하드웨어 명세 및 구현의 정확성 확인 방법을 위한 연구*

안영정[0], 김민숙, 방기석, 최진영

고려대학교 컴퓨터학과

{yjahn[0], mskim, kbang, choi} @formal.korea.ac.kr

Study for Validation of Hardware Specification and Implementation

Young-Jung Ahn[0], Min-Suk Kim, Ki-Seok Bang, Jin-Young Choi

Dept. of Computer Science and Engineering, Korea University

요 약

기능 검사(function simulation)란 하드웨어 시스템의 설계시 모델의 기능, 성능, 표준 준수 여부, 그리고 다른 상위 수준 조건의 관점에서 그 설계를 분석하는 중요한 설계 흐름이다. 하지만 복잡한 기존의 기능 검사의 절차는 사용자의 요구에 의해 하드웨어 시스템이 점점 복잡해지고 정보산업의 발전에 따라 개발 주기가 점점 빨라지는 시장의 특성으로 인해 설계자에게 많은 시간적 경제적인 부담감을 준다. 본 논문에서는 설계자에게 가중되는 부담을 극복하고 보다 효율적인 하드웨어 시스템의 모델링 및 기능 검사를 위해 오토마타 동치성 검사를 통한 하드웨어 시스템의 논리적 정확성 확인 방법론을 제안한다.

1. 서 론

정보 산업이 발전함에 따라 컴퓨터 하드웨어 및 소프트웨어 시스템의 개발 주기가 매우 짧아 지고 있다. 그리고 사용자의 요구가 다양해짐에 따라 기존에 개발된 시스템에 많은 부가적인 기능들이 추가되어 시스템이 점차 복잡해지는 추세이다. 이러한 발전에 따라 시스템의 정확성 확인을 위한 노력이 꾸준히 이루어 지고 있다. 기존의 전통적인 하드웨어 시스템의 설계 과정에서 보면 정확성 확인을 위해 기능 검사를 반드시 거치게 된다. 일반적으로 기능 검사는 하드웨어 명세 언어(Hardware Description Language)를 사용하여 시스템을 설계하고 시뮬레이터를 이용하여 테스트 벡터에 대한 출력 신호, 즉 파형을 검사하는 방법을 주로 사용한다. 이 경우 기능 검사를 할 설계자는 설계된 시스템에 대해 정확하게 파악하고 있어야 하고 기능 검사에 적용될 매우 많은 입력 신호 시나리오를 준비해야 하며 생성될 출력 파형에 대해 예지하고 출력된 파형을 분석해야 한다. 또한 기능 검사할 시스템이 복잡해 질수록 출력 신호에 대한 분석이 매우 어려워지게 된다. 만일 이러한 기능 검사를 통해서 오류가 발생하지 않았다 하더라도 그 결과는 입력 신호 시나리오의 종류에 민감하게 결정되며 오류를 발생하면 그 오류가 설계의 어느 부분에서 발생하였는지를 판단하기 위해서는 설계 명세 언어의 소스를 모두 대응 시켜 오류를 찾아가는 불편함이 있다. 결국, 기능 검사를 수행하기 위해 시간적/경제적인 부담이 가중된다.

본 논문에서는 기존의 기능 검사에서 오는 부담을 줄이기 위해 모델 구상 단계에서의 설계자가 구상한 모델의 오토마타와 하드웨어 명세 언어로 설계된 모델의 오토마타를 오토마타 동치성 검사[1]를 통해 논리적으로 올바른 모델이 설계되었는지를 증명하는 방법을 제안한다.

본 논문은 이 연구를 하게 된 동기와 현재까지 이와 관련되어 진행되고 있는 연구에 대해 논하고 오토마타 동치성 검사를 이용한 기능 검사를 수행하는 연구 방법과 그 결과 및 그 적용 사례를 보여준다. 마지막으로 본 연구의 결론과 향후 연구에 대해 논하면서 본 논문을 마무리 한다.

2. 관련 연구

기존의 하드웨어 정확성 확인 절차는 시뮬레이션 방법이 있다. 시뮬레이션의 절차는 다음 [그림 1]과 같다.

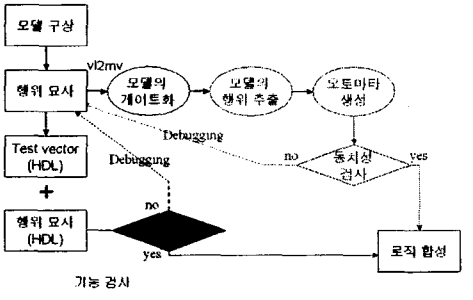
[그림 1] 시뮬레이션 절차

Simulation deriver에서는 모델에 시나리오, 즉 테스트 케이스를 추가하고 시뮬레이션 엔진을 선택한다. 시뮬레이션 엔진에는 이벤트가 발생하는 순간에 결과를 나타내주는 event driven 방식(VCS, Affirma, Verilog-XL)과 주기마다 결과를 보여주는 cycle based 방식(Cyclone-VHDL, cobra)과 두 방식 모두 적용한 hybrid 방식(VSS)이 있다.

* 본 연구는 한국과학재단 목적기초연구(R01-2000-00287)지원으로 수행되었음

3. 모델의 논리적 정확성 확인을 위한 설계 절차

일반적인 하드웨어 설계 절차의 기능 검사에서 오는 번거로움을 개선하기 위해 본 논문에서는 [그림 2]에서 보여주는 개발 절차를 제안한다. 제안된 개발 절차에서는 일반적인 개발 절차와 같이 모델 구상후 Verilog-HDL[2], VHDL 과 같은 하드웨어 명세 언어를 통해 하드웨어의 동작을 나타내는 모델을 구현한후 VIS(Verification Interacting with Synthesis) 그룹[3]에서 개발한 vl2mv(Verilog-HDL to blif-mv)[4]라는 컴파일러를 이용하여 행위 수준의 모델을 게이트 수준의 모델로 변환한다. 이때 중간 언어로는 blif-mv(Berkeley Logic Interchange Format-Multi valued Variable)[4]를 사용한다.



[그림 2] 오토마타 동치성 검사를 이용한 기능 검사 수행

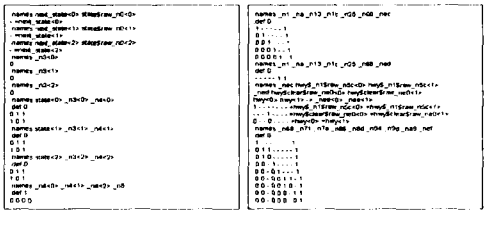
모델의 행위 추출 단계에서 게이트 수준의 모델을 BDD(Binary Decision Diagram)[5]형태로 변환한 후 깊이 우선 탐색(Depth First Search)[6]를 하면 모델의 모든 입력에 대한 모든 행위의 결과를 나타내는 행위 추출 테이블이 생성된다. 이 행위 추출 테이블은 모델이 조합 회로일때는 입력에 대한 결과로써 테스트 케이스로 이용할 수 있지만 flip-flop 이나 latch 가 이용되는 순차 회로에서는 현재 상태에서 모든 입력들을 받아 다음 상태를 나타내게 된다. 이 행위 테이블은 오토마타의 전위를 나타내며 오토마타의 전위는 오토마타의 5 개 tuple 중 초기 상태를 제외한 4 개 tuple 에 대한 정보를 가지고 있고 초기 상태는 flip-flop 이나 latch 값을 초기화하는 clear 나 reset 으로 초기 상태에 대한 정보를 추출할 수 있다. 이렇게 얻어진 오토마타의 5 개의 tuple 을 이용하여 모델을 오토마타 형식으로 바꿀 수 있다. 오토마타 생성 단계에서 얻어진 오토마타, 즉 하드웨어 명세 언어에서 추출한 오토마타와 모델 구상 단계에서의 오토마타에 대한 동치성 검사를 하는 동치성 검사 단계를 거친다. 모델 구상 단계의 오토마타는 비결정적인 유한 오토마타이고 행위 추출을 통해 얻어진 오토마타 또한 비결정적 유한 오토마타이므로 비결정적 유한 오토마타를 결정적 유한 오토마타로 변환하는 알고리즘을 이용하여 두 오토마타를 결정적 오토마타로 변환한 후에 오토마타 최소화 알고리즘을 이용하여 최소 개의 상태를 가진 결정적 유한 오토마타로 변환한다. 이렇게 변환된 두 오토마타의 동질성 검사(automata equivalence check)를 통하여 두 오토마타가 같은지 비교를 한다. 만약 구상 단계의 오토마타와 행위 추출 테이블을 통하여 생성된 오토마타가 같다면 "정규언어 L 을 받아들이는 최소 개의 상태를 가진 결정적 유한 오토마타는 유일하다"라고 증명한 [Myhill-Nerode 정리][1]에 의해서 하드웨어 명세 언어로 설계한 시스템이 설계자가 구상한 모델과 같다는 논리적인 정확성 확인 단계를 하게 된다. 그리고 구상 단계의 오토마타와 행위 추출 테이블을 통하여 생성된 오토마타가 같지 않을 시에는 틀린 부분이 쉽게 구분되고 하드웨어 명세 언어로 표현된 모델에서 틀린 부분을 쉽게 판단하여 디버깅 과정을 통해 올바른 시스템을 설계할 수 있다. 이로써 하드웨어 설계시 설계자에게 가중되는 시간적 경제적 부담을 줄이게 된다.

4. 논리적 정확성 확인 방법론 적용

본 논문에서는 간단한 하드웨어 시스템으로 제안된 하드웨어의 개발 절차에 적용하여 그 정확성을 확인하였다. 예제는 유한 상태 기계 접근을 사용해 설계한 신호등 제어기를 설계한다. 신호등 제어기는 다음과 같은 특성을 가지고 있다.

1. 주요도로는 차가 끊임없이 다니기 때문에, 주요 도로의 신호등은 가장 높은 우선 순위를 가진다. 그러므로 도로의 주요 도로의 신호는 녹색이다.
2. 간선 도로에서는 때때로 차가 교차로를 지나간다. 간선 도로의 신호등은 차가 간선 도로의 교차로에 대기중일 때 녹색으로 바뀌어야 한다.
3. 간선 도로에 차가 없으면, 간선 도로의 신호등은 노란색으로 바뀌고 다시 빨간색으로 바뀌어야 한다. 주요 도로의 신호등은 다시 녹색으로 바뀐다.
4. 간선 도로에 차가 있는지를 감지하는 센서가 있다. 센서는 제어기의 입력으로 신호 X 를 보낸다. 간선 도로에 차가 있으면 X = 1, 없으면 X = 0 이다.

위의 특성을 만족하는 신호등 시스템을 Verilog-HDL 로 명세한 후 vl2mv 를 이용해 blif-mv 로 표현한 것이다. 그림에서 나타나듯이 입력 상태와 출력 상태를 논리 테이블의 형태로 보여주게 된다. 이것을 바탕으로 게이트 수준의 시스템 모델을 할 수 있으며 입출력 테이블을 생성해 오토마타 형태의 행위 추출을 하게 된다.



[그림 3] blif-mv 로 표현된 신호등 제어기

[그림 4]은 blif-mv 파일에서 추출한 신호등 시스템의 행위 테이블이다.

Input			Output		
Clear	X	State	Hwy	Cntry	State
1	D	D	10	00	00
0	0	00	10	00	00
0	1	00	01	00	01
0	D	01	00	10	10
0	0	10	00	01	11
0	1	10	00	10	10
0	D	11	10	00	00

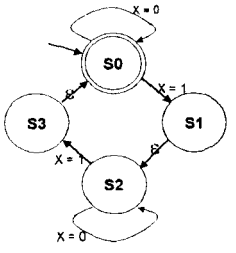
State	
00	S0
01	S1
10	S2
11	S3

Hwy, Cntry	
00	Red
01	Yellow
10	Green

D = Don't care

[그림 4] 신호등 제어기의 행위 추출 테이블

래지스터의 값을 초기화하는 Clear 가 '1'이 될 때 오토마타의 초기 상태 (q_0) 에 대한 정보를 알수있고 전위의 마지막 과정을 통해 마지막 상태 (F) 에 대한 정보를 알수 있다. 전위는 $(S0,0,S0)$, $(S0,1,S1)$, $(S1,D,S2)$, $(S2,0,S3)$, $(S2,1,S2)$, $(S2,D,S0)$ 이다. D 는 don't care 를 가리킨다. 이와 같이 추출된 오토마타의 전이, 초기 상태, 그리고 마지막 상태의 정보를 가지고 [그림 5]와 같은 오토마타를 생성한다.



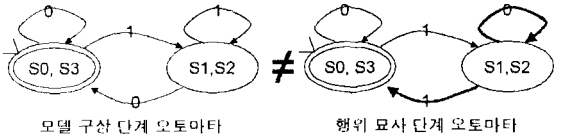
state	signals
S0	Hwy = G Cntry = R
S1	Hwy = Y Cntry = R
S2	Hwy = R Cntry = G
S3	Hwy = R Cntry = Y

[그림 5] 행위 추출 테이블로 생성된 오토마타

[그림 5]에서 생성된 비결정적 오토마타를 결정적 오토마타로 변환한 후 오토마타를 최소화 시켜 발생한 두 오토마타가 동일한지를 확인한다. [그림 6]의 오토마타 동치성 검사를 통하여 두 오토마타가 완전히 다른 행위를 하는 것을 확인할 수 있다. 그리고 어느 상태에서 오류가 발생하는지를 한 눈에 확인 할 수 있다. 따라서 매우 쉽게 소스 코드를 확인해 디버깅까지 수행할 수 있다.

I		II	
S2	S1,S2	S0	S0,S3
(II, I)	(II, I)	(II, I)	(II, I)
S2	S1,S2	S0	S0,S3

I		II	
S2	S1,S2	S0	S0,S3
(I, II)	(I, II)	(II, I)	(II, I)
S2	S1,S2	S0	S0,S3



모델 구상 단계 오토마타 ≠ 행위 묘사 단계 오토마타

S1,S2의 천이부분에 문제 발생

[그림 6] 오토마타 동치성 검사

5. 결론 및 향후 연구

하드웨어 시스템 설계 분야에서 모델링, 검증, 논리 합성, 그리고 EDA 기반 방법론에 대해서 EDA 업체들에 의해 연구가 활발히 진행되고 있다. 그러나 기능 검사를 위한 도구 개발이나 연구들은 초기의 환경을 벗어나지 못하고 있다. 입력 시나리오에 대한 테스트 케이스 생성으로 하드웨어 시스템에 대한 기능 검사를 하는 연구가 비용이 싸기 때문이다. 이러한 기능 검사는 사용자의 요구에 의해 하드웨어 시스템이 점점 복잡해지고 정보산업의 발전에 따라 개발 주기가 점점 빨라지는 시장의 특성으로 인해 설계자에게 많은 시간적 경제적인 부담감을 준다. 이러한 부담을 줄이기 위해 많은 EDA 업체들은 시뮬레이터 가속기와 같은 도구를 시장에 내놓고 있지만 여전히 설계의 정확성 확인 절차는 설계자에 의해 판단이 되도록 하는 틀을 벗어나지 못하고 있다. 본 연구에서는 설계자에게 가중되는 부담을 극복하고 보다 효율적인 하드웨어 시스템의 모델링 및 기능 검사를 위해 오토마타 동치성 검사를 통한 하드웨어 시스템의 논리적 정확성 확인 방법론을 제안하고 있다. 오토마타 동치성 검사를 통한 하드웨어 시스템의 논리적 정확성 확인 방법론은 다음과 같은 특징을 지닌다.

첫째, 하드웨어 시스템의 정확성 확인을 절차를 자동으로 수행할 수 있는 방법론을 제시함으로써 기존의 하드웨어 시스템 개발 절차를 간결하게 하였다.

둘째, 추출된 명세를 보면 설계하고자 하는 시스템의 동작을 쉽게 파악하게 한다.

셋째, 잘못된 동작이 파악될 경우 하드웨어 명세 언어의 오류 지점을 쉽게 파악함으로써 수정이 매우 용이하게 이루어지도록 도와준다. 또한 기능 검사의 비용 절감에 기여한다.

네째, BDD를 표현하는 중간 언어로 blif-mv를 선택함으로써 이 언어를 받아들이는 CTL Model Checker VIS를 이용하여 요구 명세와 설계 명세에 대한 검증에도 적용할 수 있다.

그러나 지금 현재는 설계의 행위를 추출하기 위해서 BDD를 이용하는데 만약 시스템이 복잡할 경우 상태 폭발에 따른 메모리의 한계에 부딪히게 된다.

향후 연구 과제로는 본 논문에서 제시한 방법안을 구현하여 다른 툴과의 실험적 결과를 비교하고 좀더 최적화된 알고리즘을 이용하여 모델가능한 상태의 크기를 확장 시켜주는 연구가 필요하다.

참고 문헌

[1] Harry R. Lewis, Christos H. Papadimitriou, *"Elements of the Theory of Computation, 2nd ed."*, Prentice-Hall, Upper Saddle River, New Jersey, 1998.

[2] Bob ZeidMan, *"Verilog Designer's Library"*, prentice-Hall,Inc., Upper Saddle River, New jersey, 1999.

[3] Tiziano Villa,Gitanjali Swamy,Thomas Shiple, *"VIS User' s Manual"*, University of California, Berkeley,1996.

[4] Yuji Kukimoto, BLIF-MV,The VIS Group,University of California, Berkeley, May 31, 1996.

[5] R. E. Bryant. Graph-based Algorithms for Boolean Function Manipulation. IEEE Transactions on Computers, Vol. 35, No. 6, pp.677-691, August 1986.

[6] Horowitz, Sahni, Anderson - Freed, *"Fundamental Data Structures in C"*, computer science press, 1993.