論文94-31A-6-20

고속 고장 시뮬레이션을 위한 효율적인 병렬 평가 알고리듬

(An Efficient Parallel Evaluation Algorithm for Fast Fault Simulation)

康 敏 燮*

(Min Sup Kang)

要 約

본 논문에서는 조합회로에 있어서 고장 시뮬레이션의 고속화를 위한 효율적인 병렬 평가 알고리듬을 제안한다. 제안한 알고리듬은 고장소자의 평가 및 전파에 있어서 병렬법, 연역법 그리고 동시법의 장점을 이용하고 있기 때문에 시뮬레이션의 고속화를 실현할 수 있을 뿐만 아니라 다치(multi-valued) 신호를 쉽게 취급할 수 있다. 또한, 동일한 신호선에서 발생하는 액티브(active)고장을 동일한 고장 그룹으로 할당하므로써 병렬연산의 효율을 증가시키기 위한 고장의 그룹화(fault grouping) 방법을 제안 한다. 제안한 알고리듬은 C 언어로 구현하였으며, ISCAS '85 Benchmark 회로에 대한 실험 결과 종래의 동시법과 비교하여 약 2.6배에서 8.2배 정도의 고속화가 실현되었다.

Abstract

This paper describes an efficient parallel evaluation algorithm for accelerating fault simulation, which can be applied to combinational circuits. The method is based on a combination of all the advantages in parallel, deductive and concurrent schemes in terms of evaluation and propagation of fautly gates for achieving high performance and handling multi-valued signal. We also propose a new fault grouping procedure to increase parallel operation of fault bits by packing active faults which occur in the same signal line densely into the same fault group. The algorithm has been implemented in C language on a Sun3/260, and experimental results for ISCAS' 85 benchmark circuits have been shown that this algorithm is 2.6 to 8.2 times faster than the conventional cocurrent fault simulation algorithm.

I. 서 론

반도체 설계 및 제조 공정 기술이 급속한 발전으로 인하여 단일 칩안에 수십만개 이상의 소자를 집적화 시키는 일이 가능하게 되어 VLSI(Very Large Scale Integration)의 테스트 문제가 심각하게 대두되고 있다. 고장 시뮬레이터는 VLSI의 테스트 패턴 생성 단계에 있어서 생성된 테스트 패턴의 고장 검출율(fault coverage)을 산출하는데 이용되는 CAD (Computer-Aided Design) 툴이다. VLSI의 고장

* 正會員, 대신大學校 電子計算學科
(Dept. of Computer Science, Taeshin Univ.)
接受日字 : 1993年 8月 7日

시뮬레이션을 위해서는 많은 계산 시간(계산 복잡도는 $O(n) \sim O(n^2)$이며, 여기서 n은 게이트의 수를 나타냄)과 메모리 량을 필요로하기 때문에 시뮬레이션의 고속화 문제가 중요하게 인식되어 이에 대한 연구가 계속 되고 있다. [1]

지금까지 많이 사용되어 왔던 고장 시뮬레이션 기법에는 병렬법[2], 연역법[3], 그리고 동시법[4] 등이 있는데, 이러한 시뮬레이션 기법들은 시뮬레이션 속도와 메모리 사용량에 있어서 trade-off를 가지고 있다. 고속 고장 시뮬레이션이 가능한 PVL(Parallel Value List)기법[5-6]은 신호값의 확장 및 소자의 평가에 있어서 유연성이 결여되어 있다. 이외에도 고장 시뮬레이션의 고속화를 위한 많은 기법[7-8]이 제안되었으나, 이러한 방법들은 다치 신호모델 (multi-valued signal model), 다양한 회로모델(various circuit models) 그리고 시뮬레이션 레벨 (levels of simulation) 등에 있어서 제약을 가지고 있다.

본 논문에서는 많은 고장 비트를 병렬로 연산할 수 있는 효율적인 소자 평가 알고리듬을 기본으로 하여 고장 시뮬레이션의 고속화를 위한 새로운 기법을 제안한다. 또한, 비트의 병렬 연산을 보다 효율적으로 실현하기 위하여 고장의 그룹화(fault grouping) 방법을 제안한다. 정상회로(fault free)를 위한 시뮬레이션 기법은 timing wheel을 이용한 사건 구동(event-driven)방식[10]을 채용하고 있으며, 사용된 고장 모델은 단일 축퇴고장(single stuck-at fault)이다. Ⅱ장에서는 종래의 고장 시뮬레이션 방법에 대해서 간단히 소개하고, Ⅲ장에서는 고장 시뮬레이션의 고속화를 실현하기 위해 사용된 고장 리스트의 데이타 구조를 설명하고, 또한 고장 소자의 평가 및 고장의 그룹화방법을 기술한다. Ⅳ장은 조합회로의 benchmark에 대한 실험 결과 및 고찰을 기술하며, 구현된 시뮬레이터의 평가 및 향후의 연구과제에 대해서는 Ⅴ장에서 기술한다.

Ⅱ. 종래의 고장 시뮬레이션

m개의 신호값에 대해서 고장 시뮬레이션을 수행할 경우 병렬법, 동시법, 그리고 PVL 시뮬레이션의 특징을 살펴보면 다음과 같다.

병렬법에 있어서 신호값은 일반적으로 $\lceil \log_2 m \rceil$ 비트 이상의 2진수로서 코딩한다. 이 방법은 신호값 0, 1, X를 각각 [10], [01], [00] 로 하고, 2워드를 이용해서 하나의 고장 그룹의 신호값으로 표시하고 있다. 이와 같은 방법을 이용하면, 워드 단위의 비트 연산에 의해서 한개의 고장 그룹에 속하는 복수의 고장회로에 대하여 병렬연산이 가능하게 된다. 그러나 각 pass마다 정상회로 및 검출된 고장에 대해서도 시뮬레이션을 해야하는 결점이 있다.

동시법의 경우는 정상값과 다른 신호값(고장값)을 가진 고장회로의 정보를 리스트 구조로 보존하며, 정상회로와 동일한 신호값을 가지고 있는 고장회로는 리스트로 부터 삭제되기 때문에 효율적인 시뮬레이션이 가능하다.

또한 신호값을 보존하는 형식 및 소자평가가 유연하기 때문에 다양한 시뮬레이션 모델을 쉽게 다룰 수 있다. 그러나 각각의 고장회로에 대한 많은 고장 리스트들이 회로의 신호선에 연결되어 있으므로 많은 기억용량을 필요로 한다.

병렬법과 비슷한 데이타 구조를 가지고 있는 PVL법은 고장이 존재하는 고장 그룹만을 도출하여 다수의 셀들을 리스트로 연결하는 방법을 택하고 있다. 이 방법은 병렬법과 동시법의 장점을 함께 취하고 있기 때문에 처리 속도가 빠르며, 메모리 관리도 효율적이다. 그러나 신호값의 수와 시뮬레이션 레벨에 있어서는 병렬법과 동일한 제약을 가지고 있다 제안한 알고리듬에 있어서 고장회로는 고장값에 대응하는 고장비트 표시용 플래그(flag)를 사용하여 병렬로 평가되므로 병렬법과 PVL법의 단점을 개선할 수 있다. 또한 평가된 고장 소자들은 그룹단위로 전파되므로 시뮬레이션의 고속화가 가능하며 동시법의 단점을 개선할 수 있다.

Ⅲ. 고장 시뮬레이션의 고속화

종래의 동시법에 있어서 각각의 신호선에서 발생한 신호값의 전파는 정상회로와 고장회로에 대해서 독립적으로 수행된다. 많은 고장값이 같은 시각에서 변화할 경우, 이것들을 하나의 그룹으로 묶어서 한번에 처리를 하는 것이 알고리듬의 간단화 및 고속화를 위한 효과적인 방법이라 할 수 있다.

본 논문에서 제안하고 있는 기본 알고리듬은 이와 같은 그룹화 방법을 이용하고 있다. 즉 어떤 소자의 출력에서 1개이상의 정상값과 고장값이 변화할 경우 모든 정상값과 고장값을 다음 소자의 입력측으로 전파시킨다. 이때 소자 평가는 항상 모든 정상 입력값과 고장 입력값에 대해서 행하게 되므로 각 고장값에 대응하는 비트에 대해서 병렬 연산이 가능하게 된다.

본 장에서는 고장 시뮬레이션의 고속화를 실현하기 위한 고장 리스트의 데이타 구조 및 고장 비트의 병렬 연산법을 소개하며, 또한 고장의 그룹화를 위한 효율적인 알고리듬을 기술한다.

1. 고장 리스트의 데이타 구조

그림 1 은 고장 리스트의 각 셀에 대한 데이타 구조를 나타낸다. 만약, 사용 계산기의 워드 길이를 N이라 가정한다면, N개(bit 0에서 bit (N-1)까지)의 고장 비트를 병렬로 계산할 수 있다.

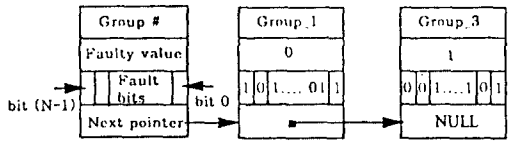


그림 1. 고장 리스트의 셀에 대한 데이타 구조
Fig. 1. Data structure for a cell of fault list.

그림 1 에서 고장 그룹 번호(fault group #)라 함은 1 워드 중의 N개의 비트에 대응하는 N개의 고장 회로의 집합을 말한다. 어떤 고장 그룹에 있어서 적어도 한개의 고장회로가 정상회로와 다른 신호값(고장값(faulty value)이 존재할 때, 그 고장 그룹 및 고장값에 대응하여 1개의 셀을 만들게 된다. 동일 그룹의 고장회로에 대해서 2개이상의 고장값이 존재할 때에는 각각에 대해서 셀을 만든다. 셀 중의 각 고장 비트(fault bits)는 그것에 대응하는 고장회로가 셀 중의 고장값을 가지고 있는지 어떤지를 나타내는데 사용된다. 즉, 한개의 셀 중에서 고장 비트값이 1일 때에는 고장값과 동일한 신호값을, 고장 비트값이 0일 때에는 고장값과 다른 신호값(정상값)을 가지고 있음을 나타낸다. 만약 셀에 있는 모든 고장 비트가 0(비트 0은 정상값과 동일함을 의미함)을 가지고 있다면 그 셀은 리스트로부터 삭제된다.

2. 소자 평가 알고리듬

본 기법에서의 소자 평가는 모든 정상 및 고장 입력값으로 부터 모든 정상 및 고장 출력값을 계산하는 방법을 취하고 있으므로 각 고장값에 대응하는 비트에 대해서 병렬 연산이 가능하게 된다. 그림 2 는 n 입력 1 출력 소자를 평가하기 위한 evaluation 프로시져를 나타낸다. 이 프로시져는 소자의 n개의 입력에 대한 정상값 x, 고장 리스트 L 그리고 소자의 입출력 함수 f를 인수로 하고, 소자의 출력측의 정상값 z와 고장 리스트 M을 리턴한다.

이 프로시져에 있어서 고장 리스트의 셀을 나타내는 레코드의 요소 bits는 고장 비트에 대응한 N개의 비트 벡터이며, 연산자 "∧", "∨" 그리고 "-"는 각각 비트에 대한 논리 화, 논리 적 그리고 논리 부정을 나타낸다. 프로시져 evaluation은 입력측에 있는 고장 리스트의 고장 그룹 번호를 오름차순으로 검색하고, 출력측의 고장 리스트를 동일한 순서로 생성한다. 따라서, 고장 리스트의 셀은 항상 오름차순으로

```
Procedure evaluation(x, L, f)
begin
 {comment: x=(x1,...,xn) is a vector
  of fault-free values for inputs.
  L=(L1,.....,Ln) is a vector of fault
  signal for inputs, f is a function
  from input values to an output value}
  z ← f(x);
  initialize falut signal list M to empty;
  for every falut group number k in L
      in increasing order do
      begin
        for every i (1 ≤ i ≤ n) do
          begin
            Ci ← {s|s ∈ Li, s.group=k};
            b ← 0;
            for each s ∈ Ci do
              b ← b ∨ s.bits;
            t.group ← k; t.value ← xi;
            t.bits ← b̄;
              Ci+ ← Ci ∪ {t}
          end;
        for each c=(c1,....,cn) such that
            ∀i,ci ∈ Ci+; ∃i, ci ∈ Ci(1 ≤
            i ≤ n) do
          begin
            z̄ ← f(c1.value,....,cn.value);
            b ← c1.bits;
            for every i (2 ≤ i ≤ n ) do
                b ← b ∧ ci.bits;
            if z̄ ≠ z and b ≠ 0 then
              if there is a cell r ∈ M
                  such thatr.group=k
                  and r.value=z̄ then
                    r.bits ← r.bits ∨ b
                else
                  begin
                    add a new cell r to M;
                    r.group ← k;
                    r.value ← z̄;
                    r.bits ← b
                  end
          end
      end;
  return z, M
end
```

그림 2. 소자평가를 위한 프로시져
Fig. 2. Procedure evaluation.

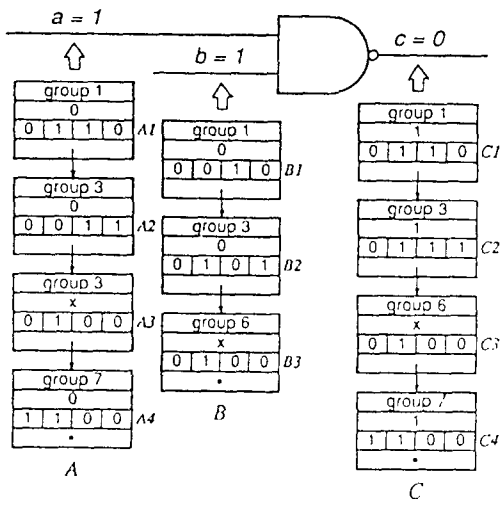


그림 3. 소자 평가의 예
Fig. 3. An example of gate evaluation.

정렬되어 있고, 프로시져 전체의 계산시간은 고장 리스트에 포함되어 있는 셀의 수에 비례한다.

그림 3 은 2 입력 NAND 게이트에 대한 소자평가 예를 나타낸다. 여기서는 연산의 간단화를 위하여 1 워드를 4비트로 하고, NAND 게이트의 입력 신호선 a, b의 정상값을 각각 1이라고 가정한다. 지금 입력측의 고장 리스트 A, B 로부터 출력측의 고장 리스트 C 를 생성하는 경우에 대해서 생각한다. 셀은 그룹 번호의 오름차 순으로 정렬되어 있으며, 셀에 대한 비트 연산처리는 그룹번호 순으로 수행하게 된다. 이 경우에 있어서 최초로 계산을 수행하는 그룹은 그룹 1이 된다.

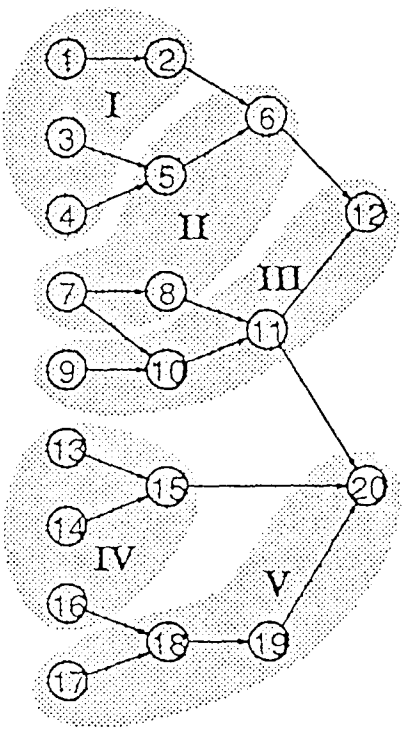
우선 a의 고장값 0과 b의 정상값 1에 대해서 계산을 행하여 리스트 C의 그룹 1을 위한 셀을 만든다. 이에 대한 c의 신호값은 1, 고장 비트는 $A1 \wedge \overline{B1}$ = [0100] 이 된다. 다음에 a의 정상값 1과 b의 고장값 0에 대해서는 고장 비트가 $\overline{A1} \wedge B1$ = [0000] 이 되기 때문에 리스트 C에 영향을 주지 않는다. 마지막으로 a의 고장값 0과 b의 고장값 0에 의해서 c의 신호값은 1, 고장비트는 $A1 \wedge B1$ = [0010] 을 얻는다. 동일한 그룹에 있는 동일 고장값에 대해서는 고장 비트의 논리 화에 의해서 1개의 셀로 묶어서 최종적으로 C의 그룹 1 과 고장값 1에 대한 고장 비트 C1 = [0110] 을 얻는다.

이하의 경우에 대해서도 동일한 방법으로 그룹 3, 6, 7의 순으로 처리를 행하여 출력측의 고장 리스트 C를 만든다. 그룹 3 에서는 리스트 A중에 신호값 0의 셀과 X의 셀이 존재하기 때문에 (a, b) = (0, 0), (0, 1), (1, 0), (X, 0), (X, 1)의 각각의 조합에 대해서 계산을 해야한다. 그룹 6 과 7 은 각각 한 개의 리스트만 가지고 있으므로 각각 (a, b) = (1, X) 와 (a, b) = (0, 1) 에 대해서만 계산하면 된다.

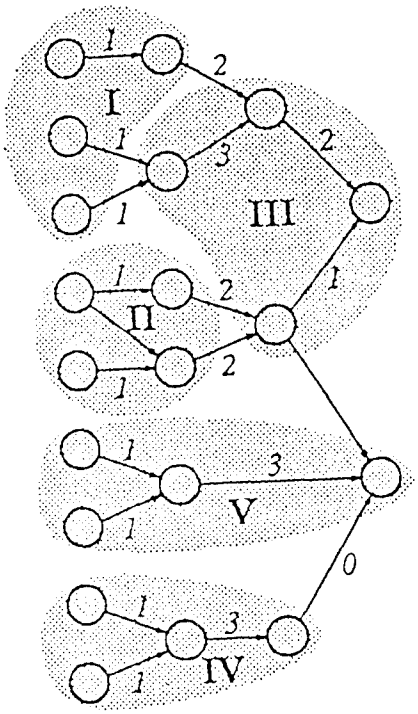
3. 고장의 그룹화

고장 비트에 대한 연산속도를 향상시키기 위하여 종래의 알고리듬들은 깊이 우선 검색(depth first search)법을 사용하여 왔다.[11] 이 방법은 출력측의 한점을 출발점으로 하여 깊이 우선 검색을 행하고 검색했던 순서대로 징점(고장 비트) 들을 그룹화한다 (그림 4(a)). 여기에서는 20개의 정점(vertex)들을 4개씩 묶어서 그룹화 하는 경우를 나타낸다. 정점에 표시된 숫자는 검색하는 순서를 나타내며, 로마숫자 I, II, … 는 그룹화를 해나가는 순서를 나타낸다.

본 논문에서는 고장 리스트 내의 각각의 셀에 대해서 동일한 신호선에서 값 1 을 가진 고장 비트(이하 액티브 비트라 함)들을 동일한 고장 그룹에 할당하여 고장 비트의 연산을 병렬로 처리할 수 있는 효율적인 그룹화 방법을 제안한다. 그림 4(b)는 제안하는 개선된 깊이 우선 검색(modified depth first search) 법을 나타내며, 그룹화를 위한 기본 개념은동일한 그룹에 포함된 2점간의 그래프(graph)에 있어서 거리



(a)



(b)

그림 4. 그룹화 알고리듬
(a) 깊이 우선 검색
(b) 개선된 깊이 우선 검색

Fig. 4. Grouping algorithms.
(a) Depth first search.
(b) Modified depth first search.

가 평균적으로 작도록 그룹화를 행하는 것이다. 여기서 각각의 기(edge)에 나타난 숫자는 알고리듬의 리턴 값(return value)을 나타낸다. 그림 4(a)와 비교할때 제안한 방법은 거리가 먼 정점은 동일한 그룹으로 할당하는 일이 거의 없기 때문에 보다 효율적으로 그룹화를 행하므로서 비트 연산의 고속화가 가능하게 된다.

다음은 간단한 논리회로의 그룹화 과정을 나타낸다. 논리회로 C와 고장집합 F가 주어졌을 경우 C내의 게이트 타입과 그 접속 정보를 이용해서 F를 이하의 프로시저에 의해 N개의 요소로 부터 구성하는 부분 집합으로 분할하고, 우선 회로 C 에 있어서 고장간의 관계는 각 고장을 정점으로 표현하는 다음과 같은 방향성 그래프 G = (V, E) 로 표현한다.

1. C에서 시뮬레이션을 위한 각 신호선의 0 및 1 축퇴 고장은 G의 정점과 1 대 1 로 대응한다.
2. G의 정점 v_1으로 부터 정점 v_2에 기가 연결되어 있는 것은 다음 조건을 완전히 만족할 때에 한한다.
   (a) C에 있어서 고장 f_1(정점 v_1에 대응)이 존재하는 신호선 a로 부터 고장 f_2(정점 v_2에 대응)가 존재하는 신호선 b까지 신호의 전파 경로가 존재한다.
   (b) 고장 f_1 의 영향이 신호선 b로 전파할때, b의 고장값은 고장 f_2의 축퇴값 과 동일한 값이 된다.
   (c) 고장 f_1 의 영향이 신호선 a에서부터 b로 전파하는 경로상의 a와 b를 제 외한 각 신호선에 있어서 f_1 의 영향에 의한 고장값과 같은 값을 가진 고장은 존재하지 않는다.

예를 들어 그림 5와 같은 간단한 논리회로에 대해서 생각한다. 그림 중에 나타난 숫자 0과 1은 각각 신호선에 있어서 0과 1 축퇴 고장을 나타낸다. 이들 고장간의 관계는 그림 6과 같은 방향성 그래프로서 표현할 수 있다. 정점 v_1과 정점 v_2 사이에 패스(path)가 존재한다는 것은 고장 f_1이 고장 f_2가 존재하는 신호선으로 전파되었음을 의미하며, 그들은 동시에 액티브 비트가 될 가능성이 있다.

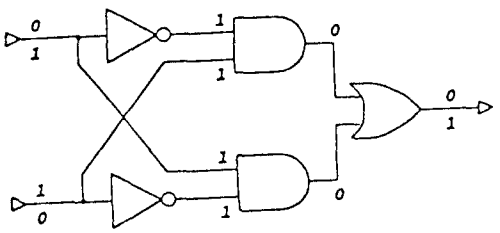


그림 5. 예제 회로
Fig. 5. An example circuit.

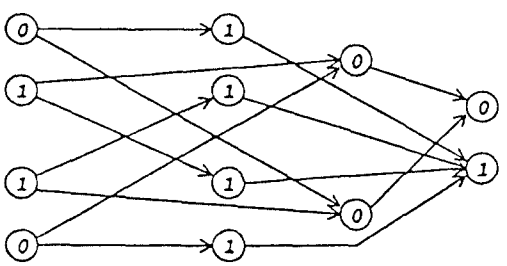


그림 6. 그림 5에 대한 방향성 그래프 G
Fig. 6. Directed graph G for Fig. 5.

```
Procedure  enumerate(v,n):
  begin
    {comment: enumerate at most n
     vertices which have paths to the
     vertex v, returnt he number of
     remaining vertices}
     if n = 0 then return 0;
     mark v as "enumerated";
     for each vertex u which has an
         edge to v and is not marked
         as "enumerated" do
          n ← enumerate(u, n);
     if n = 0 then
        remove "enumerated" mark
        from v
     else
       begin
          output v;
          n ← n - 1
       end;
     return n
  end

Procedure group(v):
  begin
    {comment: make groups of N
     vertices which have paths to
     the vertex v, return the
     number of remaining vertices}
     k ← 1;
     mark v as "counted";
     for each vertex u which has an
         edge to v and is not marked
         as "counted" do
           k ← k + group(u);
     m← k mod N;
     n← k - m;
     if n ≥ N then
          enumerate(v, n);
     return m
  end

Procedure grouping:
  begin
    initialize all vertices to "not marked";
    for each vertex v for the fault at
        the output line do
            group(v);
    for each vertex v for the fault at
        the output line do
            enumerate(v, n);
  end
```

그림 7. 프로시져 그룹핑

Fig. 7. Procedure grouping.

그림 7은 고장의 그룹화를 위한 프로시져를 나타낸다. 프로시져 grouping은 모든 정점을 "not marked"로 초기화 시킨 다음 출력선으로 부터 고장을 나타내는 모든 정점에 대해서 함수 group() 와 enumurate()를 호출한다. 함수 group()는 정점 v와 path를 가진 N 개의 정점에 대해서 그룹화를 행하고, 계속하여 그룹화를 수행할 남아있는 정점의 수를 반환한다. 함수 enumurate()는 정점 v와 directed path를 가진 n 개의 정점만을 출력하고, 남아있는 정점의 수를 반환한다. 프로시져 grouping은 그래프 G의 각 정점 및 기를 각각 2회씩 검색하기 때문에 프로시져 전체의 계산 시간은 O(|V|+|E|)이므로 회로의 소자 수에 비례한 시간으로 그룹화가 가능하게 된다. 이 알고리듬은 양방향 전송 게이트를 포함한 회로까지 쉽게 확장할 수 있다.[12]

Ⅳ. 실험 결과

제안한 알고리듬은 SUN3/260 (1워드는 32비트) 워크스테이션 상에서 C언어를 이용하여 구현하였다. 표 1 은 ISCAS'85[13] 에서 소개된 조합회로의 특성이며, 표 2 는 이회로를 이용한 실험 결과를 나타낸다. 본 실험에 이용된 입력 테스트 벡터는 랜덤하게 생성되었으며, 모든 회로에 대해서 100패턴이 사용되었다.

표 2 에 있어서 비트 병렬도(degree of parallel)는 비트의 병렬 연산을 수행한 경우에 있어서 1워드 중에 값 1을 가진 고장 비트의 수(1~32비트)의 평균치를 나타낸다. 본 논문에서 제안하고 있는 Pro1과 Pro2는 각각 고장의 그룹화를 고려한 경우와 고려하

표 1. 실험에 사용된 회로의 특성

Table. 1. Characteristics of the circuits.

Circuit Name	total gates	total lines	input lines	output lines	fanout stems	faults
c432	160	432	36	7	89	524
c499	202	499	41	32	59	758
c880	383	880	60	26	125	942
c1355	546	1355	41	32	259	1574
c1908	880	1908	33	25	385	1879
c2670	1193	2670	233	140	454	2747
c3540	1169	3540	50	22	579	3428
c5315	2307	5315	178	123	806	5350
c6288	2406	8288	32	32	1454	7744
c7552	3521	7552	207	108	1300	7550

표 2. 실험결과(Sun 3/260 상에서)
Table. 2. Experimental results on a Sun 3/260.

Circuit Name	used mem.(kb)	degree of parallel	CPU time(sec.)		
			Pro1	Pro2	Cont[14]
c432	505	2.50	7.8	9.3	64.1
c499	603	4.11	9.8	15.1	47.8
c880	808	2.29	13.0	16.7	53.2
c1355	924	4.16	43.6	74.1	113.2
c1908	976	4.71	67.4	129.4	175.2
c2670	2564	2.88	52.6	78.9	192.5
c3540	1806	3.07	197.8	336.3	472.7
c5315	2925	3.10	156.4	262.8	655.1
c6288	4306	5.44	3304.9	10046.9	18599.0
c7552	4002	3.38	263.5	495.4	647.3

* Random 100 patterns are used for all circuits

지 않는 경우를 나타내며, Cont[14]는 종래의 동시법을 사용한 구조적(architectural) 레벨의 시뮬레이터를 나타낸다.

Pro2 와 Cont의 비교에서 처리 속도는 Pro2가 약 1.3에서 6.9배 정도 빠르며, 다른 회로에 비해서 event 수가 특히 많은 c6288의 경우는 Pro2가 약2배 정도 빠르다. 그러나 Pro1과 Cont의 경우는 Pro1이 약 2.6배 에서 8.2배 정도의 고속화가 실현되었고 c6288의 경우 Pro1이 약 5.6배 정도 빠르다.

비트의 병렬연산에 의해 최대 3배 정도의 고속화가 실현되었고, 메모리 사용량은 최대 1/3 정도였다. 이미 검출된 고장에 대해서는 시뮬레이션을 수행하지 않는 방법인 fault dropping[13] 기법을 채용하면 시뮬레이션의 속도를 향상시킬 수 있지만, 본 연구에서는 이 기법을 고려하지 않았다.

V. 결 론

본 논문에서는 조합회로에 있어서 고장 비트의 병렬연산에 의한 고장 시뮬레이션의 고속화 기법을 제안하였다. 제안한 기법은 고장소자의 평가 및 전파에 있어서 병렬법, 연역법 그리고 동시법의 장점을 이용하고 있기 때문에 시뮬레이션의 고속화는 물론 메모리 사용량도 줄일 수 있었다.

ISCAS-85 회로에 대한 실험결과 종래의 동시법과 비교하여 약 2.6배 에서 8.2배 정도의 고속화가 실현되었다. 실험 결과에서 알 수 있는 바과 같이 그룹화를 고려한 소자 평가가 시뮬레이션의 고속화에 크게 기여하였다.

또한 기존의 알고리듬[5-6]과 비교하여 신호값의 확장과 소자의 평가에 있어서 유연성이 높은 특징을 가지고 있다.[15] 또한, 그룹화 방법을 고려하여 소자평가의 효율을 향상시키므로써 한층 고속화를 실현할 수 있었다. 앞으로의 연구 과제는 시뮬레이션 시간을 줄이기 위한 효율적인 fault dropping 기법의 개발 및 순서회로나 트랜지스터 레벨에서도 시뮬레이션이 가능한 알고리듬의 개발 등을 포함하고 있다.

參 考 文 獻

[1] D. Harel, "Is There Hope for Linear Time Fault Simulation?," in Proc FTCS-87, pp. 28-33, 1987.

[2] S. Seshu, "On An Improved Diagnosis Program," IEEE Trans. on Comput., vol. 14, pp. 76-79, Feb. 1965.

[3] D. B. Armstrong, "A Deductive Method for Simulation Faults in Digital Logic Circuits," IEEE Trans. on Comput., vol. 7, pp. 39-44, Apr. 1974.

[4] E. G. Ulich,and T. Baker, "Concurrent Simulation of Nearly Identical Digital Networks," in Proc. 10th Design Automat. Workshop. vol. 6, pp. 145-150, June 1973.

[5] K. Son, "Fault Simulation with The Parallel Value List Algorithm," VLSI Systems Design, pp. 229-233, Dec. 1985.

[6] P. R. Moorby, "Fault Simulation Using Parallel Value Lists," IEEE Int'l Conf. on Computer-Aided Design, Santa Clara, CA., 1983.

[7] S. Hong, "Fault Simulation Strategy for Combinationl Logic Networks," in Proc.F TCS-8, pp. 96-99, 1978.

[8] K. J. Antreich, and M. H. Schulz, "Accelerated Fault Simulation and Fault Grading in Combinational Circuits," IEEE Trans. on Computer Aided Design, vol. 6, pp. 704-712, Sep. 1987.

[9] S.Teshima, N.Chujo, et al., "Accelarated Fault Simulation by Prop-

agating Disjoint Fault-Sets," in Proc. FTCS-18, pp. 116-121, 1988.

[10] 강민섭, 이철동, 유영욱, "LOSIM: VLSI의 설계검증을 위한 논리 시뮬레이션 프로그램," 전자공학회 논문지, vol.26, no.5, pp.76-84, 1989년 5월.

[11] T. M. Niermann, W. T. Cheng, and J. H.Patel, "PROOFS: A Fast, Memory Efficient Sequential Fault Simulator," in Proc.27th Design Automat Conf., pp. 535-540, 1990.

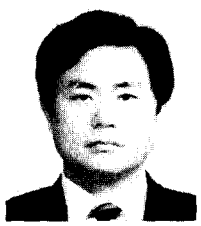
[12] M. S. Kang, "A Study on Logic Simulation and Fault analysis for VLSI Circuits." Ph. D. Dissertation, Osaka University, Dept. of Electronic Engineering, Feb. 1992.

[13] F. Brglez, and H. Fujiwara, "A Neutral Netlist of 10 Combinational Benchmark Circuits and A Target Translator in FORTRAN," IEEE Int'l Symp.Circuits and System, June, 1985.

[14] S. Davison, and J. L. Lewandowski, "ESIM/AFS - A Concurrent Architectural Level Fault Simulator," in Proc. Int'l Test Conf., pp.663-668, 1985.

[15] M. S. Kang, H. Iwashita, H. Deguchi, and I. Shirakawa, "An Extended Digital Fault Simulator for VLSI Circuirs," IEICE Trans., vol. E-74, pp. 3051-3056, 1991.

著者紹介



康 敏 燮(正會員)

1956年 出生. 1979年 광운대학교 전자통신공학과(공학사). 1984年 한양대학교 전자공학과(공학석사). 1992年 일본 Osaka 대학교 전자공학과(공학박사). 1986年 7月 ~ 1987年 6月 Osakaeo대학교 객원연구원. 1984年 5月 ~ 1993年 2月 한국전자통신연구소 선임연구원. 1993年 2月 ~ 현재 대신대학교 전자계산학과 전임강사. 주관심분야는 VLSI 설계 및 Test, Parallel processing 등임.