論文 96-21-5-13

저전력 소모 조합 회로의 설계를 위한 효율적인 알고리듬

正會員 김 형*, 최 익 성*, 서 동 욱*, 허 훈*, 황 선 영**

An Efficient Algorithm for the Design of Combinational Circuits with Low Power Consumption

Hyoung Kim*, Ick Sung Choi*, Dong Wook Seo*, Hun Heo*, Sun Young Hwang** *Regular Members*

※이 연구는 1996년도 한국 과학재단 연구비 지원에 의하여 수행된 핵심 전문 연구의 결과임(과제 번호 : 961-0919-101-2)

요 약

본 논문에서는 조합 논리 회로에서 발생되는 전력 소모를 낮추기 위한 휴리스틱 알고리듬을 제안한다. 제안된 알고리듬은 주어진 논리 함수에서 선택한 최적의 입력 변수에 대해 Shannon expansion을 수행하고 한 쪽 cofactor만 선택하였을 때 논리 함수를 구현하는 cofactor subcircuit의 게이트 수를 감소시켜 회로의 전력 소모를 낮출 수 있다. MCNC 표준 테스트 회로에 대한 실험을 통하여 제안된 알고리듬이 전력 소모면에 있어서 기존의 precomputation 논리에 바탕을 둔 전력 최소화 알고리듬에 비해 평균 48.9% 향상된 결과를 보인다.

ABSTRACT

This paper proposes a heuristic algorithm for low power implementation of combinational circuits. Selecting an input variable for a given function, the proposed algorithm performs Shannon exansion with respect to the variable to reduce the number of gates in the subcircuit realizing the cofactor function, reducing the power dissipation of the implemented circuit. Experimental results for the MCNC benchmarks show that the proposed algorithm is effective by generating the circuits consuming the power 48.9% less on the average, when compared to the previous algorithm based on precomputation logic.

I. 서 론

최근 전자 시스템에서 휴대화에 대한 요구가 크게 대두되었고 휴대용 장치의 무게와 크기는 바로 회로의 전력 소모에 직접 영향을 주는 전지에 좌우되었다. 이에 따라 전력 소모를 낮추는 문제가 중요한 설계 사양으로 부각되어, 빠른 동작 속도를 갖는 회로

*동 대학원 박사과정 재학중
**서강대학교 전자공학과 부교수
·論文番號:96125-0416
接受日字:1996年 4月 16日

설계에 관한 연구와 병행해서 전력 소모 감소에 직접 영향을 미치는 디지탈 논리 회로의 설계 방법에 대한 연구가 활발하게 진행되고 있다. 임의의 입력이 주어졌을 때 조합 논리 회로와 순차 회로에서 소모되는 평균 전력을 최소화하는 방법이 최근 발표되었다.[1][2][3]

전력 소모를 적게 하기 위한 최적화는 다양한 수준의 설계 계층(design hierarchy)에 적용될 수 있다. 예를 들어, 주어진 논리 회로에 대해 알고리듬 변환 및 구조적 변환(architectural transformation)을 수행하여 throughput의 저하없이 회로 면적, 전력 소모 등을 감소시킬 수 있는 방법이 제안되었다.[4] 특히 조합 논리 회로의 논리 최적화는 회로의 전력 소모를 크게 감소시킨다.[5] CMOS 회로에서는 회로의 확률적인 평균 스위칭 활동(switching activity)은 회로의 평균 전력 소모에 주요한 요인이며, 평균 전력 소모는 평균 스위칭 활동에 의해 계산될 수 있다. 이를 근거로 CMOS 조합 논리 회로에 대한 전력 소모를 구하는 방법이 제안되었다.[6] 조합 논리 회로에서의 전력 소모를 줄이기 위해서는 회로를 최적화하는 방법과 precomputation 논리를 이용하는 방법이 있으며 회로를 최적화하는 방법으로는 don't-care optimization, path balancing, factorization이 제안되었다.[3] Don't-care optimization은 회로 내 게이트가 갖고 있는 controllability, observability don't-care set을 이용한다. Don't-care set을 이용하여 회로 면적을 줄이고 지연 시간을 개선하는 방법이 제안되었으며[7], 게이트의 전력 소모는 게이트의 출력이 0 또는 1 값으로 전이되는 확률에 좌우되므로 이 확률을 don't-care set을 이용하여 낮추게 함으로써 스위칭 활동을 줄여 전력 소모를 감소시키는 방법이 제안되었다.[5] Path balancing은 조합 논리 회로에서 발생하는 전력 소모의 10%에서 40%까지 차지하는 spurious transition을 줄이기 위해 회로 내 각 게이트로 수렴하는 path delay를 같도록 한다. 게이트의 입력에 unit-delay 버퍼를 선택적으로 추가함으로써 회로의 critical delay는 증가되지 않고 효과적으로 spurious transition을 제거할 수 있는 반면, 이 버퍼로 인해 스위칭 활동의 감소를 일으킬 수 있는 capacitance는 증가하게 되므로 적절한 버퍼를 추가함으로 spurious transition을 제거하는 방법이 필요하다. Factorization은 technology-independent 최적화의 한 방법으로 주어진 논리식에 대해 factoring을 수행하여 트랜지스터의 수를 줄일 수 있다. Kernel은 다단계 논리 최적화를 수행하는 과정에서 추출되며, 추출된 kernel 중에서 주어진 논리식의 리터럴 수를 최대로 줄일 수 있는 kernel이 우선적으로 선택된다. 전력 소모를 목표로 할 때는 비용 함수가 리터럴 수가 아니고 스위칭 활동이므로 이 점에 착안하여 전력 소모를 줄이는 kernel 추출 방법이 제시되었다.[8] Precomputation 논리를 이용하는 경우 이 논리를 구현하기 위해서는 논리 회로에서의 여러 입력 가운데 특정 입력들을 정하게 되며, 이 입력들을 선택하기 위해 입력의 부분 집합 선택 알고리듬이 제안되었다.[9] 이 알고리듬은 주어진 회로의 여러 입력 가운데 알고리듬에 의해 선택된 입력 집합을 중심으로 예상되는 출력 값을 미리 계산하여 이 값에 따라 이어지는 클럭 주기에서는 내적인 스위칭 활동을 감소시켜 회로의 전력 소모를 줄이는 방법을 사용한다. 이 알고리듬의 단점으로는 주어진 본래 회로에 precomputation 논리를 구현하기 위한 회로가 추가됨으로 인해 회로 면적과 클럭 주기가 증가된다는 점이다.

본 논문에서는 회로 구조에 있어 precomputation 논리를 구현하기 위한 추가 회로가 필요하지 않고 precomputation 논리에 따른 입력 집합 선택 알고리듬의 경우보다 전력 소모를 적게 발생시키는 알고리듬을 제시한다. 2 장에서는 전력 소모 및 입력 변수 선택 알고리듬에 대한 저전력 설계의 기본 이론을 제시하고, 3 장에서는 기존의 저전력 소모를 위한 알고리듬을 보여준 후 제안된 알고리듬을 자세히 설명하고 4 장에서는 실험 결과를 통하여 알고리듬의 효율성을 보이며 마지막으로 결론을 보인다.

Ⅱ. 논리 회로의 저전력 설계

1. 회로 내의 전력 소모

특정한 CMOS 논리 게이트의 전력 소모는 식(1)과 같이 주어진다.[3]

$$P=\frac{1}{2}*N*C*V_{dd}^2*f+N*Q_{sc}*V_{dd}*f+I_{leak}*V_{dd} \quad (1)$$

여기서 P는 소모되는 전력, V_{dd}는 공급 전압, f는 클럭 주파수를 나타낸다. 식(1)에서 첫번째 항목은 스위칭

활동에 의한 전력 소모를 나타내는 항목으로서, 회로 노드의 충방전에 소모되는 전력이며, C는 노드 capacitance, N은 클럭 주기당 게이트의 출력 전이 횟수(transition number), 즉 스위칭 활동을 의미한다. 두 번째 항목은 전이가 일어나는 동안의 short-circuit 전력 소모를 나타내는 항목으로서 공급 전압에서 접지까지 흐르는 전류(short-circuit current)에 의해 소모되는 전력을 의미하며, Q_{sc}는 전이가 발생할 때마다 short-circuit 전류에 의해 발생되는 전하량을 가리킨다. 나머지 항목은 누설 전류(leakage current) I_{leak}에 의한 전력 소모를 나타내는데, 이 세 항목 중에서 첫 번째 항목이 대부분의 전력 소모를 차지한다. 공급 전압은 최근 저전력 마이크로 프로세서의 설계에 의해 5V에서 3V 이하까지 낮추어져 전력 소모는 크게 감소되었다. 클럭 주파수는 회로의 불필요한 스위칭을 막음으로써 감소될 수 있으며 이것은 spurious transition을 낳는 hazard를 제거시킴으로써 가능하다. 또한 논리 블럭이 활동하지 않을 때 논리적으로 특정 블록 내의 스위칭을 금지시키는 두가지 방법이 제시되었으며, 이는 먼저 특정 블럭에 일정한 입력을 공급하거나 또는 특정 블럭의 클럭 신호를 구동하지 못하게 하는 방법이다.[12]

여러 개의 게이트로 구성된 조합 논리 회로에서 소모되는 평균 전력 소모는 식(2)와 같이 주어진다.[6]

$$P_{avg} = \frac{1}{2} * E(transitions\ of\ g_i) * (V_{dd}^2/T_{cyc}) * \sum C_i$$

$$= P(f_i = 1) * (V_{dd}^2/T_{cyc}) * \sum C_i \qquad (2)$$

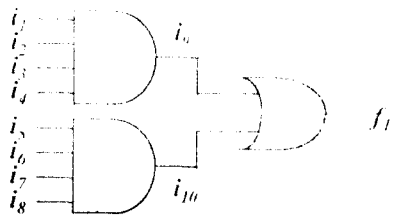
여기서 C_i는 회로 내의 i번째 게이트 g_i의 출력 capacitance, T_{cyc}은 클럭 주기이며 E는 클럭주기 당 g_i의 평균 전이 횟수(transition number), 즉 평균 스위칭 활동을 의미하며, $P(f_i=1)$은 g_i의 출력 함수(f_i)가 1이 되는 확률을 나타낸다. 회로의 평균 전력 소모는 회로 내 모든 게이트에 대한 전력 소모 합의 결과가 된다. 본 논문에서는 논리 게이트의 수를 감소시켜 전력 소모를 줄이기 위한 휴리스틱 알고리듬을 제시한다.

2. 출력 확률

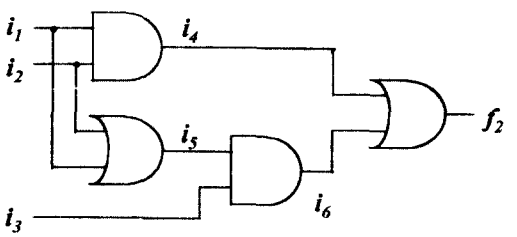
1) 정적 확률(static probability)

3개의 입력을 가진 CMOS 논리 게이트의 논리 함수가 $f = i_1 i_2 + i_2 i_3$로 주어졌다고 가정하면 게이트의 출력 평균 전이 횟수는 $E = 2 * P(f=1)$이 되며 전이횟수는 입력 신호의 확률에 의해 좌우된다. 한 입력 신호 i_j가 주어진 시간에 1이 되는 확률은 $p_j(1) = \lim_{N\to\infty} \frac{\sum_{k=1}^{N} i_j(k)}{N}$ 로 표현할 수 있으며 여기서 N은 전체 클럭 사이클 수, $i_j(k)$는 클럭 주기 k 동안 입력 신호 i_j의 값을 나타낸다. 입력 신호 i_j가 0이 되는 확률은 $p_j(0) = 1 - p_j(1)$이 되며 이 때의 $p_j(1)$, $p_j(0)$을 정적 확률이라 한다.[6] 이러한 확률을 근거로 위에서 주어진 게이트의 출력 확률을 구하면 $P(f=1) = P(i_1 i_2 + \bar{i}_1 i_2 i_3 = 1) = p_1(1)p_2(1) + p_1(0)p_2(1)p_3(1) = p_1(1)p_2(1) + (1-p_1(1))p_2(1)p_3(1)$ 이 된다. 주어진 논리 함수의 각 항목이 disjoint cube 이면 정적 확률을 근거로 출력 확률 P(f=1)의 값을 쉽게 구할 수 있다.

한 논리 신호 $i_1 = 1$의 확률을 p_1로 표현할 때 인버



(a)



(b)

그림 1. 출력 확률을 구하는 회로 예
(a) $f_1 = i_1 i_2 i_3 i_4 + i_5 i_6 i_7 i_8$를 구현하는 회로
(b) $f_2 = i_1 i_2 + i_3(i_1 + i_2)$를 구현하는 회로

Fig. 1 Circuit examples for computing output probability.
(a) Circuit realizing logic function $f_1 = i_1 i_2 i_3 i_4 + i_5 i_6 i_7 i_8$.
(b) Circuit realizing logic function $f_2 = i_1 i_2 + i_3(i_1 + i_2)$.

터의 경우 신호 $i_1=0$일 확률은 $1-p_1$이 되고 AND 게이트에서는 신호 $i_1 i_2=1$일 확률은 $p_1 p_2$가 되며 OR 게이트에서는 신호 $i_1+i_2=1$일 확률은 $p_1+p_2-p_1 p_2$가 된다.[10] 그림 1의 예제 회로에 대해 그림 1 (a) 회로의 출력에 대한 확률은 $P(f_1=1)=p_9+p_{10}-p_9 p_{10}=p_1 p_2 p_3 p_4+p_5 p_6 p_7 p_8-p_1 p_2 p_3 p_4 p_5 p_6 p_7 p_8$이 되고 그림 1 (b) 회로의 출력에 대한 확률은 $P(f_2=1)=p_4+p_6-p_4 p_6=p_1 p_2+p_2 p_3+p_3 p_1-2p_1 p_2 p_3$가 된다.

여러 개의 게이트로 구성된 조합 논리 회로에서 위와 같이 구한 출력 확률과 회로 설계에서 구해진 나머지 변수를 이용하여 식(2)을 근거로 전력 소모를 구한다.

2) 전이 확률(transition probability)

정적 CMOS 논리 회로에서는 게이트의 출력이 입력 신호가 바뀔 때만 변화하며 2개의 입력을 가진 XOR 게이트의 논리 함수가 $f=(i_1(t)i_2(t))\oplus(i_1(t+1)i_2(t+1))$로 주어졌다면 클럭 t와 t+1 동안에 게이트의 출력은 바뀌게 된다. 여기서 $i_1(t)$, $i_2(t)$, $i_1(t+1)$, $i_2(t+1)$은 클럭 t, t+1에서 게이트에 들어오는 입력 신호들이다. 함수 f의 disjoint cover는 $f=i_1(t)i_2(t)\overline{i_1(t+1)}+i_1(t)i_2(t)i_1(t+1)\overline{i_2(t+1)}+\overline{i_1(t)}i_1(t+1)i_2(t+1)+i_1(t)\overline{i_2(t)}i_1(t+1)i_2(t+1)$이 되며, 이 cover에는 입력 신호가 $0\rightarrow0$, $0\rightarrow1$, $1\rightarrow0$, $1\rightarrow1$로 전이하는 전이 확률이 사용되어야 한다. 변수 i_j의 전이 확률을 $p_j(0\rightarrow0)$, $p_j(0\rightarrow1)$, $p_j(1\rightarrow0)$, $p_j(1\rightarrow1)$로 할 때 $p_j(1\rightarrow0)$은 $p_j(1\rightarrow0)=\lim_{N\rightarrow\infty}\frac{\sum_{k=1}^{N} i_j(k)\overline{i_j(k+1)}}{N}$로 정의된다.[6] 정적 확률은 전이 확률로부터 구해질 수 있으며 그 관계는 $p_j(0)=p_j(0\rightarrow0)+p_j(0\rightarrow1)$, $p_j(1)=p_j(1\rightarrow1)+p_j(1\rightarrow0)$이 된다. 위에서 주어진 XOR 게이트의 출력 확률은 $P(f=1)=p_1(1\rightarrow0)p_2(1)+p_1(1\rightarrow1)p_2(1\rightarrow0)+p_1(0\rightarrow1)p_2(1)+p_1(1\rightarrow1)p_2(0\rightarrow1)$이 된다.

Ⅲ. 입력 선택 알고리듬

1. Precomputation 논리에 바탕을 둔 저전력 소모 알고리듬

그림 2에 레지스터 전송 수준의 기본 회로를 보인다. 블럭 C_i는 랫치 L_i, 레제스터 R_i에 의해 분리되는 조합 논리 불럭이며 함수 f로 나타내진다. 그림 3 (a)는 그림 2의 회로에 대해 precomputation 논리 구조로 변환한 회로이며 여기서 g_1, g_2는 논리 함수로서 일종의 predict 함수이다.[9] 이 함수들은 출력 함수 f의 입력으로써 표시되며 $g_1=1$이면 $f=1$이 되고 $g_2=1$이면 $f=0$이 되는 조건을 만족시킨다. 클럭 주기 t 동안 g_1 또는 g_2가 1이 되면 랫치 L_i의 load-enable 신호는 0이 되어 클럭 주기 t+1 동안 조합 논리 블럭 C_i의 입력은 바뀌지 않게 된다. 클럭 주기 t 동안 g_1이 1이면 레지스터 R_i의 입력은 다음 주기 t+1 동안 1이 되며 g_2가 1이면 R_i의 입력은 0이 된다. 같은 클럭 주기 동안에는 제시된 조건에 의하여 g_1과 g_2가 동시에 1이 될 수는 없다. g_1+g_2의 입력 조건에 따라 블럭 C_i의 입력이 스위칭 활동이 일어나지 않게 되어 블럭 C_i의 전력 소모가 감소된다. 그림 3 (b)는 다른 형태의 precomputation 구조를 가진 회로로서 클럭 주기 t 동안 g_1 또는 g_2가 1이 되면 랫치 L_{i_2}의 load-enable 신호는 turn off되어 다음 클럭 주기 t+1 동안 L_{i_2}의 출력은 스위칭 활동을 일으키지 않게 되고, L_{i_1}의 출력에 대해서만 스위칭 활동이 일어남으로써 불럭 C_i의 전력 소모는 감소된다. 이 회로에서는 먼저 g_1과 g_2에 대한 입력 변수를 선택하면 제시된 조건을 만족시키는 함수가 정해진다. 따라서 어떠한 입력 변수를 선택할 것인가가 중요한 문제가 된다.

그림 3 (a)에서 보여준 회로의 블럭 C_i의 입력 신호 집합 $I=\{i_1, ..., i_n\}$에 대한 논리 함수 f(i)를 가정할 때 입력 변수 i_j에 대한 obsevability don't-care set(ODC)은 $ODC_j=f_{i_j}\cdot\overline{f_{i_j}}+\overline{f_{i_j}}\cdot f_{i_j}$이 된다.[9]

여기서 f_{i_j}와 f_{i_j}는 i_j에 대한 f의 cofactor이고 $\overline{f_{i_j}}$와 $\overline{f_{i_j}}$는 $\overline{f}$의 cofactor이다. 주어진 입력 부분 집합이 ODC_j에 있다면 i_j가 랫치로 load 되지않게 할 수 있다. 입력 i_m, i_{m+1}, ..., i_N을 랫치에 load하지 않을 경

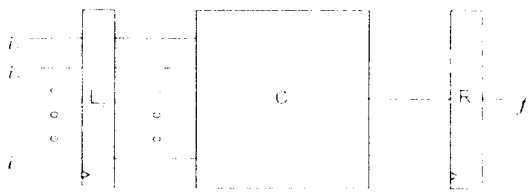
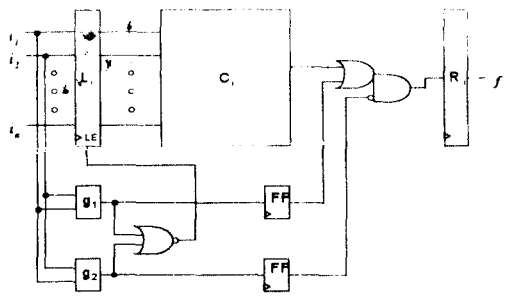
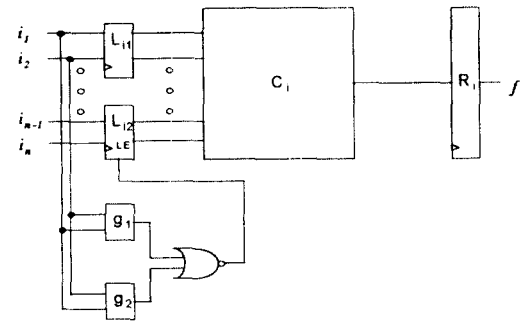


그림 2. 레지스터 전송 수준의 기본회로
Fig. 2 Register-transfer level circuit.



(a)



(b)

그림 3. Precomputation 논리를 이용한 회로
(a) Precomputation 논리 회로
(b) 다른 형태의 precomputation 논리 회로

Fig. 3 Circuit based on precomputation logic.
(a) Precomputation-based architecture.
(b) Another precomputation-based architecture.

우 함수 g는 $g=\prod_{j=m}^{N} ODC_j$로 표현된다.[9]

그림 3 (b)에서 입력의 부분 집합 $I=\{i_1, ..., i_m\}$ ($m<n$)에 있는 입력들이 g_1, g_2에 영향을 주는 변수로 선택된다고 가정할 경우, 확률 $p(g_1+g_2=1)$이 최대이면서 앞에서 제시된 조건을 만족시키는 g_1, g_2를 찾아야 한다. 이 때 g_1, g_2는 f의 universal quantification을 사용하여 결정할 수 있다. 입력 변수 i_j에 대한 f의 universal quantification은 $U_{i_j}f=f_{i_j}\cdot f_{\bar{i}_j}$로 정의할 수 있으며, $D=X-S$일 경우 $g_1=U_D f$가 되고 $g_2=U_D\bar{f}=U_{i_{m+1}}\ldots U_{i_n}\bar{f}$가 된다. 만일 함수 f와 precomputation 논리에 필요한 k개의 입력 신호를 갖는 집합 S가 주어진다면 $D=X-S$에서 $g_1=U_D f$, $g_2=U_D\bar{f}$를 계산한다. 확률 $p(g_1+g_2=1)$이 최대값이 되도록 최적의 입력 부분 집합을 선택하여 불럭 C_1에 발생하는 전력 소모를 구할 경우, 그림 2의 회로에 비해 그림 3 (b)의 precomputation 논리를 적용한 회로가 전력 소모가 적게 발생한다. 최적의 입력 부분 집합을 선택하는 알고리듬과 출력 함수가 여러 개 존재할 경우 이 함수들 중에서 precomputation 논리를 적용할 출력 함수의 부분 집합을 선택하는 알고리듬은 이미 제안되었으며[9], precomputation 논리에 필요한 입력의 수는 전체 입력의 수보다 훨씬 적어야 좋은 접근 방법이 된다.

2. Shannon expansion을 위한 입력 변수 선택 알고리듬

모든 논리 함수는 Shannon expansion으로 표시할 수 있으며 입력 신호 집합이 $I=\{i_1, ..., i_n\}$인 함수 f에 대해 입력 변수 i1에 대한 전개는 $f=i_1 f_{i_1}+\bar{i}_1 f_{\bar{i}_1}$이 된다. 여기서 f_{i_1}, $f_{\bar{i}_1}$는 i_1에 대한 f의 cofactor이다.

논리 함수 f에 대해 Shannon expansion을 수행하면 그림 4와 같은 회로 구조로 변환된다.[9][11] 입력 변수 i_1의 값에 따라 입력 랫치 L_1, L_2 중 한 랫치의 load-enable 신호를 세팅함으로써 cofactor f_{i_1}, $f_{\bar{i}_1}$ 중 한 cofactor는 disable되고 다른 한 cofactor는 계산되어 스위칭 활동은 감소한다. 입력 i_1은 해당 cofactor를 계산하는 먹스의 제어 신호로 작동한다. 선택된 입력 변수에 대해 Shannon expansion을 수행하여 cofactor subcircuit 내의 게이트 수를 줄일 수 있다면 회로의 전력 소모를 훨씬 적게 할 수 있다. 따라서 논리 함수가 주어졌을 때 최적의 입력 변수를 선택하는 알고리듬이 필요하게 된다.

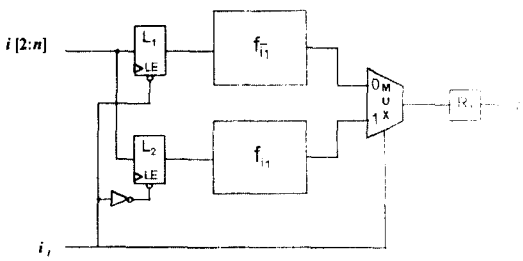


그림 4. Shannon expansion을 이용한 회로

Fig. 4 Circuit based on Shannon expansion.

```
Calc_Weight( f ):
/* f : 최적의 입력 변수를 선택하기 위한 논리 함수 */
/* Num_Cubes : 함수 f 내의 리터럴에 대해 각 리터럴을 포함하는 큐브의 수 */
/* Weight : f 내에 한 변수나 그 보수가 있을 때는 이 변수와 보수를 포함하는 큐브의 수,
            한 변수나 보수만 있을 때는 각 리터럴을 포함하는 큐브의 수 */
/* Cubes : 같은 weight 값을 갖는 변수를 포함하는 큐브들의 집합 */
/* Num_Variables : 큐브에 포함된 변수의 수 */
{
        Perform collapsing for given function f;
        for( i = 1 ; i <= n ; i++)  {          /* n : 함수 f 내 변수의 수*/
        /* 각 변수에 대한 가중치를 계산 */
          Weight( v_i ) = Num_Cubes( v_i ) + Num_Cubes( v_i' );
        }
        if( only one variable in f has maximum weight)
           v = the variable;
        else
        {
           for( j = 1 ; j <= m ; j++)  {       /* m : weight 값이 같은 변수의 수 */
                 Weight(v_j) = 0;
                 Cubes = set of cubes containing v_j;
                 for each cube c ∈ Cubes
                        Weight(v_j) = Weight(v_j) + Num_Variables(c);
           }
           v = the variable with maximum weight;
        }
        return(v);

}
```

그림 5. 최적의 입력 변수 선택 알고리듬
Fig. 5 Best input variable selection algorithm.

그림 5는 여러 입력 신호로 이루어진 회로의 논리 함수에서 최적의 입력 변수를 선택하는 알고리듬을 보여준다.

먼저 주어진 논리 함수 f의 collapsing을 수행한다. f를 이루는 리터럴 중 f 내의 큐브에 가장 많이 포함된 리터럴을 선택하기 위해 각 리터럴이 포함되는 큐브의 수를 계산한다. f 내에 한 변수와 그 보수가 있을 때는 이 변수나 보수를 포함한 큐브의 수를 합산하여 가중치를 구하고, 한 변수나 보수만 있을 때는 해당 변수나 보수가 포함된 큐브의 수로 가중치를 구한다. f 내에서 가장 큰 가중치를 가진 변수가 오직 한 개만 존재하는 경우 이 변수를 최적 입력 변수로 정한다. 이 변수를 중심으로 Shannon expansion을 적용하면 cofactor들 내 큐브의 수가 최소가 되고 한 쪽 cofactor만 구동되어 스위칭 활동이 크게 감소되므로 그만큼 전력 소모가 감소되기 때문이다. 가장 큰 가중치를 가진 변수가 두개 이상 존재하면 해당 변수들에 대해 각 변수를 포함하는 큐브들을 선택하고, 선택된 큐브를 중심으로 큐브를 이루는 변수들의 수를 구한다. 이 값이 제일 큰 변수를 최적의 입력 변수로 선택한다. 이는 이러한 입력 변수를 선택할 경우 다른 입력 변수에 비해 큐브를 구현하는 게이트의 load capacitance가 감소되어 결국 회로에서 발생하는 전력 소모가 감소되기 때문이다. 다음과 같이 두 가지 회로의 논리 함수가 주어졌을 때 제안된 알고리듬을 이용하여 최적의 입력 변수를 선택한다. 예를 들어 논리 함수

$f_1 = i_6(i_4 + \bar{i_3}(\bar{i_1} + \bar{i_2}) + i_1 i_2 i_3)$가 주어졌을 때 cofactor subcircuit 내의 게이트 수를 최소로 하는 입력 변수를 선택한다. 함수 f_1에 대해 collapsing을 수행하면 $f_1 = i_4 i_6 + \bar{i_1}\bar{i_3} i_6 + \bar{i_2}\bar{i_3} i_6 + i_1 i_2 i_3 i_6$이 되고 f_1의 각 변수에 대해 가중치를 구할 경우 변수 i_1의 가중치는 2, i_2의 가중치는 2, i_3의 가중치는 3, i_4의 가중치는 1, i_6의 가중치는 4가 된다. 이중 i_6의 가중치가 가장 크므로 이 변수를 Shannon expansion 수행을 위한 최적 입력 변수로 정한다. 다른 예로 논리 함수 $f_2 = i_5(\bar{i_1} + i_3 i_6) + (i_2 \bar{i_3} + \bar{i_2} i_3)(\bar{i_1} i_4 + i_1(i_4 \bar{i_5} + \bar{i_4} i_5))$가 주어졌을 때 이 논리 함수를 구현하는 회로의 전력 소모를 크게 감소시킬 수 있는 입력 변수를 선택한다. 함수 f_2에 대해 collapsing을 수행하면 $f_2 = \bar{i_1} i_5 + i_3 i_5 i_6 + \bar{i_1} i_2 \bar{i_3} i_4 + \bar{i_1} \bar{i_2} i_3 i_4 + i_1 i_2 \bar{i_3} i_4 \bar{i_5} + i_1 i_2 \bar{i_3} \bar{i_4} i_5 + i_1 \bar{i_2} i_3 i_4 \bar{i_5} + i_1 \bar{i_2} i_3 \bar{i_4} i_5$이 되고, f_2의 각 변수 당 가중치를 구하면 변수 i_1, i_3의 가중치는 7, i_2, i_4의 가중치는 6, i_5의 가중치는 5, i_6의 가중치는 1이 된다. 변수 i_1, i_3의 가중치가 같으므로 이 두 변수를 포함하는 큐브들의 리터럴 수의 합을 구하면 i_1를 포함하는 큐브들 내의 리터럴 수의 합은 30, i_3를 포함하는 큐브들 내의 리터럴 수의 합은 31이므로 i_3를 최적의 입력 변수로 정한다.

기존의 precomputation 논리에 따른 입력 부분 집합 선택 알고리듬은 회로 구조에 있어 precomputation 논리를 구현하기 위한 회로가 그림 2의 회로에 추가되어야 하므로 path delay가 증가되어 critical하지 않는 회로에 이 알고리듬을 적용하는 것이 바람직하며, 그림 2의 회로에 비해 precomputation 논리 회로가 전력 소모면에서 효율적이기 위해서는 이 논리를 구현함으로 발생되는 추가 회로로 인한 전력 소모 증가, 회로 면적 및 지연 시간의 증가 등이 최소의 비용을 발생하도록 추가 회로를 구현하여야 한다. 따라서 함수 f의 입력들 중에서 precomputation 논리를 구현

표 1. 벤치마크 회로에 대한 precomputation 논리 방식과 제안된 방식의 전력 소모 비교.

Table 1. Comparison in power consumption for precomputation scheme and proposed scheme.

Circuit	레지스터 전송 수준의 기본회로				Precomputation 논리 방식[9]		제안된 방식		
	입력수	출력수	리터럴수	Power (uW)	리터럴수	Power (uW)	리터럴수	Power (uW)	% Red
apex2	39	3	395	2387	399	1378	743	1385	-0.5
cht	47	36	167	1835	168	1537	149	306	80.1
cm138	6	8	35	286	38	153	30	49	68.0
cm150	21	1	61	744	62	574	66	184	67.9
cmb	16	4	62	620	72	353	44	110	68.8
comp	32	3	185	1352	198	627	63	109	82.6
cordic	23	2	194	1049	212	645	72	165	74.4
cps	24	109	1203	3726	1229	2191	1539	1717	21.6
dalu	75	16	3067	11049	3079	7344	2996	4897	33.3
duke2	22	29	424	1732	448	1328	558	828	37.7
e64	65	65	253	2039	258	513	244	323	37.0
majority	5	1	12	173	13	141	11	28	80.1
misex2	25	18	113	976	129	828	104	175	78.9
misex3	25	14	626	2350	628	1903	732	2308	-21.3
mux	21	1	54	715	54	557	60	144	74.2
pcle	19	9	71	692	74	486	69	120	75.3
pcler8	27	17	95	917	98	571	77	101	82.3
sao2	10	4	270	1191	272	422	210	438	-3.8
spla	16	46	634	2267	640	1340	776	1222	8.8
term1	34	10	625	3605	639	2133	242	500	76.6
too large	38	3	491	2718	492	1756	1499	2416	-37.6
unreg	36	16	144	1499	146	1234	84	120	90.3
Average	-	-	-	2083	-	1273	-	802	48.9

할 수 있는 최소한의 입력을 가진 입력 부분 집합을 선택하는 것이 중요하면서도 어려운 작업이 된다.

제안한 입력 변수 선택 알고리듬은 그 근거가 되는 회로 구조가 기존의 알고리듬의 회로 구조에 비해 간단하고, disable되는 입력이 기존의 알고리듬과 달리 observability don't-care set에 있을 필요가 없게 되며 또한 함수 f의 여러 입력 신호 중에서 한 입력 변수를 선택하는 일이 기존의 알고리듬보다 용이하다.

Ⅳ. 실험 결과

제안된 알고리듬의 성능 평가를 위해 MCNC 벤치마크 회로에 대해 실험을 수행하였다. 표 1은 그림 2에서 보여준 래지스터 전송 수준의 기본 회로에서 발생하는 전력 소모와 기존의 알고리듬을 이용하여 구한 전력 소모 및 제안된 알고리듬을 적용하여 구한 전력 소모를 비교하여 나타낸다. 표 1에는 벤치마크 회로로 MCNC 조합 논리 회로를 사용하였으며 이 때 사용된 공급 전압은 5 V, 클럭 주파수는 20 MHz, zero-delay model로 가정하였다. 'Powe' 열은 각 회로에서 발생하는 전력 소모를 표시하고, '% Red.' 열은 precomputation 논리 방식에 근거한 회로와 비교하여 제안된 알고리듬을 적용한 경우의 회로에서 발생하는 전력 소모 감소율을 나타낸다. 벤치마크 회로 중 cht 회로, comp 회로, majority 회로, pcler8 회로, unreg 회로는 precomputation 논리 방식이 80% 이상 전력이 적게 소모된다. 이것은 이 회로들의 논리 함수에 제안된 알고리듬을 적용할 때 함수를 구현하는 회로 내 게이트의 수가 많이 소거되는 반면, precomputation 논리 방식을 적용할 경우 여러 논리 함수 중 전력 소모의 감소를 낳는 논리 함수가 적어 회로의 전체 전력 소모가 크게 감소되지 않기 때문이다. misex3 회로, too_large 회로는 제안된 방식보다 precomputation 논리 방식이 전력 소모를 더욱 적게 발생하는데, 이것은 해당 회로들에 대해 제안된 알고리듬을 적용할 경우 리터럴 수가 상대적으로 증가되기 때문이므로 이러한 회로에는 precomputation 논리 방식을 적용하는 것이 바람직하다. 전반적으로 precomputation 논리 방식보다 제안된 방식이 평균 48.9% 전력 소모가 감소한다.

Ⅴ. 결 론

본 논문에서는 조합 논리 회로의 논리 함수가 주어졌을 때 cofactor의 논리 함수를 구현하는 subcircuit의 게이트 수를 최소로 하여 해당 회로의 전력 소모를 낮추기 위한 휴리스틱 알고리듬을 제안하였다. 제안된 알고리듬은 모든 논리함수에 적용할 수 있는 알고리듬으로 우선 주어진 논리 함수를 알고리듬에 근거하여 한 입력 변수를 선택하고 이 입력 변수에 대해 Shannon expansion을 수행하면 cofactor subcircuit의 게이트 수는 크게 줄며, 또한 한 cofactor만 선택되고 다른 cofactor는 disable되어 그만큼 논리 게이트의 스위칭 활동은 감소되어 회로에서 발생되는 전력 소모를 낮출 수 있다. 실험에서 기존의 저전력 소모를 위한 입력 부분 집합 선택 알고리듬과 비교할 때 제안한 알고리듬이 전력 소모면에서 보다 효율적임을 보인다.

참 고 문 헌

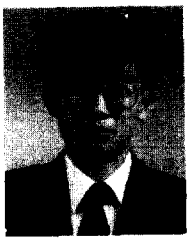
1. A. Chandrakasan, R. W. Brodersen, "Minimizing Power Consumption in Digital CMOS Circuits," Proceedings of the IEEE, Vol. 83, No. 4, pp. 498-523, April 1995.
2. L. Benini, P. Siegal, G. De Micheli, "Saving Power by Synthesizing Gated Clocks for Sequential Circuits," IEEE Design & Test of Computers, Vol. 11, No. 4, pp. 32-41, Oct. 1994.
3. S. Devadas, S. Malik, "A Survey of Optimization Techniques Targeting Low Power VLSI Circuits," in Proc. 32nd DAC, pp. 242-247, June 1995
4. A. Chandrakasan, T. Sheng, R. W. Brodersen, "Low Power CMOS Digital Design," Journal of Solid State Circuits, Vol. 27, No. 4, pp. 473-484, April 1992.
5. A Shen, S. Devadas, A. Ghosh, R. Keutzer, "On Average Power Dissipation and Random Pattern Testability of Combinational Logic Circuits," in Proc. ICCAD, pp. 402-407, Nov. 1992.
6. A. Ghosh, S. Devadas, K. Keutzer, J. White, "Estimation of Average Switching Activity in Com-

binational and Sequential Circuits," in Proc. 29th DAC, pp. 253-259, June 1992.
7. H. Savoj, R. Brayton, H. Touati, "Extracting Local Don't Cares for Network Optimization," in Proc. ICCAD, pp. 514-517, Nov. 1991.
8. K. Roy, S. Prasad, "Syclop:Synthesis of CMOS Logic for Low Power Application." in Proc. ICCD, pp. 464-467, Oct. 1992.
9. M. Alidina, J. Monteiro, S. Devadas, A. Ghosh, "Precomputation-Based Logic Optimization for Low Power," in Proc. ICCAD, pp. 74-81, Nov. 1994.
10. K. Parker, E. McClusky, "Probabilistic Treatment of Combinational Networks," IEEE Trans. Computers, Vol. C-24, No. 6, pp. 668-670, June 1975.
11. G. De Micheli, *Synthesis and Optimization of Digital Circuits*, McGraw-Hill, 1994.
12. S. Gary, P. Ippolito, G. Gerosa, C. Dietz, J. Eno, H. Sanchez, "PowerPC 603™, a Microprocessors for Portable Computers," IEEE Design & Test of Computers, Vol. 11, No. 4, pp. 14-23, Oct. 1994.

김 형(Hyoung Kim) 정회원
1979년 2월:서강대학교 전자공학과 졸업
1981년 2월:서강대학교 전자공학과 공학석사 취득
1983년 11월:럭키 금성 그룹 기획조정실 근무
1985년 1월:럭키 엔지니어링 근무
1987년 3월:금성 소프트웨어 근무
1992년 3월~현재:경민 전문대학교 전자계산학과 조교수
1993년 2월~현재:동 대학원에서 박사 과정 재학 중
※주관심분야:CAD 시스템, Logic Synthesis for Low Power, Computer Architecture 등임.

최 익 성(Ick Sung Choi) 정회원
1992년 2월:서강대학교 전자공학과 졸업
1994년 2월:서강대학교 전자공학과 공학석사 취득
1994년 2월~현재:동 대학원 박사 과정 재학 중
※주관심분야:CAD 시스템, Synthesis for Low Power, Computer Architecture 및 VLSI Testability 등임.

서 동 욱(Dong Wook Seo) 정회원
1993년 2월:서강대학교 전자공학과 졸업
1995년 2월~현재:동 대학원 재학 중
※주관심분야:CAD 시스템, Logic Synthesis for Low Power, Computer Architecture 등임.



허 훈(Hun Heo) 정회원
1992년 2월:서강대학교 전자공학과 졸업
1994년 8월:서강대학교 전자공학과 공학석사 취득
1994년 8월~현재:동 대학원 박사 과정 재학 중
※주관심분야:CAD 시스템, Testable Logic Synthesis, Logic Synthesis for Low Power 등임.

황 선 영(Sun Young Hwang)정회원
1976년 2월:서울대학교 전자공학과 졸업
1978년 2월:한국과학원 전기 및 전자공학과 공학석사 취득
1986년 10월:미국 Stanford 대학 공학박사 학위 취득
1976년~1981년:삼성 반도체 주식회사 연구원
1986년~1989년:Stanford대학 Center for Integrated Systems 연구소 책임연구원, Fairchild Semiconductor Palo Alto Research Center 기술 자문.
1989년 3월~현재:서강대학교 전자공학과 부교수
※주관심분야:CAD 시스템, Computer Architecture 및 Systems Design, VLSI 설계 등임.