상위기능 수준에서 테스트합성 기술의 개발

신상훈°, 조상욱, 오대식, 박성주
한양대학교 전자계산학과
경기도 안산시 사1동 1271, 425-791

Development of A Test Synthesis Technique for Behavioral Descriptions on High Level Designs

Sang-hoon Shin, Sangwook Cho, Daesik Oh, and Sungju Park
Dept. of Computer Science & Engineering, Hanyang University
1271 Sa-1-Dong, Ansan, Kyunggi-Do, 425-791, Korea
shshin@mslab.hanyang.ac.kr, swcho@mslab.hanyang.ac.kr, dsoh@mslab.hanyang.ac.kr, parksj@mslab.hanyang.ac.kr

Abstract

칩의 집적도에 비례한 테스트 문제의 원초적인 해결은 VHDL등으로 기술되는 상위기능 수준에서부터 고려되어야 한다. 본 논문에서는 상위수준의 기능정보에서 테스트점을 삽입 제어흐름(control flow)을 변경하여 고집적 회로의 고장점검도를 증진시키는 기술을 소개한다. while 루프와 if-then-else 제어문에 AND 및 OR 타입 등의 테스트점을 삽입하여 내부 신호의 조정도를 최적화 시킨다. 랜덤패턴 시뮬레이션을 벤치마크 회로에 적용 각 변수의 조정도를 산출하여 테스트점의 종류 및 삽입할 위치를 결정하였다. 본 연구에서 제안하는 상대적 랜덤도에 의하여 VHDL 코드에 단일 테스트점을 삽입 합성한 결과 게이트 수준 회로에 대한 고장점검도가 최대 30%까지 증진됨을 알 수 있었다.

1. 서론

칩의 집적도에 따라 설계점검 및 칩제작후의 기능점검 등은 더욱 더 어려운 문제로 부각되고 있다. 그래서 테스트 문제의 원초적인 해결을 위하여 다양한 테스트설계 기술이 널리 개발되고 있다. 상위수준에서의 테스트합성 기술은 스캔설계를 위하여 루프(loop)를 최소화하기 위한 register-module binding방법[1], 스캔설계 대신 Ad-Hoc 방법으로 추가영역을 줄이는 방법[2], 제어기의 기능파악으로 초기화시키기 힘든 플립프롭을 대상으로 스캔하는 방법 등이 Register Transfer Level(RTL)에서 개발되었다. 또한 상위 · 하위의 테스트설계 결과를 비교 · 분석한 연구[3]도 발표되었다. 상용화된 CAD 도구에서도 테스트패턴 생성 및 테스트설계 기능을 상위수준 및 논리수준에서 별도로 행하고 있다. 일반적으로 제어부(controller) 지배적인 회로에서는 데이터 통로(data path) 보다는 제어부 부위에 테스트 기능을 추가하고[4,5] 역으로 데이터 지배적인 회로에서는 데이터 통로 부위에 테스트 기능을 추가하는 방법이 최적의 효과를 얻을 수 있다[6]. 상위수준에서 부분스캔 구현시 세 개의 플립프롭만 스캔하면 되는데 상위 합성 후 생성된 게이트 회로에서는 이십 개의 플립프롭을 스캔해야 동일한 점검도를 이룰 수 있는 경우가 보고되었다[3,7,8,9,10]. 즉, 게이트수준에서 행해지고 있는 대부분의 체계적인 테스트설계 기술은 쉽게 적용할 수는 있지만 상위수준 정보와의 결합 없이는 최적의 방법이라고 볼 수 없다. 상위수준에서의 테스트합성은 물리적 결함에 대한 기능적 고장모델의 설정이 어렵지만 제어부 및 데이터 통로 등의 기능정보와 구조정보가 복합적으로 포함되어 있다.

본 논문에서는 VHDL로 기술된 기능수준의 회로의 제어문 내부 변수들의 조정도를 랜덤패턴 시뮬레이션에 의하여 산정한 후 제어문 내부의 변수에 대하여 방문 회수에 따른 상대적 랜덤도(혹은 조정도)를 산출하고 AND, OR 및 XOR 형태의 테스트점을 삽입 논리합성 후 고장점검도를 증진시키는 기술을 소개한다.

본 논문은 다음과 같은 순서로 구성되어 있다. 2절에서는 다양한 종류의 테스트점 삽입 기술을 소개하고 3,4절에서는 조정가능도 및 상대적 랜덤도(또는 조정도)에 의하여 적절한 테스트점을 선정하고 고장점검도를 증진시킬 수 있는 기술을 살펴본다. 5절에서는 3,4절에서 제안하는 조정가능도 산출을 위한 시뮬레이션 방법을 살펴본다. 6절에서는 실험결과를 소개하고 끝으로 결론 및 향후 연구계획에 대해 기술한다.

2. 다양한 종류의 테스트점 삽입

그림 1]은 본 논문에서 삽입할 4가지 다른 종류의 테스트점을 보여주고 있다.

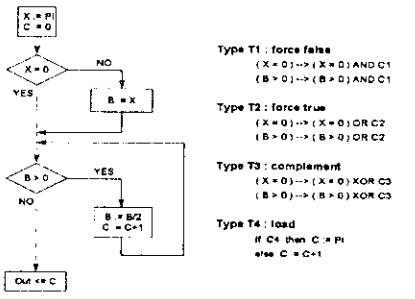

그림 1 : 테스트점 삽입

T1은 0-조정도를 증진시킬 수 있는 AND 테스트점을, T2는 1-조정도를 증진시킬 수 있는 OR 테스트점을, T3은 true와 false문 내에서 변수의 조정도를 반전시킬 수 있는 XOR 테스트점을 그리고 T4는 변수에 필요한 값을 직접 입력받을 수 있는 테스트점을 나타낸다. 그림

※ 본 연구는 정보통신연구단의 대학기초연구지원사업으로 수행하였습니다.

2]는 T2 테스트점의 게이트수준 회로도를 보여주고 있다.

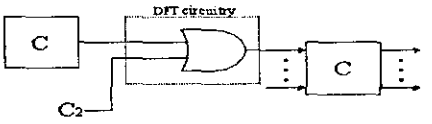

그림 2 : 게이트 수준의 테스트점 삽입

3. 테스트점 삽입에 따른 조정가능도 산출 방법

그림 3]은 최대공약수(GCD)를 구하는 회로의 기능수준 VHDL 코드와 Control Data Flow Graph(CDFG)를 보여주고 있다.

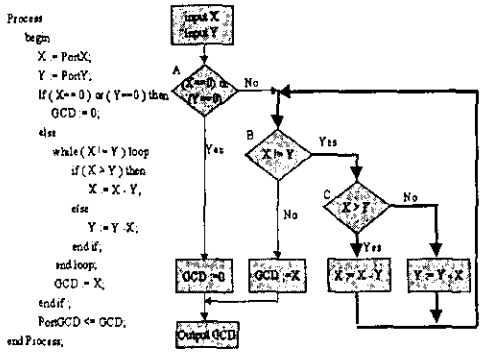

그림 3 : GCD 회로의 상위수준 기술 코드 및 CDFG

CDFG에서 A 및 C는 if-then-else를 B는 while 루프의 조건문을 나타낸다. 상위수준의 기능정보에서 제어 흐름을 증진시켜서 테스트가능도를 높이는 연구가 최근에 활발히 진행되고 있다[4,11,12]. [11]에서는 먼저 입력 단에서 출력 단에 이르는 모든 통로(path)를 열거하고 변수를 추출하며 변수(레지스터)의 크기와 정당화(Justification) 가능도를 통로에 기준 하여 측정 전체 테스트가능도를 높였다. While 루프만을 고려하는 [4]에서는 입력포트까지의 상태문장 개수를 비교하여 최대개수를 필요로 하는 B의 while 루프에 테스트점을 삽입하였다. If-then-else를 고려하는 [12]에서는 A 혹은 C의 제어문에 XOR 테스트점을 추가하였다

본 논문에서는 변수의 정당화 가능도 대신에 랜덤패턴 시뮬레이션에 의하여 조정가능도를 산출하며 while 루프만이 아닌 if-then-else문까지 고려하고 AND, OR 및 XOR 가운데 최적의 테스트점을 선별 삽입하도록 한다. 설명을 돕기 위하여 그림 3]의 소스코드에서 내부 if-then-else문을 그림 4]처럼 변경하였다. [12]에서는 조건제어문내에 관심 있는 변수 b에 대한 조정가능도를 다음과 같이 산출하였다.

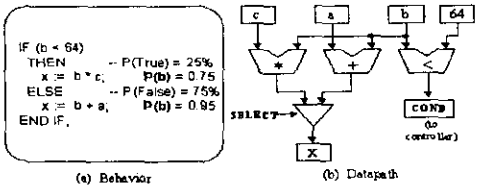

그림 4 : 기능도 및 데이터 통로도

• 방법 1: 변수 b는 8비트 레지스터로 구현될 것이며 then 이하의 true문에서는 b<64이므로 8비트 중에 상위 2비트는 항상 '00'이 되어 완전제어가 안되므로 b의 조정가능도는 개략적으로 6/8=75(%)가 된다.

상기 방법은 조정가능도(controllability)를 '0'과 '1'로 세분하지 않았으며 단지 true와 false문 내에서 변수에 대한 균등한 조정도를 유지하기 위하여 XOR 테스트점만을 사용하였다는 단점이 있다. 이러한 단점을 다음과 같은 방법으로 보완한다.

• 방법 2 : 표 1]에서 보여주는 바와 같이 b 레지스터의 각 비트에 대하여 0-조정도와 1-조정도를 산출해낸다. b 레지스터의 0-조정도는 5이며 1-조정도는 3이 되므로 그림 3]에서의 OR 테스트점을 삽입하여 1-조정도를 증진시킨다.

	b7	b6	b5	b4	b3	b2	b1	b0
0 조정도	1.0	1.0	0.5	0.5	0.5	0.5	0.5	0.5
1 조정도	0.0	0.0	0.5	0.5	0.5	0.5	0.5	0.5

표 1 : 레지스터의 각 비트에 대한 0 및 1 테스트가능도

4. 상대적 랜덤도 및 조정도 산출 방법

레지스터 단위의 조정도를 산출하기 위해서는 랜덤패턴 시뮬레이션을 하고 제어문의 방문 회수와 각 변수가 취하는 값을 관찰하면 된다. 레지스터의 랜덤도는 레지스터가 가질 수 있는 값 가운데 실제 시뮬레이션 상에서 나타나는 값의 비율로 정의한다. 테스트점을 삽입할 위치 선정과 테스트점의 종류를 결정하기 위하여 상대적 레지스터 조정도 및 랜덤도를 다음과 같이 정의한다.

- RRC : 상대적 레지스터 조정도(Relative Register Controllability)
- RRR : 상대적 레지스터 랜덤도(Relative Register Randomness)
- OC : 테스트점 삽입 전의 방문 회수(Original Counter)
- TC : 테스트점 삽입 후의 방문 회수(Test mode Counter)
- RC : 레지스터 조정도(Register Controllability)
  : $\sum 0.5 - (1 - \text{조정도})$
- RR : 레지스터 랜덤도(Register Randomness)
  : 독립적인 값 / $2^{\text{레지스터 길이}}$

$$RRC = \sum_{i=1}^{k}\left[\frac{TC}{OC} \times RC\right] \qquad (i=1,2,3,\ldots,k) \qquad [식\ 1]$$

$$RRR = \sum_{i=1}^{k}\left[\frac{OC}{TC} \times RR\right] \qquad (i=1,2,3,\ldots,k) \qquad [식\ 2]$$

```
Fancy : PROCESS
        :
  BEGIN
    :
    WHILE ((counter < b) and C1) LOOP
      WhileCounter <= WhileCounter + 1;
      ..... While Loop 회수 : 26344
      ..... While Loop 회수 : 15034
      IF (a <= counter) THEN
        TrueCounter1 <= TrueCounter1 + 1;
        ..... True 방문 회수 : OC = 9537
        ..... True 방문 회수 : TC = 4544
        temp6a := b;
        ..... RC = 0.7453 , RR = 106/256=0.4141
        ..... RC = 0.8070 , RR = 57/256=0.2227
        ..... RRC[1] = 0.3845 , RRR[1] = 0.4674
      ELSE
        :
    END LOOP;
END PROCESS Fancy;
```

그림 5 : Fancy의 테스트점 삽입에 대한 RRC와 RRR 계산 예

식 1,2]는 K개의 변수에 대한 상대적 조정도 및 랜덤도의 합을 계산

하는 식이다. 그림 5]은 테스점 삽입 전 제어문의 방문 회수 및 각 변수가 취한 값(가는 글자)에 대한 테스트점 삽입 후(굵은 글자)의 상대적인 값을 산정하기 위해 변경된 Fancy 벤치마크 회로의 소스코드와 시뮬레이션 결과를 보여준다. RRC와 RRR 계산 및 테스트점 선정에 대한 상세한 설명은 실험결과에서 하기로 한다.

원래의 회로 및 테스트점 삽입에 의하여 변경된 회로에 대한 RRC와 RRR이 계산되면 식 3]과 같이 변경에 따른 이윤 값을 산정 최대 이윤을 내는 곳에 테스트점을 삽입하도록 한다.

$$PF(\text{테스트점}) = \sum \text{조정도(변경후)} - \sum \text{조정도(변경전)} \quad [\text{식 } 3]$$

5. 랜덤패턴 시뮬레이션 방법

본 논문에서 사용한 랜덤패턴 시뮬레이션 방법은 다음과 같다. 그림 6]는 GCD회로에 랜덤패턴 생성기를 추가한 블럭도로써 변수에 대한 0-조정도와 1-조정도를 산출하기 위하여 변경 설계하였다.

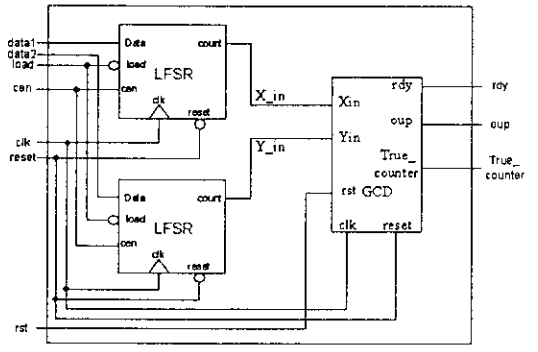

그림 6 : 랜덤패턴생성기가 추가된 GCD 블럭도

GCD 내부 제어문 주위에 출력변수를 추가하여 조정도를 측정하여 식 1,2]과 같은 상대적 조정도와 랜덤도를 산출할 수 있게 했다.

또한 기능수준의 VHDL 소스코드에 테스트점을 삽입하는 전체 알고리듬은 아래와 같다.

- *VHDL 회로에 따른 랜덤패턴 생성기 설계*
- *랜덤패턴 생성기와 VHDL 소스코드를 통합 설계*
- *조건문의 조정가능도 측정(시뮬레이션)*
- *삽입할 테스트점의 종류 결정 및 VHDL 소스코드에 삽입*
- *Synopsys CAD 도구 이용 회로 합성*
- *Synopsys CAD 도구 이용 고장점검도 및 추가영역 평가*

6. 실험결과

설계순서는 상위수준에서 기술된 회로를 분석한 다음 제어점을 삽입하고 상위수준의 합성도구를 사용하여 합성하였다. 합성된 게이트 수준의 회로도에 대하여 순차회로 테스트패턴 생성기를 사용 고장점검도의 증진여부를 검증하여 보았다.

SUN Ultra 1에서 Synopsys CAD 도구를 이용하여 VHDL 시뮬레이션, 합성 및 고장점검도를 산출하였다. 삽입하는 위치에 따라서 고장점검도의 차이가 많음을 알 수 있고 일반적으로 테스트회로를 삽입하지 않은 회로에 비하여 고장점검도가 상당히 증진됨을 표 2, 3]과 같이 알 수 있다.

테스트점	GCD		BARCODE		FANCY
	while(x>y)	while(y/=0)	while((white = 255) or (black=255))	while((actnum = num) and (white=255))	while(counter < b)
c1	73.31	77.82	84.06	70.85	81.13
c2	69.31	73.01	75.05	72.17	72.12
c3	73.36	74.83	70.87	68.88	81.63
미사용	69.40		65.68		48.15

표 2 : while 문에 테스트점을 삽입한 회로의 고장점검도 비교

테스트점		고장점검도
GCD	IF (x/=0) AND (y/=0) THEN 에 c3	71.77
BARCODE	IF video=wh THEN 에 c3	72.77
	IF flag=wh THEN 에 c3	83.69
FANCY	IF (a <= counter) THEN 에 C3	46.04
	IF (a > b) THEN 에 C3	54.76
DISPLAY	IF (secs = 9) THEN 에 C3	85.42
	IF mins = 9 THEN 에 C3	90.45
	미사용	89.04

표 3 : IF 문에 테스트점 XOR를 삽입한 회로의 고장점검도 비교

테스트점을 삽입하지 않은 GCD 본래의 회로는 69.40% 고장점검도인데 비하여 while 루프에 AND 테스트점을 삽입할 경우는 최대 77.82%의 고장점검도를 IF 문에 XOR 테스트점 삽입할 경우는 71.77%로 고장점검도가 증진되었다. BARCODE 회로도 테스트점 삽입에 의하여 고장점검도가 증진되었으나 FANCY와 Display 회로는 항상 테스트점이 고장점검도를 증진시키지는 않음을 알 수 있다. 이러한 테스트점의 종류 및 삽입할 위치를 체계적으로 설정하기 위해 랜덤패턴 시뮬레이션으로 GCD 회로 변수들의 각 비트를 관측하였다. 변수의 각 비트가 '0'과 '1'이 될 확률을 산출하고 이상적인 0.5확률과의 차이를 합산한 결과는 표 4]와 같다.

테스트점 / 변수	미사용	while(x>y) and c1	while(y/=0) and c1
x := x-y	1.4870	0.8384	1.2816
h := x	2.1422	1.1966	1.9535
x := y	1.5147	0.8305	1.3430
y := h	2.1422	1.1966	1.9535
Total	7.2860	4.0621	6.5316

표 4 : 테스트점 삽입에 의한 각 비트별 관측된 결과와 이상적인 값의 차이에 대한 합

표 4]에서 보는 바와 같이 테스트점을 삽입한 경우 좀더 균등하게 나타남을 알 수 있다. 표 5]는 루프의 방문 회수에 따른 상대적 레지스터 조정도(RRC)를 나타내며 C1 테스트점의 경우 미사용 시에 비해서 훨씬 더 조정도가 향상되었다. 표 2]을 참조하면 C1에 의하여 고장점검도가 9% 정도 증진되었으며 변수의 각 비트가 균등하게 나타남에 따라 고장점검도가 증진됨을 확인할 수 있었다

테스트점 / 변수	미사용	while(y/=0) and C1	while(y/=0) or C2
x := x-y	1.2976	1.0929	1.2976
h := x	2.2240	1.7829	3.4003
x := y	1.4424	1.0734	2.4728
y := h	2.2240	1.7829	3.4003
Total	7.1880	5.7321	10.571

표 5 : GCD의 상대적 레지스터 조정도(RRC)

FANCY	lfsr		while C1(and)		while C2(or)	
	절대적	상대적	절대적	상대적	절대적	상대적
temp6a_if	0.7453	0.7453	0.8070	0.3845	0.4741	0.5846
temp6a_else	0.1106	0.1106	0.1313	0.0819	0.1313	0.1309
temp1b_if	0.0955	0.0955	0.0745	0.0422	0.0745	0.0809
temp1b_else_if	0.4316	0.4316	0.5205	0.3945	0.5205	0.3945
temp1b_else_else	0.7453	0.7453	0.8539	0.7785	0.8539	0.8081
temp4_if	2.0000	2.0000	2.0000	1.8967	2.0000	2.0000
temp4_else	0.2599	0.2599	0.1343	0.0762	0.1491	0.1615
temp3_if_if	1.0000	1.0000	1.1176	0.6007	1.1176	0.9801
temp3_if_else	0.9577	0.9577	0.9483	0.6803	0.9483	1.8555
temp3_else	0.3744	0.3744	0.4124	0.2242	0.4124	0.3806
temp1a_if	0.1545	0.1545	0.1869	0.1063	0.2684	0.2912
temp1a_else_if	0.6724	0.6724	0.6780	0.5471	0.6780	0.5471
temp1a_else_else	0.6930	0.6930	0.8778	0.6897	0.8778	0.6897
counter_while	0.4601	0.4601	0.6761	0.3858	0.3300	0.3573
Total	8.7003	8.7003	9.4186	6.8886	8.8359	9.2620

표 6 : Fancy의 상대적 레지스터 조정도(RRC)

표 6]은 Fancy 회로에 대하여 테스트점 C1을 넣었을 때 절대적 레지스터 조정도가 미사용 시보다 나빠졌으나 상대적인 레지스터 조정도는 명확히 증진됨을 보여준다. 지금까지는 비트별로 얼마만큼 '0'과 '1'의 조정도가 균등하게 분포되는가를 국부적으로 관찰했다. 이제는 각각의 비트가 아닌 비트 전체인 변수에 대해 관찰해 본다. Fancy 회로 변수들에 대해서 상대적으로 원래의 회로보다 얼마나 다양한 값을 가지는가를 관측하였다. While 루프에서 방문되는 회수에 비해서 다양한 값을 가지게 되면 적은 시간에 많은 패턴을 발생시킬 수 있다. 따라서 이것은 적절한 제어점이 될 것이다.

테스트점 / 변 수	미사용	while and C1	while or C2
temp6a_if	9537:106/256 =0.414	4544: 57/256 =0.467	11760:116/256 =0.368
temp6a_else	16807:199/256 =0.777	10490:160/256 =1.001	16761:160/256 =0.627
temp1b_if	25932:198/256 =0.773	14673:161/256 =1.112	28149:161/256 =0.579
temp1b_else_if	95: 95/256 =0.371	72: 73/256 =0.376	72: 73/256 =0.376
temp1b_else_else	317:106/256 =0.414	289: 89/256 =0.381	300: 89/256 =0.367
temp4_if	213: 3/256 =0.012	202: 3/256 =0.012	213: 3/256 =0.012
temp4_else	26131:233/256 =0.910	14832:194/256 =1.335	28308:228/256 =0.822
temp3_if_if	2934: 25/256 =0.098	1577: 17/256 =0.124	2573: 17/256 =0.076
temp3_if_else	4203: 71/256 =0.277	3015: 58/256 =0.316	8224: 58/256 =0.116
temp3_else	19207:166/256 =0.648	10442:142/256 =1.020	17724:142/256 =0.601
temp1a_if	26143:210/512 =0.410	14873:158/512 =0.542	28360:199/512 =0.359
temp1a_else_if	145:144/512 =0.281	117:118/512 =0.286	117:118/512 =0.286
temp1a_else_else	56: 57/512 =0.111	44: 45/512 =0.112	44: 45/512 =0.112
counter_while	26344:255/256 =0.996	15034:238/256 =1.629	28521:255/256 =0.920
Total	6.492	8.713	5.620

표 7 : Fancy의 상대적 레지스터 랜덤도(RRR)

식 2]에 따른 Fancy 회로에서의 상대적 레지스터 랜덤도(RRR)는 표 7]과 같다. While 구문에 테스트점 C1을 삽입한 경우 미사용한 경우에 비해 상대적 랜덤도는 월등히 증가한다. 즉 적은 시간 내에 다양한 패턴을 많이 생성시키므로 테스트패턴 생성이 용이해 질 것이다. 반면에 테스트점 C2를 삽입한 경우는 많은 시간을 할당하여 얻어지는 패턴이 C1에 비해 상대적으로 적으므로 적절한 제어점이 되지 못함을 알 수 있다.

7. 결론 및 향후 연구 계획

본 논문에서는 기능수준의 VHDL 제어문에 테스트점을 삽입하여 합성 후 게이트수준에서의 고장점검도를 증진시키는데 목표를 두고 있다. 랜덤패턴 시뮬레이션에 의하여 0-조정도와 1-조정도를 산출하고 이윤함수를 정의하여 체계적으로 신속 정확하게 테스트점을 선정할 수 있는 기술을 제안한다. 조정도 산출을 위하여 GCD 회로에 랜덤패턴 생성기를 포함한 회로를 설계하여 시뮬레이션을 수행한다. 그리하여 변수들이 각 비트별로 '0'과 '1'이 균등하게 나타남과 변수에 대하여 방문 회수에 따른 상대적 랜덤도(또는 조정도)를 산출하고 AND, OR 및 XOR 형태의 테스트점을 삽입하였다. If-then-else에는 XOR 테스트점을, while 루프에는 AND 테스트점 주로 사용하여 상위합성 benchmark 회로에서 Synopsys CAD 도구로 합성 후 테스트패턴을 생성해 본 결과 고장점검도가 삽입 위치에 따라 큰 차이가 있음을 알 수 있었다. 본 논문에서 제안하는 상대적 랜덤도에 의하여 VHDL 코드에 단일 테스트점을 삽입 합성한 결과 게이트수준 회로에 대한 고장점검도가 최대 30%까지 증진됨을 알 수 있었다. 향후 계획으로서 테스트점의 추가 및 게이트 수준의 구조분석에 의한 부분스캔 설계와 통합 사용으로 최소 영역으로 최대의 고장점검도를 이루도록 하는 기술을 개발할 것이다.

참고문헌

[1] Mujumdar A., Saluja K. and Jain R., "Incorporating Testability Considerations in High-Level Synthesis," Proceedings, ICCAD, pp.272-279, 1992.

[2] Dey S. and Potkonjak M., "Non-Scan Design-For-Testability of RT-Level Data Paths," Proceedings, Design Automation Conf., pp.640-645, 1994.

[3] Chickermane V., Lee J. and Patel J., "A Comparative Study of DFT Methods Using High-Level and Gate-Level Descriptions," Proceedings, ICCAD, pp.620-624, 1992.

[4] Hsu F. F., Rudnick E. M., and Patel J. H., "Enhancing High-Level Control-Flow for Improved Testability," Proc. Int'l Conf. on Computer-Aided Design, pp.322-328, Nov. 1996.

[5] Dey S., Gangaram V. and Potkonjak M., "A Controller-Based Design-For-Testability Technique for Controller-Data Path Circuits," Proceedings, ICCAD, pp.534-540, Nov. 1995.

[6] Wagner K. D. and Dey S., "High-Level Synthesis for Testability: A Survey and Perspective," Proceedings, Design Automation Conf., pp.131-136, Jun. 1996.

[7] Chickermane V. and Patel J. H., "An Optimization Based Approach to the Partial Scan Design Problem," Proceedings, Int'l Test Conf., pp.377-386, Oct. 1990.

[8] Lee D. H. and Reddy S. M., "On Determining Scan Flip-Flops in Partial Scan Designs," Proceedings, ICCAD, pp.322-325, Nov. 1990.

[9] Park S. J. and Akers S. B., "A Graph Theoretic Partial Scan Design By K-Cycle Elimination," Proceedings, Int'l Test Conf., pp.303-311, Oct. 1992.

[10] Abramovici M., Kulikowski J. J. and Roy R. K., "The Best Flip-Flops to Scan," Proceedings, Int'l Test Conf., pp.116-173, Oct. 1991.

[11] Chen C., Karnik T., and Saab D. G., "Structural and Behavioral Synthesis for Testability Techniques," IEEE Trans. on Computer-Aided Design, Vol. 13, No. 6, pp.777-785, Jun. 1994.

[12] K. A. Ockunzzi and C. A. Papachristou., "Testability Enhancement for Behavioral Descriptions Containing Conditional Statements", Proceedings, Int'l Test Conf., pp.236-245, Nov. 1997.