論 文
53P-4-10

효율적인 Partial Scan 설계 알고리듬

An Efficient Algorithm for Partial Scan Designs

金 倫 弘† · 申 載 興*
(Yun-Hong Kim · Jae-Heung Shin)

Abstract - This paper proposes an implicit method for computing the minimum cost feedback vertex set for a graph. For an arbitrary graph, a Boolean function is derived, whose satisfying assignments directly correspond to feedback vertex sets of the graph. Importantly, cycles in the graph are never explicitly enumerated, but rather, are captured implicitly in this Boolean function. This function is then used to determine the minimum cost feedback vertex set. Even though computing the minimum cost satisfying assignment for a Boolean function remains an NP-hard problem, it is possible to exploit the advances made in the area of Boolean function representation in logic synthesis to tackle this problem efficiently in practice for even reasonably large sized graphs. The algorithm has obvious application in flip-flop selection for partial scan. The algorithm proposed in this paper is the first to obtain the MFVS solutions for many benchmark circuits.

Key Words : Partial scan, Feedback vertex set, Sequential circuit, Logic synthesis

1. 서 론

순서회로(sequential circuit)에 포함된 플립플롭들 중의 일부분만을 스캔할 수 있도록 만드는 설계방식을 부분 스캔(partial scan)이라고 한다.[10] 스캔할 플립플롭을 효과적으로 선택한다면, 부분 스캔은 최소한의 면적 부담과 성능 저하, 그리고 전체 스캔(full scan)에 비하여 테스트용이도(testability)의 큰 차이 없이 테스트 패턴 인가 시간이 단축되는 장점을 갖는다. 그동안 플립플롭 설계와 테스트 패턴 인가 방식에 관한 많은 연구가 이루어져 부분 스캔의 유용성을 입증해 주고 있다.[1,4,10] 부분 스캔의 회로 설계는 본 논문의 주제에서 벗어나기 때문에 회로설계에 관하여 자세히 살펴보지는 않는다.

본 논문에서는 플립플롭을 효과적으로 선택하기 위한 알고리듬을 다룬다. 논문 [4]에서는 순서회로에 대한 테스트 씨이퀀스(test sequence)가 매우 길어지는 이유는 바로 순차회로의 순환 구조 때문이라는 사실을 밝혀내고 이를 근거로 한 플립플롭 선택 알고리듬을 제안하였다. 이 알고리듬에서는 각 플립플롭마다 자기를 포함하는 사이클(cycle) 수를 계산한다. 플립플롭을 스캔할 수 있도록 만드는 것은 바로 그 플립플롭을 포함하는 모든 사이클을 끊는다는 것을 의미한다. 플립플롭은 최종 회로에서 최소한의 피드백(feedback)만이 남도록 선택된다.

이러한 접근방식을 기본으로 한 여러가지 변형된 방식들이 다른 연구자들에 의해서 제안되었다. 이 간단하고 휴리스틱한 방법은 상당히 실용적인 성공을 거두었고 지금까지 다른 유사 알고리듬보다 월등한 성능을 발휘하고 있다.

본 논문은 그래프에서 최소비용 피드백 정점 집합(minimum cost feedback vertex set, 이하 MFVS)을 정확히 계산하는 방법을 제안한다. 그래프가 주어지면, 이에 대한 부울 함수가 유도되는데 이 함수를 만족하는 값들이 바로 그 래프에 대한 MFVS가 된다. 이 방법은 부울 함수를 구성하면서 그래프의 모든 사이클을 일일이 고려하지 않는다. 이 함수는 MFVS를 결정하기 위해 사용되는데, 최소 비용으로 부울 함수를 만족하는 값들을 계산하는 것이 NP-hard한 문제이기는 하지만, 논리합성(logic synthesis) 분야에서 개발된 기법들을 사용하면 비교적 큰 규모의 그래프에 대해서도 효과적으로 이 문제를 해결할 수 있다. 본 논문에서는 ISCAS89 벤치마크 회로들 중의 많은 회로들에 대한 MFVS를 계산해 본다.

휴리스틱하지 않은 정확한 접근방법들도 제안되었다. Smith와 Walford[9]는 MFVS 선택 문제에 대한 탐색공간을 줄여나가는 알고리듬을 제안하였다. 이들은 그래프에서 어떤 정점이 MFVS가 될 수 있는가에 대한 간단한 충분조건을 제안한 것이다. 이 알고리듬에서는 한번에 한 정점씩 충분조건을 만족하는지 여부를 검사하고, 다시 두 정점씩 그리고 그 이상씩을 검사한다. 어떤 정점 집합이 조건을 만족하면, 그래프에서 이 정점들을 제거하여 그래프를 변경한다. 변경된 그래프에 대해서 다시 똑같은 과정이 반복된다. 그들이 제안한 조건이란 다음과 같다: A를 정점 집합이라 하고, A를 포함하는 모든 루프(loop)에 있는 정점 집합을 B라 한다.

† 교신저자, 正會員 : 상명대학교 천안캠퍼스 工科大學 컴퓨터시스템공학과 敎授 · 工博
* 正 會 員 : 동서울대학 컴퓨터시스템과 助敎授 · 工博
接受日字 : 2004年 10月 27日
最終完了 : 2004年 11月 17日

그래프에서 A를 제거했을 때 B에 포함된 정점들로 구성된 모든 루프가 끊어진다면, A는 최소 FVS가 된다. 이 알고리듬에는 두가지 문제점이 있다: (1) 최악의 경우에 이 알고리듬은 지수함수적으로 증가하는 모든 가능한 정점 조합 수를 고려해야 한다. (2) 최소 FVS 해의 모든 부분집합을 얻으려고 한다면 이 알고리듬은 유용하지 못하다. 논문 [3]에서는 정점이 선택될 때마다 잘 알려진 그래프 축소/분할 방법을 적용하는 간단한 branch-and-bound 알고리듬이 발표되었다. 흥미롭게도 이 알고리듬은 짧은 시간 안에 모든 ISCAS89 벤치마크 회로에 대한 MFVS를 계산할 수 있었다. 이것은 벤치마크 회로에 해당되는 그래프가 분할하기 매우 쉬운 구조를 갖고 있다는 것을 보여준다.

MFVS를 계산하기 위해 제안된 다양한 근사 알고리듬들 중에서는, Lee와 Reddy[6], Park와 Akers[8]의 알고리듬이 주목된다. 두 알고리듬 모두 처음에는 알려진 그래프 축소 기법을 적용하여 그래프를 전처리한다. Lee와 Reddy는 축소된 그래프에서 최소 FVS을 선택하기 위한 많은 휴리스틱 조건을 시도하였다. 이 휴리스틱 조건에서는 FVS에 추가하기 위해 많은 루프에 포함된 정점들과 많은 에지(edge)를 갖는 정점들을 선택한다. 또한 경로 길이 축소 문제를 최소 FVS 선택 문제로 변환함으로써 사이클 없는 최종 그래프에서 경로 길이가 어떻게 짧아 질 수 있는가를 보였다. 그들의 휴리스틱은 결국 벤치마크 회로에 대한 최소에 매우 가까운 FVS를 선택하게 된다. Park과 Akers가 제안한 알고리듬은 Smith-Walford 알고리듬과 매우 유사하다. 그들의 알고리듬은 그래프에서 기본 사이클들을 확인함으로써 시작된다. 기본 사이클의 정의는 Smith와 Walford의 충분조건과 바로 일치한다. 기본 사이클이란 내부에 더 이상의 다른 사이클이 존재하지 않는 최소의 사이클을 의미한다. 기본 사이클은 각 정점에 대해 파악되고, 가장 많은 수의 기본 사이클에 포함된 정점이 FVS에 추가된다. 이 휴리스틱 또한 벤치마크 회로에 대해 최소에 매우 가까운 FVS 선택이 가능하다.

2. 최소비용 피드백 정점 집합(MFVS)

G(V, E)를 방향성 그래프라고 하자. 이때 V는 정점 집합이고 E는 방향성 에지들의 집합이다. 각 에지는 순서쌍 (vi, vj), vi, vj ∈ V이 된다. 이 에지를 정점 vj에 대해서는 입력에지(in-edge), vi에 대해서는 출력에지(out-edge)라고 부른다. 그래프에서 경로는 에지와 정점이 교대로 나타나는 순서로 표현된다. 경로에서 처음과 마지막의 정점이 같다면, 그 경로는 순환적이다라고 한다. 정점 집합을 실수로 바꾸어주는 비용 함수가 있다고 가정하면, MFVS 문제는 다음과 같이 정의된다: 그래프 G로부터 어떤 정점 집합을 제거하였을 때 G 안의 모든 사이클이 제거되는 최소 비용 정점 집합, Vmin, 을 구하라. 이 문제는 NP-hard하다는 것이 증명되었다.[5]

이 문제는 부분 스캔에서 플립플롭 선택을 위해 널리 사용되는 Cheng-Agrawal 휴리스틱의 근간을 이루고 있기 때문에, 부분 스캔을 연구하는 사람들에게는 주관심사이다. 이 휴리스틱은 다음과 같다:

회로가 주어지면, 각 플립플롭은 하나의 정점으로, vi에 해당하는 플립플롭으로부터 vj에 해당하는 플립플롭까지의 경로가 존재하면 두 정점 vi과 vj 사이의 에지로 표현된 그래프 G를 구성한다. 플립플롭과 G의 정점 간에는 일대일 대응관계에 있기 때문에, 그래프에서 명확히 정점을 실제 플립플롭으로 볼 수 있다.

Cheng과 Agrawal은 사이클이 없는 그래프를 갖는 회로를 테스트 하는 것이 쉽다는 사실을 이용하였다. 이 경우의 테스트는 조합회로의 테스트와 유사하다. 이것은 바로 전체 스캔의 경우가 된다. 사이클이 존재하는 회로에 대해서, 그래프에 사이클이 없도록 플립플롭 집합이 선택되었다면, 이 플립플롭들은 부분 스캔에서 스캔 플립플롭으로 사용되기 위한 이상적인 후보가 된다. 이 집합은 바로 그래프 G에 대한 FVS이다. G를 계산할 때, Cheng과 Agrawal은 자체 루프, 즉 (vi, vi) 형태의 에지를 무시했다. 그 이유는 이런 루프들은 테스트 생성에 큰 문제를 일으키지 않기 때문이다. 부분 스캔의 사용은 전체 스캔에 따른 면적과 지연 부담을 줄이려는 의도에서 시작되었으므로, FVS를 선택할 때 이 사실을 항상 염두에 두는 것이 중요하다. 따라서 우리가 해결해야 하는 실제 문제는 최소 비용의 FVS 문제가 된다. FVS의 비용은 지연/면적 제한을 반영하는 것이다.

이 문제는 NP-hard하기 때문에, Cheng과 Agrawal은 이 목적을 위한 휴리스틱 알고리듬을 제안하였다.[4] 이 알고리듬은 가장 많은 사이클 상에 놓인 정점을 선택하고, 이를 FVS에 추가하고, 그래프에 사이클이 없어질 때까지 이 과정을 반복한다. 각 정점을 통과하는 사이클 수를 계산하는 것은 상당한 시간적 부담이 된다. 이를 극복하기 위해, 일정한 수의 사이클만이 고려되고, 결국 최종 결과의 정확도가 떨어질 수도 있다. 또한 이 알고리듬은 계산된 최종 결과의 최적화 정도에 대한 아무런 얘기도 해주지 못한다.

본 논문에서는 정확하면서도 대규모 그래프에 대해서도 실용적인 최소 비용의 FVS를 계산하는 알고리듬을 제안한다. 이 알고리듬의 주요 아이디어는 모든 사이클을 직접적으로 고려하지 않는다는 것이다. 이것은 그래프 G에 대한 부울 함수 f(G)를 polynomial한 시간 내에 계산함으로써 이루어진다. 부울 함수 f(G)를 만족하는 값들은 G의 FVS와 일대일로 대응된다. 이 함수는 그래프의 모든 사이클을 간접적으로 고려함으로써 구성된다. 물론 부울 표현식의 크기가 G의 크기에 따라 polynomial할 것이라고 보장을 할 수는 없다. 그러나 논리합성 분야에서 사용하기 위해 개발된 효과적인 부울 표현/처리 기법을 활용하면 이 문제을 쉽게 다룰 수 있다. 사실 f(G)를 위해 이 표현방식을 사용함으로써 최소 비용의 FVS는 표현식 크기에 선형적으로 비례하는 시간 내에 얻을 수 있다.

```
dfs_initialize(G)
{
    foreach vertex u ∈ V {
        status[u] = unvisited;
        function[u] = 0;
    }
    foreach vertex u ∈ V {
        if (status[u] == unvisited)
            dfs_initialize_visit(u);
```

```
    }
}

dfs_initialize_visit(u)
{
    status[u] = visiting;
    foreach edge(u, v) {
        if (status[v] == unvisited)
            dfs_initialize_visit(v);
        if (status[v] == visiting) {
            add_to_set(B, u);
            function[u] = 1;
        }
    }
    status[u] = visited;
}
```

그림 1 초기화 알고리듬

Fig. 1 Initialization algorithm

```
compute(G, B)
{
    for i = 1 to |B| {
        foreach vertex v ∈ V
            status = unvisited;
        foreach vertex v ∈ B
            compute_step(v);
    }
    f = 0;
    foreach vertex v ∈ B {
        f = f ∪ f[v];
    }
}

compute_step(v)
{
    status = visiting;
    g = 0;
    foreach edge(u, v) {
        if (status[u] == unvisited)
            compute_step(u);
        g = g ∪ f[u];
    }
    f[v] = v ∩ g;
    status = visited;
}
```

그림 2 반복계산 알고리듬

Fig. 2 Iteration computation algorithm

본 알고리듬을 설명하기 위해 그림 3 (a)의 예제를 고려해 보자. 그래프의 각 정점에 논리변수를 할당하고 정점의 이름으로 사용한다. 각 정점 u에 대해 논리함수 f(u)를 저장한다. f(u)는 반복적으로 계산된다. 우선, 그림 1의 dfs_initialize(G) 알고리듬에서 기술된 것 처럼, 그래프 G에 대해 깊이우선(depth first) 검색을 함으로써 시작한다. 이 알고리듬은 2개의 귀환 에지, 즉 그래프에서 사이클을 유발시키는 에지를 찾아낸다. 깊이우선 검색에 의해 부과된 귀환 에지와 위치순서를 반영하여 다시 그려진 그래프는 그림 3 (b)와 같다. 정점 c와 d는 집합 B에 추가되는데, B는 모든 귀환 에지를 포함하고 있다. B에 포함된 모든 정점을 제거하면 그래프에 있는 사이클이 모두 제거되는데, 일반적으로 이것은 부분 최적화된 것이다. 초기화 과정에서 f(c) = 1, f(d) = 1로 할당한다, 즉 집합 B에 있는 모든 정점에 대한 논리함수는 상수 1이 된다. 이 초기화 과정이 수행되면, 그림 2의 compute(G, B) 알고리듬에서 그래프에 대한 반복적인 처리 준비가 된 것이다. B에 있는 정점들에 대한 논리함수의 새로운 값들은 이전 값들을 이용해서 계산된다. 이 계산은 재귀적인 알고리듬인 compute_step(v)에서 알 수 있듯이 B의 정점들로부터 시작해서 깊이우선의 역순으로 이루어진다. 따라서 정점 v에 대한 논리 함수는 (u, v) ∈ E인 모든 정점에 대한 논리 함수가 모두 계산된 후에만 계산된다. 한 정점에 대한 논리 함수는 식 1에 의해서 계산된다.

$$f(v) = v \cap (\cup_{(u,v)\in E} f(u)) \qquad \text{(식 1)}$$

f(v)는 v를 통과하는 모든 경로를 간접적으로 저장한다. 식 1은 한 정점을 통과하는 경로의 집합은 입력에지와 이 정점을 통과하는 경로라는 사실을 자연스럽게 표현하고 있다. 그림 3 (c)는 첫 번째 실행 후의 논리함수의 값들을 보여준다. 가장 긴 단순 경로는 B의 모든 정점을 통과할 수 있기 때문에, 그런 모든 경로를 찾아내려면 k번 반복을 해야 한다. 이 때 k는 B에 있는 정점의 개수이다. 실제로 k번 모두 반복되지 않을 수도 있다. 많아야 k 번 반복한 후에, B의 k 개 정점에 대한 함수들은 회로의 모든 사이클에 대한 정보를 담고 있다. f를 이 함수들의 합집합이라고 하자. 어떤 함수도 간단화하지 않았다고 가정하면, f는 k · |E|에 비례한 시간 내에 계산된다, 여기서 k는 V의 요소 개수이며 |E|는 그래프의 에지 수이다. k는 |V|에 의해 제한되므로, f를 생성하기 위한 복잡도는 O(|V| · |E|)이다. 실제로는 O(|E|)에 더 가깝다. 그림 3의 예제에서 f = bcd+bca이다. 이것은 바로 2개의 기본 사이클이 bcd와 bca라는 것을 말해준다. 물론, f를 sum-of-product형식으로 표현했다면, 모든 기본 사이클이 열거될 것이다. 이것이 모든 사이클 집합보다는 다루기가 쉽겠지만 가능하면 피하는 것이 바람직하다.

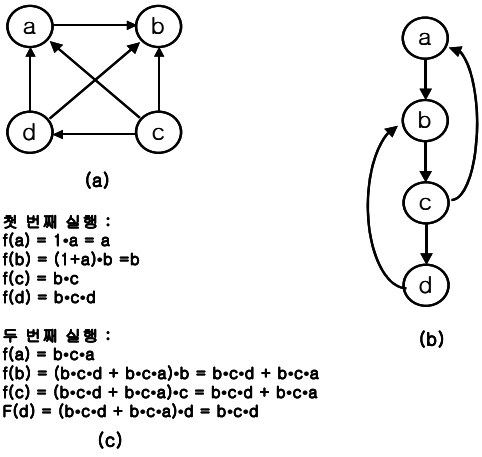

그림 3 예제 그래프

Fig. 3 Example graph

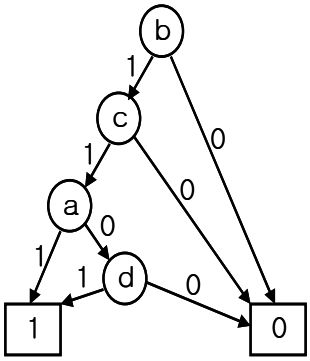

그림 4 예제 BDD
Fig. 4 Example BDD

다행히 BDD(Binary Decision Diagram)는 sum-of-product 형식보다 더 간결하게 논리함수를 표현할 수 있는 방법이다. 본 알고리듬은 이 표현방법을 사용하여 구현되었다.

예제에 대한 함수 f의 BDD는 그림 4와 같다. BDD는 상수 1과 0 값을 갖는 2개의 종단 정점을 갖고 있다. 종단 정점을 제외한 각 정점은 변수로 표현되고 2개의 가지를 갖는다. 1(0) 가지는 그 변수의 값이 1(0)일 때 함수가 어떤 값을 갖게 되는지를 말해준다. 주어진 입력에 따른 함수 값의 계산은 BDD에서 각 정점에 있는 변수의 값에 따라 가지를 따라 내려감으로써 진행된다. 도착한 종단 정점이 주어진 입력에 대한 함수 값이다.

```
min_cost(f)
{
    if (f == 1) {
            f.cost = 1;
            f.assignment = {};
    }
    if (f == 0) {
            f.cost = ∞;
            f.assignment = {};
    }
    if (f->1branch.cost < f->0branch.cost) {
            f.cost = f->1branch.cost;
            f.assignment = f->1branch.assignment;
    } else {
            f.cost = f->0branch.cost + cost(f.variable);
            f.assignment = f->0branch.assignment ∪
                                        f.variable;
        }
}
```

그림 5 BDD를 이용한 최소비용 할당 값
Fig. 5 Minimum cost assignment using BDD

f′으로 표시되는 f의 보수는 유용한 함수인데, f′을 만족시키는 값에 대한 보수값들은 그래프에 대한 FVS에 해당된다. 위의 예제에서 f = bcd+bca이고, f′=(b′+c′+d′) · (b′+c′+a′)이다. 따라서 b′은 f′을 만족시키고, 이는 b를 제거하면 그래프에 있는 모든 사이클을 끊게 된다는 것을 의미한다. 일반적으로 함수를 만족하는 입력값들은 몇가지가 존재하고 따라서 FVS가 몇가지 존재한다. 이중 가장 적은 비용이 요구되는 것을 하나 결정하는 것이 중요하다. 이것은 BDD 표현법에 관련된 대부분의 비용함수에 대해 쉽게 이루어진다. 0 종단 정점에 대해서는 만족시킬 할당값이 없기 때문에 ∞의 비용이 할당된다. 1 종단 정점에는 1의 비용이 할당된다. 이 초기 할당값으로부터 최소 비용 할당값이 어떻게 얻어지는지를 그림 5가 보이고 있다. 여기서 f.variable은 f의 root에 있는 변수를, f.cost는 f에 대한 최소 비용 할당값을, f.assignment는 최소비용 할당값을 가르킨다. cost(variable)는 할당값에서 이 변수를 포함시키는 비용이다. 이 알고리듬은 각 에지를 정확히 한 번씩 통과하기 때문에, BDD의 크기에 선형적으로 비례한다.

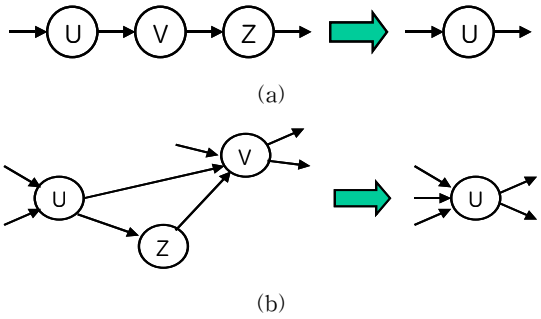

그림 6 체인 결합의 예
Fig. 6 Examples of chain merging

2.1 그래프의 축소

MFVS를 계산하기 위해 기존의 그래프 분할과 그래프 압축 방법을 사용하게 되면 상당한 성능 향상을 얻을 수 있다.[7, 9] 우선 전체 그래프에 대한 MFVS는 부분 그래프에 대한 최소비용 피드백 집합의 합이다. 따라서 부분 그래프들을 개별적으로 고려하여도 최적화에는 별 손실이 없다. 이 사실로부터 결국 큰 그래프가 다루기 쉬운 더 작은 그래프로 쪼개어질 수 있기 때문에 상당한 절감을 할 수 있다는 결론을 얻을 수 있다. 둘째로 단일 입력 정점과 출력 정점을 갖는 정점은 생략될 수 있어서 MFVS 알고리듬은 최적도에 별 손실 없이 압축된 그래프에 적용될 수 있다. 이 과정은 그림 6에서 보인 것 처럼 반복적으로 적용될 수 있다. 그러나 만약 그래프 축소 과정에서 자체 루프가 생겼다면 이 자체 루프에 있는 정점은 어떤 해에도 발생되고 그래프로부터 바로 삭제될 수 있는 문제가 발생할 수도 있다.

2.2 대규모 그래프의 처리

원래의 그래프가 너무 커서 다룰 수 없다면, 플립플롭들을 선택하고 이때마다 그래프 분할과 축소 방법을 적용하여 가장 큰 부분 그래프가 다룰 수 있는 크기로 충분히 분할될 때까지 반복한다. 이를 위한 한가지 방법은 branch-and-bound

알고리듬에 의해 정확히 수행하거나, 휴리스틱하게 초기 플립플롭을 선정한다. branch-and-bound 접근방법에서는 검색을 bound하기 위해서 본 알고리듬이 주요하게 사용된다.

또다른 휴리스틱은 다음과 같다: 귀환 에지 노드들에 대한 표현식의 논리합은 항상 피드백 정점 문제에 대한 해를 찾기에 충분한 정보를 갖고 있다. 실행 중 어느 순간에 메모리 한계상황이 되면, 그때까지 계산된 표현식으로부터 가장 작은 수의 정점을 갖는 완전 해를 선택하거나 해의 일부분을 선택해서 거기서 계속 진행할 수 있다. 이 휴리스틱은 둘다 부분 최적화된 최종 해를 찾아낸다. 두 번째 선택사항으로는 현재 가능한 해 중에서 가장 많은 수의 해에서 발생한 정점들의 부분 해를 선택할 수 있다.

3. 실험결과

본 논문에서 제안한 알고리듬에 대한 실험에서는 ISCAS89 벤치마크 회로[2]를 사용하였다. 이들 중에서 가장 큰 회로는 1500개 이상의 플립플롭을 갖고 있으며, 이를 이용하면 본 알고리듬에 대한 좋은 평가를 할 수 있다. ISCAS89 벤치마크 회로의 몇 회로들은 본 알고리듬이 다루기에 지나칠 정도로 소규모 회로이기 때문에, 이런 회로에 대한 결과는 생략하였으며, 비교적 큰 회로들 중에서 본 알고리듬이 MFVS를 찾을 수 있는 회로들에 대한 실험 결과만을 나타내었다. 특히 s38584와 s15850의 두 회로에 대한 최적 해를 얻는 것은 가능하지 않기 때문에 이에 대한 실험 결과를 나타내지 않았다. 벤치마크 회로에 대한 특성은 표 1과 같다.

표 1 벤치마크 회로 특성

Table 1 Characteristics of benchmark circuits

회로	주입력 수	주출력 수	게이트 수	플립플롭 수
s13207	31	121	7875	669
s1423	17	5	635	74
s35932	35	320	15998	1728
s38417	28	106	22263	1636
s5378	35	49	2195	164
s838	35	2	390	32
s9234	19	22	5556	228
s953	16	23	395	29

표 2에서, 휴리스틱 루프 제거 방법은 Lee와 Reddy[6]가 제안한 휴리스틱 루프 제거 알고리듬을 적용한 결과이다. 또한 표 2에서 FF수란 스캔 플립플롭으로 선택된 플립플롭 수에 해당하고, CPU 시간은 해를 계산하는데 필요한 시간에 해당한다. [6]의 방법에 의해 얻어진 CPU 시간은 일반적으로 매우 작다.

본 접근방법은 루프 제거 알고리듬에 대한 모든 최소 정점을 갖는 모든 해를 제시할 수 있다는 장점이 있다. 이것은 다음의 이유로 인해 매우 유용하다: 한 플립플롭을 스캔하는 상대적인 비용이 여러 가지 요인에 의한다는 것은 잘 알려져 있다. 예를 들어, 한 플립플롭을 스캔하는 것은 그것을 통과하는 경로에 지연을 유발하기 때문에, 두 플립플롭 중에서 하나를 선택해야 한다면, 일반적으로 항상 더 짧은 경로를 갖는 플립플롭을 선택할 것이다. 플립플롭과 관련된 상대적인 비용을 갖는 회로가 주어지면, 본 알고리듬은 가능한 수백만가지의 해들 중에서 최소 비용의 해를 선택할 수 있다. 표 2에서 제공된 자료는 모든 플립플롭에 동일한 비용이 할당된 특별한 경우를 고려한 것이다. 동일하지 않은 비용을 할당한다고 해도 더 복잡해지지는 않는다.

표 2 실험 결과

Table 2 Experimental results

회로	휴리스틱 루프제거 방법에 의한 FF수	본 알고리듬	
		FF 수	CPU 시간(s)
s13207	59	59	31
s1423	21	21	3469
s35932	306	306	91
s38417	374	374	306
s5378	30	30	8
s838	0	0	1
s9234	53	53	1770
s953	5	5	1.5

4. 결 론

본 논문에서는 그래프에서 최소비용 피드백 정점 집합인 MFVS(minimum cost feedback vertex set)를 정확히 계산하는 방법을 제안하였다. 임의의 그래프가 주어지면, 이에대한 부울 함수가 유도되는데 이 함수를 만족하는 값들이 그래프에 대한 MFVS가 된다. 본 알고리듬에서는 부울 함수를 구성하면서 그래프의 모든 사이클을 일일이 직접 고려할 필요가 없다. 이 함수는 MFVS를 결정하기 위해 사용되는데, 최소 비용으로 부울 함수를 만족하는 값들을 계산하는 것이 NP-hard한 문제이기는 하지만, 논리합성 분야에서 널리 사용되는 BDD를 이용하여 비교적 큰 규모의 그래프에 대해서도 효과적으로 이 문제를 해결할 수 있었다. C언어로 구현된 본 알고리듬은 ISCAS89 벤치마크 회로들에 대한 실험을 통하여 기존의 알고리듬에 비하여 개선된 성능을 나타냄을 확인할 수 있었다.

참 고 문 헌

[1] V.Agrawal and K.Cheng, "A complete solution to the partial scan problem," in Proc. Int'l Test Conf., pp.44-51, September 1987.

[2] F.Brglez, D.Bryan, and K.Kozminski, "Combinational Profiles of Sequential Benchmark Circuits," in Proc. Int'l Symp. Circuits and Systems, pp. 1929- 1934, May 1989.

[3] S.T.Chakradhar, A.Balakrishnan, and V.D. Agrawal, "An exact algorithm for selecting partial scan

flip-flops," in Proc. Design Automation Conf., pp.81-86, June 1994.
[4] K-T Cheng and V.D. Agrawal, "An economical scan design for sequential logic test generation," in Proc. Fault Tolerant Computing Symposium, pp.28- 35, June 1989.
[5] M.R. Garey and D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-completeness, W.H. Freeman and Company, 1979.
[6] D. Lee and S.M. Reddy, "On determining scan flip-flops in partial scan designs," in Proc. Int'l Conf., on Computer-Aided Design, pp.322-325, November 1990.
[7] E. Lloyd and M. Soffa, "On locating minimum feedback vertex sets," Journal of Computer and System Science, No.37, pp.292-311, 1988.
[8] S. Park and S.Akers, "A graph theoretic approach to partial scan design by k-cycle elimination," in Proc. Int'l Test Conf., pp.303-311, October 1992.
[9] G. Smith and R. Walford, "The identification of a minimal feedback vertex set of a directed graph," IEEE Trans. on Circuits and Systems, Vol. CAS-22, No.1, January 1975.
[10]E.Trischler, "Incomplete scan design with an automatic test generation methodology," in Proc. Int'l Test Conference, pp.153-162, November 1980.

저 자 소 개

김 윤 흥 (金 倫 弘)

1986년 한양대 전자공학과 졸업, 1988년 한양대학교 대학원 전자공학과 졸업(공학석사), 1992년 동대학원 전자공학과 졸업(공학박사), 1993년 3월 ~ 현재 상명대학교 천안캠퍼스 공과대학 컴퓨터시스템공학과 교수. 2002년 8월 ~ 2003년 7월 미국 Northeastern Univ. 방문교수. 관심분야는 VLSI CAD, Embedded System, RTOS

Tel : 041) 550-5358
Fax : 041) 550-5386
Email : yunkim@smu.ac.kr

신 재 흥 (申 載 興)

1986년 한양대 전자공학과 졸업, 1991년 동대학원 전자공학과 졸업(공학석사), 1997년 동대학원 전자공학과 졸업(공학박사), 1997년 3월 ~ 현재 동서울대학 컴퓨터시스템과 조교수.

Tel : 031) 720 - 2175
Fax : 031) 720 - 2294
E-mail : jhshin@haksan.dsc.ac.kr