DOI: 10.3745/KIPSTD.2010.17D.2.111

모델 검사를 위한 Simulink 디버거의 기능 개선

김 성 조[†] · 이 홍 석[††] · 최 경 희[†††] · 정 기 현[††††]

요 약

본 논문에서는 Simulink로 모델 검사를 위한 향상된 기능을 가진 디버거의 구현에 대해 기술한다. Simulink에서 기본적으로 제공되는 디버그 기능은 복잡한 시나리오나 복잡한 모델을 검사할 때 단순 반복적인 작업이 다수 요구되었다. 이를 개선하기 위해서 본 연구에서는 임의의 시나리오에 따른 시뮬레이션 결과와 예상한 결과를 확인할 수 있는 기능, 원하는 시점에서 시스템의 변화를 확인하는 기능, 임의의 혹은 전체 시나리오에 대한 시스템의 Coverage Report 기능 등이 구현된 Simulink 디버거에 대해 소개하고 구현에 대한 이슈를 기술한다. 이 프로그램을 Matlab에서 제공하는 자판기 모델에 적용해서 그 유용성을 확인했다.

키워드 : 모델 검사 도구, Simulink, 모델링, 디버깅 도구

Improvement of a Simulink Debugger Capacity for Model Verification

SeongJo Kim[†] · Hongseok Lee[††] · Kyunghee Choi[†††] · Kihyun Chung[††††]

ABSTRACT

In this paper, we describe the implementation of debugger that has advanced features for verifying Simulink model. The debugger provided in Simulink has some boring and repetitive work when verifying complicated Simulink models or complicated scenarios. In order to resolve the problems, this paper addresses the issues on the implementation of debugger that provides features such as a convenient feature to compare the simulation output to the expected output for specific input, to monitor system's behavior at specific time, and coverage report function in some or all input scenarios. The proposed debugger is applied to the vending machine model provided by Matlab, demonstrating its feasibility.

Keywords : Model Checking Tool, Simulink, Modeling, Debugging Tool

1. 서 론

Matlab은 알고리즘을 검증하는데 유용한 도구로서 그 중에서 Simulink는 IP(Intellectual Property) 형태의 수많은 컴포넌트를 가지고 블록 다이어그램 형태의 설계가 가능하다[5]. 이와 같은 장점은 Matlab/Simulink가 시스템 모델링의 많은 부분에서 사용되도록 한다. 최근의 소프트웨어 개발은 알고리즘 검증 단계에서의 Simulink 모델을 그대로 HW 설계 및 SW 설계에서 사용하고 최종적으로 검증에까지 사용하는 추세이다[1].

그러나 사용자가 Simulink로 모델링을 할 때에는 다음과 같은 문제점을 가지고 있다[2]. 첫째, Simulink에서 기본적으로 제공하는 디버거는 모든 실행 주기에 대해서 Chart Entry, Event Broadcasting, State Entry에서 Breakpoint를 설정할 수 있도록 제공한다. 그런데 모델 검사를 목적으로 하는 사용자는 모든 시점에 대해 모델의 동작을 확인하고 싶은 것이 아니라 특정 시점에서 모델의 동작을 확인하고 싶은 경우가 있다. 이런 경우에 기존의 디버거를 활용하여 검사를 하는 것은 매우 불편한 일이다. 둘째, Simulink는 기본적으로 모델을 디자인하고 시뮬레이션 하는 도구이기 때문에 모델이 올바르게 동작하는지 확인하기 위해서 입력에 따른 예상된 결과를 확인하는 기능이 없다. 그런데, 모델을 디자인하다 보면 불가피하게 잘못 기술하게 되기 쉬운데, 이를 검사하기 위해서는 특정 시나리오에 따라 모델이 예상된 결과대로 나오는지 확인해야 할 필요가 있다. 현재 Simulink에서는 이런 기능을 별도로 제공하지 않는다.

이와 같은 문제점을 해결하기 위해서 본 연구에서는 작성된 모델의 검사를 위해 다음과 같은 기능을 가진 디버거를 제안하고 구현 이슈들에 대해 기술한다.

† 준 회 원 : 아주대학교 전자공학과 석사과정
†† 준 회 원 : 아주대학교 전자공학과 박사과정
††† 정 회 원 : 아주대학교 정보통신전문대학원 교수
†††† 정 회 원 : 아주대학교 전자공학부 교수

논문접수 : 2009년 6월 5일
수 정 일 : 1차 2010년 2월 11일, 2차 2010년 3월 22일
심사완료 : 2010년 3월 22일

1) 모델 검사 기능 - 임의의 시나리오와 그에 따른 예상 결과 데이터를 읽어서 모델을 시뮬레이션 하면서 그 결과가 맞는지 틀렸는지의 여부를 비교할 수 있다.
2) 특정 시점에 대한 시뮬레이션 잠시 멈춤 기능 - 특정 시간의 모델의 동작을 확인하기 위해서 시간 별로 시뮬레이션 잠시 멈춤을 설정하거나 해제할 수 있다.
3) 모델 검사한 데이터의 Coverage Report 기능 - 임의의 혹은 전체 시나리오가 어떤 테스트 Coverage를 만족하는지 확인할 수 있다.

이 논문의 중요한 공헌은 모델링, 시뮬레이션에서 강력한 기능을 제공하는 Simulink의 약점인 모델 디버깅을 위한 일부 기능을 본 연구에서 제안한 방법을 통해서 극복하였으며, Simulink모델 기반의 테스팅이나 모델 검사를 더욱 편리하게 할 수 있도록 한 점이다.

이 논문의 구성은 다음과 같다. 2장에서는 Simulink/Stateflow가 무엇인지 간략하게 소개를 하고, Simulink를 사용한 관련연구에 대해서 기술한다. 3장에서는 기존의 Simulink 및 기존 관련도구와 본 연구의 Simulink 디버거의 차이점을 기술한다. 4장에서는 본 논문에서 제시한 모델 검사를 위한 향상된 Simulink 디버거의 기능 구현을 설명한다. 4.1절에서 모델 검사 기능을 설명하고, 4.2절에서 이 때 예로 사용된 자판기 모델에 대해 기술한다. 4.3절에는 임의의 시점에 시뮬레이션 중인 Model을 잠시 멈추는 방법에 대해서 설명하고, 4.4절에는 이 도구에서 제공되는 Coverage 범주 및 Coverage Report에 대해서 설명할 것이다. 그리고 마지막으로 5장에서는 결론으로 끝을 맺는다.

2. Simulink/Stateflow 및 관련 연구

Simulink/Stateflow는 시스템을 모델링, 시뮬레이션 및 분석하는 도구로 선형 시스템과 비선형 시스템을 지원한다. 또한 연속적인 시간으로 모델링 되어 있는 시스템이나 혹은 비연속적으로 모델링 되어 있는 시스템을 모두 지원한다. Simulink를 사용하여 사용자는 프로그램에서 제공하는 다양한 블록 또는 사용자 임의의 블록을 사용하여 모델링을 할 수 있다. Simulink는 시스템 개발에 필요한 다양한 기능을 제공한다. 산업 분야별로는 항공우주, 자동차, 생명공학, 의학, 통신, 전자, 금융 서비스, 산업 자동화, 반도체 등 사용되지 않는 곳이 없을 정도로 광범위하게 사용되는 매우 강력한 툴이다.

Simulink로 시스템을 모델링하여 시뮬레이션 하는 방법은 두 단계를 거친다. 첫 번째 단계는 도구에서 제공하는 블록을 사용하여 모델의 그림을 그리는 것이고, 두 번째 단계는 모델의 입력단을 통해 임의의 입력을 넣어서 모델이 계산한 결과를 분석하는 단계이다. 시뮬레이션 한 결과는 시간에 따른 값으로 나오며 그 결과를 시각화 할 수 있는 블록이 제공되어 바로 확인할 수 있다.

Stateflow는 Harel이 1987년 제안한 Statechart[3]의 Simfulink버전으로 시스템을 유한 상태 기계로 표현하고자 할 때 사용되는 방법이다[6]. Stateflow는 Simulink내에 포함되어 있는 개념으로 Simulink내의 Stateflow블록으로 표현 된다. Stateflow는 이벤트에 의해 시스템의 상태가 결정되는 reactive system을 모델링하기에 적합한 도구이다. 유한 상태 기계는 동작하는 모델을 상태로 표현하며, 그 상태가 지속되고 있을 때 혹은 어떤 상태에서 다른 상태로 전이가 일어남에 따른 시스템의 행동을 표현할 수 있다. Statechart와 비슷하게, Stateflow는 전통적인 상태 다이어그램에서 아래의 5가지 기능들이 추가되었다[7]. 첫째, 계층적 표현이 가능해졌다. 둘째, 병렬적 표현 기능을 지원한다. 셋째, Matlab언어를 사용하여 함수를 사용할 수 있다. 넷째, 테이블이나 진리 표(truth table)를 삽입할 수 있는 기능을 제공한다. 그리고 마지막으로 Temporal logic에 대한 기능을 지원한다.

3. Simulink 및 기존 관련 도구와 개발된 Simulink 디버거의 비교

Simulink는 모델링 및 시뮬레이션 하는 도구로 여러 분야에서 사용되고 있다. 하지만, Simulink의 부족한 점을 보완하기 위해 다른 도구와 연동하거나 혹은 모델링을 Simulink로 하고 그 모델을 다른 도구에서 사용하기 위해 인터페이스 하거나 혹은 다른 도구에서 사용할 수 있도록 새로운 모델을 생성하는 연구들이 있다.

Simulink를 다른 도구와 연동하여 모델링 한 연구로 [10]에서는 6의 자유도를 가지는 모델에 대해 Simulink와 ADAMS를 모델링 도구로 같이 사용하여 중력을 보상한 PID제어에 대한 연구가 있다. [11]에서는 Simulink와 다른 프로그램과의 인터페이스를 하는 연구를 했는데, Simulink로 표현한 모델과 전력 시스템을 계획하는 도구인 PSS/E라는 프로그램과 인터페이스를 하여 교류 발전 및 관리시스템을 모델링하고 co-simulation하는 연구를 진행했다. [12]에서는 모델 기반의 테스팅을 위해 Simulink모델을 이용하였으며, Simulink 모델에 존재하지 않는 요구사항의 기능을 보완하기 위해 T-VEC이라는 프로그램에서 Simulink모델과 요구사항을 관리하며, 이를 바탕으로 테스트 케이스를 자동으로 생성하는 연구를 진행하였다. 그리고 Simulink모델을 기반으로 다른 도구에서 사용할 수 있도록 모델을 변환한 연구가 있는데, [13]에서는 Matlab/Simulink모델로부터 SystemVision에서 사용되는 VHDL-ASM 모델을 생성하는 연구를 진행했다. 본 연구에서는 다양한 분야에 쓰이는 Matlab/Simulink 모델의 입출력 데이터의 포맷을 각 시스템에 특정하지 않은 범용적인 형태로 주고 받을 수 있게 하였다. 범용적인 형태의 입력과 예상되는 출력을 넣어주면, 향상된 Simulink 디버거에서 예상되는 출력이 달라지는 곳을 표시하여 쉽게 알 수 있게 하였다.

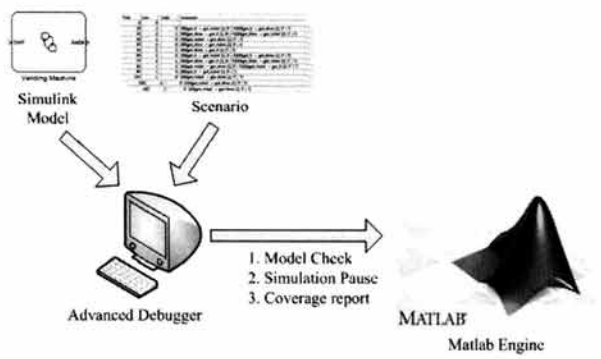
Simulink 모델을 수행한 이후에 Simulink에서 제공되는

디버거를 이용했을 때, 모든 실행 주기의 Chart Entry, Event Broadcasting, State Entry에서 Breakpoint를 설정하게 된다. 이 때 모든 실행 주기가 아닌 특정 시간에 모델을 멈출 수가 없으며, 이러한 기능을 제공해주는 도구를 발견하지 못하였다. 본 연구에서는 Simulink의 특정 블록을 응용하여 본 기능을 구현하였다.

Simulink 모델을 수행시킨 후에 사용된 입력들이 어떤 테스트 Coverage를 만족시키는지 계산을 수행하며, 더 높은 Coverage를 구하기 위한 다양한 방법들이 연구되고 있다[14]. 이 때 Simulink에서 제공되는 Coverage의 계산을 편리하게 수행해 주는 도구를 발견하지 못하였다. 본 연구에서는 복수 개의 입력 Set을 설정해주면 해당 입력 Set들을 차례대로 모두 수행시켜 누적된 Coverage를 편리하게 구할 수 있게 하였다.

4. 향상된 Simulink 디버거

본 연구에서의 Simulink의 모델 검사 기능을 보완하기 위한 프로그램의 전체 구조는 (그림 1)과 같다. 입력 데이터로 Simulink 모델 파일과 그 모델의 상태를 변화시키기 위한 시나리오 파일을 필요로 한다. 시나리오 파일에는 입력의 시퀀스만 존재할 수도 있고, 추가적으로 입력에 따른 예상되는 출력 시퀀스가 존재할 수도 있다. 입력의 시퀀스만 존재하는 경우는 단지 모델을 시뮬레이션 하기 위한 용도로 사용될 수 있으며, 입력에 따른 예상되는 출력 시퀀스 파일은 모델을 검사하기 위한 용도로 사용될 수 있다. 향상된 Simulink 디버거의 기능은 Model 검사기능, 임의의 시점에 시뮬레이션 중인 모델의 중지 기능, 시나리오에 대한 Coverage 기능이 있다. 이 기능에 대한 설명을 위해 본 연구에서는 Matlab에서 제공하는 자판기 모델[8]을 적용한 것을 설명할 것이다.



(그림 1) 향상된 Simulink 디버거의 프로그램 구조

4.1 Model 검사기능

Simulink에서 Model의 동작이 맞는지 틀리는지의 여부를 확인하기 위해서 Model에 입력을 넣어주어야 하는데, Model에 입력을 넣어주기 위한 방법은 두 가지가 있다. 첫 번째 방법은 Signal Builder블록을 사용하는 방법이다. Signal Builder블록을 이용하는 것의 장점은 간단한 조작으로 쉽게 입력이 가능하다는 점이다. 하지만, Signal Builder의 단점은 외부에서 생성한 시나리오의 경우 Signal Builder를 이용하여 변환하는 것이 불가능하고, 새로운 시나리오가 추가되는 상황일 경우 데이터를 변경하는 것이 쉽지 않다. 그리고 입력 data의 내용을 별도로 보관하는 것이 불가능하다.

Model에 입력을 넣어주기 위한 두 번째 방법은 Inport블록을 이용하여 시스템에 입력을 넣는 방법이다. Inport블록을 이용하기 위해서는 약간의 설정을 필요로 하는데, Simulink 메뉴의 Configuration Parameters의 Data Import/Export설정을 변경하고 Import할 때의 입력 변수 이름을 지정하면 되는데, 이는 구현한 외부 프로그램 상에서 조작이 가능하다. 이 방법의 장점은 외부에서 작성된 시나리오를 불러들여서 사용할 수 있다는 점과, 데이터의 편집이 용이하며, 모델과 별도로 시나리오를 저장, 보관할 수 있다는 장점이 있다. 본 연구에서는 두 방법 중에서 Inport블록을 이용하는 접근 방법을 택하였다.

Simulink는 모델을 시뮬레이션 할 때 고정된 시간을 주기로 수행하는 방식이 있고, 고정되지 않은 시간을 주기로 수행하는 방식이 있다. 고정된 시간을 주기로 수행하는 방식은 이산적인 성격을 가지는 시스템을 대상으로 사용되어야 하며, 수행 주기가 고정적이지 않는 시뮬레이션 방식은 연속적인 성격을 가지는 시스템을 대상으로 사용되어야 한다. 만약 이산적인 성격을 가지는 시스템을 고정되지 않은 시간을 주기로 수행하는 시뮬레이션을 수행하게 되면 모델을 작성한 사용자의 의도대로 동작하지 않을 수 있다. 본 연구에서는 고정적인 시간을 주기로 수행하는 방식에 대해서만 지원한다. 즉 모델을 실행하기 위한 타입은 Fixed-step이어야 한다.

본 연구에서 개발한 프로그램에서 모델 검사를 위해 로드되어 사용되는 데이터에는 {Fixed Step size, Test case Set}의 내용이 들어있다. Fixed Step size는 모델을 시뮬레이션 시킬 때 수행하는 주기이다. Test case Set은 두 부분으로 구성된다. 첫 번째는 {Time_Step, Input_Name(1), ... ,Input_Name(n), Blank, Output_Name(1), ..., Output_Name (m)}으로 Time Step 및 각각의 입출력 변수의 이름을 입력하는 부분이고, 두 번째는 {Time_Value, ValueInput(1), ... ,ValueInput(n), Blank, ValueOutput(1), ... ,ValueOutput(m), Comment}의 리스트로 구성되어 있다. Time_Value는 각 변수의 값이 시스템의 입력으로 들어가는 시간을 의미하고, ValueInput(k)는 Input_Name(k)가 시스템의 입력으로 들어가는 값을 의미하고, ValueOutput(m)는 입력 값에 따른 시스템의 예상되는 Output_Name(m)의 값을 나타낸다. 그리고 Comment는 해당 입력에 대한 주석을 위해 사용자가 입력할 수 있는 부분이다. 그리고 입력 변수와 출력 변수에 대한 것을 구별할 수 있도록 하기 위해서 입력 변수들과 출력 변수들 사이에는 공백으로 분리되어 있다. 위 내용에 대한 입력 예제는 (그림 2)와 같다.

모델은 요구사항을 명확히 정한 후에 디자인 단계에서 만

들어지게 된다. 이 때 초기에 만들어진 모델이 요구사항을 정확하게 반영하고 있는지를 검사하기 위해서 다양한 시나리오를 모델에 넣어서 돌려보게 된다. 이러한 시나리오의 생성은 [12]에서 T-VEC라는 프로그램을 이용하여 자동으로 생성한 사례가 있다. 본 연구의 Simulink 디버거는 이러한 자동화 툴에서 생성된 시나리오, 수동으로 생성된 시나리오 모두 (그림 2)와 같은 데이터 형태로 넣어주면 동작이 가능하다.

(그림 2)에서 Time Step이 10이란 말의 의미는 모델을 Fixed Step으로 돌리며 10의 주기를 가지고 수행하라는 의미이다. 이 설정은 Simulink의 Configuration parameter창에서 변경하게 되는데, 외부 프로그램에서 이 설정을 제어할 수 있다. Configuration parameter에 대한 그림은 (그림 3)과 같다. 모델의 검사 방법은 시뮬레이션을 수행한 결과와 시나리오에 적힌 예상된 결과를 비교하여 결과가 다르면 그 부분을 표시하는 방식이다. 만약 예상되는 결과에 대하여 확인하고 싶지 않을 경우에는 예상되는 출력의 값에 아무 값도 적어주지 않고 빈 공란으로 남겨놓으면 되는데, 프로그램이 그 시간에 예상되는 출력의 결과에 대해서는 어떤 값이 나와도 상관없다는 의미로 인식하여, 예상되는 결과와 실제값을 비교하지 않는다.

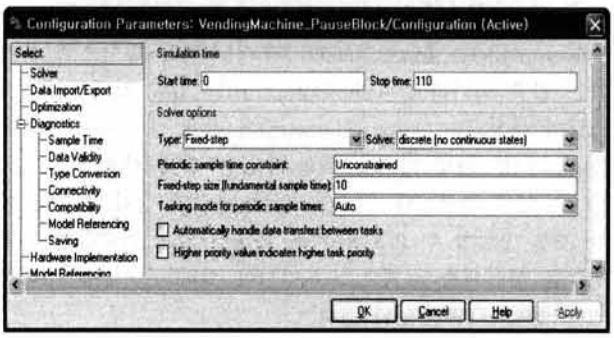
자판기 모델을 예로 들어서 설명하면 (그림 2)에서 Test Case #1은 테스트 시나리오의 첫 번째 이름을 나타내고, Test Case #2는 테스트 시나리오의 두 번째 이름을 나타낸다. 각각의 테스트 시나리오 내에서는 입력되는 시간의 값, 입력 변수의 값, 예상되는 출력 변수의 값, 그리고 주석으로 이루어져 있다. 그래서 (그림 2)의 첫 번째 테스트 시나리오인 Test Case #1에서는 시간이 10일 때 coin을 2로 입력하게 되며, 그에 따른 예상 결과는 soda가 0이 된다는 의미이다.

Time Step 10

Test Case #1

Time	coin	soda	Comment
0	0		
10	2	0	$$$got_0 -> got_nickel [1] (F \| F)$$$got_0 -> got_dime [2] (T | T)
20	2	1	$$$got_dime -> got_0 [1] (F \| F)$$$got_dime -> got_nickel [2] (T | T)
30	1	0	$$$got_nickel -> got_dime [1] (T | T)
40	3	0	$$$got_dime -> got_nickel [2] (F | F)
50	1	1	$$$got_dime -> got_0 [1] (T | T)
60	2	0	$$$got_0 -> got_nickel [1] (F \| F)$$$got_0 -> got_dime [2] (T | T)
70	2	1	$$$got_dime -> got_0 [1] (F \| F)$$$got_dime -> got_nickel [2] (T | T)
80	2	1	$$$got_nickel -> got_dime [1] (F \| F)$$$got_nickel -> got_0 [2] (T | T)
90	1	0	$$$got_0 -> got_nickel [1] (T | T)
100	1	0	$$$got_nickel -> got_dime [1] (T | T)

(그림 2) 테스트 시나리오 예제



(그림 3) Configuration Parameter 화면

(그림 2)와 같이 저장된 테스트 시나리오를 모델에서 실행시킬 수 있는 구조체의 형태로 변화시켜 주어야 한다. 이 구조체의 이름을 'indata'로 정의하였다. (그림 4)는 작업을 수행하는 pseudocode이다.

(그림 4)의 pseudocode를 보면 먼저 테스트 시나리오 파일을 로드한다. 이 때 같은 시간에 들어가는 입력의 개수와 입력되는 시간대의 개수를 저장해준다. 그림 2의 테스트 시나리오에서는 입력의 개수는 1이고 입력되는 시간대의 개수는 10이다. 그 후에 로드된 Simulink 모델 파일에서 Inport 이름들을 저장한다. Simulink 모델을 구동시키는 데이터를 만들 때 시간에 따른 입력의 순서는 Simulink 모델의 Inport 이름의 순서를 따르게 된다. 이렇게 만들어진 시간에 따른 데이터를 Simulink의 Inport 이름의 순서에 따라서 넣어준 후에, 데이터의 진행방향을 transport 해서 Base Workspace에 넣어준다. 마지막으로 모델을 구동시킬 총 시간값을 읽어서 모델의 구동시간으로 셋팅한다. (그림 4)의 작업들은 Matlab의 스크립트 파일에서 수행하게 된다. 본 연구에서는 C#을 기본으로 하여 Matlab의 Engine을 호출해서 (그림 4)와 같은 기능을 수행하는 Matlab의 스크립트 파일을 실행시키는 방법으로 Simulink에 접근하였다.

위의 입력된 파일을 프로그램에서 불러들인 화면이 (그림 5)와 같다. FileName은 불러들인 파일의 이름을 나타내고 위 예에서는 Vending_Maching_TC.csv파일이다. 그리고 No는 파일 내의 테스트 시나리오의 번호를 의미하는데, 차례대로 숫자를 매긴다. 그리고 Step#는 각 테스트 시나리오 별로 입력된 "시간, 입력 값들, 예상되는 출력 값들"의 레코드의 개수를 나타낸다.

(그림 5)의 테스트 시나리오 중 첫 번째 시나리오를 선택해서 수행한 결과가 (그림 6)과 같다. 수행 결과 8번째 step에서 모델의 예상 수행 결과와 실제 시뮬레이션 결과가 다르게 나와서 틀린 부분이 색으로 칠해져 있다. 이와 같이 모델의 예상된 결과와 실제 수행된 결과가 다르게 나올 경우 해당 부분에 대한 모델의 움직임을 관찰해 볼 필요가 있는데, 이 방법에 대한 내용은 4.3절에서 다룬다.

1. Open Excel File
2. Read Simulink Input
3. Determine MainTimeTerm
4. Make indata
5. Mapping InputPort with InputData
6. Transfer indata
7. Copy data to Base Workspace
8. Set StopTime

(그림 4) Indata로 변환하는 pseudocode

FileName	No	Step #
		19
Vending_Machine_TC.csv	1	11
Vending_Machine_TC.csv	2	3
Vending_Machine_TC.csv	3	5

(그림 5) 테스트 시나리오 파일을 로드한 화면

Step	Time	input coin	Expected_Results soda	Simulated_Results soda	Information
1	0	0		0	
2	10	2	0	0	$$$got_0 -> got_nickel [1] (F \| F)$$$got_0 -> got_dime [2] (T | T)
3	20	2	1	1	$$$got_dime -> got_0 [1] (F \| F)$$$got_dime -> got_nickel [2] (T | T)
4	30	1	0	0	$$$got_nickel -> got_dime [1] (T | T)
5	40	3	0	0	$$$got_dime -> got_nickel [2] (F | F)
6	50	1	1	1	$$$got_dime -> got_0 [1] (T | T)
7	60	2	0	0	$$$got_0 -> got_nickel [1] (F \| F)$$$got_0 -> got_dime [2] (T | T)
8	70	2	1	0	$$$got_dime -> got_0 [1] (F \| F)$$$got_dime -> got_nickel [2] (T | T)
9	80	2	1	1	$$$got_nickel -> got_dime [1] (F \| F)$$$got_nickel -> got_0 [2] (T | T)
10	90	1	0	0	$$$got_0 -> got_nickel [1] (T | T)
11	100	1	0	0	$$$got_nickel -> got_dime [1] (T | T)

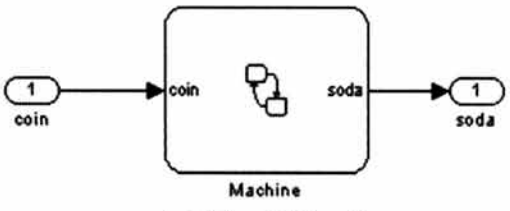
(그림 6) 그림 5의 첫 번째 테스트 시나리오에 대한 화면

4.2 자판기 모델

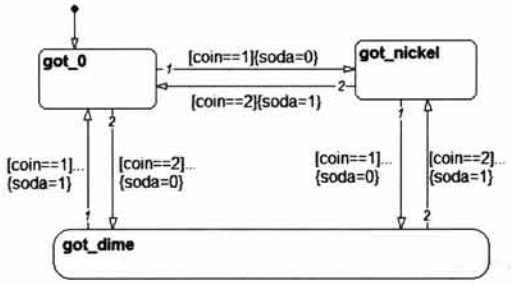
(그림 6)의 테스트 시나리오가 적용된 자판기 모델의 최상위 모델은 (그림 7)과 같고, Machine 차트의 내부는 그림 8과 같다. 입력은 0, 1, 2가 있을 수 있다. 0은 동전을 넣지 않는 것이고, 1은 5센트를 기계에 넣는 것이고, 2는 10센트를 기계에 넣는다는 의미이다. 출력은 0 또는 1이 있을 수 있는데, 0은 음료수가 나오지 않는다는 의미이고, 1은 음료수가 나온다는 의미이다.

(그림 8)의 Machine차트에서 got_0, got_nickel, got_dime 상태는 현재 자판기에 입력된 현재 금액이 각각 0, 5, 10센트라는 의미이다. 투입된 동전의 총 합이 15센트가 넘으면 soda가 1이 되며, 그 금액이 차감된 상태가 기계의 상태가 된다. 즉 got_0상태에서 coin이 1인 경우(5센트 입력) got_nickel상태로 전이하고, coin이 2인 경우(10센트 입력) got_dime상태로 전이한다. soda는 어떤 상태로 전이가 일어나도 값이 0이 된다. got_nickel상태에서 coin이 1인 경우 got_dime상태로 전이가 일어나며, 이때의 soda는 여전히 0인 상태가 된다. 그러나 coin이 2인 경우 got_0상태로 전이가 일어나게 되며 soda는 1이 된다. got_dime상태에서 coin이 1인 경우 got_0상태로 전이가 일어나며 soda는 1이 되고, coin이 2인 경우 got_nickel상태로 전이가 일어나고, soda는 1이 된다.

(그림 7)과 같은 일정한 특성을 가진 모델을 작성한 후에는 해당 모델의 특성을 잘 실현할 수 있는 시나리오를 작성하게 된다. 시나리오의 작성법으로 T-VEC와 같은 툴을 이용하여 자동으로 생성하는 방법이 있으며[12], 사용자가 요구사항을 분석해서 수동으로 만들 수도 있다. 본 연구의 Simulink 디버거는 후자의 사용자가 직접 만든 시나리오의 검증에 유용하게 사용될 수 있다. 예를 들어 자판기의 요구사항을 바탕으로 (그림 7)과 같은 자판기 모델을 생성한다. 이렇게 생성된 자판기 모델에 (그림 2)와 같은 자판기의 요구사항의 입력과 예상되는 결과를 넣어보면서 처음에 생각



(그림 7) 자판기 모델



(그림 8) Machine 차트의 내부 Stateflow 모델

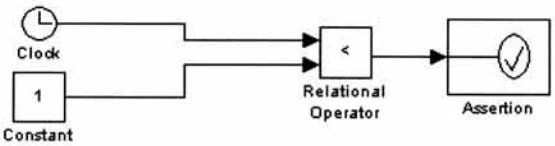
했던 요구사항과 같이 모델이 정확하게 동작하는지를 점검하게 된다.

4.3 임의의 시점에 Model을 중지시키기

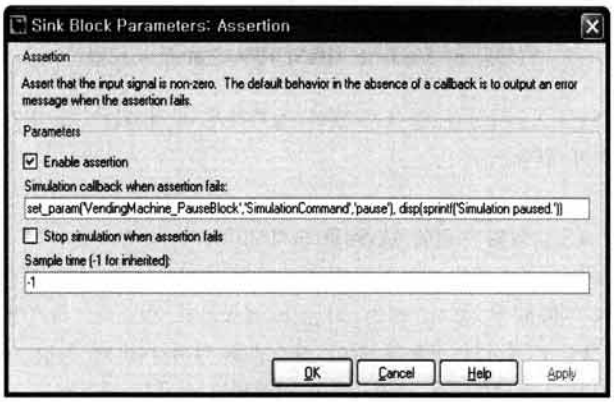
Simulink에서 기본적으로 제공되는 디버거는 임의의 시간에 Model을 중지시키는 기능을 제공하지 않는다. 하지만, (그림 6)에서의 상황과 같이 시스템을 디버깅할 때 특정 시간에 시뮬레이션을 중지시키고 시스템의 상태를 관찰하는 기능은 필수적으로 필요한 기능이다. 그래서 본 연구에서 구현한 프로그램에서는 모델이 시뮬레이션 하는 도중에 사용자가 원하는 특정 시간에 시뮬레이션이 잠시 멈추는 기능을 지원하기 위해서 멈춤 동작을 유도하는 특정 블록을 모델 내에 삽입하는 전략을 선택하였다. 즉 시뮬레이션을 동작시키기 위한 모델을 읽어서 그 모델에 새로운 블록을 삽입하여 새로운 모델을 만들어서 실행시키는 것이다. 물론 본 프로그램에서는 원 모델이 손상되도록 하지 않는다. 모델이 시뮬레이션 도중 멈출 수 있도록 하는 블록은 (그림 9)와 같다.

Clock 블록은 현재 Simulink의 시간을 나타내고, Constant 블록은 멈추고자 하는 시간을 나타낸다. Assertion블록은 들어오는 입력이 거짓일 경우 어떤 행동을 취하게 할 수 있는데, 구현한 프로그램에서는 시뮬레이션이 멈추는 행동을 취하도록 하였다. 즉 현재 시간과 Constant 블록의 값을 비교해서 시간이 크거나 같으면 시뮬레이션은 멈춤 상태가 된다. Constant블록의 값은 외부에서 변경이 가능하며, 직접적으로 제어가 가능하기 때문에 사용자가 원하는 시점에 모델이 멈춤 상태가 되도록 하는 것이 가능하다. (그림 10)은 Assertion 블록을 이용하여 모델의 잠시 멈춤 상태를 설정하는 것에 대한 그림이다. (그림 9)의 Constant 블록의 값은 동적으로 변해야 하는데, 그 값은 각 Time에 멈출 지의 여부를 체크하는 체크박스에 의해 결정된다.

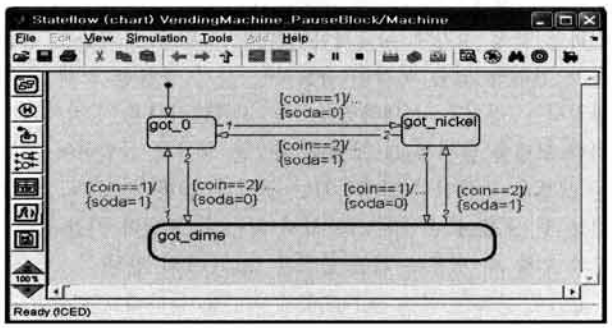
자판기 모델을 예로 들어 설명하면 (그림 6)의 모델 검사 과정에서 8번째 step인 Time이 70에서 잘못된 결과가 나왔기 때문에 사용자는 7번째 step에서 시뮬레이션을 멈추도록 하고 8번째에서 모델이 어떻게 동작하는지 알고 싶을 것이다. 7번째에서 8번째로 넘어가는 때의 모델의 동작을 보기 위해 7번째에서 breakpoint를 선택하고, 시뮬레이션을 돌린 후에 (그림 11)과 같이 Stateflow 모델 창을 띄우고 시뮬레이션을 다시 수행시켜 8번째 입력으로 인한 모델의 행동이 어떻게 동작하는지를 확인할 수 있다. 이와 같은 방식으로



(그림 9) 원하는 시점에 모델을 멈추게 하는 기능을 가진 서브 시스템의 내부구조



(그림 10) 모델을 잠시 멈추게 하는 설정 방법



(그림 11) 테스트 시나리오의 7 step까지 수행된 Stateflow 모델

테스트 시나리오로 모델을 검사하고, 원하는 시점에서 시뮬레이션을 멈추도록 할 수 있다.

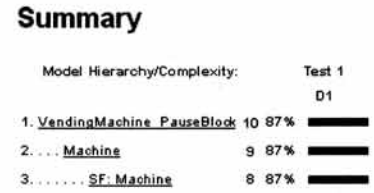
4.4 Coverage Report

Simulink는 모델이 시뮬레이션으로 수행된 결과를 바탕으로 Coverage 측정을 할 수 있는 도구를 제공한다. 일반적으로 모델을 검사할 때 Coverage 측정을 하는 이유는 작성된 모델을 시뮬레이션 시킬 때 얼마나 많은 부분을 시뮬레이션 하면서 커버했는지를 확인하기 위함이다. 만약 모델에서 사용한 블록의 개수가 100개인데 시뮬레이션 기간 중에 활성화 되어 동작을 수행한 블록의 개수가 30개 미만이라면 모델을 충분히 검사했다고 볼 수는 없을 것이다. 이와 같이 Coverage Report는 모델을 검사할 때 충분히 검사했는지의 척도로 사용될 수 있다.

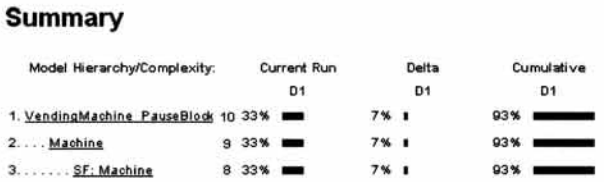
Simulink에서 제공하는 Coverage 메트릭은 5개가 있는데, Decision Coverage, Condition Coverage, MCDC Coverage, Look-up Table Coverage, Signal Range Coverage가 이에 해당한다[4, 9]. 모든 Coverage 메트릭들은 공통적으로 많은 입력들을 넣어줌에 따라 커버되는 비율이 높아진다. 본 연구에서는 5개의 Coverage 메트릭 중에서 간단한 모델에서도 나타나는 Decision Coverage의 변화를 관찰하였다. 본 연구에서 구현한 Simulink 디버거는 Simulink에서 제공하는 Coverage 보고서 기능을 사용하여, 시나리오 전체 혹은 임의의 시나리오를 선택하여 선택된 시나리오가 얼마나 모델을 커버하는지를 확인할 수 있다.

자판기 모델을 예로 들어 설명하면 (그림 5)의 첫 번째 시나리오 하나를 실행시킨 결과 (그림 12)와 같이 87%를 커버하는 것을 볼 수 있다. 여기서 D1은 Decision coverage를 뜻한다. 자판기 모델에서는 Decision coverage로만 측정이 가능하고 다른 Coverage는 측정이 불가능하기 때문에 Decision Coverage만 표시되었다. 여기서 두 번째 시나리오를 실행시켜서 누적시킨 결과 (그림 13)과 같이 7%가 추가로 커버 되어서 93%가 되었고, 세 번째 시나리오를 선택해서 실행시키고 누적시킨 결과 (그림 14)와 같이 7%가 새로 커버되어서 100%가 완성되었음을 알 수 있다. 본 연구에서 구현한 Simulink 디버거에서는 이와 같이 원하는 시나리오들을 선택해주면, (그림 4)에 기술된 pseudocode에 기술된 것처럼 각 테스트 시나리오마다 데이터를 불러들이고 모델을 실행시킨다. 구현된 Simulink 디버거에서 유저가 선택한 모든 시나리오의 Coverage 계산을 일괄적으로 수행해준다.

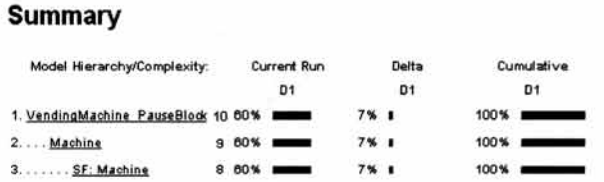
기존의 체킹 도구에서 도구 자체의 정적인 분석 기법으로 모델을 분석했다면, 본 연구의 Simulink 디버거는 Simulink에서 제공되는 Coverage Report를 활용하여 다양한 분석을 할 수 있다. 자동 생성된 툴에서 만들어진 테스트 시나리오



(그림 12) 첫 번째 시나리오를 수행했을 때의 Coverage Report 결과



(그림 13) 두 번째 시나리오를 추가로 수행했을 때의 Coverage Report 결과



(그림 14) 세 번째 시나리오를 추가로 수행했을 때의 Coverage Report 결과

를 개발된 Simulink 디버거를 이용해 생성된 Coverage Report에서 어느 정도의 Coverage가 나오는지를 볼 수 있다. 이 정보를 이용해서 자동 생성된 툴에서 나온 시나리오 데이터의 품질 및 모델의 품질을 분석할 수 있다.

5. 결 론

본 논문에서는 모델 검사를 위한 향상된 Simulink 디버깅 도구의 구현에 대해 기술하였다. Simulink에서 제공되는 다양한 기능을 이용해서 복잡한 요구사항을 편리하게 작성할 수 있다. 기존의 도구에서 모델의 정확한 동작을 검사하기가 불편했던 점을 개선하기 위해서 테스트 시나리오를 통한 자동 검사 도구 기능, 모델을 시뮬레이션 하는 도중 원하는 시점에서 모델을 멈추게 하는 기능, 테스트 시나리오에 대한 모델의 테스트 Coverage 기능을 통해 기존의 Simulink 도구만으로 사용하기 불편한 모델의 행동을 검사하는 기능에 개선을 했다. 본 논문에서 구현한 도구를 이용하여 Matlab에서 제공하는 자판기 모델에 적용했으며, 적용 결과 모델 검사 작업을 더 편리하게 할 수 있음을 보였다. 추가적으로 개발된 도구를 이용하여 자동차 전자제어 장치 모델의 복잡한 검증 작업을 편리하게 수행할 수 있었다.

참 고 문 헌

[1] 송문빈, 송태훈, 오재곤, 정연모, "효율적인 통합시뮬레이션에 의한 스피커 연결 시스템의 SoC설계," 전자공학회논문지, 제43권 SD편, 제10호, pp.671-676, 2006.

[2] 김성조, 정기현, 최경희, "Simulink 기반의 Testing Framework," 제30회 한국정보처리학회 추계학술발표대회 논문집, 제15권, 제2호, pp.539-542, 2008.

[3] David Harel, "Statecharts: A visual formalism for complex systems," Science of Computer Programming, Vol8, Issue3, pp.231-274, 1987.

[4] Kelly Hayhurst, et al, "A Practical Tutorial on Modified Condition/Decision Coverage," NASA/TM-2001-210876, 2001.

[5] Mathworks, http://www.mathworks.com/applications/dsp_comm/xilix_ref_guide.pdf

[6] Mathworks, http://www.mathworks.com/access/helpdesk/help/toolbox/stateflow/index.html?/access/helpdesk/help/toolbox/stateflow/gs/bqdfwc7-1.html

[7] Mathworks, http://www.mathworks.com/access/helpdesk/help/toolbox/stateflow/index.html?/access/helpdesk/help/toolbox/stateflow/gs/f14-37240.html

[8] Mathworks, http://www.mathworks.com/access/helpdesk/help/toolbox/stateflow/index.html?/access/helpdesk/help/toolbox/stateflow/ug/bqvoz7x.html

[9] Mathworks, http://www.mathworks.co.kr/products/simverification/description5.html

[10] Chifu Yang et al, "Modeling and Simulation of 6-DOF Parallel Manipulator Based on PID Control with Gravity Compensation in Simulink/ADAMS," International Workshop on Modeling, Simulation and Optimization, pp.391-395, 2008.

[11] Koo, K.L. "Modeling and co-simulation of AC generator excitation and governor systems using Simulink interfaced to PSS/E," IEEE Power Systems Conference and Exposition, Vol2, pp1095-1100, 2004.

[12] Lisa M. Boden, Robert D. Busser, Mark R. Blackburn, Aaron M. Nauman, "Extending Simulink Models With Natural Relations To Improve Automated Model-Based Testing," 29th Annual IEEE/NASA Software Engineering Workshop, pp.325-332, 2005.

[13] da Silva, et al, "Generating VHDL-AMS Models of Digital-to-Analogue Converters From MATLAB/SIMULINK," Thermal, Mechanical and Multi-Physics Simulation Experiments in Microelectronics and Micro-Systems, pp.1-7, 2007.

[14] R. Alur, A. Kanade, S. Ramesh, and K. C. Shashidhar. "Symbolic analysis for improving simulation coverage of Simulink/Stateflow models," In de Alfaro and Palsberg, pp.89-98, 2008.

김 성 조

e-mail : mitnic@empal.com

2007년 아주대학교 전자공학부(학사)

2007년 아주대학교 정보 및 컴퓨터공학부(학사)

2008년~현 재 아주대학교 전자공학과 석사과정

관심분야 : 임베디드 시스템, Simulink, 모바일 플랫폼

이 홍 석

e-mail : myhong5@ajou.ac.kr

2003년 아주대학교 정보 및 컴퓨터공학부(학사)

2003년 아주대학교 전자공학부(학사)

2005년 아주대학교 전자공학과(석사)

2005년~현 재 아주대학교 전자공학과 박사과정

관심분야 : 정형 검증, 명세 기술, 임베디드 시스템

최 경 희

e-mail : khchoi@ajou.ac.kr
1976년 서울대학교 수학교육과(학사)
1979년 프랑스 그랑데꼴 Enseeiht대학 (석사)
1982년 프랑스 Paul Sabatier대학 정보공학부(박사)
1982년~현 재 아주대학교 정보통신전문대학원 교수
관심분야 : 운영 체제, 분산시스템, 실시간 및 멀티미디어시스템 등

정 기 현

e-mail : khchung@ajou.ac.kr
1984년 서강대학교 전자공학과(학사)
1988년 미국 Ilinois주립대 EECS(석사)
1990년 미국 Purdue대학 전기전자공학부 (박사)
1991~1992년 현대반도체 연구소
1993년~현 재 아주대학교 전자공학부 교수
관심분야 : 컴퓨터구조, VLSI 설계, 멀티미디어 및 실시간 시스템 등