PLA의 고장 분석 및 검출

Fault Analysis and Detection in Programmable Logic Arrays

조 병 호* 서울 대학교
황 희 용 전자계산기 공학과
진 병 문

I. 서론

PLA(programmable Logic Array)는 Fleisher (11)가 제안한 이후로 마이크로 프로그래밍, 순차제어기, 함수 발생기, 코-드 변환 등 여러 분야에서 응용되고 있으며(17), VLSI 의 제조에도 그 중요성을 인정받고 있다.(12)

Fleisher 는 PLA 의 구조를 개념상 2-level AND-OR 배열로서 다입력-다출력의 회로로 구성하고, 이에 대한 여러가지의 실현 기술을 제시하였다.

또한, PLA 는 메모리와 흡사한 규칙적인 구조로 구성되어 있으며, AND-OR 배열내의 연결고리를 끊음으로서 조합 스위칭 함수를 프로그램할 수 있다. 특히 일반 논리회로 설계자가 프로그램할 수 있도록 고안된 것이 FPLA (Field PLA) 이다.

이러한 PLA 는 다른 논리회로와 마찬가지로 정상적으로 작동하는가를 테스트 하는 것이 필요하다. PLA 가 개념상 AND-OR 의 조합이기 때문에 이미 널리 연구되어온 조합 논리회로 테스트 방법(1,2, 9,12)에 의해 테스트되어 질지도 모르겠으나, PLA 는 메모리와 흡사한 규칙적인 구조로 인하여 조합 논리회로에서는 볼 수 없는 프로그래밍 고장(17,18)이 생기므로 비효율적이다(6). 특히 FPLA 에서는 프로그래밍 고장이 주종을 이루고 있다. (6,13,17,18).

PLA 에서의 고장을 검출하기 위하여 다음과 같은 방법들이 제안되었다. Ostapko 와 Hong (6)은 PLA 의 논리적인 구성을 AND 배열의 입력으로 2-입력 디코우더 회로를 거치는 모델을 설정하여 고장을 검출했으며, Smith (13)는 프로그래밍 고장을 위주로 test set 을 구하는 방법을 제안하였다. Agawal (17)은 완전 단일 접속 고장에 대한 test set이 대부분의 다중 고장을 검출할 수 있다는것을 보여주고 있다. 또 Lindbloom (8)은 PLA 의 AND-OR 배열에서 사용된 접점만을 고려하여 heuristic 하게 test set 을 구하는 방법을 제안하였으며, Daehn (18)은 하드웨어적으로 고장을 검출하는 방법을 제안하였다.

본 논문에서는 PLA 의 개념적인 회로를 이용하여 고장의 모형을 Stuck-at- 고장, Bridging 고장, 프로그래밍 고장등으로 확장하여 설정하고 이들을 분석하였다. 또, 이에 대한 test set을 만드는 데는 path sensitization 개념, #-연산, intersection 을 사용하여 최적의 test set을 만드는 생성 알고리즘을 제안하고, 생성되는 test set의 최대 크기를 보인다.

II. PLA의 고장 모형화

일반적인 PLA 의 구조는 입력 인버어터와 AND-OR 배열로 구성되며 여기에 출력 인버어터를 추가하기도 한다. 이러한 PLA 에서 프로그램표는 그림1.과 같으며 이에 등가한 AND-OR 배열은 그림 2.와 같다. 이러한 PLA 에서 발생되는 고장은 다음과 같이 크게 3종류로 분류되며 이들을 고장 모형으로 설정한다.

1. Stuck-at- 고장 : AND 게이트, OR 게이트 및 입력 인버어터에서의 입출력 Stuck-at-0 와 Stuck-at-1 고장이다.

2. 프로그래밍 고장
   - AND게이트 입력이 AND 게이트와 정상적으로는 연결되어야할 것이 연결되지 않은 경우
   - AND 게이트 출력이 OR 게이트와 정상적으로는 연결되어야할 것이 연결되지 않은 경우
   - AND게이트 입력이 AND 게이트와 정상적으로는 연결되지않아야할 것이 잘못 연결된 경우
   - AND 게이트 출력이 OR게이트와 정상적으로는 연결되지 않아야할 것이 잘못 연결된 경우

3. Bridging fault : 프로그래밍할 때 인접한 2개의 선이 접선되는 경우

$f_1 = x_1\bar{x}_2 + x_1x_3$

$f_2 = x_1x_3 + x_2x_3$

x	x	x	f	f
1	0	d	1	0
1	d	1	1	1
d	1	1	0	1

(d : don't care 상태)

그림 1. 프로그램표

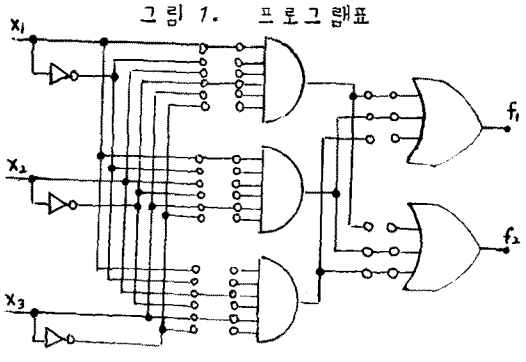

그림 2. 등가 AND-OR배열

Ⅲ. 고장 분석 및 검출 여부

논리회로에서 존재하는 고장을 검출하기 위해서는 테스트입력을 인가했을 때 그 출력치가 정상출력치와 같지 않을 경우에 그 고장은 검출 가능하며, 2개의 고장 α 및 β 에 대하여서 각각의 모든 입력들에 대해 그의 출력함수가 똑 같으면 ($f_\alpha = f_\beta$) 2개의 고장은 등가라는 고장 등가 개념(3,4,5,7,10)을 이용해서 테스트할 고장을 최적화하면 다음과 같다.

1. Stuck-at 고장에서는 AND 배열의 출력 s-a-0 고장과 입력 s-a-1 고장 및 OR 배열의 출력 s-a-1 고장과 입력 s-a-0 고장을 검출해야 한다.

2. 프로그래밍 고장에서는 AND 배열의 don't care 입력에 대해 d→1 로 만들어 테스트하고, 또 d→0 로 만들어 테스트하면 된다.

3. Bridging fault 에서는 실제 회로의 구성에 따라 AND bridging 과 OR bridging 으로 분류되며 wired logic 함수로 동작한다. 대부분의 bridging fault는 Stuck-at- 고장 검출방법과 프로그래밍 고장 검출 방법에 의해 검출되고, AND 게이트 입력에서 2개의 프로그래밍 점이 모두 연결되어지면서 AND bridging 되는 경우와 OR 게이트 입력에서 2개의 프로그래밍 점이 모두 연결되어지면서 OR bridging 되는 경우는 그 출력치가 정상 출력치와 똑 같으므로 검출 불가능하다. AND 게이트 입력에서 한 입력은 연결되고 인접한 다른 입력은 연결되지 않아야하는 곳에서 OR bridging 될 경우와 OR 게이트 입력에서 한 입력은 연결되고 인접한 다른 입력은 연결되지 않아야하는 곳에서 AND bridging이 되는 경우는 Stuck-at 고장 검출 방법이나 프로그래밍 고장 검출 방법에 의해서는 검출되지 않으므로 별도로 고려해야 되겠다. 그런데 일반적으로 bridging fault 는 실제 회로의 구성에 따라 AND bridging 이나 OR bridging 중 어느 한가지 형태의 Bridging fault만이 발생하므로(2) 이에 대한 test set 만을 구하면 될것이다.

Ⅵ. Test Set 생성 방법

Test set 을 만드는 기본 알고리즘은 path sensitization 개념을 사용하며, 이것은 고장이

발생함으로 인한 영향이 출력에 전달되도록 테스트를 생성하는 방법이다. path sensitization을 하기 위하여 Roth(14)가 처음 제안한 #-연산을 입력벡터에 반복 적용함으로서 테스트를 쉽게 얻을수 있다. 이렇게 하여 구한 test set 에 대해 intersection을 함으로서 최적의 test set 을 구할 수 있다.

PLA 의 구조가 m 개 입력, n 개 적항, l 개의 출력으로 구성되어 있다고 할 때, AND 배열을 X, OR 배열을 Y 라 하고, L 배열은 Y 배열에서 $y_j^i=1$ 일때 $y_j^k=1$ ($i\neq k$)이 되는 적항에 대응하는 X 배열의 입력벡터들이라 한다. 또 Γ 배열은 Y 배열에서 $y_j^r=1$ 이 되는 적항에 대응하는 X 배열의 입력벡터들이라 한다면 분석된 각 고장들을 검출할 수 있는 test set 은 다음과 같다.

1. T_A (AND 게이트 출력 s-a-0 를 검출하는 test set)는 AND 배열의 각 적항에 대해 $\chi^i \# L_k^i$ 로 정의된다

2. T_B (AND 게이트 입력 s-a-1 을 검출하는 test set)는 χ_j^i 가 0 혹은 1 인 entry 에 대해 $\hat{\chi}^i \# L_k^i$ 로 정의되며 $\hat{\chi}^i$ 는 χ^i 에서 χ_j^i entry 를 보수화한 것이다.

3. T_C (OR 게이트 입력 s-a-0 를 검출하는 test set)는 OR 배열의 각 entry 에 대응하는 적항에 대해 $\chi^i \# L_j^i$ 로 정의된다.

4. T_D (OR 게이트 출력 s-a-1 을 검출하는 test set)은 Y 배열의 각 열에 대해 (universal test set) $\# \Gamma_j$ 로 정의된다. 단, Universal test set 은 모든 입력조합 즉, dd.. d (m 개) 이다.

5. T_E (AND 배열의 프로그래밍 고장을 검출하는 test set)는 $\chi_j^i=d$ 일때 $\chi_j^i=1$ 로 만들어 $\chi^i \# L_k^i$ 와 $\chi_j^i=0$ 로 만들어 $\chi^i \# L_k^i$ 로서 정의된다.

6. Bridging fault 는 AND 배열에서 OR bridging된 경우는 don't care entry 를 중심으로 하여 인접해 있는 entry 를 보수화하고 don't care입력은 1로 만든것을 χ^i 라하면 test set T_R 은 $\chi^i \# L_k^i$ 로서 정의되고, 만약 OR 배열에서 AND bridging된 경우 test set T_N 은 $\chi^i \# L_k^i \# \chi^{i+1}$ (단 $y_j^i=1$ 과 $y_j^{i+1}=0$) 및 $\chi^i \# L_k^i \# \chi^{i-1}$ (단 $y_j^i=1$ 과 $y_j^{i-1}=0$)로서 정의된다.

상기의 각 test set 을 구할 때, L_k^i 의 선택은 Y 배열의 열에서 1 의 갯수가 적은 것부터 선택한다. 이렇게 하여 구한 각 test set 에 대해 intersection을 하면 최적화된다. 즉 $T=T_A \cap T_B \cap T_C \cap T_D \cap T_E \cap T_R \cap T_N$ 가 최적의 test set 이 된다.

마지막으로 모형화한 고장에 대한 최대 test set 의 크기는 $(1+\alpha)mn+nl+2m+2n+1-2$ 보다 작다는 것을 얻을 수 있다. (단, α = AND 배열에서 don't care entry의 비율 <1)

V. 결론

본 논문에서는 프로그래밍 고장은 물론 s-a-0 및 s-a-1 고장뿐만아니라 bridging fault 까지를 분석하여 검출할수 있는 알고리즘을 제안하였으며, test set 의 크기는 exhaustive test set 이 2^m개인데, 본 논문의 알고리즘을 사용하면, 최대 $(1+\alpha)mn+nl+2m+2n+1-2$ 개로 줄어든 것을 보였다. 또 test set 을 최적화하기 위해서는 반복해서 intersection 을 사용함으로서 컴퓨터 프로그래밍이 용이하도록 하였다. 즉 test set 을 생성하는데 #-연산과 intersection을 반복 적용하였기 때문에 알고리즘이 더욱 더 규칙적이고 단순하다.

참고문헌

1) A.D.Friedman and P.R.Menon, "Fault Detection in Digital Circuits" Prentice-Hall, Inc, Englewood Cliffs. 1970.

2) A.D.Friedman and M.A.Breuer, "Diagnosis and Reliable Design of Digital system". Woodland Hills. CA.Comput. Sci. 1976.

3) A.Goundan & J.P.Hayes, "Design of Totally Fault Locatable combinational Networks". IEEE Tran. on comp. Vol.

c-29. pp 33-43. Jan. 1980.

4) A.Goundan & J.P.Hayes, "Identification of Equivalent Faults in Logic Networks". IEEE.Tran. on comp. Vol.c-29 pp 978-985. Nov. 1980.

5) B.K.Roy, "Diagnosis and Fault Equivalence in Combinational Circuits" "IEEE. Tran. on comp. pp 1421-1426. Sept. 1974.

6) D.L.Ostapko & S.J.Hong, "Fault Analysis and Test Generation for Programmable Logic Arrays (PLA's). IEEE Tran. on comp. Vol c-28. pp 617-626. Sept. 1979.

7) D.R.Schertz and G.Mtze, "A New Representation for Faults in Combinational Digital Circuits". IEEE Tran. on comp. Vol c-21. pp 858-866. Aug. 1972.

8) E.B.Eichelberger & E.Lindbloom, "A Heuristic Test-Pattern Generator for Programmable Logic Arrays". IBM J.Res. Develop. Vol.24. pp 15-22. Jan.1980.

9) E.I.Muehlorf & A.D.Savkar, "LSI Logic testing-An Overview". IEEE.Tran. on comp. Vol c-30. pp 1-16. Jan. 1981.

10) E.J.McCluskey & F.W.Clegg, "Fault Equivalence in Combinational Logic Networks". IEEE.Tran. on comp. Vol c-20 pp 1286-1293. Nov 1971.

11) H.Fleisher. L.I.Maissel, "An Introduction to Array Logic" IBM J.Res. Develop. Vol. 19, pp 98-109. Mar. 1975.

12) H.Fleisher, "Introduction to Special section on Programmable Logic Arrays". IEEE. Tran. on comp. Vol c-28. Sept 1979.

13) J.E.Smith, "Detection of Faults in Pro rammable Logic Arrays" IEEE.Tran. on comp. Vol c-28 pp 845-853 Nov. 1979.

14) J.P.Roth, "Al ebraic topological methods for the synthesis of switching system I". Tran. Amer.Math.Soci. Vol88 pp 301-327 July 1958.

15) K.C.Y.Mei, "Bridging and Stuck-At F Faults". IEEE. Tran. on comp Vol c-23 pp 720-727 July 1974.

16) M.Morris Mano, "Digital Logic and Computer Design" Prentice-Hall 1979.

17) V.K.Agarwal, "Multiple Fault Detection in Programmable Logic Arrays". IEEE. Tran. on comp. Vol c-29 pp 518-522. June 1980.

18) W.Daehn & J.Mucha, "A Hardware Approach to self-testing of Large Programmable Logic Arrays". IEEE.Tran. on comp. Vol c-30 pp 829-833 Nov. 1981.