論文98-35C-9-1

VLSI 논리회로의 동적 임계경로 선택 알고리듬 (DYSAC)

(Dynamic Critical Path Selection Algorithm (DYSAC) for VLSI Logic Circuits)

金 東 郁 *, 趙 原 逸 *, 金 宗 賢 *

(Dong-Wook Kim, Won-Il Cho, and Jong-Hyeon Kim)

요 약

본 논문에서는 대형 디지털 회로에 대하여 임계경로를 탐색하는 시간을 줄이고 기존의 방법에서 임계경로를 찾지 못했던 회로에서도 정확히 임계경로를 찾을 수 있는 임계경로 탐색 알고리듬(DYSAC)을 제안하였다. 또한 이 탐색 알고리듬의 내부에서 사용되는 경로부각기준(DYPSEC)을 함께 제안하였다. DYSAC는 각 노드에 레벨을 부과하기 위한 레벨지정 부알고리듬과 최장의 부각가능한 경로를 찾는 임계경로 탐색 부알고리듬으로 구성되었다. 제안된 알고리듬은 SUN Sparc 환경에서 C-언어로 구현되어 ISCAS'85 벤치마크회로에 적용, 제안된 알고리듬의 정확한 동작여부를 확인하였다. 또한 실험결과를 기존의 방법들과 비교하였는데, 그 결과 제안된 알고리듬이 임계경로를 찾는 능력과 임계경로를 찾는데 걸리는 시간 모두에서 기존의 방법들보다 월등히 우수함을 보였다.

Abstract

This paper is to propose an algorithm named as DYSAC to find the critical path(the longest sensitizable path) in a large digital circuit, whose purpose is to reduce the time to find critical path and to find critical paths of the circuits for which the previous methods could not find one. Also a set of path sensitization criteria named as DYPSEC is proposed, which is used to select a path from input to the output inside the DYSAC. The DYSAC consists of two sub-algorithms; the level assignment algorithm to assign a level to each node and the critical path selection algorithm to select the sensitizable path. The proposed algorithm was implemented with C-language on SUN Sparc and applied to the ISCAS'85 benchmark circuits to make sure if it works correctly and finds the correct critical path. Also, the results from the experiments were compared to the results from the previous works. The comparison items were the ability to find the critical path and the speed, in both of which the proposed algorithm in this paper shows better results than others.

I. 서 론

최근의 급속한 집적도 증가는 설계단계에서의 검증작업을 더욱더 어렵게 하고 있으나, 그 중요성은 날로 증가하고 있다. 설계단계에서의 검증은 설계사양의 함수적인 조건의 준수여부를 확인하는 것 이외에도 많은 부분이 포함되어 있으며, 이 사양들을 만족할 때까지 설계와 검증단계는 반복하여야 한다[1] [2]. 고집적도와 함께 발전하고 있는 매우 중요한 요소가 회로의 동작속도이고, 고속동작이 수행될수록 샘플링 시간과 관련하여 회로의 지연시간에 의한 논리적 오류의 발생이 증가하고 있다[1] [2].

* 正會員, 光云大學校 電子材料工學科

(Dept. of Electronic Materials Eng., Kwangwoon Uiversity)

接受日字:1998年4月6日, 수정완료일:1998年8月18日

이에 최근 재부각되고 있는 연구분야중 하나가 지연 관련 분야이며, 이것은 보통 주어진 회로의 최장 부각 가능한 경로(the longest sensitizable path)로 정의되는 임계경로(critical path)의 지연시간을 기준으로 수행된다. 즉, 임계경로의 지연시간은 회로의 동작속도를 결정하며, 이 시간에 의해 그 회로의 동작주파수가 결정되는 것이다. 따라서 주어진 회로에서 임계경로를 찾고, 이 임계경로를 부각시키는 입력조합을 구하는 것이 매우 중요한 일이다. 최근의 지연시간에 대한 설계검증 및 테스트 연구의 주류는 바로 이 임계경로를 찾는 것이며, 지금까지 많은 연구가 진행되어 오고 있다[1] [3] [4] [5] [6] [7] [8]. 이들의 대부분은 best first search 방법을 기초로 하고 있는데, 임계경로를 탐색하는 알고리듬 자체가 대상회로의 최장경로 후보들을 추출한 후 그 경로의 부각여부를 판별하는 두 개의 알고리듬으로 구성되어 있으며, 그 전에 예비단계를 거쳐 탐색에 필요한 여러 가지 요소들을 미리 계산한다.

임계경로 선택 알고리듬은 보통 각 방법에 맞는 부각기준을 마련하고 있는데[1] [3] [4] [5] [6] [7] [9], 서로 다른 기준에 의해 기술되어 있으므로 부각기준 자체의 평가는 불가능하고 단지 각 부각기준을 사용한 임계경로 선택 알고리듬의 성능으로 알고리듬 자체와 함께 평가된다. 본 논문의 목적은 주어진 게이트-레벨 회로에 대해 임계경로 탐색의 실패율을 최소로 함과 동시에 임계경로를 탐색하는데 소요되는 재원과 시간을 축소하는 것이다. 기존의 방법은 탐색 알고리듬을 수행하기 전에 많은 양의 선처리과정을 거치게 될 뿐 아니라, 임계경로 탐색에 있어서도 회로의 입력과 출력사이를 여러번 왕복하여야 하는 단점들이 있었다. 본 논문에서는 이런 단점들을 보완하는 DYSAC (DYnamic Search Algorithm for Critical path)이라 명명한 임계경로 탐색 알고리듬을 DYPSEC (DYnamic Path SEnsitization Criteria)이라 명명한 경로부각기준과 함께 제안한다. 제안된 알고리듬은 C-언어로 실현되어 벤치마크 회로들(ISCAS'85)을 대상으로 동작여부를 확인하고, 기존의 방법들과 임계경로의 탐색능력과 탐색시간 등을 비교한다.

Ⅱ. 정의

먼저 본 논문에서 사용될 용어들을 정의하면 다음과 같다.

[정의 1] 논리회로는 게이트들과 게이트들 간의 연결선으로 이루어진 방향성 그래프(directed graph)로 나타낼 수 있고, 이 때 게이트는 정점(vertex)으로, 연결선은 에지(edge)로 각각 나타낼 수 있다. 또한 게이트 G와 연결선 f의 지연시간을 각각 $d(G)$와 $d(f)$로 나타낸다.

[정의 2] 회로상에서 경로(path) P는 게이트와 연결선을 반복하는 시퀀스로 나타낸다. 즉, $P=(G_0,f_0,G_1, f_1, \cdots, G_{m-1}, f_{m-1}, G_m, f_m)$. 여기서 G_0는 주입력(primary input), fm은 주출력(primary output)을 각각 나타낸다. 또한 $Pk=(G_0, f_0, G_1, f_1, \cdots, G_{k-1}, f_{k-1})$을 f_{k-1}까지의 P의 부분경로라 정의한다.

[정의 3] 주어진 경로 P상의 게이트 G_i의 입력중 그 경로상에 있는 입력을 경로입력(on-input; c)이라 하고 경로상에 있지 않은 입력을 비경로입력(off-input 또는 side-input; nc)이라 한다.

[정의 4] 주어진 게이트에 대하여 다른 입력의 값에 무관하게 그 게이트 출력값을 결정하는 입력값을 제어입력값(controlling value)이라 하고, 그 반대의 값을 비제어입력값(noncontrolling value)라 한다.

[정의 5] 입력벡터 v가 분석대상회로에 인가되었을 때 그 응답으로 연결선 f 또는 게이트 G의 출력이 가지는 안정된 논리값을 f 또는 G의 안정값(stable value)이라 하고 $sv(f, \boldsymbol{v})$ 또는 $sv(G, \boldsymbol{v})$로 나타낸다. 또한 f나 G가 안정값을 가지는데 필요한 시간을 안정시간(stable time)이라 하고 $st(f, \boldsymbol{v})$ 또는 $st(G, \boldsymbol{v})$로 각각 나타낸다.

[정의 6] 주어진 회로의 입력에서 게이트 G_i까지의 거리(distance)를 G_i의 레벨이라 정의하고, $level(G_i)$로 나타낸다. 게이트 G_i의 입력이 게이트 K_1, K_2, $\cdots$, K_p의 출력으로 구성되어 있을 때 $level(G_i)$는,

$$level(i) = \text{가중치}(G_i) + \max_j [\, level(K_j) \,] \qquad (1)$$

로 표현된다. 여기서 가중치(Gi)는 필요에 맞는 값을 가질 수 있다. 예를 들면, 각 게이트의 지연시간을 게이트의 가중치로 사용할 수 있고, 단위지연 모델의 경우는 1을 각 게이트의 가중치로 사용하여 각 게이트의 깊이를 계산할 수 있다.

[정의 7] 경로부각을 위해서 일정한 부각기준(X_

CRT)을 사용하는데, 그 기준을 만족하는 입력벡터의 집합을 V(X_CRT)로 표기한다.

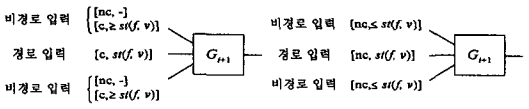
[정의 8] 주어진 회로에 특정 부각기준을 적용하였을 때, 하나 이상의 경로에 대해 그 부각기준을 만족하는 하나 이상의 입력벡터가 존재하면, 그 부각기준은 임계경로 문제에 대해 정확하다고 한다.

Ⅲ. 동적 경로부각 기준

본 장에서는 본 논문에서 제안할 임계경로 탐색 알고리듬에 사용할 경로부각 기준을 제안한다.

1. 발표된 경로부각기준

본 논문의 제안에 앞서 기존에 발표된 기준들 중 지표가 될 만한 기준들을 먼저 소개한다. 현재까지 발표된 부각기준중 가장 정확한 결과를 보이는 것으로 알려진 것이 '정확한 경로부각 기준(exact path sensitization criteria, 차후로는 EXT_CRT로 약칭함)[11]'이다. 이 기준을 그림 1에 나타내었는데, 경로입력이 제어입력인지의 여부에 따라 두 경우로 나누어 정의하고 있다. 먼저 경로입력 f가 비제어입력인 경우 모든 비경로입력 f_i는 $st(f, v) \le st(f_i, v)$를 만족하여야 한다. 경로입력 f가 제어입력이면 비경로입력중 모든 제어입력 f_i는 $st(f, v) \ge st(f_i, v)$의 조건을 만족하거나, 모든 비경로입력을 비제어입력이어야 한다. 이상의 조건을 만족하는 입력벡터 v가 최소 한 개 존재할 때 그 경로를 정확한 부각경로라고 한다. 이 부각기준은 임계경로 문제에 대해 정확하다고 증명되었으며[1], 가장 정확한 경로부각의 결과를 나타내기 때문에 경로부각 기준을 제정할 때 그 지표가 되고 있다.

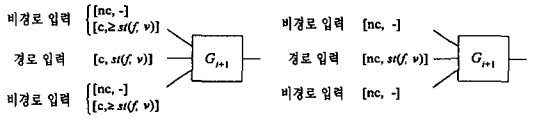


(a) (b)

그림 1. 정확한 경로부각 기준 (EXT_CRT)
(a) 경로입력이 제어입력인 경우
(b) 경로입력이 비제어입력인 경우

Fig. 1. Exact path sensitization criteria.
(a) The case that the on-input is a controlling one (b) The case that the on-input is a noncontrolloing one

입력벡터 v가 경로 $P=(G_0, f_0, G_1, f_1, \cdots, G_{m-1}, f_{m-1}, G_m, f_m)$를 부각시킨다면 G_{i+1}의 경로입력은 제어입력중 가장 빠르거나 비제어입력중 가장 느린 것임에 틀림없다. 만약 경로입력이 비제어입력이라면 경로입력이 가장 느린 비제어입력일 필요는 없으므로 이 조건은 완화될 수 있다[1]. 이 부각조건을 '완화된 경로부각 기준(looser path sensitization criteria, 차후로는 LOS_CRT라 약칭함)'라 하며, 그림 2에 이 부각기준을 나타내었다. 따라서 V(LOS_CRT)$\supseteq V$(EXT_CRT)를 만족한다. 이 기준은 EXT_CRT보다 제한성이 약하기는 하지만 그 결과는 동일하다는 것이 증명되었다[1].



(a) (b)

그림 2. 완화된 경로부각 기준 (a) 경로입력이 제어입력인 경우 (b) 경로입력이 비제어입력일 경우

Fig. 2. Looser path sensitization criteria.
(a) The case when the on-input is a controlling one (b) The case when the on-input is a noncontrolling one

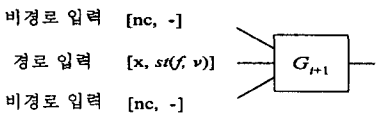


그림 3. 정적 경로부각 기준

Fig. 3. Static path sensitization criteria.

이 두 기준보다 훨씬 제한적인 조건이 비슷한 시기에 발표되었는데[3], 그 이름을 '정적 경로부각 기준(static path sensitization criteria, 향후 STA_CRT라 약칭함)'이라 하며, 그림 3에 나타낸 것과 같이 경로를 부각시키는 조건을 모든 비경로입력이 비제어입력일 때로 제한하고 있다. 따라서 V(STA_CRT) $\subseteq V$(EXT_XRT)를 만족한다. 이상의 세 기준을 비교하면, V(STA_CRT)$\subseteq V$(EXT_CRT)$\subseteq V$(LOS_CRT)를 만족한다.

주어진 게이트에서 입력들의 위치가 그 게이트의 지연을 좌우할 수 있다[10]. 예를 들어, 2-입력 NAND 게이트에서 두 nMOS중 아래에 위치한 MOS가 도통되기 전에는 출력이 논리 0으로 떨어질 수 없다. 따라서 위상이 낮은 입력이 위상이 높은 입력보다 지연시

간에 더 큰 영향을 미친다. 이런 위상을 고려한 부각기준[6]이 발표되었는데, 그림 4에 이 부각기준을 나타내었다. 즉, 게이트 G_{i+1}의 비경로입력중 경로입력보다 상위에 위치한 입력(상위 비경로입력)에는 제약을 두지 않고 하위에 위치한 비경로입력(하위 비경로입력)은 반드시 비제어입력이라야 한다는 제약을 두고 있다. 이 기준에서 경로입력을 제어입력과 비제어입력으로 분할하지 않았고, 앞의 기준들은 상,하위의 구분을 하지 않았으므로 앞의 기준들과의 직접적인 관계를 규명할 수는 없다.

그 외에 여러 가지 부각기준들이 발표되었는데, [4]에서는 각 게이트와 연결선까지의 최장지연시간과 최단지연시간을 계산하여 비제어입력의 신호는 최장지연시간보다 늦게 도달하여야 하고 제어입력의 신호는 최단지연시간보다 빨리 도달하여야 한다고 기준을 정의하고 있다. 또한 이 최장 및 최단시간을 동적인 시간변수로 정의하여 설정한 부각기준[8], 경로입 력을 활성화하기 위한 조건을 기준으로 설정한 부각기준[7] 등이 발표된 바 있다.

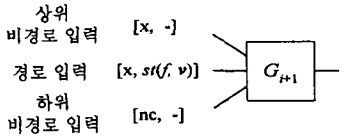


그림 4. 위상을 포함한 경로부각 기준
Fig. 4. A path sensitization criteria considering topology.

2. 동적 경로부각 기준(DYPSEC)

앞 절에서 언급한 LOS_CRT에서 임의의 게이트가 두 개 이상의 입력을 가진 경우 모든 제어입력이 게이트의 출력을 결정하는 부각입력으로 동작한다. 예를 들어, 그림 5의 회로에 v=(0, 0, 0) 입력이 가해졌다고 가정하자. 설명을 간단화하기 위하여 각 게이트의 지연시간을 1, 각 연결선의 지연시간을 0으로 가정하면, EXT_CRT 또는 LOS_CRT는 $P_1=(B,b,G_1,f,G_4,h,G_5,i,G_6,k)$와 $P_2=(C,c,G_2,g,G_4,h,G_5,i,G_6,k)$가 모두를 임계경로로 결정한다. 그러나 본 논문의 성격상 정확한 하나의 임계경로가 필요할 뿐이고, 상위입력보다 하위입력에 제한성이 강하므로, 본 논문에서는 P_2만을 임계경로로 선택한다.

이러한 조건을 만족하는 경로부각 기준을 본 논문에서는 제안하는데, 그림 6에 그 기준을 나타내었다. 이 기준을 '동적 경로부각기준(DYnamic Path SEnsitisation Criteria, DYPSEC)'이라 명명하였으며, 이 기준을 만족하는 경로를 '동적 부각경로'라 한다. 이 기준 역시 EXT_CRT나 LOS_CRT와 같이 경로입력이 제어입력일 때와 비제어입력일 때를 구분하고 있다. 즉, G_{i+1}의 경로입력 f_i가 비제어입력일 때는 모든 비경로입력이 비제어입력이고, f_i가 제어입력일때는 상위 비경로입력 f_u는 $level(f_u, v) \geq level(f_i, v)$를 만족하고 하위 비경로 입력 f_l은 $level(f_l, v) > level(f_i, v)$를 만족하도록 하는 입력벡터 v가 존재하면, 그 경로는 동적 부각경로라고 한다. 앞의 기준들과는 달리 DYPSEC에서는 레벨을 사용하고 있는데, 앞에서 정의한 바와 같이 지연정보를 가지는 어떤 값이든 레벨로 사용할 수 있다.

<정리 1> DYPSEC에서 레벨로 안정값을 사용하면, 임계경로 문제에 대해 정확하다.

(증명) 앞에서 언급한 바와 같이 EXT_CRT와 LOS_CRT는 동일한 결과를 가져오고, EXT_CRT가 임계경로 문제에 대해 정확하므로, LOS_CRT 또한 임계경로 문제에 대해 정확하다. 그림 2와 그림 6을 비교하면 V(LOS_CRT)$\supseteq V$(DYPSEC)임을 확인할 수 있고, DYPSEC와 LOS_CRT는 단지 주어진 게이트 G의 두 개 이상의 입력(여기서는 두 개만을 고려하여도 충분함. 이 두 입력을 f_i, f_j라 함)이 제어입력이고 동일한 레벨을 가질 때만 서로 다름을 알 수 있다 (그림 5 참조).

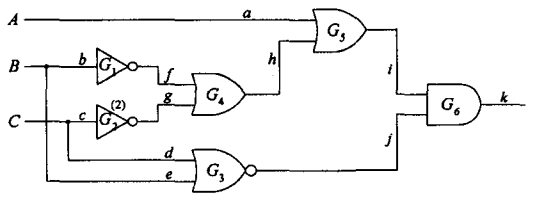


그림 5. v=(0, 0, 0)에 대한 동적 부각경로
Fig. 5. Dynamic sensitizable path under v=(0, 0, 0).

이 경우 LOS_CRT는 f_i, f_j 모두를 경로입력으로 간주하나, DYPSEC에서는 최하위입력(fj)만 경로입력으로 간주한다. G의 입력까지의 부분경로를 고려하면, 이 부분경로의 임계경로는 분명 f_i와 f_j까지의 경로이고, G와 그 후의 경로는 G의 기준에 영향을 받지 않으므로, DYPSEC를 만족하는 경로는 LOS _CRT를 만족하는 경로의 부분집합임에 틀림없다. 또한 LOS_CRT를

만족하는 경로가 임계경로 문제에 대해 정확하므로, DYPSEC를 만족하는 경로는 분명 임계경로 문제에 대해 정확하다.

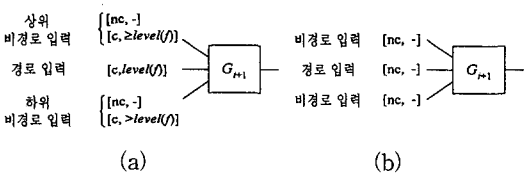


그림 6. 동적 경로부각 기준(DYPSEC)
(a) 경로입력이 제어입력인 경우 (b) 경로입력이 비제어입력인 경우

Fig. 6. Dynamic path sensitization criteria(DYPSEC). (a) The case when on-input is a controlling one (b) The case when on-input is a noncontrolling one

DYPSEC의 만족여부를 판별하는 절차를 그림 7에 나타내었다.

Ⅳ. 동적 임계경로 탐색 알고리듬(DYSAC)

앞에서 언급한 경로부각 기준들과 함께 많은 수의 임계경로 탐색 알고리듬이 제안되었는데[1] [2] [3] [11], 이들의 대부분은 두 단계로 구성되었다. 먼저, 분석대상회로의 최장경로의 목록을 만드는데, 여기에는 depth-first 탐색방법[12]와 breadth-first 탐색방법[13]이 사용되었으나, 최근에는 best-first 탐색(BFS) 방법을 주로 사용하고 있다. 그 다음은 생성된 목록의 경로들에 대해 최장의 경로부터 그 경로가 부각가능한지를 판별하게 된다.

BFS방법(다른 방법도 유사함)은 많은 선처리과정이 필요하고 정확한 결과를 얻기 위해서는 많은 메모리 양을 사용하여야 할 뿐 아니라 경로를 내림차순으로 배열하여야 하는 등의 처리과정을 거쳐야 함으로써, 많은 양의 재원과 CPU시간이 필요하다. 또한 두 번째 단계인 경로의 부각여부를 결정하는 단계에서도 다수의 순방향 추적과 역방향 추적을 수행하여야 하며, 입력벡터에 의존함으로써 상당량의 계산시간이 소요된다. 본 장에서는 한 번의 순방향 추적(레벨 지정 알고리듬)과 역방향 추적(부각경로 탐색 알고리듬) 만으로 임계경로를 찾을 수 있고, 입력벡터에 의존하지 않는 탐색 알고리듬을 제안한다. 이 알고리듬은 그림 8에 나타내었는데, 먼저 분석대상회로(Circuit Under Analysis, CUA)를 [정의 1] 의 방향성 그래프로 변환하고, 레벨 지정 알고리듬(*LevelAssigment()*)과 부각경로 탐색 알고리듬(*SensPathSearch()*)을 차례로 수행한다(그림 8).

```
DYPSEC( )
Inputs : G=F(f, f1, f2, · · · fn);
  // f is on-input, fi(0≤i≤n) are side-inputs //
Output : YES/NO;
begin
  if f has noncontrolling vaule then
    if all fi have noncontrolling value then
  return YES;
    else return NO;
  else if all fi have noncontrolling value then
    return YES;
  else if f has the highest level then
     if an fi with the same level resides in upper side
        then return YES;
    else return NO;
  else return NO;
end
```

그림 7. DYPSEC 판별 절차

Fig. 7. Procedure to check if DYPSEC is satisfied.

1. 레벨 지정 알고리듬

본 논문에서는 임계경로 탐색에 사용되는 각 게이트 및 연결선의 가중치(식(3) 참조)로 여러 가지 가능한 양을 사용할 수 있도록 하기 위해서 안정시간 대신 [정의 6] 의 레벨을 사용한다. 즉, 레벨의 값으로 지연시간을 계산하기 위해서는 안정시간을 사용할 수 있으며, 또 각 게이트의 가중치를 1로, 연결선의 가중치를 0으로 사용하면 레벨은 입력에서 그 지점까지의 깊이(depth)를 나타내게 할 수 있다. 그 외에 필요에 따른 값을 가중치로 사용함으로서 연동적으로 분석의 목적에 부합하도록 할 수 있다.

레벨 지정 알고리듬은 먼저 가상정점 S를 생성시켜 모든 주입력이 S로부터 분기되도록 한다. S를 제외한 모든 정점은 알고리듬이 진행되면서 세 종류로 변환되는데, 여기서는 '흰색', '회색', '검정색'으로 표현하기로 한다. 즉, 모든 정점의 초기값은 '흰색'이고, 알고리듬에 의해 방문되면 '회색'으로 변화하며, 레벨값이 할당되면 '검정색'이 된다. 따라서 알고리듬은 정점 S에서 시작하여 모든 정점이 검정색이 되면 끝난다. 알

고리듬의 진행은, 검정색 정점에서 후행(주출력쪽)정점을 방문하여 식 (1)에 따라 레벨을 할당하고 그 정점을 검정색으로 변환하는 방법으로 진행된다. 후행정점 중 흰색과 회색이 공존하면 회색정점을 먼저 방문하고, 방문된 정점의 선행(주입력쪽)정점이 모두 검정색일 때만 레벨을 지정한다. 이 알고리듬은 그림 9에 나타내었다.

```
DYSAC( )
input : CUA
output : a critical path
begin
Convert CUA into a directed graph, G;
LevelAssignment();
SensPath Search();
end
```

그림 8. 동적 임계경로 탐색 알고리듬
Fig. 8. DYSAC.

```
LevelAssignment()
input : directed graph, G
output : level-assigned graph G
initialization : assign white to all vertices
visit S, assign '0', and convert to black;
for all vertices
begin
  if all the vertices in graph are black then
    return;
  else
    if any successor X is gray then
      visit the gray vertex X;
    else
      visit a white vertex X;
    if all the predecessors of X are black then
      Assign level and convert X to black;
    else
      for all non-black predecessor
        visit, assign level, and convert to black;
        visit X, assign level, and convert to
        black;
end
```

그림 9. 레벨 지정 알고리듬
Fig. 9. Level assignment algorithm.

이 알고리듬의 진행을 설명하기 위해 그림 10에 간단한 한 예제회로를 나타내었으며, 이 회로에 대해 가상정점 S가 추가된 방향성 그래프를 그림 11에 나타내었다. 이 예제에서 설명을 간단히 하기 위해 모든 게이트의 가중치를 1로, 그리고 모든 연결선의 가중치를 0으로 한다. 레벨 지정 알고리듬은 먼저 S의 레벨로 '0'을 할당하고 S를 검정색으로 변환한다. 다음은 후행정점을 방문하는데, 이 때는 모든 후행정점이 흰색이므로 PI1을 먼저 방문한다. PI1의 선행정점이 모두 검정색이고, 식 (1)의 $max_j\,[\,level(K_j)\,]$가 0이므로 PI1의 레벨은 1이 된다. 마찬가지로 PI2와 PI3 모두 레벨이 1로 할당된다. 다음은 PI1에서 후행정점을 방문하게 되는데, 이 때 F, E 모두 그 선행정점이 검정색이 아니므로 PI2를 먼저 검색한다. PI2에서도 B의 선행정점이 모두 검정색이 아니므로 A부터 검색한다. A의 선행정점이 모두 검정색이므로 식 (1)에 의해 레벨값 2가 할당되고 이 정점은 검정색으로 변환된다. 그 후 다시 B를 방문하여 식 (1)에 의해 3이 레벨로 지정되고, 마찬가지로 C가 3으로 레벨이 지정된다. 그 다음은 B의 후행정점 D가 지정되고 E, F, G, H, I, PO1, PO2의 차례로 레벨이 지정된다.

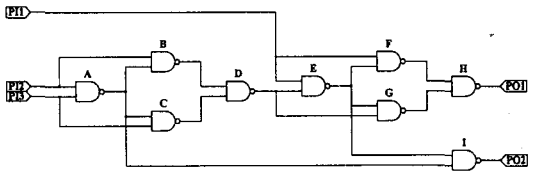


그림 10. 예제회로
Fig. 10. Example circuit.

2. 부각경로 탐색 알고리듬

레벨 지정이 끝난 그래프를 대상으로 그 다음은 경로부각 탐색 알고리듬이 적용된다. 이 알고리듬은 레벨이 가장 큰 주출력으로부터 시작하여 입력측으로 동적 경로부각 기준(DYPSEC)을 만족하는 최고 레벨의 경로를 찾는 것이다.

[정의 9] 게이트 G의 반전패리티 P_G는 게이트 G의 반전 유무를 표시하며, G가 NAND, NOR 또는 NOT의 형태이면 $P_G=1$, 그 외의 행태이면 $P_G=0$이다.

[정의 10] 게이트 G의 출력 논리값이 X로 주어졌을 때 이에 해당하는 G의 한 입력값을 역방향 논리할당값(Reverse logic assignment value, RLA)이라 하며, 그 값은 다음과 같이 계산된다.

$$RLA(G)=X\oplus P_G \tag{4}$$

그래프내의 한 정점 G의 값(해당 게이트의 출력값)이 주어졌을 때 부각경로탐색 알고리듬은 G의 선행정점중 레벨이 가장 높고 위상이 가장 낮은 정점을 방문한다. 그 정점에 미리 지정된 논리값이 없을 경우 [

정의 10] 에 의한 RLA를 그 정점의 논리값으로 지정하고, 만약 미리 지정된 논리값이 있을 경우 그 논리값이 RLA와 같으면 나머지 정점들을 G의 비제어 입력으로 할당하고, 미리 지정된 논리값이 RLA와 다르면 G의 다른 선행정점을 찾아 RLA값을 지정하고, 나머지 할당되지 않은 선행정점에 비제어입력값을 지정한다. 이 알고리듬은 역방향 추적중 처음 주입력의 값을 성공적으로 지정하면 더 이상의 역방향 추적을 중단하고 그 때까지 할당된 값에 따라 결정되는 논리값만 지정하고 끝난다. 이 알고리듬은 그림 12에 나타내었다.

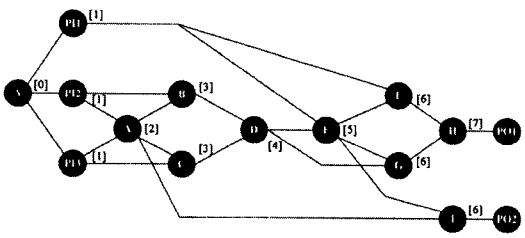


그림 11. 그림 10 회로에 대한 방향성 그래프 및 레벨 지정 알고리듬 수행결과

Fig. 11. The directed graph for the circuit in Fig. 10 and the results from the level assignment algorithm.

앞의 예제로 이 알고리듬을 적용한 예는 그림 13에 나타내었다. 먼저 그림 11의 레벨에 의하면 PO1이 높은 레벨을 가지므로 PO1에서 시작한다. PO1에 "0" 또는 "1" 어떤 값을 할당하여 시작하여도 무방하나, 여기서는 "0"을 할당하는 것으로 한다. G와 F의 레벨이 같으므로 하위의 G에 *RLA(H)*="1"을 할당하고 F에 비제어입력인 "1"을 할당한다. 이 할당으로 부각부분경로는 (*PO1, H, G*)가 된다. G의 두 선행정점 중 E가 높은 레벨을 가지므로 E에 *RLA(G)*="0"을 할당하고 D에 "1"을 할당한다. 이로써, PI1의 "1"의 값을 가지고 D의 다른 팬아웃 또한 "1"을 가진다. 이 값들에 의해서 논리값 할당에 충돌(conflict)이 발생하지 않았으므로 이 할당은 유효하며, 이로써 (*PO1, H, G, E, D*)가 부각부분경로가 된다. D의 두 입력의 레벨이 동일하므로 하위입력인 C에 *RLA(C)*="0"를 할당하고 B에 "1"을 할당한다. 그 다음 C의 두 입력중 A가 높은 레벨을 갖고 있으므로 A에 *RLA(C)*="1"을 할당하고 PI3에"1"을 할당하며, 이로써 부각부분경로는 (*PO1, H, G, E, D, C, A*)가 된다. 여기서, A의 두 입력이 동일한 레벨을 갖고 있으므로 하위 입력인 PI3에 *RLA(A)*="0"을 할당하면 미리 할당된 "1"과 충돌이 발생한다. 따라서 PI3에 "0"을 할당하지 못하고 PI2에 "0"을 할당하여 이 입력을 경로입력으로 한다. 경로입력이 주입력을 만났으므로 알고리듬은 끝나게 되고 최종적인 부각경로는 (*PI2, A, C, D, E, G, H, PO1*)이 되며, 이 경로를 부각시키는 주입력 벡터는 (PI1, PI2, PI3)=(1, 0, 1)이다.

```
SensPathSearch()
input : level-assigned directed graph
output : critical path, corresponding input vector
initialization : assign 'X' to all the edges
begin
  take the highest-leveled output Z as vertex under
  analysis; visit the highest-leveled and topologically
  lowest predecessor Y;
  Logic Assign(Y);
  if Y is a primary input then
    stop;
  else
    if DEC='YES' then
      take Y as Z;
      repeat SensPathSearch();
    else
      take the next highest-leveled output as Z;
      repeat SensPathSearch();
end
LogicAssign()
input : vertex Z and the predecessors Ys
Output : DEC='YES' or 'NO', assigned logic values
begin
  if the highest-leveled Y has logic value 'X' then
    assign RLA(Z) to Y;
  else
    ;
  assign noncontrolling value to all the unspecified Ys;
  if DYPSEC() is satisfied then
    DEC='YES';
    take the on-input as Y;
    return;
  else
    assign controlling value to all the unspecified Ys;
      if DYPSEC is satisfied then
        DEC='YES';
        take the on-input as Y;
        return;
      else
        DEC='NO';
        return;
end
```

그림 12. 부각경로 탐색 알고리듬

Fig. 12. Sensitizable path search algorithm.

즉, 이 알고리듬은 레벨 지정 알고리듬으로 정방향 추적을 한 번 수행하고, 부각경로 탐색 알고리듬으로 역방향 추적을 한번 수행하며, 그 결과 주입력에서 주출력까지의 임계경로와 그 임계경로를 부각시키기 위한 주입력 벡터가 생성된다.

V. 실현 및 결과의 비교

본 논문에서 제안한 알고리듬은 SUN Sparc4 워크스테이션에서 C-언어로 구현되었다. 이 알고리듬의 동작과 비교를 위해서 대상회로로 ISCAS'85의 벤치마크회로를 선택하였다. 또한 본논문에서는 순차회로는 고려하지 않았으며, 순차회로의 경우 메모리 소자들의 출력을 2차 입력으로 사용하고 메모리소자들의 입력을 2차 출력으로 사용하는 주사방법을 적용, 내부의 조합회로의 임계경로를 구할 수 있으리라 사료된다.

본 논문에서 제안한 DYSAC는 두 개의 부알고리듬으로 되어 있고, 그 중 레벨 지정 알고 리듬은 하나의 선처리 과정이라 볼 수 있다. 기존에 발표된 모든 방법에도 이와 유사한 선처리 과정이 있으나, 그 과정에 대한 데이터를 공개하지 않은 관계로 본 논문에서도 이 두 부알고리듬을 분리하였다. 비교항목으로는 기존에 발표된 논문에서 CPU시간만을 고려하고 있으므로 여기서도 CPU시간만을 고려하기로 한다.

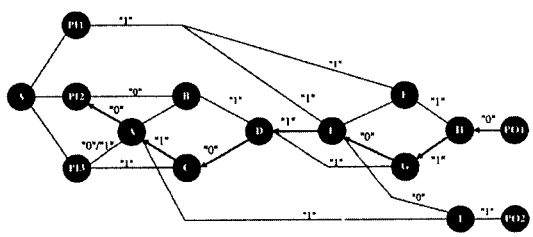


그림 13. 그림 11을 그림 12 알고리듬에 적용한 예
Fig. 13. An example applying Fig. 11 to Fig. 12.

먼저, 선처리 과정인 레벨 지정 알고리듬의 수행결과를 표 1에 나타내었다. 이 표에서 '<1'은 1초 미만의 CPU시간을 의미하며, 모든 CPU시간 측정은 초단위로 수행하였다. 본 논문의 알고리듬중 부각경로 탐색 알고리듬의 수행결과 및 비교결과를 표 2에 나타내었는데, 현재 CPU 시간에 대한 수행결과가 발표된 것은 표 2에 나타낸 두 알고리듬 뿐이어서 이 두 알고리듬과 비교하였다. 이 두 알고리듬의 CPU시간은 발표된 논문[1]의 자료를 인용했으며 이 논문에서는 SUN Sparc 워크스테이션에서 C-언어로 알고리듬을 구현하였다. 표 2에서도 모든 CPU시간은 초단위로 측정하였는데, 그것은 기존의 두 알고리듬이 초단위로 측정하였기 때문이다. 마찬가지로 '<1'은 CPU시간이 1초미만을 나타내고 있다.

표 2에서 볼 수 있듯이, 본 논문에서 제안한 알고리듬의 수행시간이 C432를 제외한 모든 회로에서 월등히 적은 CPU시간을 보이고 있다. C432회로는 Exclusive-OR 회로인데, Exclusive-OR 게이트는 0과 1이 모두 제어입력이나, 본 논문에서는 임의로 0을 제어입력으로 정하여 CPU시간이 다소 길어진 것으로 사료된다. 표 2에서 볼 수 있는 또 하나의 관건은 C6288이다. 이 회로는 그 특성상 매우 많은 거짓경로를 갖고 있어 기존의 두 방법은 이 회로의 임계경로를 찾지 못하여 CPU시간을 측정할 수 없었으나, 본 논문의 방법은 1초의 CPU시간내에 임계경로를 찾는 것으로 나타났다. 또한 C1908의 경우 기존의 방법보다 월등히 우수한 결과를 나타내고 있는데 그 이유는 C1908회로에는 부각이 불가능한 최장경로가 매우 많아서 기존 방법의 경우 임계경로를 찾기 위해 많은 경로 검사가 필요하기 때문이다. 따라서 본 논문에서 제안한 알고리듬은 CPU시간면에서 월등한 개선을 보일 뿐 아니라 임계경로를 찾는 능력면에서도 기존의 방법보다 우수한 것을 알 수 있다.

표 1. 레벨 지정 알고리듬의 결과
Table 1. Results from level assignment algorithm.

대상회로	CPU시간(sec)
C17	<1
C432	<1
C499	<1
C880	1
C1355	<1
C1908	2
C2670	14
C3540	7
C5315	17
C6288	1
C7552	40

표 2. 부각경로 탐색 알고리듬 수행의 결과 및 비교

Table 2. Results from ensitizable path search algorithm and comparison.

대상회로	CPU시간(sec)		
	EXT_CRT 알고리듬	LOS_CRT 알고리듬	DYSAC (본 논문)
C17	<1	<1	<1
C432	<1	<1	1
C499	2	3	<1
C880	<1	<1	<1
C1355	4	4	<1
C1908	660	1427	<1
C2670	<1	6	<1
C3540	32	97	<1
C5315	2	3	1
C6228	-	-	1
C7552	2	21	1

VI. 결 론

본 논문에서는 대형 논리회로에 사용 가능한 임계경로 탐색 알고리듬(DYSAC)을 제안하였다. 또한 이 알고리듬에서 경로의 부각여부를 판별하기 위한 경로 부각기준(DYPSEC)을 같이 제안하였다. 임계경로 탐색 알고리듬은 레벨지정 알고리듬과 부각경로탐색 알고리듬의 두 부알고리듬으로 형성되었으며, 워크스테이션 환경에서 C-언어로 구현되어 실행결과를 기존의 방법과 비교하였다.

비교결과 특별한 경우를 제외한 모든 경우에 있어서 기존의 방법에 비해 알고리듬 수행시간에서 월등히 우수한 결과를 보였으며, 기존의 방법에서 임계경로를 확인하지 못하는 회로의 임계경로를 찾음으로서 임계경로를 탐색하는 능력에서도 월등한 결과를 보였다. 따라서 본 논문의 결과는 현존의 임계경로 탐색 알고리듬보다 훨씬 빠른 시간내에 더 정확한 임계경로를 찾을 수 있어, 설계단계에서 설계된 회로의 성능을 검사하는데 기존의 방법보다 훨씬 효율적으로 사용될 수 있을 것으로 기대된다.

참 고 문 헌

[1] H. C. Chen and D. H. Du, "Path sensitization in critical path problem", IEEE Trans. on Computer-Aided Design, vol. 12, no. 2, pp. 196-207, Feb. 1993.

[2] C. L. Fang and W. W. Jone, "Timing optimization by gate resizing and critical path identification", IEEE Trans. on Computer-Aided Design, vol. 14, no. 2, pp. 201-217, Feb. 1995.

[3] H. Chang and J. A. Abraham, "VIPER: An efficient vigorously sensitizable path extractor", 30th Design Automation Conf., pp. 112-117, 1993.

[4] D. H. Du, et al., "On the general false path problem in timing analysis", 26th Design Automation Conf., pp. 555-560, 1989.

[5] J. Benkoski, et al., "Timing Verification using statically sensitizable paths", IEEE Trans. on Computer-Aided Design, vol. 9, no. 10, pp. 1073-1084, Oct. 1990.

[6] D. Brand and V. Iyengar, "Timing analysis using functional analysis", IBM Tomas J. Watson Research Center Technical Report, 1993.

[7] P. McGeer and R. Brayton, Efficient algorithm for the longest viable pathe in a combinational network, 26th Design Automation Conf., pp. 561-567, 1989.

[8] S. Perremans, et al., "Static timing analysis of dynamically sensitizable paths", 26th Design Automation Conf., pp. 568-573, 1989.

[9] J. P. Roth, "Diagnosis of automata failures: A calculus and a new method", IBM J. of Res. and Dev., pp. 278-281, Oct. 1996.

[10] A. Pierzynska and S. Pilarski, "Pitfalls in Delay Fault Testing" IEEE Trans. on Computer-Aided Design, vol. 16, no. 3, pp. 321-329, March 1997.

[11] H. C. Chen, et al., "Critical path selection for performance optimization", IEEE Trans. on Computer-Aided Design, vol. 12, no. 2, pp. 185-195, Feb. 1993.

[12] K. J. Ousterjout, "A switch-level timing verifier for digital MOS VLSI", IEEE

Trans. on Computer-Aided Design, vol. 4, no. 7, pp. 336-349, July 1985.

[13] R. B. Hitchcock, et al., "Timing analysis of computer hardware", IBM J. of Res. and Dev., vol. 26, pp. 100-105, Jan. 1982.

저 자 소 개

金 東 郁(正會員)

1960년 8월 23일생. 1983년 2월 한양대학교 전자공학과 졸업(공학사). 1985년 2월 한양대학교 대학원 졸업(공학석사). 1991년 9월 Georgia 공과대학 전기공학과 졸업(공학박사). 1992년 3월 ~ 현재 광운대학교 전자재료공학과 부교수. 광운대학교 신기술연구소 연구원. 1997년 12월 ~ 현재 광운대학교 반도체설계 지역센터 운영위원. 주관심분야는 디지털 VLSI Testability, VLSI CAD, 소자 및 회로 모델링

金 宗 賢(正會員)

1973년 4월 22일생. 1997년 2월 광운대학교 전자재료공학과 졸업(공학사). 1997년 3월 ~ 현재 광운대학교 전자재료공학과 석사과정. 주관심분야는 디지털 VLSI Testability, VHDL, VLSI CAD

趙 原 逸(正會員)

1996년 2월 광운대학교 전자재료공학과 졸업(공학사). 1998년 2월 광운대학교 대학원 전자재료공학과 졸업(공학석사). 1998년 2월 ~ 6월 (주) HVD. 1998년 7월 ~ 현재 로직캠프 주임연구원. 1998년 8월 ~ 현재 단국대학교 전기공학과 박사과정 재학중