화 되고 반복 수에 의해 평균화된다. 즉, *i*번째 반복의 천이수 matrix의 한 entry 값은 $H(x_i)$가 되고 이러한 시뮬레이션은 모든 entry가 수렴할 때까지 - 즉 연속되는 *i*번째와 *(i+1)*번째 두 개의 반복에 대한 entry 값 $H(x_i)$와 $H(x_{i+1})$이 근소한 차이를 갖게 될 때까지 - 계속된다. 즉, 레지스터 변수간의 스위칭 동작율은 *i*번째와 *(i+1)*번째의 두개의 변수들 간에서 얻어지고, FU의 스위칭 동작율은 FU에 연속적으로 입력되는 *i*번째와 *(i+1)*번째의 두개의 변수들 간에서 얻어진다. 변수 사이의 스위칭 동작은 연산의 표 1에서 보듯이 n이 연산의 개수라 하면 이 연산을 하는데 필요한 계산 량은 n^2가 된다. 수렴하는데 필요한 시뮬레이션의 반복수가 S라 하면 수행하는데 필요한 시간은 $O(S* n^2)$가 된다. 이러한 기능적 시뮬레이션은 논리적 수준의 시뮬레이션보다 상대적으로 소요되는 CPU 시간이 짧으므로 빠른 스위칭 동작 계산 결과를 얻을 수 있다. 표 1은 그림 3.2(b)의 CDFG에 대한 정규화된 스위칭 동작율을 보인다. 100번(a)의 반복 수에서는 수렴이 되지 않았으나 1000번(b)에서는 거의 수렴되어 2000번(c)의 시뮬레이션의 결과와 거의 비슷한 데이터를 보여주고 있다. 3.4절 리소스 할당에서 사용될 FU의 스위칭 동작 확률을 표 1과 유사하게 얻어진다.

2. 저전력을 위한 스케줄링 방법

저전력을 위한 스케줄링 목적은 리소스(resource)

표 1. 그림 3.2(b)에 대한 변수간의 단위 스위칭 확률(1 bit 당)
Table 1. Unit (per 1 bit) switching probability between variables in Figure 3.2(b).

변수 \ 변수	a	b	c	d	e	f	g	h
a	0.00000	-	-	-	-	-	-	-
b	0.18750	0.00000	-	-	-	-	-	-
c	0.17750	0.16750	0.00000	-	-	-	-	-
d	0.17500	0.16750	0.18250	0.00000	-	-	-	-
e	0.19875	0.20375	0.19375	0.21875	0.00000	-	-	-
f	0.22125	0.20875	0.19625	0.21125	0.17250	0.00000	-	-
g	0.18000	0.27250	0.24500	0.23500	0.24625	0.22375	0.00000	-
h	0.37750	0.38250	0.40750	0.40000	0.39875	0.37125	0.35500	0.00000

(a) 그림 3.2(b)의 a,b,c,d,에 대한 100개의 임의의 입력 값에 대한 평균치

변수 \ 변수	a	b	c	d	e	f	g	h
a	0.00000	-	-	-	-	-	-	-
b	0.17353	0.00000	-	-	-	-	-	-
c	0.17105	0.17228	0.00000	-	-	-	-	-
d	0.17177	0.17355	0.17118	0.00000	-	-	-	-
e	0.21133	0.21150	0.21227	0.21135	0.00000	-	-	-
f	0.21248	0.21170	0.20988	0.21210	0.18710	0.00000	-	-
g	0.18788	0.26650	0.24972	0.24730	0.25740	0.24965	0.00000	-
h	0.37102	0.39020	0.40198	0.40255	0.40470	0.34635	0.35430	0.00000

(b) 그림 3.2(b)의 a, b, c, d,에 대한 1000개의 입력 값에 대한 평균치

변수 \ 변수	a	b	c	d	e	f	g	h
a	0.00000	-	-	-	-	-	-	-
b	0.17244	0.00000	-	-	-	-	-	-
c	0.17153	0.17159	0.00000	-	-	-	-	-
d	0.17105	0.17189	0.17045	0.00000	-	-	-	-
e	0.21110	0.21084	0.20888	0.21217	0.00000	-	-	-
f	0.21070	0.20861	0.21130	0.21155	0.18910	0.00000	-	-
g	0.18706	0.26653	0.24884	0.25026	0.25711	0.25236	0.00000	-
h	0.37089	0.38868	0.40301	0.40174	0.40389	0.34559	0.35220	0.00000

(c) 그림 3.2(b)의 a, b, c, d,에 대한 2000개의 입력 값에 대한 평균

즉, FU(functional units)의 입력 변수의 재사용의 가능성을 최대로 하는 것이다. 이것은 한 입력 변수를 같은 FU에서 연속적으로 수행시킴으로써 FU의 한 입력 변수를 고정시켜 (2.2절 참조) 스위칭 변화가 일어나지 않도록 스케줄링하고 이에 따라 리소스 할당을 하는 것이다. 그림 3.1은 두 가지의 스케줄링과 FU 할당 방법을 보여주고 있다. 여기서 몇 개의 연산의 결과가 한 개 이상의 연산자의 입력으로 들어감을 볼 수 있다. 즉, 1의 연산의 결과가 2와 4의 입력으로 들어가고 2의 연산의 결과가 3과 5의 입력으로 들어간다. 저전력을 구현하기 위해 연산자 5를 제어구간 4에 스케줄링하고 리소스 할당 과정에서 3과 5를 같은 FU를 사용함으로써 2개의 adder인 A1과 A2의 한쪽의 입력을 고정시켜 스위칭 동작이 일어나지 않도록 하여 저전력을 구현한다. 다음 장에서 제안된 알고리즘을 상세히 기술한다.

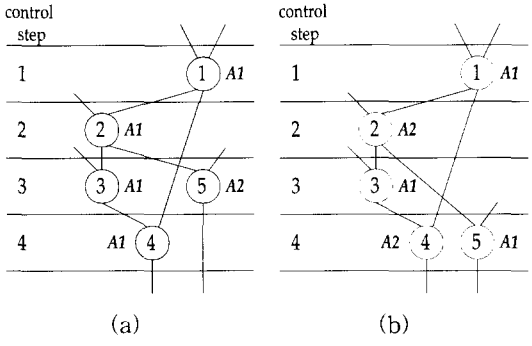
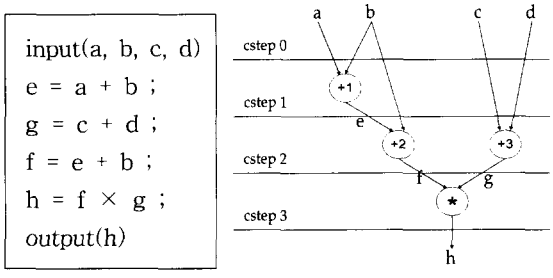


그림 3.1. (a) 저전력을 고려하지 않은 스케줄링 (b) 저전력을 고려한 스케줄링

Fig. 3.1. (a) Scheduling without considering Lower Power (b) Considering Low Power.

3. 저전력을 위한 레지스터 할당 방법



(a) 회로의 동작기술 (b) Control Data Flow Graph

그림 3.2.회로의 동작(a)과 CDFG

Fig. 3.2.Circuit Behaviour and its CDFG.

제안된 알고리즘은 최소 흐름 비용 알고리즘을 적용하여 저전력 레지스터 할당을 수행한다. 회로의 CDFG적 표현에서 변수들은 그래프내의 노드가 되며 이들은 방향성(directed) 에지로 연결된다. CDFG의 시뮬레이션을 통해 얻은 에지의 가중치는 에지로 연결된 노드 즉, 변수들이 같은 레지스터에 공유되었을 때의 스위칭 동작율이 된다. 사용 가능한 레지스터의 최소 수는 왼쪽 에지 알고리즘(left edge algorithm) [21] 에 의해 CDFG에서의 변수의 생존 구간을 분석한 후 필요한 레지스터 수를 결정한다. 즉, 모든 변수들의 생성시간(birth_time)에 따라 오름차순으로 정렬하여 변수들이 가장 많이 살아 있는 구간에서의 변수들의 개수가 레지스터의 최소 수로 결정된다.

	a	b	c	d	e	f	g
cstep 0	■	■	■	■			
cstep 1		■			■		
cstep 2						■	■

그림 3.3.그림 3.2에 대한 변수들의 생존 구간 분석

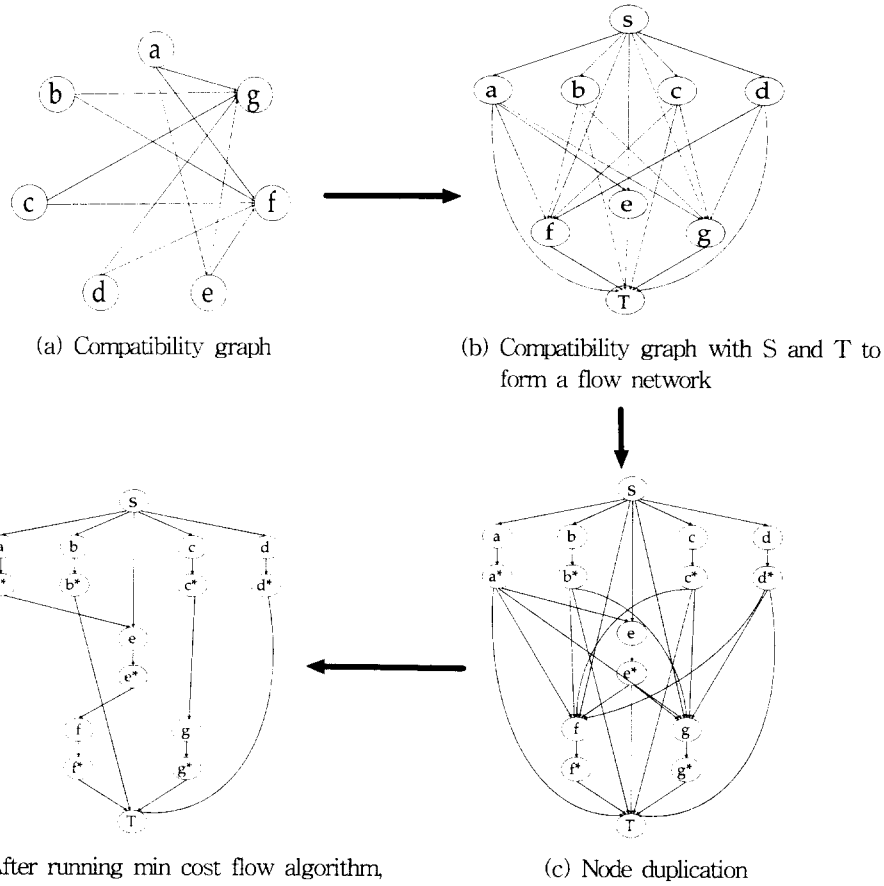
Fig. 3.3.Lifetime Interval of Variables in Figure 3.2.

그림 3.3에서, cstep 0에 변수들이 가장 많이 살아 있으므로 사용해야 하는 최소의 레지스터 수는 4개이며 사용할 수 있는 최대 수는 변수들의 개수인 8개과 같다. 그러므로 가용한 레지스터 수를 k라고 하면, $4 \le k \le 8$이된다. 생존 구간이 겹치지 않는 변수 (그래프에서 노드)들은 방향성 에지로 연결이 된다. 이것을 호환 그래프(compatibility graph, 또는 양립그래프) [22] 라고 한다. 호환그래프는 다음의 조건을 만족하는 것을 알 수 있다. *If (a,b), (b,c) are edges of graph G, then (b,c) is in G.* 그러한 조건을 만족하는 그래프를 transitively oriented graph라고 한다. 호환 그래프에서 에지로 연결된 변수들은 레지스터를 공유할 수 있음을 의미한다

(그림 3.4a 참조). 이렇게 구성된 호환 그래프에서 전체 비용을 최소화하는 k개의 disjoint 클릭(clique)을 구하는 문제로 정형화될 수 있다. 클릭이란 완전부 그래프(complete subgraph)로서 모든 노드들간에 에지로 연결된 그래프란 뜻이다.

그러한 네트워크에 최소 비용 흐름 알고리즘 [10] [23] 을 적용하기 위하여 출발점(S)과 도착점(T)과 각각의 변수를 나타내는 노드들이 방향성으로 연결된

PATH 1 : S→a→e→f→T -----> REG1 : a, e, f
PATH 2 : S→b→T -----> REG2 : b
PATH 3 : S→c→g→T -----> REG3 : c, g
PATH 4 : S→d→T -----> REG1 : d



(a) Compatibility graph

(b) Compatibility graph with S and T to form a flow network

(c) Node duplication

(d) After running min cost flow algorithm, showing 4 disjoint paths

그림 3.4. 레지스터의 호환 그래프에서 네트워크의 형성
Fig. 3.4. Network Formulation for Register Compatibility Graph.

다 (그림 3.4b 참조). 형성된 네트워크에 일반적인 최소 비용 흐름 알고리즘을 그대로 적용하면 모든 노드가 포함되지 않거나 한 노드에서 들어오거나 나가는 경로가 두 개 이상이 될 수 있다. 레지스터 할당을 위해 한 노드에서 입, 출력되는 흐름(flow)량은 한 개이어야 하고 모든 노드가 포함되어야 한다. 이러한 문제점을 해결하기 위해 새로운 가중치 함수를 이용 네트워크를 변형하여야 한다. 그림 3.4c에서와 같이 우선 S, T를 제외한 각각의 노드를 이중화(duplication)시켜 제한 용량(capacity)을 1로 주면 노드의 입, 출력 흐름량은 한 개로 정해진다. 이중화된 노드들 예를 들면 a와 a* 사이의 에지의 제한용량을 1로 함으로써 흐름이 각 노드를 한번씩만 지나게 된다.

그리고 모든 노드가 포함되기 위하여 제한용량과 에지의 가중치는 그림 3.5와 같이 주어진다. 에지 가중치는 스위칭 동작율을 정수화시켜 이들을 최대의 값에

서 뺀 다음 음의 부호를 취함으로써 모든 노드를 지나가게 한다. 예를 들어 에지 가중치에 스위칭 동작 확률을 그대로 주면 모든 흐름의 양이 한 노드만을 거친 채 도착점으로 가게 된다. 그러나, 그림 3.4처럼 가중치를 결정하면 가중치의 상대적 크기는 유지되고 모든 노드를 거치게 된다. 예를 들면 W_a+N을 3 및 2로, M을 5라고 하면 가중치는 $W=-(M-W_a\times N)$에 의하여 -2와 -3으로 변화된다. 즉 -2 > -3의 상대 크기 비교는 본래의 3 > 2와 바뀌지 않게 된다. 레지스터를 4개 사용한다면 흐름 량은 4로 하여 최소 비용 흐름 알고리즘을 수행하면 그림 3.4d와 같이 다음과 같은 4개의 분리(disjoint)된 경로(즉, 클릭) 가 주어진다: (S, a, e, f, T), (S, b, T), (S, c, g, T), (S, d, T). 여기서 (S, a, e, f, T)의 경로 비용은 B(S, a) + B(a, e) + B(e, f) + B(f, T)가 된다. 이 결과는 스위칭 동작이 최소화된 레지스터 할당을 의미하고 그림 3.4와 같이 레지스터에 변수들이 저장된다.

그림 3.5에서 노드 T는 변수 h에 해당된다.

입력(input) : 레지스터 공유를 위한 변형된 CDFG
출력(output) : 분리된 k(=레지스터 수)개의 경로

$$C_e=1, B_e=L, \quad \forall e=(S,i),(i,T)$$
$$C_e=1, B_e=W, \quad \forall e=(i,j)$$

i, j : 일반적인 노드
e : 노드를 연결하는 에지
S, T : 출발점과 도착점
C_e : 에지 e에 부여된 제한 용량 (capacity)
B_e : 에지 e의 비용 (cost)
$W = -(M - W_a\times N)$
W_a : 두 노드간의 스위칭 확률
N : 스위칭 확률을 정수화하기 위한 값
M : $W_a\times N$의 최대 값과 크거나 같은 정수
L : W의 최대 값과 크거나 같은 정수

그림 3.5.레지스터 할당을 위한 가중치 비용과 용량
Fig. 3.5.Weight and Capacity for Register Allocation.

4. 저전력을 위한 리소스 할당 방법

최소 비용 흐름 알고리즘을 적용한 리소스 할당 과정은 레지스터 할당 과정과 유사한 방법을 사용한다. 레지스터 할당과정에서 각 변수들이 노드로 표현되는 것과 마찬가지로 리소스 할당에서는 리소스들이 노드로 표현되며 에지의 가중치는 리소스의 두 입력의 스위칭 동작이 최소화 되도록 정한다.

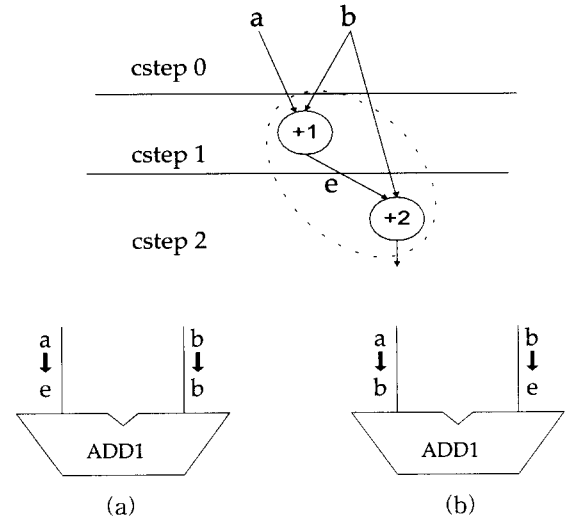


그림 3.6. 두 연산자를 공유시 발생하는 입력
Fig. 3.6. Input to the Shared Resource.

예를 들어 그림 3.2의 CDFG에서 +1과 +2를 ADD1로 공유한다고 하면 ADD1의 각각의 입력은 그림 3.6과 같이 두 가지의 경우가 생긴다: 즉, {(a, e), (b, b)}와 {(a, b), (b, e)}. 표 2와 같이 모듈 할당 과정의 +1과 +2 사이의 가중치는 두 가지 중 작은 스위칭 확률, 즉, (스위칭 확률(a, e) + 스위칭 확률(b, b))과 (스위칭 확률(a, b) + 스위칭 확률(b, e)) 중 작은 것을 취함으로써 두 리소스를 공유시 모듈의 전력소모를 줄이도록 한다.

표 2. 저전력 리소스 할당의 위한 가중치 결정
Table 2. Weight Assignment for Low Power Resource Allocation.

리소스 공유 경우	연속된 입력의 쌍	스위칭 율의 합 +멀티플렉서 가중치(N=K=1 경우)	선택
+1 , +2 공유시	(a-e) (b-b)	0.2113	(a-e) (b-b)
	(a-b) (b-e)	0.3850+2	
+1 , +3 공유시	(a-c) (b-d)	0.3446+2	(a-d)(b-c)
	(a-d) (b-c)	0.3440+2	

또한 전력 소모에 영향을 미치는 캐패시턴스는 FU, 멀티플렉서, 버스들의 수에 의하여 결정된다. 그러므로 이들의 수를 줄임으로써 캐패시턴스를 줄인다. 본 논문에서는 레지스터와 리소스 사이의 멀티플렉서의 수를 줄인다.

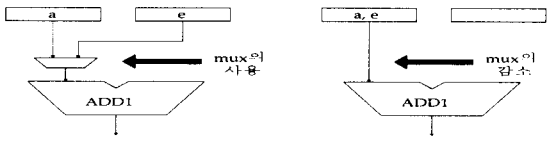


그림 3.7.변수의 저장에 따른 멀티플렉서의 증가와 감소

Fig. 3.7.The number of Multiplexor depending on Variable Assignment to Registers.

만약 다른 레지스터에 저장되어 있는 변수들이 한 리소스의 입력 단을 공유한다면 그 리소스의 입력 단에는 멀티플렉서가 필요하게 된다.

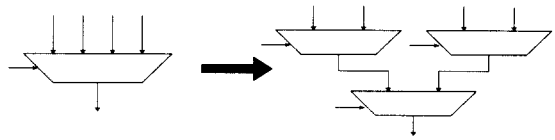
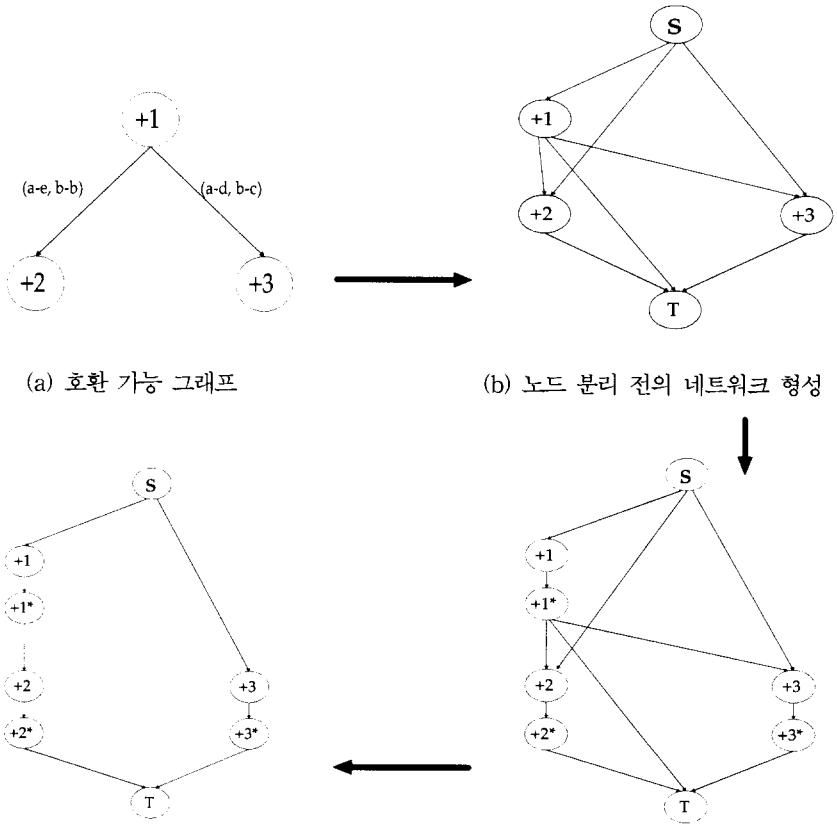


그림 3.8. 4입력 멀티플렉서의 구성도

Fig. 3.8. Configuration of 4-input Multiplexor.

그림 3.7을 살펴보면 만약 a, e가 다른 레지스터에 할당이 되어 있고 +1, +2가 같은 adder의 입력 단으로 들어가게 되면 그 입력 단에는 멀티플렉서가 필요



(a) 호환 가능 그래프

(b) 노드 분리 전의 네트워크 형성

(c) 노드 분리 후 흐름 알고리즘의 적용 결과

(d) 리소스 할당을 위한 최소 비용 네트워크 형성

그림 3.9. 리소스 할당을 위한 최소 흐름 비용 알고리즘 적용 과정

Fig. 3.9. Min Cost Flow Algorithm for Resource Allocation.

하게 된다. 여기에서 다른 레지스터에 저장된 변수들이 같은 리소스의 같은 입력 단으로 주어진다면 두 리소스가 공유하는데 필요한 가중치를 크게 함으로써 멀티플렉서의 수를 줄이도록 한다. 4개의 서로 다른 변수가 한 하드웨어의 입력을 공유한다면 그림 3.8과 같이 멀티플렉서의 수와 제어신호가 늘어난다. 그러므로 스위칭 확률에 큰 차이가 없다면 멀티플렉서의 수가 줄어들도록 최소 비용 흐름 알고리즘을 수행한다.

리소스 할당을 위한 가중치 비용과 용량은 그림 3.4의 레지스터를 위한 가중치 비용과 용량과 유사하며 그 차이는 다음과 같다.

W : -[M - ($W_a \times N$ + $W_{mux} \times k$)]
(edge weight)
W_{mux} : 0 만약 변수 i, j가 같은 레지스터에 할당되고 모듈의 동일 입력 단으로 할당될 때
: 1 만약 변수 i, j가 다른 레지스터에 할당되고 모듈의 동일 입력 단으로 할당될 때,
여기서 k는 정규화하기 위한 상수이다.

모듈 할당을 위한 최소 비용 흐름 알고리즘을 적용하기 위해 두 개(k=2)의 덧셈기를 사용한다고 하면 2개의 흐름을 보내어 최소비용 알고리즘을 적용하면 (S, 1, 2, T), (S, 3, T) 의 2개의 분리된 경로가 발생한다.

PATH 1: S→1→2→→T ---> adder 1 : +1, +2
PATH 2: S→3→T ---> adder 2 : +3

그러므로 adder 1의 각각의 입력의 순서는 a→e, b→b가 되며 adder2는 입력 c 와 d에 대한 연산을 수행한다. 이들의 정보로 최종적으로 이 회로의 데이터 경로를 구성하면 그림 3.10이 된다.

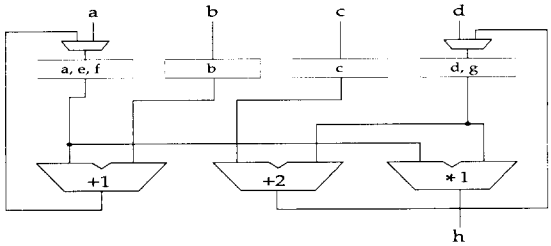


그림 3.10. 레지스터와 리소스 할당 후의 데이터 경로
Fig. 3.10. Data Path after Register and Resource Allocations.

Ⅳ. 실험 결과

본 논문에서 제안된 알고리즘은 C 언어로 구현되었고 최소 비용함수 프로그램을 사용하였다. 또한 합성된 데이터 경로를 합성하기 위해 Ultra Sun Sparc 2에서제안한 알고리즘에 의해 레지스터에 공유되는 변수들이 정해지고 이에 따라 데이터 경로가 정해진다. 정해진 데이터 경로가 동작하도록 콘트롤러를 구성하고 이에 따라 설계하였다. 설계는 Mentor tool에서 VHDL로 설계하여 QuickSim을 이용하여 검증하였다. 속도를 위해 최장 경로(Critical Path)의 Time Step을 할당하였다. 또한, 스케줄링 과정에서 ALAP(as late as possible) 스케줄링과 저전력을 구현하기 위한 스케줄링을 하였다. 또한, 최소의 리소스를 사용하여 저전력을 위한 레지스터 및 리소스 할당 과정을 최소 비용 흐름 알고리즘을 사용하였다. 벤치마크 회로는 UC Berkeley의 Low power 설계용 상위수준합성 시스템인 Hyper-LP [24] 에서 테스트 예제로 사용하는 DSP 필터를 사용하였다.

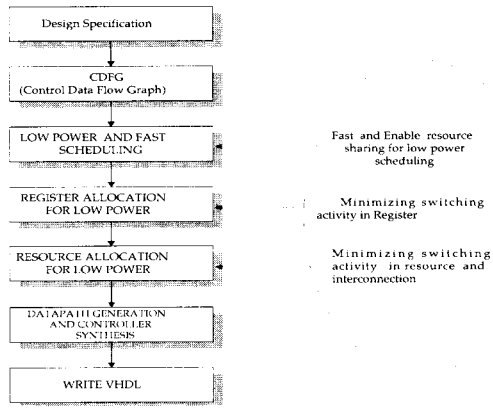


그림 4.1. 시스템 구성
Fig. 4.1. System Configuration.

표 3. 벤치마크 회로의 특징
Table 3. Characteristics of Benchmark Circuits.

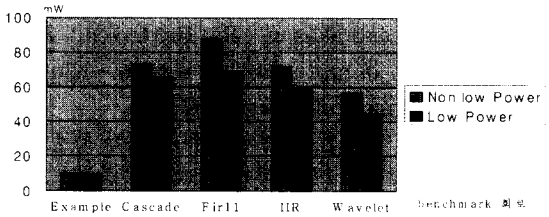
benchmark	Number of Operations			critical path	resource allocation
	mult(×)	add(+)	sub(-)		
cascade	7	7	-	6	+(2), *(2), reg(7)
fir11	11	10	-	11	+(2), *(2), reg(7)
iir5	10	10	-	8	+(2), *(2), reg(7)
wavelet	7	8	3	8	+(2), *(1), reg(7)

표 3에서 임계경로(critical path)는 회로의 최장 경로를 나타내며 자원할당은 이들을 구현하기 위해 사용된 리소스 및 레지스터의 수를 나타낸다. 또한 데이터 경로는 16 비트로 구성되어 있으며 덧셈기, 곱셈기, 뺄셈기는 Signed Magnitude형식으로 구현되었다.

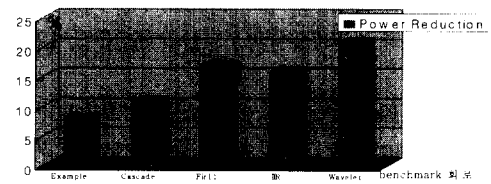
표 4. 구현된 저전력 회로의 전력 평가 비교
Table 4. Comparison between Circuits w/ and w/o applying Our Algorithm.

benchmark	w/o Low Power			w/ Low Power			Power
	FU	REG	TOTAL	FU	REG	TOTAL	Reduction
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 4 및 그림 4.2는 본 논문에서 제안된 저전력 알고리즘을 고려하여 합성된 회로(w/ Low Power)와 Hyper-LP를 통하여 합성된 회로(w/o Low Power)에 대한 전력 비교를 보인다. 실험을 통해 제안된 알고리즘을 사용함으로써 평균 15%의 전력 감소를 얻었다.



(a)



(b)

그림 4.2.Benchmark 회로의 전력 소모(a)와 감소율(b)
Fig. 4.2.Power Consumption and Reduction Rates of Benchmark Circuits.

우선 스위칭 확률을 계산하기 위해 random 입력 발생기로 입력을 발생시켜 평균을 내어 얻었다. 이 결과에 따라 제안된 알고리즘을 이용하여 상위 레벨 합성을 수행하여 데이터 경로와 콘트롤러를 구성하여 Hyper-LP에서 제공된 전력 측정[24]을 이용하였다. 제안된 알고리즘과의 성능 비교를 위하여 Hyper-LP의 상위수준 합성 시스템[24]를 수행한 결과 데이터 경로와 콘트롤러를 구성하여 Hyper-LP에서 전력 측정을 하였다. 표 4의 전체 전력 소모 (TOTAL)와 FU과 레지스터(REG)의 합과 차이에서 발생하는 전력 소모는 Control, Clock, Bus, Mux에서 소모되는 전력 소모의 합을 의미한다. 특히, 저전력 스케줄링과 리소스 할당이 적용된 Wavelet Filter의 경우에는 가장 큰 (21%) 전력 감소율을 보였다. 심사위원의 지적 데로 참고문헌[7]에서 제안된 알고리즘과의 성능 비교를 위한 Benchmark test(switching 수 비교)를 못해 본 것이 아쉬움으로 남는다.

V. 결 론

본 논문에서는 저전력 하드웨어 설계를 위한 상위 수준 합성 시스템의 설계와 구현에 대해 기술하였다. 제안된 저전력 데이터 경로 알고리즘은 스케줄링, 레지스터 할당 과정, 연산기 할당과정에서 저전력 소모를 추구하였다. 스위칭 동작의 최소화를 위해 각 변수간의 해밍 거리(Hamming distance)를 목적 함수로 모델링하였다. 주어진 최소의 Time Step과 리소스 자원을 가지고 전력 소모를 최소화하였다. 스케줄링 과정에서 가능한 경우에 한쪽 입력을 고정시켜 리소스 할당 시 전력소모를 줄였으며 레지스터와 리소스 할 당시 일어나는 스위칭 동작을 최소화하였다.

이 과정에서 최소 비용 흐름 알고리즘을 선택하여 빠른 시간에 최적의 결과를 얻도록 하였다. 즉, 변수의 개수가 증가함에 따라 계산 양이 지수적으로 증가하지 않고 일정한 증가를 보이기 때문에 많은 회로 소자의 경우에도 빠른 시간 내에 최적의 결과를 가져 올 수 있다. 제안된 알고리즘은 UC Berkeley의 상위수준합성 시스템인 Hyper-LP를 이용하여 시뮬레이션한 결과 회로의 전력 소모는 평균 15%의 전력 감소를 보였다. 본 논문에서 제안한 알고리즘은 고성능 저전력 데이터 경로 설계의 구현을 위하여 DSP, Microcontroller 및 ASIC의 설계에 효과적으로 적용 할 수 있다. 끝으로 본 논문의 질적 향상을 위해서 많은 조언을 해주신 편집위원 및 심사위원께 감사드립니다.

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 digital 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] E. Musool and J. Cortadella, "Scheduling and resource binding for low power", *Int'l Symp on Synstem Synthesis*, pp. 104-109, Apr. 1995.

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

[7] J-M. Chang and M. Pedram, "Register Allocation and Binding for Low Power Scheduling", *32rd Design Automation Conference*, pp. 29-35, 1995.

[8] R. S. Martin, J. P. Knight, "Optimizing Power in ASIC Behavioral Synthesis", *IEEE Design & Test of Computers*, pp. 58-70, 1995.

[9] R. Mehra, J. Rabaey, "Exploiting Regularity for Low Power Design", *IEEE Custom Integrated Circuits Conference*, pp.177-182. 1996.

[10] T. C. Hu, *Combinatorial Algorithms*, Addison Wesley pp. 24-27, 83-86, 1982.

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

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

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

[14] 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.

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

[16] M. C. Mcfarland, A. C. Parker, R. Camposano, "The high level synthesis of digital systems," *Proceedings of the IEEE.* Vol 78. No 2, Feb., 1990.

[17] A. Chandrakasan, S. Sheng, R. Brodersen, "Low power CMOS digital design,", *IEEE Solid State Circuit*, April, 1992.

[18] A. Chandrakasan, R. Brodersen, *Low power digital CMOS design*, Kluwer Academic Publishers, 1995.

[19] 조준동, 임세진, *소자의 스위칭 동작 최소화를 통한 디지털 회로 저전력 상위 레벨 최적화에 대한 연구*, 서울대 반도체 공동연구소, 10월, 1997년

[20] 조준동, "알고리즘 및 아키텍쳐 수준 저전력 설계자동화," *전자공학회 CAD 기술 특집*, 12월, 1997

[21] F. J. Kurdahi, A. C. Parker, "REAL: A Program for Register Allocation", 24th Design Automation Conference, ACM/IEEE, pp. 210-215, 1987.

[22] C. J. Tseng, D. P. Siewiorek, "Automated Synthesis of Data Paths in Digital Systems," *IEEE Transactions on Computer Aided Design.* Vol 5, pp. 379-395, July 1986.

[23] 임세진, *스위칭 동작 최소화를 통한 저전력 ASIC 회로설계의 상위레벨 최적화에 관한 연구*, 성균관대학교 전자공학과 석사학위논문, 10

월 1997년

[24] A.P. Chandrakasan, M. Potkonjak, J. Rabaey, R. Brodersen, Hyper-LP: A Design System for Power Minimization using Architectural Transformations. ICCAD-92 IEEE International CAD, Santa Clara, pp. 300-303, November 1992.

저 자 소 개

林 世 鎭(正會員)

1996년 성균관대학교 전자공학과 학사. 1998년 성균관 대학원 전자공학과 석사. 1998년 3월 ~ 현재 삼성전자(주) 통신연구소 주임연구원. 관심분야는 저전력 설계, CAD, 정보통신 ASIC 개발

趙 浚 東(正會員) 第 35卷 C編 第 11號 參照