論文95-32B-5-1

대량의 병렬성을 이용한 고속 자동 테스트 패턴 생성기

(A Fast Automatic Test Pattern Generator Using Massive Parallelism)

金 榮 禹 *, 林 寅 七 *

(Young-Oo Kim, and In-Chil Lim)

요 약

본 논문에서는 신경회로망의 대량의 병렬성을 이용하여 디지탈 조합논리회로에 대한 테스트 패턴을 고속으로 생성하는 테스트 생성 방식을 제안한다. 전처리 단계에서는 지배자(dominator) 개념을 적용하여 각 신호선이 주출력으로 도달하기 위해 꼭 거쳐야 하는 게이트에 대한 정보를 분석한다. 주어진 stuck-at 고장에 대한 테스트 패턴을 생성하기 위해 필수적으로 할당되어야 할 신호선을 미리 파악함으로써, 고장 주입시에 여러 뉴런의 활성치를 미리 고정할 수 있게 되어, 테스트 패턴 생성 시스템의 안정화 속도를 향상시킬 수 있으므로 고속으로 테스트 패턴을 생성한다. Stuck-open 고장은 구해진 stuck-at 고장에 대한 초기화 패턴을 구하는 방법에 의해 테스트 패턴을 생성한다.

Abstract

This paper presents a fast massively parallel automatic test pattern generator for digital combinational logic circuits using neural networks. Automatic test pattern generation neural network(ATPGNN) evolves its state to a stable local minima by exchanging messages among neural network modules. In preprocessing phase, we calculate the essential assignments for the stuck-at faults in fault list by adopting dominator concept. It makes more neurons be fixed and the system speed up. Consequently, fast test pattern generation is achieved. Test patterns for stuck-open faults are generated through getting initialization patterns for the obtained stuck-at faults in the corresponding ATPGNN.

I. 서 론

VLSI 기술이 발전함에 따라 단일 칩상에 집적되는 소자수의 증가로 인하여 VLSI 테스트가 큰 문제로 대두되고 있다.

그것은 많은 소자들이 서로 복잡하게 연결되어 있어서 제한된 수의 외부 핀을 통하여 칩 내부 신호선을 제어하고 관측하는 데 한계가 있기 때문이다[1].

일반적으로 조합 논리회로에 대한 테스트 패턴 생성(Test Pattern Generation)은 대상회로(Circuit Under Test : 이하 CUT)의 탐색공간(search space)에서 정상회로와 고장회로의 출력이 서로 다르게 되어 고장 여부를 구분할 수 있는 입력 조합을 찾아내는 탐색 문제(search problem)로 생각할 수 있다[2]. 과거 수년간 조합 논리회로에 대한 탐색 속도

* 正會員, 漢陽大學校 電子工學科
(Dept. of Elec. Eng. Hanyang Univ.)
接受日字: 1994年9月13日, 수정완료일: 1995년4월27일

를 빠르게 하기 위하여 많은 테스트 생성 알고리듬이 제안되고 개선되어 왔으나[2], NP완전(NP-complete) 문제로 알려진[3] 테스트 생성 문제를 단일 프로세서를 갖는 컴퓨터로 푸는 것은 여전히 상당한 시간과 계산량이 필요하다[2,4]. 이에 따라 병렬처리 및 고속화 알고리즘 등을 이용한 테스트 생성 방식들이 제안되었다[2].

신경회로망은 순회판매원 문제 등의 NP완전한 최적화 문제에 대한 해법이 알려진[6,7] 이후, 대량의 병렬성을 이용한 고속 최적화 기법으로 많은 연구가 되어 왔다. 테스트 생성 문제에 관련한 연구는 디지탈 논리회로에 대한 대응 신경회로망 구성법과 이를 이용하여 stuck-at 고장에 대한 테스트를 생성하기 위한 신경회로망을 구성하는 방법 및 테스트 생성 방식이 제안된[8,9,10] 이후, 고장 진단 방식[11], 무정의조건까지를 고려한 3치 테스트 생성 방식(3-valued model)[12], k-out-of-n 설계 규칙을 이용한 테스트 생성 방식[13] 등이 제안되었다. 그러나, 일반 조합 논리회로에 대한 적용이 어렵거나[13], fan-out에서의 여러 상이한 고장을 하나의 고장으로 취급하는[8,9,10] 문제점들을 갖고 있었다.

본 논문에서는 부분 신경회로망들이 상호간의 제한 아래에서 전체적인 안정화에 의해 조합논리회로에 대한 테스트를 고속으로 생성하는 방식을 제안한다. CUT를 논리 게이트 단위의 신경회로망 모듈로 구성하고, 모든 모듈들이 상호간의 메시지 교환을 통해 안정화의 제한조건을 모두 만족하면서 안정화될 때 자동 테스트패턴 생성 신경회로망(Automatic Test Pattern Generation Neural Network : 이하 ATPGNN)은 안정화된다. 이때 CUT의 입력선에 대응되는 뉴런의 활성치(activation value)에 의하여 주입된 stuck-at 고장에 대한 테스트를 생성한다. 전처리 단계에서는 지배자(dominator) 개념을 적용하여 각 신호선이 주출력으로 도달하기 위해 필수적으로 거쳐야 하는 게이트에 대한 정보를 분석한다. 분석된 정보는 ATPGNN으로의 고장 주입시에 여러 뉴런의 활성치를 미리 고정하여, 고장이 주입된 ATPGNN의 안정화 속도를 향상시킨다. 또한 stuck-open 고장은 초기화 패턴(initialization pattern)과 특정 stuck-at 고장에 대한 테스트 패턴을 적용함으로써 검출할 수 있으며,[5] 본 논문에서는 stuck-at 고장에 대한 테스트 패턴을 구한 ATPGNN의 상태를 발전시켜 초기화 패턴을 구함으로써 stuck-open 고장에 대한 테스트 패턴을 구하는 방법을 제안한다. 본 논문에서 제안한 방식은 여러 예제 회로에 대한 실험을 통하여 그 유효성을 입증한다.

Ⅱ. ATPG 신경회로망의 구성

하나의 입력 패턴에 의한 정상회로의 출력 패턴과 고장회로의 출력 패턴이 서로 다를 때, 그 입출력 패턴을 테스트 패턴이라 한다. 따라서 조합논리회로에서 특정 부분의 고장을 검출하는 테스트 패턴을 생성하기 위해서는 입력이 정상적으로 출력측까지 전달되는 것과 전달중에 고장이 개입하여 잘못된 신호가 출력측에 전달되는 것을 모델링하고, 그 두 경로를 통한 출력이 서로 다르도록 하여야 한다. CUT에 대한 ATPGNN은 하나의 입력 패턴이 정상회로와 고장회로에서 서로 다른 출력 패턴을 내도록 하기 위하여 정상적인 신호의 전달을 모델링하는 무고장부(fault-free part), 고장신호의 전달을 모델링하는 고장부(faulty part), 이들 두 부분에 동일한 입력이 가해지도록 연결하는 입력부(input part), 그리고 두 부분의 출력이 서로 다르도록 제한을 가하는 출력 인터페이스부(output interface part)의 네 부분으로 구성된다. ATPGNN의 구조는 그림 1과 같다.

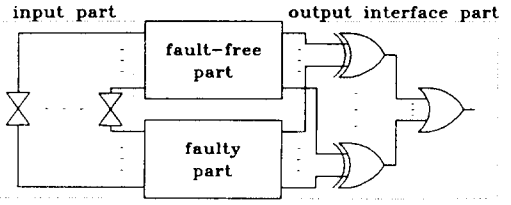


그림 1. ATPG 신경회로망의 구성
Fig. 1. The structure of ATPG neural network.

논리회로에 대한 신경회로망은 각 논리게이트에 대응하는 신경회로망 모듈의 뉴런이 인접 모듈의 뉴런과 활성치가 동일하게 되도록하는 제한하에서 안정화됨으로써 올바른 동작을 이룰 수 있다. 그림 2는 각 논리게이트에 대한 신경회로망 모듈의 예이다. 모듈 내의 각 뉴런들의 시간 발전 규칙은 식 (1)과 같다.

$$net_i = \sum_{i \neq j} T_{ij} V_j + \theta_i \begin{cases} > 0 \text{ 일 때}, V_i(t+1) = +1, \\ < 0 \text{ 일 때}, V_i(t+1) = -1, \\ = 0 \text{ 일 때}, V_i(t+1) = V_i(t). \end{cases} \quad (1)$$

여기에서 T_{ij}는 뉴런 j에서 뉴런 i로의 연결강도(connection strength)를 나타내며, V_j는 뉴런 j의 활성치(activation value)를 나타낸다. 또한 θ_i는 뉴런 i의 내부 임계치(internal threshold)를 나타내며,

neti는 뉴런 i에 대한 자극의 총합이다.

각 뉴런간의 연결강도 및 임계치는 각 뉴런들에 결정 초평면(decision hyperplane)이 존재하여야 하고 일치상태(consistent state)에서만 에너지 기저상태를 갖는다는 제한조건으로부터 유도하였다. (1)에서 뉴런의 활성치가 +1인 것은 논리 1 값을, -1인 것은 논리 0 값을, 0인 것은 무정의조건(don't care condition)임을 나타내며, ATPGNN의 초기 상태는 값이 주입되는 뉴런을 제외하고는 모든 뉴런의 활성치가 0으로 초기화되는 것으로 한다.

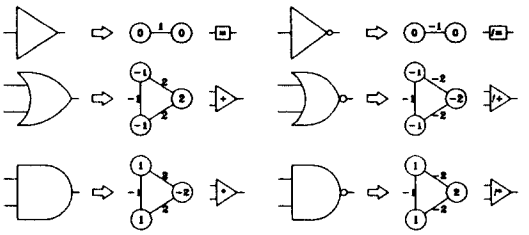


그림 2. 각 논리 게이트에 대한 신경회로망 모듈 및 심볼

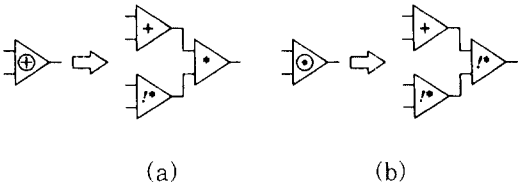
Fig. 2. Neural network modules and symbols for logic gates.

XOR 게이트나 XNOR 게이트는 4개의 뉴런을 갖는 신경회로망으로 구성이 가능하다.[8,9] 그러나 이 경우에는 하나의 상태에서 인접 상태로의 천이가 임의적이지 못하고 특정한 인접 상태로의 천이만이 발생하는 문제가 있다. 예를 들어 XOR 게이트의 경우, 두 입력을 각각 I_1, I_2라 하고, 출력을 O라 할 때, (I_1, I_2, O) = (-1, -1, +1)은 일치상태가 아니므로 (+1, -1, +1), (-1, +1, +1), (-1, -1, -1) 중의 한 상태로 천이할 확율이 각각 1/3이 되어야 하지만, 항상 셋 중의 특정 상태로만 천이하게 된다. 따라서 본 논문에서는, 임의의 상태 천이를 이루게 하기 위하여, XOR 게이트와 XNOR 게이트를 다음의 식에 의해 그림 3과 같이 3개의 게이트 모듈의 조합으로 구성한다.

$$A \oplus B = (A+B)\cdot(A'+B') = (A+B)\cdot(A\cdot B)'$$
$$A \odot B = (A \oplus B)' = ((A+B)\cdot(A\cdot B)')'$$

또한 3입력 이상인 게이트들도 2입력 게이트의 조합으로 구성한다.

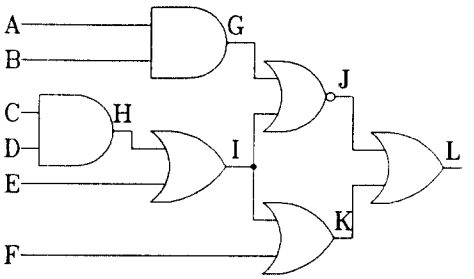
그림 4(b)는 그림 4(a)의 예제회로[2]에 대한 ATPGNN을 나타내며, 그림 4(c)는 신호선 I->K(I에서 K로의 fan-out 신호선)에 s-a-0 고장을 주입한 초기상태이다. 기본적으로 초기의 ATPGNN은 출력인터페이스부의 최종단 뉴런의 활성치는 +1, 고장부의 고장이 가정된 신호선에 대응하는 뉴런의 활성치는 가정된 고장값(s-a-0에 대해서는 -1, s-a-1에 대해서는 +1), 무고장부의 뉴런에는 가정된 고장의 반대값(s-a-0에 대해서는 +1, s-a-1에 대해서는 -1)을 할당한 상태이다. 그림 4(c)에서 검게 칠해진 부분의 뉴런은 그 활성치가 고정되었음을 나타낸 것이다. 뉴런의 활성치는 그 뉴런으로의 입력과 다른 뉴런의 출력과의 연결을 차단함으로써 고정시킬 수 있다.



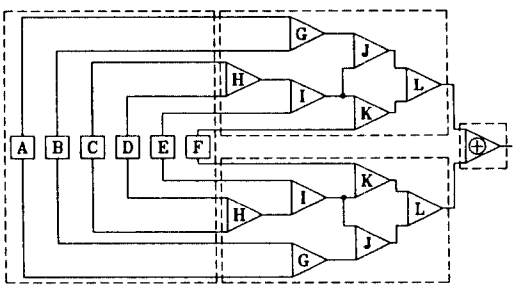
(a) (b)

그림 3. (a) XOR 게이트에 대한 심볼 및 구성,
(b) XNOR 게이트에 대한 심볼 및 구성

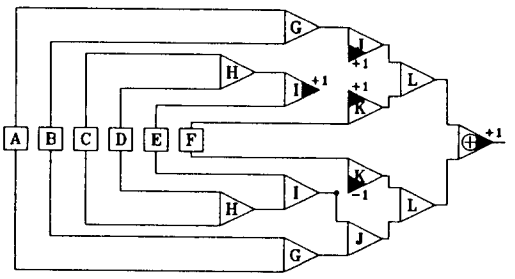
Fig. 3. (a) Symbol and organization of XOR gate,
(b) and those of XNOR.



(a)



(b)



(c)

그림 4. (a) 예제회로의 논리도,
(b) (a)에 대한 ATPGNN,
(c) 신호선 I->K에 s-a-0 고장을 주입한 초기 ATPGNN

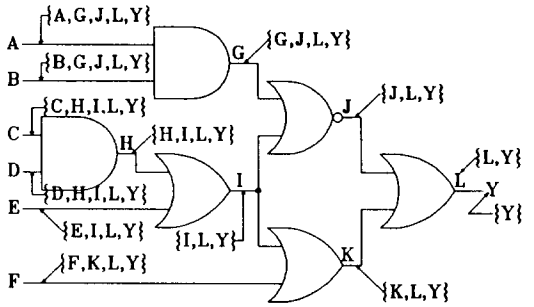
Fig. 4. An example circuit. (a) logic diagram, (b) ATPGNN, (c) and the initial ATPGNN with stuck-at-0 fault on signal line I->K.

Ⅲ. 상태공간의 축소

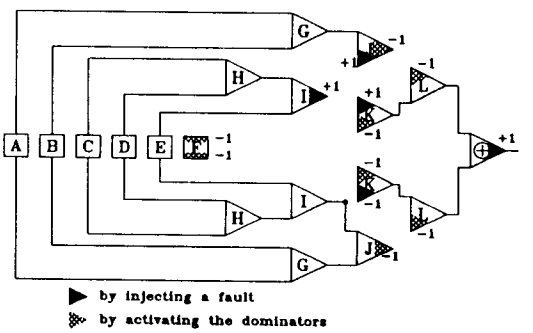
ATPGNN을 이용한 테스트 패턴 생성 문제는 ATPGNN이 내재된 극소점 중 하나로 수렴해 가는 상태공간 탐색 문제로 생각할 수 있다. 이때 탐색의 출발점은 고장이 주입된 초기상태로, 출력 인터페이스부의 최종 출력 뉴런의 활성치가 +1이고, 고장부와 무고장부의 고장주입 뉴런의 활성치가 +1,-1인 상태가 된다.

그림 4의 예제회로는 결정되어야 할 뉴런의 활성치가 모두 31개(입력부 12 + 고장부 8 + 무고장부 8 + 출력 인터페이스부 3)이다. 신호선 I->K에 s-a-0 고장이 주입된 초기상태에서는 이 중 5개(무고장부의 I, I->J, I->K 뉴런에 각각 +1, 고장부의 I->K 뉴런에 -1, 출력 인터페이스부 최종단 뉴런에 +1)의 뉴런의 활성치가 고정되므로 이후 시간발전을 통해 나머지 26개의 뉴런의 활성치를 결정함으로써 테스트 패턴을 생성한다. 이것은 326개의 상태 중에 존재하는 극소점(여러개가 있을 수도 있다.) 중에 하나를 찾는 것이 되므로 탐색공간이 매우 크다. 초기상태에서 활성치가 고정되는 뉴런의 수가 많을수록 탐색공간의 크기는 줄게 되어 테스트 패턴 생성을 고속화할 수 있다. 따라서 고속의 테스트 패턴 생성을 위해서는 ATPGNN의 여러 안정화 상태 중 공통적인 뉴런의 활성치를 초기에 검색하여 할당할 필요가 있다.

회로 내의 가정된 고장을 검출하기 위해서 필수적으로 값이 할당되어야 하는 신호선을 필수할당선이라 한다. 필수할당선은 고장이 발생한 신호선에 고장 신호를 발생시키기 위해 값이 할당되어야 할 필수제어할당선과 발생한 고장 신호를 최소한 어느 한 주출력까지 전파시키기 위해 꼭 할당되어야만 하는 필수관측할당선으로 나뉜다. 회로 내부의 각 게이트로부터 주출력에 도달하기 위해 반드시 통과해야 할 게이트를 지배자(dominator)라 한다. 필수관측선은 지배자를 계산하는 방법을 적용하여 구할 수 있다. 다출력 조합 논리회로에서 지배자를 계산하는 알고리듬은 다음과 같다.



(a)



(b)

그림 5. (a) 예제회로에 대한 지배자
(b) 필수할당선에 값을 할당한 초기 ATPGNN

Fig. 5. (a) Dominators for example circuit, (b) initial ATPGNN with essential assignment.

[지배자 계산 알고리듬]

[단계 1] 모든 주출력 중에서 한 Fi를 선택한다.

[단계 2] 출력 F_i에 대한 지배자 집합을 초기화한다.

$$Dom(F_i) = F_i$$

[단계 3] 주입력측으로 진행하며 각 신호선 l에 대한 다음 연산을 수행한다.

$Dom(\ell) = \{\ell\} \cup$ {앞 신호선들의 지배자 집합의 intersection}

[단계 4] 처리해야 할 주출력이 남아 있으면 단계 1로 간다.

[단계 5] 종료

그림 5(a)는 그림 4(a)의 회로에 지배자 계산 알고리듬을 적용하여 지배자를 계산한 예이다. 신호선 I->K에 s-a-0을 가정하면, 이들 게이트를 반드시 통과해야 함을 알 수 있다. 즉,

$$Dom(\text{I->K}) = \{\text{K, L, Y}\}$$

이다. I->K의 고장은 K, L, Y를 모두 통과하여야 주출력에 도달된다. 따라서 이들 지배자의 입력은 전달치(propagation value)로 할당되어야 한다. 고장이 게이트 K를 통하여 전달되기 위해서는 신호선 F에 -1(논리 0)이 할당되어야 하며, 게이트 L을 통과하기 위해서는 게이트 J의 출력에 -1이 할당되어야 한다. 이러한 할당들을 모두 고정시키면 그림 4(c)의 초기 ATPGNN는 그림 5(b)와 같이 구성될 수 있다.

그림 6은 각 게이트 모듈에서 뉴런들의 활성치가 전파되어 할당되는 경우이다. 이들은 필연적인 전파이므로 ATPGNN을 시간발전시키기 이전에 값을 전파시키고 고정한다.

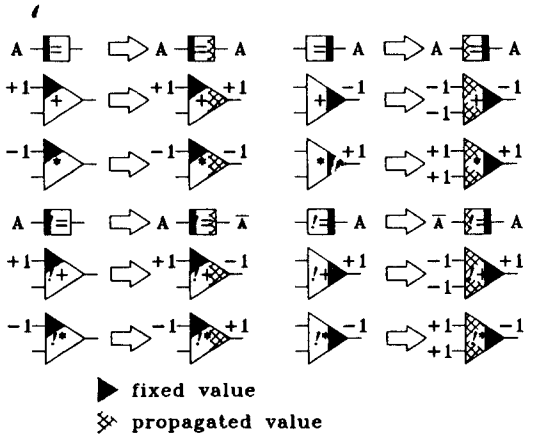
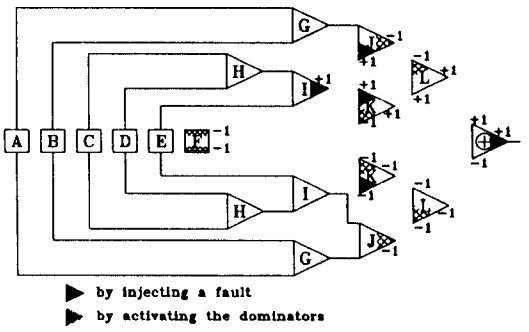
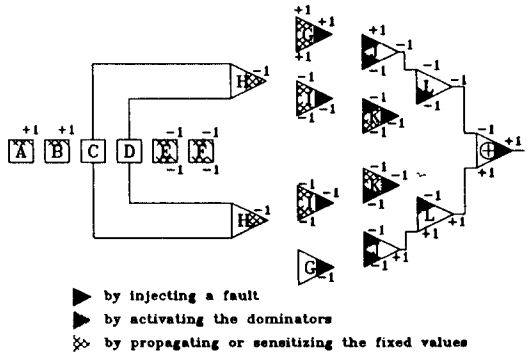


그림 6. 선행처리 단계에서의 필수할당의 전파
Fig. 6. Propagation of essential assignment in preprocessing phase.

이상의 과정을 통해 그림 4 회로의 신호선 I->K의 s-a-0 고장에 대한 필수할당을 하면 그림 7(a)와 같은 초기 ATPGNN을 얻을 수 있다. 그림 7(a)의 ATPGNN은 활성치가 미확정된 뉴런의 수가 15개(입력부 10 + 고장부 3 + 무고장부 2 + 출력인터페이스부 0)이므로 탐색 대상 상태수는 3^{15}으로 줄게 된다. 실제로 신호선 I-K의 s-a-0 고장에 대한 테스트 패턴은 모두 90개가 존재하며, 그림 7(a)의 초기 ATPGNN은 90개의 극소점({(A,B,C,D,E,F)|(X,X,X,X,+1,-1) or (X,X,+1,+1,-1,-1), where X = -1 or 0 or +1})을 갖고 있다. 초기 ATPGNN은 시간발전을 통해 90개의 극소점 중의 하나에서 안정화된다.



(a)



(b)

그림 7. 예제회로의 필수할당선에 값을 할당하고 전파한 초기 ATPGNN
(a) I->K의 s-a-0, (b) G의 s-a-0

Fig. 7. Initial ATPGNN of the example circuit with essential assignment and propagation for (a) s-a-0 at I->K, and (b) s-a-0 at G

그림 7(b)는 예제회로의 신호선 G에 s-a-0 고장을 주입하고 필수할당을 전파한 것이다. 이 경우는 탐색 대상 상태 수가 3^{22}에서 3^{4}으로 줄게 되며, 5개의 극소점 중 하나의 극소점에서 안정화되므로 매우 빠른 탐색이 가능하게 된다.

Ⅳ. Stuck-at 고장의 주입 및 테스트 패턴 생성

그림 8은 stuck-at 고장에 대한 ATPG 시스템의 흐름도이다. CUT의 구성은 YACC를 이용하여 구성한 하드웨어기술언어로 기술되어 시스템에 제공되며, 신경회로망 컴파일러는 신경회로망 데이타베이스에 저장된 게이트 모듈에 대한 신경회로망 구성을 참조하여 ATPGNN을 구성한다. 그림 9는 그림 4의 회로에 대한 회로기술이다.

3입력 이상의 게이트는 2입력 게이트의 조합으로 분해되며, XOR, XNOR 게이트는 그림 3과 같이 분해하는 전처리 과정을 거친다. 또한 회로기술로부터 고장결합(fault collapsing)을 통하여 테스트 패턴을 요하는 고장을 결정하고, 지배자 계산을 통하여 가정된 고장에 대한 필수 할당을 계산하여 고장사전에 등록한다. 고장사전은 ATPGNN을 생성할 때 고장리스트가 ATPGNN에서의 뉴런의 위치와 할당 값으로 매핑되어 수정 고장사전으로 구성된다.

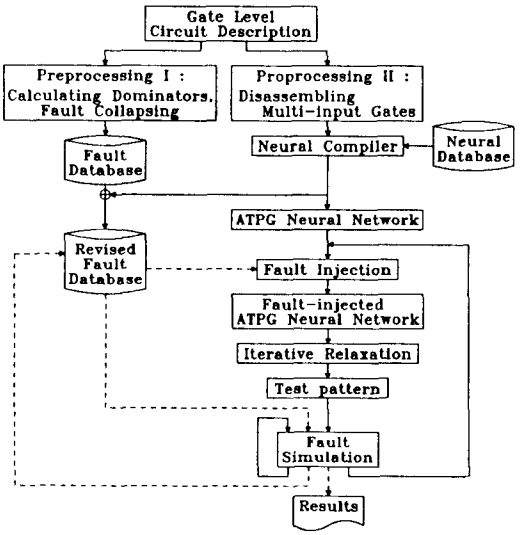


그림 8. Stuck-at 고장에 대한 ATPG 시스템의 흐름도

Fig. 8. The flow of ATPG system for stuck-at faults.

고장 주입 과정에서는 고장사전에 등록된 고장에 대한 할당치를 ATPGNN의 해당 뉴런에 할당하여 고장이 주입된 초기 ATPGNN을 구성하고, 이는 시간발전을 통하여 안정화되도록 진행된다. 그림 10에 그림 7(a)의 초기 ATPGNN이 시간이 경과함에 따라 안정화되어가는 과정을 보였다.

```
#C14 circuit : example of Fig.4.
C14(A, B, C, D, E, F : OUT_L)
{
    AND
        G(A, B : J)
        H(C, D : I);
    OR
        I(H, E : J, K)
        K(I, F : L)
        L(J, K : OUT_L);
    NOR
        J(G, I : L);
}
```

그림 9. 예제회로에 대한 회로기술

Fig. 9. Circuit description of the circuit of Fig.4.

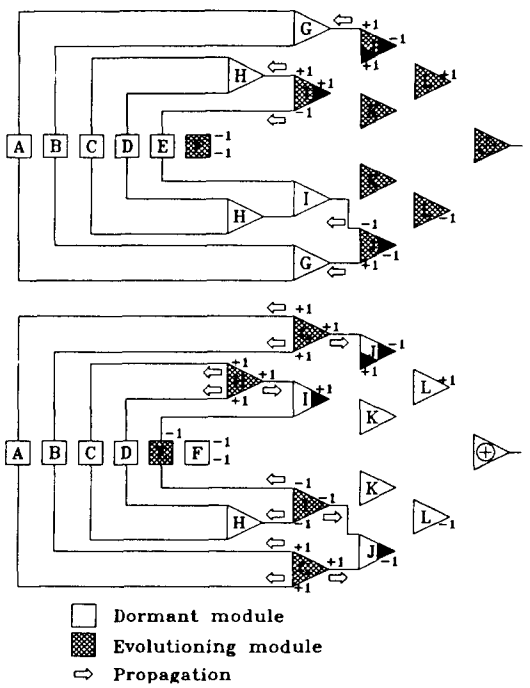


그림 10. 안정화 과정의 일부

Fig. 10. A portion of stabilization process.

ATPGNN이 안정화되면, 무고장부의 입력 뉴런 및 출력 뉴런에 나타난 활성치를 읽음으로써 테스트 패턴

을 생성한다. 만약 가정된 고장이 검출될 수 없는 고장(undetectable fault)이면 시간발전을 통해 안정화되지 못하고 불안정한 상태를 떠돌게 되는데, 본 논문에서는 ATPGNN이 일정한 시간 내에 안정화되지 못할 경우의 고장을 검출 불능 고장으로 간주하였다. 본 논문의 시뮬레이션에서는 (게이트모듈의 수)^2 만큼의 모듈 시간발전을 임계치로 하였다.

검출 불능 고장은 전처리 과정에서도 검출될 수 있다. 예를 들어 그림 4의 예제회로에서 신호선 I->J에 s-a-0 고장이 발생한 경우를 가정하면, Dom(I->J) = {J,L,Y}이므로 게이트 L을 통해 신호가 전달되기 위해서는 게이트 K에 -1이 필수적으로 할당되어야 한다. 이러한 필수 할당은 고장신호가 고장부와 무고장부에서 모두 출력측으로 전파되기 위한 것이므로, 고장부와 무고장부에 동시에 할당되어야 한다. 게이트 K에 -1 값을 만들기 위해서는 신호선 F와 게이트 I가 모두 -1 값을 가져야 한다. 그러나 고장을 만드는 과정에서 이미 무고장부의 게이트 I는 +1 값으로 고정하였으므로 양자간에 모순이 발생한다. 이러한 경우는 고장을 주출력까지 전파시킬 수 없는 고장이 된다. 본 논문의 ATPG 시스템에서는 전처리 과정에서 발견된 검출 불능 고장을 고장사전에 기록함으로써 불필요한 탐색 시도를 줄이고 있다.

하나의 테스트 패턴이 생성되면, 동일한 테스트 패턴으로 검출될 수 있는 다른 고장이 있는지를 살펴보기 위해 고장 리스트의 고장에 대하여 고장 시뮬레이션을 수행한다. 고장 시뮬레이션은 생성된 테스트 패턴의 활성치를 입력 뉴런 및 출력 뉴런에 고정하고 시뮬레이션의 대상이 되는 고장을 주입한 후, 입력에서 출력측의 정방향으로 활성치를 전달시킬 때, 고정된 출력 뉴런의 활성치와 어긋나지 않는가를 살핌으로써 이루어진다. 만약 어긋나지 않는다면, 대상 고장은 동일한 테스트 패턴으로 검출되는 고장이므로 이를 고장 리스트에서 제외시킨다.

생성된 테스트 패턴으로 고장 리스트의 모든 고장에 대한 고장 시뮬레이션을 종료하면, 고장 리스트의 다음 고장을 ATPGNN에 주입하고 그 이하 단계를 수행한다. 고장 리스트가 빈 리스트(empty list)가 되면 테스트 생성을 종료한다.

V. Stuck-open 고장에 대한 테스트 패턴 생성

디지탈 회로가 CMOS(Complementary Metal Oxide Semiconductor)로 구성될 경우에는 stuck-open 고장을 검출하기 위해 초기화 패턴(initialization pattern)과 전파 패턴(propagation pattern)의 두 연속 패턴이 필요하게 된다.[5,14] 초기화 패턴은 가정된 고장 위치에 초기의 천이값을 설정하는 역할을 하며, 전파 패턴은 초기 천이값을 출력까지 전달함으로써 고장의 효과를 확인하는 역할을 한다.

전파 패턴은 고장 위치에서의 논리 레벨의 대응 stuck-at 고장에 대한 테스트 패턴을 구함으로써 얻을 수 있으며, 초기화 패턴은 해당 stuck-at 고장 위치에 고장값이 설정되도록 패턴을 구함으로써 얻을 수 있다.[5]

그림 11(a)는 2입력 CMOS NAND 게이트이며, 각 트랜지스터에서의 stuck-open 고장에 대한 테스트 패턴은 표 1과 같다. 또한 그림 11(b)는 2입력 CMOS NOR 게이트로, 표 2에 각 트랜지스터에서의 stuck-open 고장에 대한 테스트 패턴을 나타내었다.

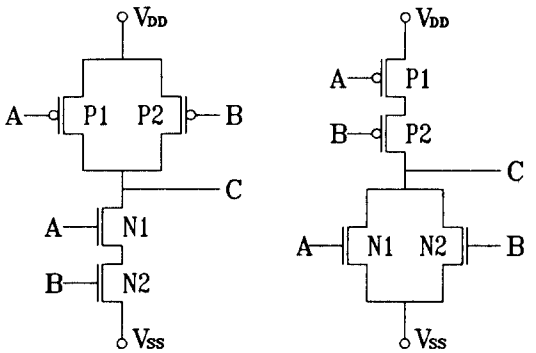


그림 11. 2입력 CMOS 게이트, (a) NAND 게이트, (b) NOR 게이트

Fig. 11. Typical two-input CMOS gates, (a) NAND gate, (b) NOR gate.

표 1. 2입력 CMOS NAND 게이트의 각 트랜지스터에 대한 stuck-open 고장의 테스트 패턴

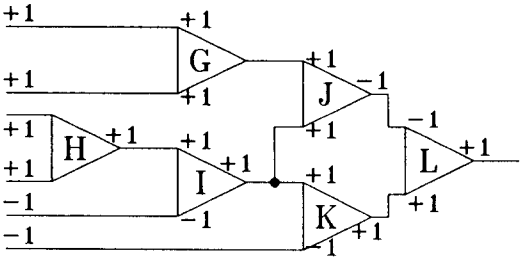
Table 1. Test patterns for stuck-open faults of two-input CMOS NAND gate.

트랜지스터	초기화 패턴	전파패턴
P1	11	01(stuck-at 1 at A)
P2	11	10(stuck-at 1 at B)
N1	OX	11(stuck-at 0 at A)
N2	XO	11(stuck-at 0 at B)

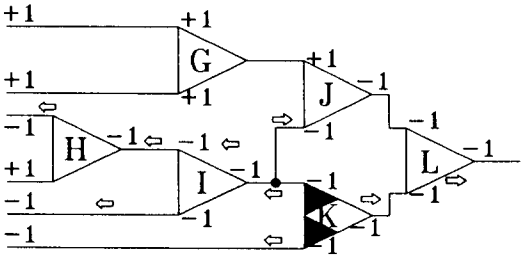
표 2. 2입력 CMOS NOR 게이트의 각 트랜지스터에 대한 stuck-open 고장의 테스트 패턴

Table 2. Test patterns for stuck-open faults of two-input CMOS NOR gate.

트랜지스터	초기화패턴	전파패턴
P1	1X	00(stuck-at 1 at A)
P2	X1	00(stuck-at 1 at B)
N1	00	10(stuck-at 0 at A)
N2	00	01(stuck-at 0 at B)



(a)



(b)

그림 12. 그림 4(a) 회로 I->K 신호선의 n-Tr.에서의 stuck-open 고장
(a) 전파패턴을 구한 ATPGNN의 무고장부
(b) 초기화패턴을 구한 ATPGNN의 무고장부

Fig. 12. Stuck-open fault at n-Tr. of signal line I->K in the circuit of Fig. 4(a). Fault free part of ATPGNN (a) after evolution for propagation pattern, and (b) after evolution for initialization pattern.

Stuck-open 고장에 대한 테스트 패턴을 구하는 순서는 다음과 같다.

1) 표 1, 표 2를 참조하여 ATPGNN의 고장 위치에 stuck-at 고장을 주입한 후, 시간발전을 통해 대상 stuck-open 고장에 대한 전파패턴을 구한다. 이 과정은 Ⅳ절의 stuck-at 고장 검출 과정과 동일하다. 2) 과정 1)에서 안정화된 ATPGNN 무고장부의 고장 위치에 표 1, 표 2의 초기화 패턴을 고정하고, 그 값을 전파하여 대상회로에 대한 초기화패턴을 구한다.

그림 4(a) 예제 회로의 I->K 신호선의 n-Tr.에 stuck-open 고장을 가정하면, 그에 대한 전파패턴은 I->K 신호선의 stuck-at-0 고장과 동일하므로 Ⅳ절과 같은 방법으로 구할 수 있으며, 그 패턴은 (111100:D)이다. 여기서 D는 고장이 없는 경우 '1'을, 고장이 있는 경우 '0'을 출력함을 의미한다.

그림 12(a)는 그림 4(a) 회로의 I->K 신호선에 stuck-at-0 고장을 가정한 후, 시간발전을 거친 ATPGNN의 무고장부이다.

이 상태에서 K 게이트의 입력을 OR 게이트의 n-Tr.에 대한 초기화패턴인 (00)을 인가하여 전파하면, 그림 12(b)와 같이 되며, 이때의 초기화 패턴인 (110100:0)을 얻을 수 있다.

Ⅵ. 실험 결과 및 고찰

본 논문에서 제안된 ATPG 시스템은 SUN4 SPARC station 1에서 C 언어로 구현되었다. 표 3은 여러 회로의 stuck-at 고장에 대해 본 시스템을 적용한 결과이다. 전처리 과정에서 지배자를 계산하여 고장 주입 단계에서 적용함으로써 고장 검출 시간에 많은 향상이 있음을 볼 수 있다. 그것은 지배자 계산을 통해 초기에 많은 뉴런의 활성치를 고정하여 탐색 영역을 축소함으로써, ATPGNN이 시간발전을 통해 안정화되는데 필요한 시간을 단축하여 얻은 효과이다. 또한, 테스트 패턴 생성시에 가장 많은 시간이 소모되는 검출불능 고장 검출이 전처리 과정에서, 지배자를 구할 때 필요조건간의 모순을 발견함에 의해, 다수 완수되므로 그에 대한 탐색 시간이 단축되었다. 전처리에 따른 오버헤드는 표 3에 나타난 바와 같이 전체 테스트 생성 시간에 비해 매우 적으므로 무시할 수 있다.

표 3의 결과에서 보면 각 모듈의 시간발전이 동시에 병렬적으로 수행될 경우, 테스트 생성 시간은 게이트 모듈의 수에 거의 선형적이므로 병렬처리시의 테스트 생성 시간은 대략 O(n)이 됨을 알 수 있다.

표 4는 stuck-open 고장에 대해 본 시스템을 적용한 결과이다.

표 3. Stuck-at 고장에 대한 실험 결과
Table 3. Experimental Results for stuck-at faults.

Circuits	ECT	F/A	SCH	SN7	C17	C14	ALU2	BCD
# inputs	6	3	4	9	5	6	8	4
# outputs	1	2	1	3	2	1	3	4
# signals	23	16	25	49	17	14	89	28
# gate modules (i)	57	25	31	73	31	21	114	45
# total faults	46	32	50	98	34	28	178	56
# faults detected	38	32	43	98	34	27	178	56
# undetectable faults	8	0	7	0	0	1	0	0
# test patterns	5	25	9	38	19	8	33	20
(with preprocessing)								
total time	38.3	1.7	19.6	17.1	0.8	1.7	226	3.9
total time (detected)	9.0	1.7	1.1	17.1	0.8	0.3	226	3.9
ave. time/fault (sec)	0.83	0.05	0.39	0.17	0.02	0.06	1.27	0.07
ave. time/detec. (ii)	0.24	0.05	0.03	0.17	0.02	0.01	1.27	0.07
(ii) / ((i) * (i))	7.39	8.00	3.12	3.19	2.08	2.27	9.77	3.46
(with preprocessing)								
total time	6.9	1.2	1.5	3.2	0.4	0.3	42.5	0.9
ave. time/fault (sec)	0.15	0.02	0.11	0.09	0.01	0.02	0.24	0.03
ave. time/detected	0.09	0.02	0.02	0.09	0.01	0.01	0.24	0.30
preprocessing time	0.2	-	-	-	-	-	0.4	-

- : 0.1초 이하

cf) ECT : ECAT circuit[14],
F/A : Full adder circuit,
SCH : Schneider circuit[14],
SN7 : SN7480 gated full adder[15],
ALU2 : A 2 bit Arithmetic Logic Unit,
BCD : BCD to excess-3 code converter

표 4. Stuck-open 고장에 대한 실험 결과
Table 4. Experimental Results for stuck-open faults.

Circuit	SCH	SN7	C17	C14	BCD
# total faults	30	93	18	28	46
# faults detected	21	93	18	23	46
# undetectable faults	9	0	0	5	0
ave. time/fault (sec)	0.12	0.10	0.02	0.02	0.02
ave. time/detected	0.01	0.10	0.01	0.01	0.02

Ⅶ. 결 론

본 논문에서는 신경회로망 모듈 상호간의 메시지 교환에 의해 전체 신경회로망이 안정화됨으로써 조합논리회로에 대한 테스트 패턴을 생성하는 테스트 패턴 생성기를 제안하였다. 전처리 단계에서는 각 신호선이 주출력으로 도달하기 위해 꼭 거쳐야 하는 게이트에 대한 정보를 분석하고, 주어진 고장에 대한 테스트 생성을 위해 필수적으로 할당되어야 할 신호선을 구하였다. 이를 통해 테스트 패턴 생성 시스템이 고장 주입 후 테스트 패턴을 찾는 탐색 영역을 축소하였으며, stuck-at 고장에 대한 테스트 패턴 생성을 고속화하였다. 또한 stuck-at 고장에대한 테스트 패턴을 구한 시스템으로부터 stuck-open 고장에 대한 테스트 패턴을 생성하였다.

본 논문의 테스트 패턴 생성 시스템은 여러 예제 회로에 대한 실험을 통하여 대상회로의 크기가 증가하더라도 병렬처리를 할 경우 테스트 생성 시간이 회로 크기에 대해 O(n)이 됨을 보였다.

앞으로는 회로내의 게이트 모듈의 수를 줄이는 collapsing 및 하드웨어 실장에 대한 연구가 이루어져야 할 것이다.

참 고 문 헌

[1] T.W.Williams and K.P.Parker, "Design for Testability A Survey," Proc. IEEE, vol. 71, 98-112, Jan. 1983.

[2] H.Klenke et al. "Parallel-Processing Techniques for Automatic Test Pattern Generation," Computer, Vol.25, No.1, pp.71-84, Jan. 1992.

[3] O.H.Ibrra and S.K.Shani, "Polynomially Complete Fault Detection Problems," IEEE Trans. on Computers, pp.242-249, March 1985.

[4] H.Fujiwara and T.Shimono, "On The Acceleration of Test Generation Algorithms," IEEE Trans. Computer vol. C-32, pp.1137-1144, Dec. 1983.

[5] L.N.Leddy et al. "COMPACTEST-II : A Method to Generate Compact Two-Pattern Test Sets for Combitional Logic Circuits," Proc. ICCAD-92, Santa Clara, USA, pp.568-574, Nov. 1992.

[6] J.J.Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computional Abilities," Proc. Nat'l. Acad. Sci. USA, Vol.79, pp.2554-2558, April 1982.

[7] J.J.Hopfield and D.W.Tank, ""Neural" Computation of Decisions in Optimization Problems," Biological Cybernetics, Vol.52, pp.158-159, 1985.

[8] S.T.Chakradhar et al. "Automatic Test Generation Using Neural Networks," Proc. IEEE Int'l Conf. Computer-Aided Design, pp. 416-420, 1988.

[9] S.T.Chakradhar et al."Toward Mas-

sively Parallel Automatic Test Generation," IEEE Trans. on Computer-Aided Design, pp.981-994, Sep. 1990.

[10] S.T.Chakradhar et al. Neural Models and Algorithms for Digital Testing, Kluwer Academic Publishers, 1991.

[11] S.S.Kalkunte et al. "A Neural Network Approach for High Resolution Fault Diagnosis in Digital Circuits," Proc. IJCNN, Beijing, China, pp.I-83-88, Nov. 1992.

[12] A.Majumder and R.Dandapani, "Neural Networks as Massively Parallel Automatic Test Pattern Generators," Proc. IEEE ICNN, San Francisco, USA, pp.1724-1730, April 1993.

[13] M.Ali, T.Nakagawa, and H.Kitagawa, "An Approach to Automatic Test Pattern Generation Using Strictly Digital Neural Networks," Proc. IJCNN, Baltimore, USA, pp. -474-479, June 1992.

[14] H.Fujiwara, Logic Testing and Design for Testability, The MIT Press, 1985.

[15] R.G.Bennetts, Design of Testable Logic Circuits, Addison-Wesley, 1984.

저 자 소 개

金 榮 禹(正會員) 第 31卷 B編 第 5號 參照.
현재 한양대학교 대학원 전자공학과 박사과정

林 寅 七(正會員) 第 30卷 B編 第 2號 參照
현재 한양대학교 전자공학과 교수