Domino CMOS 회로의 고장 시뮬레이터

°박동규, 이주희, 이홍주, 임인칠
한 양 대 학 교

Falut Simulator for Domino CMOS Circuits

°D.G.Park, J.H.Lee, H.J.Lee, I.C.Lim
Hanyang University

ABSTRACT

This paper proposes fault simulation algorithms for Domino CMOS circuits.

The inputs having fanouts are described correctly in the algorithms by modeling the functional block in the Domino CMOS circuits as Modified dependency matrix.

The proposed algorithms generate easily the test sequence which can detect the s-a-0, s-a-1, stuckopen faults in the Domino CMOS circuits.

Ⅰ. 서 론

VLSI 회로의 집적도가 증가함에 따라 VLSI의 구성소자로 비교적 낮은 전력소모와 고집적도의 성질을 가진 CMOS회로의 사용범위가 점차 확대되고 있다.

특히 Domino CMOS와 NORA[6,7]등과 같은 다이나믹 CMOS(dynamic CMOS)는 기존의 CMOS보다 칩면적이 작고 동작속도도 1.5내지 2배 정도 빠른 장점을 갖는다.

Domino CMOS회로에는 기존의 고장형태인 s-a-0와 s-a-1고장 이외에 Stuck-open(이하 s-op)고장이 존재하며, 이러한 고장이 발생할 경우 CMOS회로의 Charge store 기능으로 인하여 전상태 값을 그대로 갖게되므로, 조합논리회로가 순서논리회로와 같은 동작을 행한다. 따라서 이와 같은 s-op고장의 검출은 하나의 테스트 패턴으로는 불가능하고 두개의 테스트 시이퀀스가 필요하게 된다.[1,2,3,4]

R.Chadramouli[3]는 경로 활성화 방법을 이용하여 s-op 고장을 검출하는 방법을 제안하였고, 또한 Z.G.Vranesic[1,2] 등은 연결 그래프를 이용한 CMOS 회로의 테스트 패턴 생성방법을 제안하였다.

그러나 경로 활성화 방법은 단일경로 활성화로 고장검출이 불가능한 트랜지스터들에 대하여 다중경로 활성화로 다시 테스트 패턴을 구해야하므로 알고리즘 수행이 복잡하며, 연결그래프는 대규모 회로에서는 적용이 곤란하다.

한편 El-zig와 Su[8]는 MOS complex gate network의 Stuck-at 고장을 검출하기 위하여 basic cell로 모델링하여 테스트하는 방식을 제안하였으나, 이 알고리즘은 MOS complex gate network을 basic cell로 partitioning해야하는 단점이 있다.

본 논문에서는 이러한 점을 해결하기 위하여 modified dependency matrix로 Domino CMOS회로의 함수블럭을 모델링하므로써 Domino CMOS회로의 함수블럭내에 fanout을 가진 입력을 정확하게 기술하는 방법을 제안한다. 또한 Domino CMOS회로의 s-op고장 및 s-a-0, s-a-1고장을 검출할 수 있는 테스트 시이퀀스를 생성하는 고장 시뮬레이션 알고리즘을 제안한다.

Ⅱ. 고장 모델과 Modified dependency Matrix

Domino CMOS회로는 그림 1과 같이 함수 형성을 위한 NMOS 함수블럭과 클럭킹 게이트 그리고 인버어터로 구성되어 있다.

이러한 구조로 인하여 하나의 클럭 펄스에 의해 함수값을 얻을 수 있으므로 타이밍 문제에 있어서 다른 다이나믹회로보다 안정하다는 잇점이 있다. 이 기본 구조에서 클럭 펄스 C가 0일때는 Cp가 on이 되어 출력 F가 0가 되고, 클럭 펄스 C가 1일때는 Cn이 on이 되어 함수블럭과 인버어트에 의해 출력이 결정된다.

Domino CMOS회로내에 존재하는 고장은 NMOS 함수블럭내에 트랜지스터들의 s-a-0, s-a-1, node break 고장 및 클럭킹 게이트, 인버어터의 s-op 고장들이다.

* 과학기술처 연구비에 의한 다목적 공동설계 연구 (MPC 86) 개발 결과임

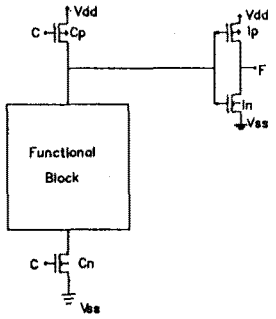

그림 1. Domino CMOS 회로의 기본구조

Fig.1. Basic structure of Domino CMOS circuit.

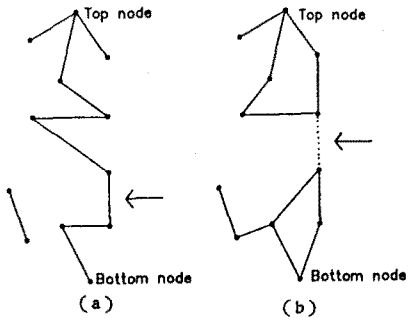

그림 2. 임의의 입력 T에 대한 NMOS 함수블럭의 연결 그래프

Fig.2. connection graph of NMOS function block about arbitrary input T.

그림 2는 그림 1과 같은 Domino CMOS 회로의 NMOS 함수블럭에 인가되는 임의의 입력 T에 의해서 표현되는 연결 그래프이다.

그림 2.(a)의 그래프는 인가된 입력 T에 의해 실제 Domino CMOS 회로의 NMOS 함수블럭에서 전류가 discharge 되어질 경로가 형성되어 출력값이 1이 됨을 알 수 있다.

그런데 화살표로 표시된 edge로 나타내지는 트랜지스터에서 s-a-0 고장이 발생되면 실제 Domino CMOS 회로의 NMOS 함수블럭내에 경로가 형성되지 않으므로 출력 값이 0이 되게된다.

그러므로 그림 2.(a)의 그래프에서 가정된 s-a-0 고장은 고장검출이 가능하고 node break도 같은 개념으로 고장검출이 가능하다.

그림 2.(b)의 그래프는 인가된 입력 T에 의해 실제 Domino CMOS 회로의 NMOS 함수블럭에서 전류가 discharge 되어질 경로가 끊어져서 출력값이 1이 됨을 알 수 있다.

그런데 만약 점선으로 표시된 트랜지스터에서 s-a-1 고장이 발생되면 실제 Domino CMOS 회로의 CMOS 함수블럭에서 전류가 discharge 되어질 경로가 활성화되어서 출력값이 1이 된다.

그러므로 그림 2.(b)에서 가정된 s-a-1 고장의 검출이 가능하다.

일반 CMOS 회로에서 트랜지스터의 s-op 고장은 출력 F를 0으로 만드는 테스트 패턴 T0 들과 출력 F를 1로 만드는 테스트 패턴 T1 들을 교대로 배열한 테스트 시이퀀스에 의해 검출된다.

일반 CMOS 회로에서는 대개 T0가 여러개이나, Domino CMOS 회로는 그림 1과 같은 회로자체의 구조적 특성때문에 테스트 패턴 T0는 클럭 C가 0이고 함수블럭으로 인가되는 모든 입력이 0인 경우의 테스트 패턴이 된다. 그리고 T1 들은 Domino CMOS 회로의 NMOS 함수블럭에서 오직 한개의 경로만을 활성화시키는 입력들로서 이때 활성화되는 경로상에 놓인 트랜지스터들은 고장검출이 가능하다.

그래서 Domino CMOS 회로에서 트랜지스터의 s-op 고장은 고정된 T0를 출력 F를 1로 만드는 테스트 패턴 T1 들 사이에 배열한 테스트 사이퀀스에 의해 검출이 가능하게 된다.

그리고 Domino CMOS 회로의 NMOS 함수블럭이 unate-function인 경우에 s-op 고장검출 테스트 패턴 T1 들로서 s-a-0 고장검출이 가능하다.

기존의 회로 시뮬레이터인 SPLICE에서는 signal flow graph를 사용하여 network ordering을 가능하게 하였다.[5] 그러나 MOS 트랜지스터의 게이트에 두개 이상의 fanout을 가진 입력이 인가될때 0과 1로서만 구성된 dependency matrix로는 회로기술이 불가능하다. 그래서 본 논문에서는 dependency matrix의 입력노드 부분을 1로 표시하지 않고 입력들을 나타내는 정수값으로 수정함으로써, 두개 이상의 fanout을 가진 입력들이 MOS 트랜지스터의 게이트에 인가될 때도 정확하게 표현되도록 dependency matrix

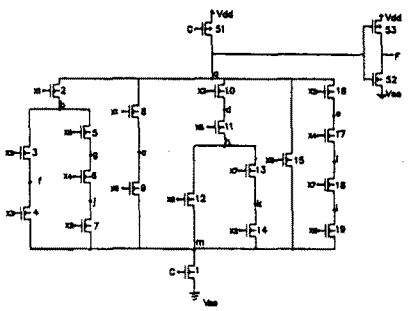

그림 3. Domino CMOS 회로

Fig.3. Domino CMOS circuit

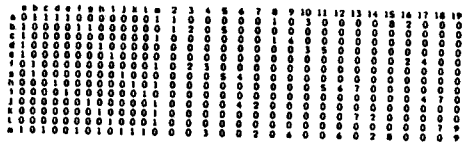

표 1 . modified dependency matrix

Table 1 . modified depency matrix

를 수정한 modified dependency matrix에 의해 Domino CMOS 회로의 NMOS 함수블럭을 모델링하였다.

예를 들어 그림 3과 같은 Domino CMOS 회로에 대해서 그 그림의 함수블럭을 modified dependency matrix로 나타내면 표 1과 같다.

Modified dependency matrix는 함수블럭에 인가되는 입력값에 따라 0과 1로서만 구성된 원래의 dependency matrix를 본 논문에서 제안된 경로형성 알고리즘에 적용시키면, 인가되는 입력값에 의하여 함수블럭 내에 어떤 경로가 형성되는지를 알 수 있다.

위와 같은 개념에 의하여 modified dependency matrix를 이용한 경로형성 알고리즘은 다음과 같다.

〔경로형성 알고리즘〕

〈용 어〉

* top node : 함수블럭 내의 제일 위쪽 노드
* bottom node : 함수블럭 내의 제일 아래쪽 노드
* 노드 집합 : 임의의 노드에 대한 노드 집합은 dependency matrix 상에서 그 노드의 행을 선택하여 선택된 행에서 입력노드들을 제외한 각각의 트랜지스터간의 노드들 중, 값이 1인 노드들의 집합을 의미한다.
* 입력 집합 : 임의의 노드에 대한 입력집합은 dependency matrix 상에서 그 노드들의 행을 선택하여 선택된 행에서 값이 1인 입력 노드들의 집합을 의미한다.

단계 1 . top 노드에 대한 입력집합과 노드집합을 구한다.

단계 2 . top 노드의 노드집합에서 첫번째 노드를 선택하여 그 노드 대한 입력집합과 노드집합을 구한다. 선택된 노드가 bottom 노드이면 단계 8로 간다.

단계 3 . 단계 1에서 구한 입력집합과 단계 2에서 구한 입력집합을 비교하여 같은 원소가 있는지 검사한다.

단계 4 . 단계 3에서 같은 원소가 있으면 단계 2에서 구한 노드집합에서 첫번째 노드를 선택하여 그 노드에 대한 입력집합과 노드집합을 구한다. 선택된 노드가 bottom 노드면 단계 8로 간다 같은 원소가 없으면 단계 2에서 구한 노드를 그 노드를 포함하고 있는 top 노드의 노드집합에서 제거한다음 단계 2부터 다시 시작한다.

단계 5 . 단계 2에서 구한 입력집합과 단계 4에서 구한 입력집합을 비교하여 같은 원소가 있는가를 조사한다.

단계 6 . 같은 원소가 있으면 단계 4에서 구한 노드집합에서 첫번째 노드를 선택하여 그 노드에 대한 입력집합과 노드집합을 구한다. 선택된 노드가 bottom 노드면 단계 8로 간다. 같은 원소가 없으면 단계 4에서 구한 노드를 단계 2에서 구한 노드집합에서 제거하고 제거된 노드집합에서 다른 노드를 선택하여 이 노드에 대한 입력집합과 노드집합을 구한 다음 구해진 집합은 단계 4에서 구한 집합과 같이 생각하여 단계 5부터 다시 시작한다.

단계 7 . 단계 4와 단계 5에서 구해진 입력집합들을 단계 2와 단계 4에서 구한 입력집합들에 대하여 단계 5와 단계 6을 수행한것처럼 단계 6에서 선택되는 노드가 bottom 노드가 될때까지 되풀이 한다.

단계 8 . 단계 1에서 단계 7까지 임의의 노드집합에서 선택된 노드가 bottom 노드면 그 bottom 노드가 선택될때까지 선택된 노드들을 1로 하고 선택되지않은 노드들을 0로 하는 path matrix를 만들고, bottom 노드부터 각 선택된 노드들을 포함한 노드집합에서 노드를 선택할 때의 역순으로 차례대로 각 선택된 노드들을 제거한다음, top 노드의 노드집합이 공집합이 될때까지 단계 1부터 단계 8까지를 되풀이한다.

Ⅲ. 고장 시뮬레이션

Domino CMOS 회로의 s-a-1 고장을 검출할 수 있는 테스트 시이퀀스를 생성해 주는 고장 시뮬레이션 알고리즘은 다음과 같다.

* s-a-1 고장 시뮬레이션 알고리즘

단계 1 . fanout을 가진 입력들을 큰 순서에 따라 선택해서 선택된 입력을 게이트로 하는 MOS 트랜지스터들이 같은 row에 위치할 수 있도록 MOS 트랜지스터들의 row위치를 변경시킨다.

단계 2 . 각 row를 선택해서 선택된 row에 있는 MOS

트랜지스터들의 게이트에 인가되는 입력들에 값으로 1을 인가한다.

선택된 row와 겹쳐지는 구간들중 선택된 row에 놓인 트랜지스터들을 포함하지 않는 경로들은 활성화되지 않도록 그 경로 놓인 트랜지스터들의 입력에 0을 인가해준다.

경로를 끊어지도록 하기 위해 입력들의 값을 인가할때 이미 인가된 입력들과 모순이 발생되는 경우에는, 경로와 같은 구간에 있고 선택된 row에 놓인 트랜지스터들을 uncovered set U1에 넣는다.

단계3. 선택된 row를 포함한 구간을 제외하고 top-node에서 구간까지, 또는 구간에서 bottom node까지 path를 형성시키도록 그 경로들상에 놓인 트랜지스터들의 입력에 1을 인가한다. 만약 이미 인가된 입력들에 의해서 경로를 형성시킬 수가 없다면, 단계2에서 선택된 구간에서 활성화되지 않아야되는 경로들에 놓인 트랜지스터들중 단계2의 조건과 단계3의 조건을 만족하는 트랜지스터들을 선택한다.

단계4. 단계3에서 선택된 트랜지스터가 있으면 입력에 0을 가하고 없으면 path가 형성되지 않는 구간내에 있는 선택된 row에 놓인 트랜지스터들을 uncovered set U1에 넣는다.

단계5. 인가되지 않은 입력들에 don't care X를 넣는다.

결정된 입력을 test pattern으로 T1에 넣고 선택된 row가 N1(전체 row수)보다 작으면 단계3으로 간다.

선택된 row가 N1이면 단계6로 간다.

단계6. uncovered set U1이 공집합이 아니면 U1이 트랜지스터들이 같은 row에 가장 많이 오도록 그 트랜지스터들의 row위치를 변경시킨다. 선택된 row를 포함하는 구간을 통과하는 경로를 형성시켜서 s-a-1고장검출이 가능한 트랜지스터들을 U1에서 빼주고 이때 입력들을 test pattern으로 T1에 넣는다.

U1이 공집합이 될때까지 위의 과정을 반복한다.

Domino CMOS회로의 s-op고장을 검출할 수 있는 테스트 시이퀀스를 생성해 주는 고장 시뮬레이션 알고리즘은 다음과 같다.

* s-open고장 시뮬레이션 알고리즘

단계1. modified dependency matrix의 입력 노드들을 나타내는 부분을 0이 아닌 정수부분은 모두 1로 변화시켜준 matrix를 생성한다.

단계2. 단계1에서 구한 matrix를 경로 형성 알고리즘에 적용시켜 함수블럭 내에서 형성되는 모든 경로들을 구한다. 구해진 모든 경로에 대하여 단계3부터 단계8까지 수행한다.

단계3. 단계2에서 구해진 경로가 통과하는 트랜지스터들을 구하여 그 경로에 대한 트랜지스터 집합속에 넣는다.

단계4. 단계3에서 만들어진 트랜지스터 집합들에 영향을 미치는 입력을 전체 입력 중에서 선택해서 트랜지스터 집합내에 트랜지스터들의 상태가 on이 되도록 그 입력의 값을 결정한다.

단계5. 단계4에서 결정되지 않은 입력의 갯수가 k개라면 0부터 2**k -1까지 값들을 2진수의 값들로 바꾸어 결정되지 않은 입력의 값에 각각 차례로 집어넣은 다음 단계6부터 단계8까지 수행한다.

단계6. 단계4와 단계5에서 구해진 입력 값으로 modified dependency matrix를 변환시켜 dependency matrix로 만든 다음 dependency matrix를 경로 형성 알고리즘 적용시켜 함수블럭내에 어떤 경로가 형성되는가를 검사한다.

단계7. 단계6에서 오직 하나의 경로만이 형성되면 그때 입력값을 테스트 matrix에 저장한다. 그리고 그 경로가 통과하는 트랜지스터들을 테스트 집합에 넣고 전체 고장집합에서 테스트 집합을 빼준다. 단계9로 간다.

단계8. 만일 단계6에서 형성된 경로가 2개 이상이면 단계6에서 형성된 모든 경로가 공통으로 통과하는 트랜지스터들이 있으면 그 트랜지스터들을 save집합에 넣고 그때 입력값은 save matrix에 저장한다.

단계9. 전체 고장집합이 공집합이 아니면 고장집합내의 트랜지스터들을 하나 선택한 다음 단계8에서 만들어진 save집합들 중에서 선택된 트랜지스터를 포함한 것이 있는지 검사한다. 포함한 것이 있으면 그 save집합을 만들수 있게한 입력값을 테스트 matrix에 저장하고 그 save집합의 값은 테스트 집합에 넣은 다음 전체 고장집합에서 빼준다. 포함한 것이 없으면 선택된 트랜지스터를 no-detect집합에 넣고 고장집합에서 그 트랜지스터를 빼준다.

단계10. 테스트 matrix에 입력값을 입력패턴 T1으로 하여 테스트 시이퀀스를 만든다.

위의 알고리즘들에 의하여 Domino CMOS회로내의 모든 트랜지스터에 대한 s-a-0, s-a-1 및 s-op고장 검출이 가능하다. 경로활성화 방법으로 테스트 패턴을 구하는 것은 알고리즘 수행이 복잡하다. 그러나 s-open

고장 시뮬레이션 알고리즘에서는 단일경로 활성화로 고장검출이 불가능한 트랜지스터들에 대해서도 다시 다중경로활성화를 수행할 필요없이 간단하게 테스트 패턴을 구할 수 있다. 또한, 연결그래프를 이용한 #operation으로 테스트 패턴을 구하기 위해서는 만일 입력이 n개라면 연결그래프상의 각 경로에 대하여 2**n개의 입력패턴으로 #operation을 수행해야 하므로 대규모 회로에서는 적용이 곤란하다. 그러나, 이 알고리즘에서는 함수블럭내의 임의의 경로가 형성하기 위한 그 경로가 통과하는 트랜지스터들의 게이트에 인가되는 입력갯수를 k라하면, 이 k개의 입력값을 그 경로가 형성되도록 결정시켜 줌으로써 2**(n-k)개의 입력패턴으로 #operation을 수행하여 테스트패턴을 구할 수 있다. 그래서 이 알고리즘은 기존의 연결그래프를 이용한 #operation으로 적용이 곤란했던 대규모 회로에서도 적용이 가능하다.

표1의 입력화일을 제안된 알고리즘들에 적용시키면 그 일부가 표2와 같다.

```
      X1  X2  X3  X4  X5  X6  X7  X8  X9  C
       0   0   0   0   0   0   0   0   0   0
       1   1   1   0   0   0   0   0   0   1
 1    52   2   3   4     (s-op) (s-a-0)
       0   0   0   0   0   0   0   0   0   0
51    53     (s-op)
       1   1   0   1   1   0   0   0   0   1
 1    52   2   5   6   7   (s-op) (s-a-0)
       0   0   0   0   0   0   0   0   0   0
51    53     (s-op)
       1   0   0   0   0   1   0   0   0   1
 1    52   8   9     (s-op) (s-a-0)

                      o
                      o
                      o

       0   0   0   0   0   0   0   0   0   0
51    53     (s-op)
       0   1   1   1   0   1   1   0   0   1
 2     8  11  18     (s-a-1)
       1   1   0   1   0   0   0   0   1   1
 4     5  10  15  19     (s-a-1)

                      o
                      o
                      o
```

표2. 표1에 대한 고장 시뮬레이션의 일부

Table 2. parts of Fault simulation result about Table 1.

Ⅳ. 결 론

본 논문에서는 Domino CMOS회로에서 함수블럭을 modified dependency matrix로 모델링하여 회로내 모든 트랜지스터에 대한 s-a-0, s-a-1, s-op고장을 검출할 수 있는 고장 시뮬레이션 알고리즘들을 제안하였다.

제안된 알고리즘들을 VAX 11/750 상에서 C-language로 실현하여 임의의 Domino CMOS회로에 적용시켜 본 결과 고장회로에 트랜지스터에 대한 s-a-0, s-a-1, s-op고장을 검출할 수 있었다.

s-a-1고장 시뮬레이션 알고리즘은 함수블럭을 basic cell로 분할하지 않고도 s-a-1고장검출이 용이하였고, s-open고장 시뮬레이션 알고리즘에서는 단일경로 활성화로 고장검출이 불가능한 트랜지스터들에 대해서도 다시 다중경로활성화를 수행할 필요없이 간단하게 테스트 패턴을 구할 수 있었다.

앞으로 Domino CMOS회로 설계방식이 점차 많이 사용됨에 따라 고장 시뮬레이터에 의한 테스트 시이퀀스의 생성이 많이 사용될 것이라 기대된다.

참 고 문 헌

1. Chiang, Kuang-Wei and Vranesic, Zvonko G. "On Fault Detection in CMOS Logic Networks", IEEE 20th Design Automation Conference, pp. 50-56, 1983.
2. Chiang, Kuang-Wei and Vranesic, Zvonko G. "Test Generation for MOS Complex Gate Networks", The 1982 International Symposium on Fault-Tolerant Computing, Snata Monica, California, June 1982.
3. Chandramouli, R. "On Testing Stuck-open Faults", The 1983 International Symposium on Fault Tolerant Computing, Milano, Italy, June 28-30, 1983.
4. Neil Weste, Kamran Eshraghian, "Principles of CMOS VLSI Design A Systems Perpective," ADDISON-WESLEY PUBLISHING, 1985.
5. A.R. Newton, A. Sangiovanni-Vincentelli, "Relaxation-Based Electical Simulation", IEEE Trans. Computer-Aided Design, Val. CAD-3, pp 308-331, Oct 1984.
6. V.G. Oklobdzija and P. GKovijanic, "On Testability of CMOS-Domino Logic", Proc. 1984 Int. Symp. Fault-Tolerant Computer, pp. 50-55, June 1984.
7. N.F. Goncalves and H.J. Deman, "NORA: A Race-free Dynamic CMOS Technique for Pipelined Logic Stryctures", IEEE Journal of Solid-State Circuits, Vol. SC-18, No. 3, pp. 261-266, June 1983.
8. Y.M. El-Zig & S.Y.H. Su. "Fault Diagnosais of MOS Combinational Networks". IEEETC, Feb., 1982.
9. 박동규, 신용철, 조상복, 임인칠, " Domino CMOS 회로의 고장 시뮬레이터(Ⅰ), (Ⅱ)", 대한 전자공학회 추계종합학술 발표대회 논문집, Vol. 8, No.2, 11월, 1985.