울산대학교 공학연구논문집 제31권 1호 pp. 115~128, 울산대학교.
Journal of Engineering Research Vol. 31, No. 1, pp. 115~128, 2000
University of Ulsan.

저전력 회로설계를 위한 상태할당 알고리즘

김민경 · 김종수
전기전자 및 자동화공학부

<요 약>

본 논문에서는 디지털 순서 회로 설계를 위한 새로운 상태할당 알고리즘을 제안하였다. 제안된 알고리즘은 상태변수의 변화를 최소로 하는 코드를 할당하여 상태천이시의 스위칭 회수를 줄임으로서 동적 전력 소비를 감소시키는 알고리즘이다. Markov의 확률함수를 이용하여 Hamming거리가 적도록 상태천이도에서 각 상태를 연결하는 에지(Edge)에 가중치(Weight)를 정의한 다음, 가중치를 이용하여 각 상태들간의 연결성을 고려하여 인접한 상태들간에는 가능한 적은 비트 천이를 가지도록 하는 코드할당 알고리즘이다. 제안된 알고리즘의 성능을 분석하기 위하여 LG Synthesis Library의 벤치마크회로를 이용하여 실험한 결과 기존의 Lakshmikant의 알고리즘보다 최고 57.42%의 스위칭 변화를 감소시킬 수 있었다.

State Assignment Algorithms for Low Power Dissipation

Min-Kyoung Kim · Jongsoo Kim
School of Electrical Engineering and Automation

<Abstract>

In this paper, two state assignment algorithms, which assign unique code to each state in a finite state machine, were proposed. The aim of these algorithms is to reduce the average switching activity on state transitions based on the Markov probabilistic description model. Thus, the Hamming distances between neighbor nodes tied by directional edges could be maintained as minimal values. The algorithms reduced the switching activity up to 57.42% compared to the Lakshmikant's algorithm under the Logic Synthesis Benchmarks.

1. 서 론

상태할당과 관련하여 기존에 수행되었던 연구들은 회로의 지연시간을 짧게 유지하면서 회로가 차지하는 면적을 감소시키는 방법들이었으나, 근래에 저전력에 대한 요구가 높아지면서 스위칭 변화를 감소시켜 저전력 회로를 구현하려는 연구가 활발해지고 있다[1-14].

PLA(Programmable Logic Array)에 상태할당 알고리즘을 구현하기 위해서는 구조적인 제한 때문에 상태를 최소화시킬 필요가 있다. 따라서 분할을 이용하여 상태변수의 의존도를 줄이도록 상태할당을 하고 동시에 common cube의 크기와 빈도 수를 최소화시킴으로써 논리 회로를 최소화한 후 literal의 수가 최소화되게 하거나[12], 상태할당 전에 미리 예측이 가능한 symbol을 최소화시키기 위해서 입력과 출력을 고려하여 할당하는 방법도 제안되었다[13]. 2단 논리회로와 같은 간단한 형태의 논리를 고려하는 방법에서부터 다층 논리와 같이 복잡한 논리회로를 고려하여 면적을 감소시킴으로서 전력을 줄여보려는 접근법도 연구되었다. 그러나, VLSI 회로설계가 점차 상위수준으로 기술됨에 따라서 상위수준에서 저전력 회로설계를 구현할 수 있는 방법에 대한 관심이 높아지면서 스위칭 변화를 감소하여 상태코드를 할당하는 방법이 제안되었다.

본 논문에서는 저전력 회로를 구현하기 위하여 상태 코드가 이웃하는 상태들과 가능한 한 비트씩만 변화하도록 상태할당을 하는 알고리즘에 관한 연구이다. 상태 코드가 가질 수 있는 코드의 길이는 $\log_2 N$(N=상태수)에서 one-hot code라 부르는 상태수 N까지이다. 각 상태에 코드를 할당하는 방법은 여러 가지 방법이 있지만 코드 길이를 중심으로 최대 길이인 상태수를 초기 상태 코드 길이로 두고 여러 가지 논리를 적용하여 1 비트씩 감소시켜 나가는 방법이 있고, 이와는 반대로 최소 길이인 $\log_2 N$를 초기상태 코드 길이로 두고 1 비트씩 증가시키면서 스위칭이 줄어드는 코드길이를 찾아 할당하는 방법이 있다. 참고문헌 [3]에서 이용한 방법은 one-hot code를 사용하여 초기 상태할당을 한 다음, 상태 천이 확률을 근거로 상태 비트 동일화 및 비트 수정 단계를 거쳐 최종 상태코드를 할당하는 방법으로서 최대 길이에서 최소길이로의 접근법이다. 이외에 상태를 할당하는 방법으로 ILP(Integer Linear Programming)를 이용하기도 한다. 실질적으로 ILP는 모든 변수들을 방정식에 대입하여 해를 구함으로서 알고리즘으로 구현하는데 쉽고 효과적이다. 이외에 유전자 알고리즘을 사용하여 상태할당을 구하거나[14], 특정분야에서만 효과적인 알고리즘들도 있는데 이 방법은 특정 입력 시퀀스가 정하여지면 그 입력 시퀀스를 분석하여 일정한 기준을 만들어 내고 거기에 맞추어 상태 코드를 simulated annealing 방식으로 코드를 업데이트시킨다[11].

비동기 회로에서는 racing이 발생하지 않도록 하기 위하여 racing 발생하는 상태에서는 이중의 상태 코드를 할당하여 각각의 경우에 비트 변화가 1로 발생하는 쪽으로 천이가 발생하도록 여분의 상태할당을 이용한다[6]. 이 방법은 면적이 늘어나는 단점이 있지만 안정적으로 동작하면서 스위칭 확률이 매우 작다는 장점을 가진다.

본 논문의 구성은 다음과 같다. 2장에서는 전력소비모델을 이용하여 상위수준에서 저전력을 구현하기 위해서 어떤 요소들을 감소시켜야 하는지에 대하여 언급한다. 3장에서는 상태 할당을 하기 위해서 스위칭 정도를 정의하는 확률 모델을 세우며, 4장에서는 확률모델을 바탕으로 가중치를 구하기 위한 과정을 설명한다. 5장에서는 제안한 상태할당의 기본원리와 상태할당 수행시 고려할 점인 cost함수를 설명하고, 6장에서는 저전력을 구현하기 위

한 상태 할당 알고리즘을 제안한다. 그리고, 7장에서는 실험결과를 벤치마크회로를 실험한 결과를 기존의 알고리즘과 비교 분석하고, 마지막으로 8장에서는 결론과 앞으로의 연구방향에 대하여 언급한다.

2. 전력소비모델

다음의 식은 CMOS회로의 전력소비 모델이다.

$$P_{total} = \frac{1}{2} V_{dd}^2 \cdot f \sum_{nodes} C_i \cdot P_i \qquad (1)$$

P_{total} 은 소모되는 전체 전력을 나타내며, f 는 주파수, V_{dd}는 공급전압, C_i는 i 번째 노드에서의 부하용량, P_i는 CMOS회로에서 입력의 스위칭에 따라 공급전압으로부터 충전되거나 접지로 방전될 때 발생되는 스위칭으로 동적 전력소모를 나타낸다. 따라서, P_i를 노드 i에서 스위칭 확률이라 부른다. 식 (1)에서 다음과 같은 사실을 알 수 있다.

- 총 소비 전력 P_{total}은 공급 전압과 공급 주파수에 비례한다.
- 총 소비전력은 각 노드에서의 부하용량과 스위칭 확률의 곱에 비례한다.

CMOS 회로에서 소모되는 전력은 스위칭 전력, 단락회로 전력과 누설전력이다. 이중에서도 VLSI 회로에서 전력소모의 대부분을 차지하는 것이 스위칭 전력이다. 저전력 회로를 구현하기 위해서는 f, V_{dd}, C_i, P_i를 각각 낮출 수 있는데, 상위 수준에서는 자원공유를 최소화하여 데이터 상관을 보존하며 스위칭 변화를 감소시키는 방법으로 저전력을 구현한다. 그러나 식 (1)에 의하면 전력은 공급전압의 제곱에 비례하므로 공급전압을 줄이는 것이 효과적이다. 실제로 파이프라인 등을 사용하여 회로의 속도나 시간당 처리량을 높인 다음, 공급 전압을 낮추어 처리량을 유지하는 방법도 있다[2]. 그러나 잡음과 같은 영향이 심각한 경우에는 동작전압을 줄일 수가 없다. 소비전력이 주파수와 직접 비례하기 때문에 속도를 감소시키는 방법도 있으나 신호처리와 같은 실시간 응용 분야에서는 주파수를 낮게 할 수가 없다. 마지막으로 부하용량 C_i를 감소시킴으로서 전력소비를 줄일 수 있으나 이는 하위 수준에서 최종적으로 결정되며, 회로공유와 대치되므로 부가적인 전력소비를 초래한다.

3. 확률모델

2장에서 언급한 바와 같이 주어진 환경에서 전력 소모는 평균 스위칭 변화에 비례한다. 평균 스위칭 변화는 스위칭 확률이나 스위칭 천이확률로 근사화할 수 있다[1]. 현재 대부분의 전력측정 접근 방식은 pattern 의존성 문제를 해결하기 위하여 확률개념을 사용한다. 보통 zero delay 모델이 사용되고 공간적, 시간적으로 서로 독립이라고 가정한다[9]. 본 논문에서는 확률모델로 그림 1과 같은 Markov model을 사용하였다. 이 Markov model은 순

차특성에 있어서 동적 변화뿐만 아니라 공간적 상관 관계에 있어서도 유연한 구조를 가진다. 시스템에 관한 정보와 입력에 대한 확률이 주어지면 상태천이도에 대한 천이확률을 계산할 수 있다. 입력에 대한 확률은 시스템에 대한 상위수준 기술에서 직접 얻을 수도 있고 설계자로부터 직접 정보를 얻을 수도 있으며 천이확률은 그림1과 같이 상태의 입력에 의존한다.

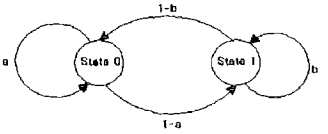

그림 1. 기본적인 Markov Chain

입력에 대한 확률을 얻은 다음 Markov chain의 형태로 변환하는데 상태 천이도가 사용된다[10]. Markov chain model은 다음 상태가 다른 어떤 시간적 제약도 받지 않고 단지 현재 상태에 의해서만 결정되는 것을 나타낸다. 상태 S_i에서 상태 S_j로 천이할 조건부 상태천이확률을 다음과 같이 표현할 수 있다.

조건부 상태천이확률

$$p_{i,j} = Prob\left(\frac{\text{다음상태} = S_j}{\text{현재상태} = S_i}\right) \tag{2}$$

그러나, 조건부 상태천이확률은 입력 값의 변화만을 고려하여 계산한 값이기 때문에 시스템의 스위칭 정도를 정확하게 나타낼 수 없다. 따라서 입력 변화에 의한 상태 천이뿐만 아니라 내부의 천이를 동시에 고려한 총 상태천이확률을 다음과 같이 정의할 수 있다.

총 상태천이확률

$$P_{i,j} = p_{i,j} \cdot P_i \tag{3}$$

식 (3)의 총 상태천이확률을 얻기 위해서는 조건부 상태천이확률 $p_{i,j}$ 와 상태확률 P_i를 모두 알아야 한다. 상태확률 P_i는 상태 i에 머무는 확률이다.

Homogeneous Markov chain은 다음과 같은 성질을 가진다.

$$B = \begin{bmatrix} p_{0,0} & p_{0,1} & p_{0,2} & \cdots \\ p_{1,0} & p_{1,1} & p_{1,2} & \cdots \\ \cdots & \cdots & \cdots & \\ p_{i,0} & p_{i,1} & p_{i,2} & \cdots \\ \cdots & \cdots & \cdots & \end{bmatrix} \tag{4}$$

B는 조건부 상태천이확률 행렬, $p_{i,j}$는 상태 i에서 상태 j로 조건부 상태천이확률이다. 확률의 공리로부터 다음의 관계를 얻을 수 있다.

$$\sum_j p_{i,j} = 1 \tag{5}$$

즉, 행렬 B에서 각 행의 합은 1이다. A를 상태확률 P_i를 원소로 가지는 정상 상태 확률 벡터라 하면 다음과 같은 관계를 얻을 수 있다.

$$A^T \cdot B = A^T \tag{6}$$

$$\sum_{i=0}^{\text{상태수}} P_i = 1 \tag{7}$$

위의 식을 살펴보면 정상상태 확률벡터를 계산해 내는 것은 조건부 상태천이확률 행렬의 고유벡터(eigenvector)를 계산해 내는 것과 같다. 위의 식 (6)과 식 (7)을 연립으로 풀어서 정상 상태 확률을 구한 후, 식 (3)에 대입하면 필요로 하는 총 상태천이확률 $P_{i,j}$를 구한다.

4. 상태 천이도(State Transition Diagram)

이장에서는 앞에서 기술한 확률모델을 표1의 상태천이표를 이용하여 설명하도록 한다.

표 1. 상태 천이표

입력	현재상태	다음상태
11	S1	S1
01/10/00	S1	S2
11	S2	S1
01/10/00	S2	S3
00	S3	S3
10/01/11	S3	S4
10/11	S4	S2
01/00	S4	S3

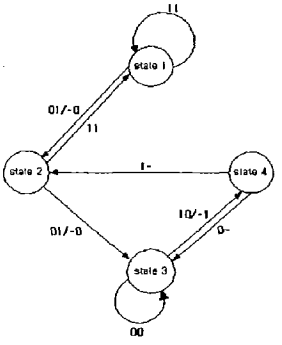

그림 2. 방향성STD
(State Transition Diagram)

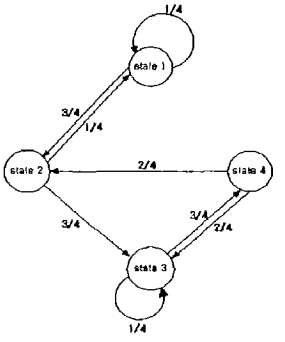

그림 3. 조건부 상태
천이확률 STD

입력은 표1과 같이 2-bit이며 4개의 상태로 구성되었으며, 입력만을 고려한 초기의 상태

천이도가 그림 2이다. 입력에 관한 영향을 논리적인 수치로 나타내기 위하여, 입력을 바탕으로 현재 상태에서 다음 상태로 천이하는 조건부 상태천이확률을 계산한다. 현재 상태에서 다음 상태로 천이할 때 발생할 수 있는 입력의 종류는 입력 비트의 수가 2비트이므로 총 4가지이다. 그 중에서 실제로 천이가 발생하는 입력이 확률이 된다. 즉, 그림 2에서, state 1에서 state 2로의 천이는 입력이 총 4가지 중에서 01/10/00과 같은 세 가지 경우에 발생하므로 조건부 상태 천이 확률은 3/4가 되고 state 2에서 state 1로의 천이는 입력이 11인 경우 발생하므로 1/4가 된다. 그림3은 이러한 조건부 상태천이확률이며 식 (8)은 조건부 상태천이확률 행렬 B이다.

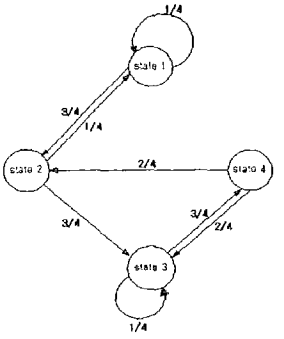

그림 4. 조건부 상태 천이확률 STD

$$B=\begin{bmatrix} \frac{1}{4} & \frac{3}{4} & 0 & 0 \\ \frac{1}{4} & 0 & \frac{3}{4} & 0 \\ 0 & 0 & \frac{1}{4} & \frac{3}{4} \\ 0 & \frac{2}{4} & \frac{2}{4} & 0 \end{bmatrix} \tag{8}$$

이 값들을 식 (6)과 (7)에 대입하면 다음의 방정식을 얻을 수 있다.

$$\begin{aligned} P_1 &= \frac{1}{4}P_1+\frac{1}{4}P_2 \\ P_2 &= \frac{3}{4}P_1+\frac{2}{4}P_4 \\ P_3 &= \frac{3}{4}P_2+\frac{1}{4}P_3+\frac{2}{4}P_4 \\ P_4 &= \frac{3}{4}P_3 \\ P_1+P_2+P_3+P_4 &= 1 \end{aligned} \tag{9}$$

(9)의 방정식을 풀면, 다음과 같이 상태확률 P_i를 얻을 수 있다.

$P_1=\frac{2}{29}$, $P_2=\frac{6}{29}$, $P_3=\frac{12}{29}$, $P_4=\frac{9}{29}$

이들 상태확률 값에 조건부 상태천이확률을 식(3)에 대입하면 그림 4와 같이 총 상태천이확률을 계산할 수 있다.

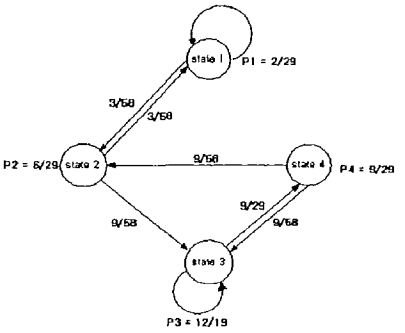

그림 5. 상태확률과 총
상태천이확률의 STD

본 논문의 목적은 상태들 간에 코드 변화가 가장 적도록 코드를 할당하는 것이다. 즉, 천이확률이 가장 높은 상태를 상태 1이라 하면 실제로 상태 1과 연결된 상태들에 도달하였을 때 그 다음 상태가 바로 상태 1로 천이할 경우가 가장 많다는 뜻이므로 상태 1에 가장 비트 변화가 적은 코드를 할당하는 것이다. 각 상태들 간의 천이 정도만을 고려하기 때문에 그림 4에서 상태 1에서 상태 2로의 천이와 상태 2에서 상태 1로의 천이를 동일한 것으로 간주한다. 따라서 새로 작성하는 상태 천이도에는 방향성을 고려하지 않고 각 상태들 간의 천이 정도는 방향성 상태 천이확률의 합을 정수화하여 나타낸다. 즉, 그림 5와 같이 상태 1과 상태 2사이의 가중치는 상태 1에서 상태 2로의 천이확률 3/58과 상태 2에서 상태 1로의 천이확률 3/58을 합하여 정수화한 6으로 계산할 수 있다.

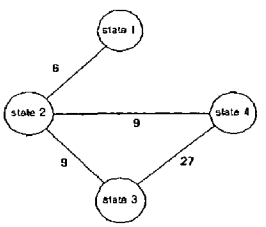

그림 6. 가중치로
표시된 STD

5. COST 함수

이 절에서는 상태천이도의 가중를 이용하여 각각의 상태에 코드를 할당한 후 이웃하는 상태들과의 스위칭확률을 측정하는 함수에 대하여 설명한다.

순서논리회로에서 저전력을 구현하기 위해서는 상태를 디코딩하는 논리회로의 크기도 고려하여 야 하는데 이는 상태 비트수가 커지면 디코딩 로직이 차지하는 면적도 커지게 되기 때문이다. 따라서, N 개의 상태를 가진 시스템에서 최소의 비트수를 가지면서 각 상태간 스위칭 확률을 작게 가지도록 상태 코드를 할당하여야한다. 상태코드의 스위칭확률은 각각의 상태가 어떤 방식으로 연결되어 있는 지와 얼마만큼의 천이가 발생하는 가에 의하여 결정될 것이다.

상태할당 알고리즘을 소개하기 전에 그림6의 최소 clique구조를 살펴보자. 이러한 삼각형 구조에 상태할당을 하면 Hamming distance가 1이 되도록 모든 상태를 할당할 수가 없다. 그림 6의 state 1에 01을 할당할 경우, state 2에서 Hamming distance가 1이 되는 코드는 11과 10으로 2 가지이다. 그러나 둘 중 어느 하나를 할당해도 상태 1과의 상태 3사이에는 Hamming distance가 1 이상이 된다. 상태 1과 상태 3 사이 또는 state 2와 state 3 사이에 2이상의 Hamming distance가 발생한다. 이러한 현상은 비트를 증가시켜 상태 코드를 할당하더라도 해결할 수 없다.

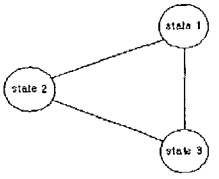

그림 7. 기본적인 최소 clique

따라서, 본 논문에서는 clique이 많은 상태천이도에서 가능하면 각각의 상태간의 Hamming distance가 2이상이 되지 않게 cost 함수를 이용하여 계산한다.

Cost 함수는 상태 i 와 j 사이의 스위칭 정도를 나타내는 가중치와 상태코드 비트 변화의 곱으로 다음 식 (10)과 같이 정의한다.

$$cost(i,\ j) = weight \times \sum_{k=1}^{state\_code\ bit} \mathrm{mod}\,(C_i(k), C_j(k)) \tag{10}$$

여기서, $C_i(k), C_j(k)$는 node i, j의 코드 비트이므로 mod 함수는 Hamming distance와 동일하다. 전체 상태 천이도의 cost는 식 (11)과 같이 각 상태의 연결성을 고려한 부분 cost의 합이다.

$$cost = \sum_{edge\ count} cost(i,\ j) \tag{11}$$

결국, 식 (11)은 스위칭 정도를 나타내므로 이 cost 값을 최소화하는 것은 전력소비량을 감소하는 것과 동일하다.

시작 상태의 선택은 최종 cost에 큰 영향을 주기 때문에 본 논문에서는 초기 상태를 가장 먼저 할당하는 기존의 알고리즘과는 다르게 여러 상태들 중에서 가장 천이확률이 높은 상태를 시작 상태로 선택한다[4]. 이러한 시작상태를 선택하는데는 두 가지 방법이 있는데 (1)어떤 상태로 향하는 에지들에 부여된 상태간 천이확률의 합이 가장 큰 상태를 선택하는

방법과 (2)단순히 에지의 천이확률이 가장 높은 에지를 선택하는 방법이 있다. 본 논문에서는 단지 가중치만을 고려해서 높은 가중치를 가지는 에지들 순서로 상태할당을 하는 Lakshmikant[4]의 알고리즘과는 달리 가중치와 연결성을 동시에 고려하여 상태할당을 하였다.

6. 알고리즘

본 논문에서는 두 가지 방법으로 상태할당을 수행한 후, cost가 가장 작은 상태할당을 채택하였다. 전체적인 알고리즘은 초기 할당 상태를 기준으로 가중치가 가장 높은 상태를 먼저 할당하는 방식이다. 할당된 상태와 연결된 에지들을 추적하면서 할당될 수 있는 코드의 집합 중 할당된 상태와 비트 변화가 가장 적은 코드를 현재 할당되지 않은 가장 높은 가중치의 상태에 할당한다. 세부적으로 상태를 찾는 방법에 따라 두 가지로 나눈다.

그림 7은 알고리즘의 flow chart로 입력 변화에 대한 상태 천이 정보를 입력받아서 상태 천이확률을 계산하고 방향성 상태 천이도를 구한다. 방향성 상태 천이도에서 확률 모델에서처럼 방향성을 제거하면서 얻은 가중치를 각각의 상태에 대한 기본적인 정보로 이용한다. 이후 상태 코드 비트를 변화시키면서 각 상태 비트수에서의 할당을 수행한 후 cost가 가장 작은 비트수의 상태할당을 선택한다.

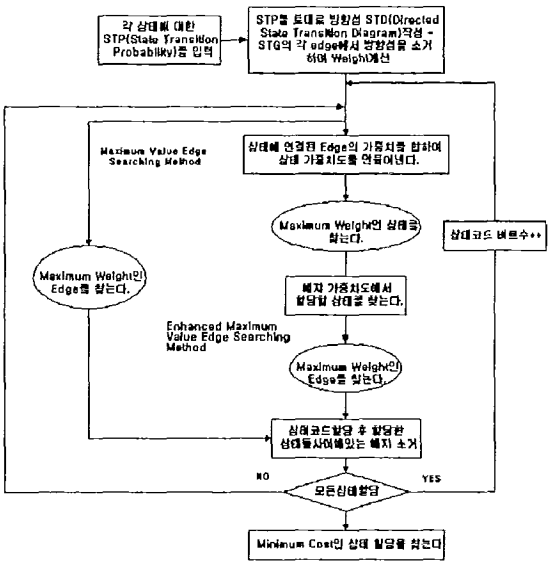

그림 8. 알고리즘 Flow Chart

6-1 Maximum Value Edge Searching(MVES)

MVES알고리즘은 상태 천이확률이 가장 높은 에지를 찾는다. 천이확률이 가장 높은 에지를 선택하여 그 에지에 연결된 상태 2개를 할당하는데 가장 먼저 할당되는 상태코드는 십진수 0과 1이다. 이 MVES 알고리즘은 이미 할당된 에지와 연결된 에지들을 고려 대상으로 하여 그 중에서 가장 천이확률이 높은 에지를 먼저 할당하는 방식이다. 그림 8의 예를 들어 할당의 작업을 수행해 보면 다음과 같다.

그림 8의 state 3과 state 4가 27로 가장 높은 가중치를 가진다. state 3과 state 4에 상태코드 0과 1을 할당한다. 이진으로 표시하면 비트 길이에 따라 처음에는 00과 01로 할당된다. State 3과 state 4를 연결하는 가중치가 27인 에지 $e_{3,4}$는 소거되고 $e_{3,4}$탐색 집합에는 $e_{2,3}$, $e_{2,4}$가 대입된다. 탐색 집합에서 가장 높은 가중치를 가지는 에지를 다음 할당할 에지로 선택해야 하나 두개의 에지의 가중치가 동등하므로 어느 것을 선택하여도 상관없다. 그러나, 이들 에지가 동일한 상태를 가리키고 있으므로 다음 할당 순서로 state 2를 할당한다. 할당할 상태가 정하여지면 할당할 상태와 연결된 상태 중에 이미 할당한 상태의 코드들과 할당할 수 있는 코드집합에서 cost가 가장 작은 코드를 찾는다. 가장 작은 비트 변화를 가지는 코드가 10과 11 두 개가 있다. 프로그램에서는 10이 할당 가능한 코드 집합 내에서 앞쪽에 위치하므로 state 2의 상태 코드로 10을 할당한다. State 2, 3, 4는 할당되었으므로 이들 사이에 있는 에지 $e_{2,3}$, $e_{2,4}$를 소거한다. 다음 단계로 탐색 집합에는 $e_{1,2}$가 있는데 이는 state 1를 가리키므로 인접한 상태 코드와 가장 적은 cost를 가지는 코드인 11을 할당한다. 최종 cost는 식 (10)과 식 (11)을 이용하여 다음과 같이 계산된다.

$$cost = 6\times \mathrm{mod}(11,\ 10) + 9\times \mathrm{mod}(10,\ 01) + 9\times \mathrm{mod}(10,\ 00) + 27\times \mathrm{mod}(00,\ 01)$$
$$= 6\times 1 + 9\times 2 + 9\times 1 + 27\times 1 = 60$$

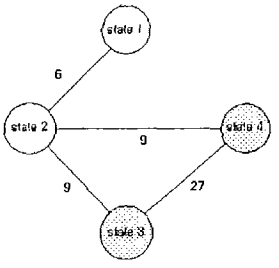

그림 9. MVES 할당

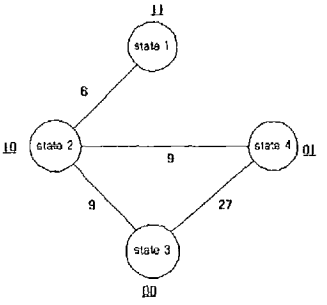

그림 10. MVES의 최종 상태할당코드

상태수가 4개이므로 상태 코드는 2비트에서 4비트의 범위에서 가질 수 있는데 이 경우는 비트 사이즈가 커지더라도 할당된 상태 코드에는 변화가 없다. 이 경우 Cost에 변화가 없으므로 가장 적은 비트인 2비트를 선택한다.

MVES의 상태할당 과정은 다음과 같다.

1. 입력 에지 중에서 가장 높은 가중치를 가지는 에지를 찾아낸다.

2. 처음 할당할 상태를 선택함에 있어서 동등한 가중치의 에지가 여러 개라면 이에 연결된 두 개의 상태에 연결된 에지가 많은 에지를 찾는다. 가능하면 많은 상태간에 distance를 작게 가지도록 할당을 하기 위해서는 많은 에지를 먼저 고려하여 주는 것이 효율적이다.
3. 에지에 연결된 두 개의 상태를 할당한다.
4. 할당된 두 개의 상태 사이에 있는 에지를 소거한다.
5. 할당될 가능성이 있는 에지를 수집하여, 다음으로 할당 가능성 있는 에지를 탐색 집합에 대입한다. 탐색 집합에 대입된 에지는 이미 할당된 상태에 연결된 에지들이다. 다음 할당할 상태는 이전에 할당한 상태(연결되어 있는)의 코드와 비교하여 distance가 적은 코드를 할당하여야 하므로 이미 할당한 상태에 연결되어 있는 에지들을 탐색 집합에 대입한다.
6. 탐색 집합에서 가장 높은 가중치를 가지는 에지를 찾는다.
7. 6번에서 선택한 가중치를 가지는 에지가 여러 개일 경우, 가능하면 이미 할당된 상태가 많이 연결된 상태를 먼저 할당한다. 그 이유는 이미 할당된 상태가 많이 연결된 상태는 상태간의 적은 distance를 가지는 코드의 종류가 적기 때문에 먼저 할당하여 주는 것이 좋다. 만일 다른 상태를 먼저 할당하게 되면 이 상태에 할당할 적합한 코드를 다른 상태에 할당하는 수가 있다.
8. 선택한 에지를 할당한다. 그 에지에 연결된 할당되지 않은 상태는 하나이므로 그 상태를 할당한다.
9. 할당된 상태들 사이에 있는 에지를 소거한다.
10. 할당될 가능성 있는 에지가 변하였으므로 탐색 집합을 다시 만든다.
11. 이런 방식으로 하여 모든 상태를 다 할당할 때까지 5번에서 10번까지를 반복하여 수행한다.
12. 1 번부터 11 번까지, 상태 비트수를 변화시키면서 반복한다.

6-2 Enhanced Maximum Value Edge Searching (EMVES)

EMVES 알고리즘은 MVES 방법과 같으나 초기 할당 상태를 다음과 같은 과정을 거쳐서 선택한다. 각각의 상태에 연결된 에지들의 가중치를 더하여 상태에 가중치를 부여한다. 이렇게 부여된 수치들을 상태 가중치라 하고, 가장 큰 상태 가중치를 가지는 상태를 초기 할당할 상태로 선택하는 것이 MVES 방식과의 차이점이다.

그림 11을 보면 state 3과 state 4가 36으로 상태 가중치가 가장 높다. 이 경우 다음 시작 상태의 선택은 연결된 에지들의 수로 결정된다. 그러나 이 역시 동일하므로 무작위로 선택한다. 프로그램 수행시 state 3을 00으로 할당한다. 탐색 집합에는 에지 $e_{2,3}$과 에지 $e_{3,4}$가 대입되고 탐색 집합 중에서 에지 $e_{3,4}$가 가중치 27로 가장 높으므로 state 4를 다음 할당할 상태로 선택한다. State 4는 01로 할당되고 다음 탐색 집합에는 $e_{2,4}$와 $e_{2,3}$이 대입된다. 이 경우는 가중치가 동일하나 모두 하나의 상태를 가리키므로 state 2를 10으로 할당하고 나머지 state 1을 11로 할당한다. 할당한 결과는 그림 9에서 할당된 결과와 동일하게 나타난다.

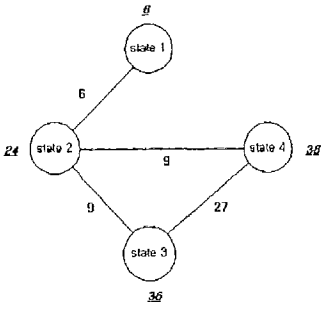

그림 11. 에지 가중치를
상태 가중치로 전환

EMVES의 상태할당 과정은 다음과 같다.

1. 입력 상태로 향하는 모든 에지가 가지는 가중치를 합하여 그 상태의 가중치로 한다.
2. 상태 가중치 중에서 가장 높은 가중치를 가지는 상태를 찾아낸다.
3. 선택한 상태를 할당한다. 처음 할당할 상태를 선택하는데 있어서 동등한 가중치의 상태가 여러 개라면 되도록 많은 에지가 연결된 에지를 찾는다. 가능하면 많은 상태에 Hamming distance가 작게 되도록 할당하기 위해서는 많은 에지를 먼저 고려하여 주는 것이 효율적이다.
4. 할당될 가능성이 있는 에지를 수집하여, 다음으로 할당 가능성 있는 에지를 탐색 집합에 대입한다. 탐색 집합에 대입된 에지는 이미 할당된 상태에 연결된 에지들이다. 다음 할당할 상태는 이전에 할당한 상태(연결되어있는)의 코드와 비교하여 distance가 적은 코드를 할당하여야 하므로 이미 할당한 상태에 연결되어 있는 에지들을 탐색 집합에 대입한다.
5. 탐색 집합에서 가장 높은 가중치를 가지는 에지를 찾는다.
6. 5번에서 선택한 가중치를 가지는 에지가 여러 개일 경우, 가능하면 이미 할당된 상태가 많이 연결된 상태를 먼저 할당한다. 그 이유는 이미 할당된 상태가 많이 연결된 상태는 상태간의 적은 distance를 가지는 코드의 종류가 적기 때문에 먼저 할당하여 주는 것이 좋다. 만일 다른 상태를 먼저 할당하게 되면 이 상태에 할당할 적합한 코드를 다른 상태에 할당하는 수가 있다.
7. 선택한 에지를 할당한다. 그 에지에 연결된 할당되지 않은 상태는 하나이므로 그 상태를 할당한다.
8. 할당된 상태들 사이에 있는 에지를 소거한다.
9. 할당될 가능성 있는 에지가 변하였으므로 탐색 집합을 다시 만든다.
10. 이런 방식으로 하여 모든 상태를 다 할당할 때까지 5번에서 10번까지를 반복하여 수행한다.
11. 1 번부터 10 번까지, 상태 비트수를 변화시키면서 반복한다.

7. 실험결과

제안한 상태할당 알고리즘의 성능을 평가하기 위해서 Logic Synthesis [15]에서 제안한 회로들을 benchmarks들로 사용하였다. 이 회로들은 대표적인 예제로 알고리즘들의 성능 평가를 위해서 사용된다.

MVES와 EMVES 알고리즘으로 상태할당을 처리한 결과는 표 2 및 표 3과 같다. 클릭의 형태를 가지는 회로는 결과가 동일하였고, 임의의 상태에 많은 에지들이 향하는 경우가 아니지만 상태들간의 에지에 가중치가 높은 경우, 즉, 어떤 두 상태들간의 천이 확률이 높은 경우가 많은 경우는 MVES 알고리즘이 효과적임을 알 수 있다. EMVES 알고리즘은 MVES 알고리즘과 cost에서는 거의 동일한 영향을 나타냄을 표를 통해서 알 수 있으나, 임의의 상태로 향하는 에지의 수가 많은 상태가 하나만 존재한다면 EMVES 알고리즘이 효과적이다.

본 논문에서는 Lakshmikant가 제안한 상태할당 알고리즘과 비교하였다. 기존의 다른 알고리즘들이 면적을 고려하여 상태할당을 하는 과거의 방법을 유지하면서 저전력을 구현하는 상태할당을 구현하였기 때문에 비교 환경이 다르다. 제안한 알고리즘은 Lakshmikant의 알고리즘에 비하여서 스위칭확률을 최대 57.42%정도 감소시킬 수 있었고, 상태가 많아질수록 cost를 효과적으로 떨어뜨릴 수 있음을 볼 수 있었다.

8. 결 론

본 논문에서는 상태천이확률을 이용하여 스위칭 변화를 감소시키는 상태할당 알고리즘을 제안하였다. 제안한 알고리즘은 시스템의 입력에 대한 상태 변화를 가지고 복잡하게 얽혀 있는 상태간의 연결성을 따져서 스위칭이 작게 발생하는 코드를 할당하였다. 그 결과 기존의 알고리즘과 비교하여 최대 57.42%의 스위칭 변화의 감소 효과를 얻을 수 있었다.

본 연구에서는 racing과 hazard를 고려하지 않았으나 비동기회로 시스템 설계에서 racing과 hazard 문제로 인한 심각한 문제들이 많이 발생하므로 이에 대한 연구가 필요하다. 또한, 입력에 관한 상태천이만을 고려하였고 출력에 관한 사항은 고려하지 않았다. 상태코드가 변경되면 조합논리회로 및 출력에 영향을 주게 되므로 이 부분에 대한 종합적인 연구가 필요하다.

표 2. MVES 알고리즘

	상태수	에지수	(cost/bit)		천이 감소율
			Lakshmikant	제안	
bbtas	6	6	174/3bit	123/3bit	29.31%
dk15	4	6	2013/3bit	1701/3bit	15.5%
s8	5	7	1504/3bit	1354/3bit	10%
tav	4	4	24/2bit	16/2bit	33.33%
train11	11	14	59669/4bit	52896/4bit	11.35%
bbara	10	22	2901/4bit	2870/4bit	1.07%
modulo12	12	12	9174/4bit	5004/4bit	45.45%
train4	4	4	5003/2bit	5003/2bit	0%
planet1	48	63	2795/6bit	1190/6bit	57.42%
sse	16	25	121/4bit	109/4bit	9.1%
sand	32	46	1024/5bit	581/5bit	43.26%

표 3. EMVES 알고리즘

	상태수	에지수	(cost/bit)		천이 감소율
			Lakshmikant	제안	
bbtas	6	6	174/3bit	123/3bit	29.31%
dk15	4	6	2013/3bit	1701/3bit	15.5%
s8	5	7	1504/3bit	1354/3bit	10%
tav	4	4	24/2bit	16/2bit	33.33%
train11	11	14	59669/4bit	52303/5bit	12.34%
bbara	10	22	2901/4bit	2829/4bit	2.48%
modulo12	12	12	9174/4bit	5004/4bit	45.45%
train4	4	4	5003/2bit	5003/2bit	0%
planet1	48	63	2795/6bit	1190/6bit	57.42%
sse	16	25	121/4bit	109/4bit	9.1%
sand	32	46	1024/5bit	581/5bit	43.26%

참고문헌

[1] L. Benini and G. De Micheli, "State assignment for low power dissipation", IEEE Journal of Solid-State Circuits. vol. 30. March 1995.

[2] 임세진, 조준동, "스위칭동작 최소화를 통한 저전력 데이터 경로의 최적화" 전자공학회 논문지 1999년 4월 제 36 권 C 편 제 4호

[3] 구경회, 조경록, "상태천이확률을 이용한 비동기 회로의 저전력 상태할당 알고리즘", 전자공학회논문지 1997년 12월 제 34 권 C 편 제 12호

[4] Lakshmikant Bhupathi and Liang-Fang Chao, "Exploiting skewed stateprobabilities for low power state assignment", Proceedings of the IEEE International Symposium on Circuits and Systems, vol. 4, 1996.

[5] 신용섭, "다단논리합성과 상태할당", 1991, 포항공과대학 석사논문

[6] Vitit Kantabutra and Andreas G. Andreou, "A state assignment approach to asynchronous CMOS circuit design", IEEE Transactions on Computers, vol. 43, no. 4, April, 1994.

[7] Paul Landman, Renu Mehra, and Jan M. Rabaey, "An integrated CAD environment for low-power design", IEEE Design and Test of Computers, vol. 13, no. 2, June 1996.

[8] Jordi Cortadella and Michael Kishinevsky, "A region-based theory for state assignment in speed-independent circuits", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 16, no 18, August 1997

[9] Radu Marculescu and Diana Marculescu, "Sequence compaction for power estimation: theory and practice", IEEE Transactions On Computer-Aided Design of Integrated Circuits and Systems, vol. 18, no. 7, July 1999

[10] Akhilesh Tyagi, "Entropic bounds on FSM switching", IEEE Transactions on VLSI Systems", vol. 5, no. 4, December 1997

[11] M. Koegst, G. Franke, and K. Feske, "FSM state assignment for low power and power estimation under user-specified input sequences", BEC' 96

[12] 이기중, 황선영, "다단 논리 회로로 구현된 FSM의 효율적인 상태할당 알고리즘", 한국 정보 과학회 논문지, vol. 18, no. 2, March, 1991

[13] 양세양, 김진옥, "유한상태기의 면적 최적화를 위한 상태할당 문제에의 해석적 접근", 한국 정보 과학회 논문지, 제 21권 제 7호, 1994. 7

[14] A. E. A. Almaini, J. F. Miller, P. Thomson, and S. Billina, "State assignment of finite state machines using a genetic algorithm", IEEE Proc. Computer Digital Techniques, vol 142, no. 4, July 1995

[15] Proc. International Workshop on Logic Synthesis, 1989-1993