Simulink 에 적용되는 Testing Framework

김성조*, 정기현*, 최경희**
*아주대학교 전자공학과
**아주대학교 정보통신전문대학원
e-mail : rlat@ajou.ac.kr

A Testing Framework in Simulink

Kim Seong-Jo*, Jung Ki-Hyun*, Choi Kyung-Hee**
*Division of Electronics Engineering, Ajou University
**Graduate School of Information and Communication, Ajou University

요 약

본 연구에서는 Simulink 에서 테스트케이스를 적용하는 과정을 자동화시켜 주었다. Simulink 는 자동 생성된 많은 수의 테스트케이스를 dynamic 하게 적용하는 작업에서 반복적이고 수동적인 다수의 작업이 필요하다. 그래서 Simulink 를 이용한 테스트에 많은 인력과 시간이 필요한 문제점이 있다. 본 연구의 Framework 에서는 파일에 저장된 테스트케이스를 Simulink 에 바로 적용할 수 있게 해주었다. 먼저 Framework 에서 사용된 테스트케이스의 입력방법을 알아보고, Framework 에서의 테스트 수행과정을 나타내고 결과를 분석하였다.

1. 서 론

소프트웨어 테스트 시에 적용되는 테스트케이스를 생성하는 상용화된 툴이 몇 가지 있다. Borland Together Architect, Rational Software Architect, Telelogic TAU, Telelogic Rhapsody or Artisan Studio 등의 모델 기반의 CASE 툴에서 테스트케이스를 생성하는 Qtronic[1], Statemate 에서 만들어진 모델의 테스트케이스 생성툴이 있다[2]. 그리고 Reactis Tester 라는 툴은 Simulink 와 Stateflow 에서 테스트케이스를 생성해준다[3]. 이러한 툴들을 이용하면 모델이 변화됨에 따른 테스트케이스들을 쉽게 생성할 수 있다. 이렇게 만들어진 많은 테스트케이스들의 검증은 필수적인 작업이다.

Matlab 은 알고리즘을 검증하는데 유용한 도구로서 그 중에서 Simulink 는 IP(Intellecutual Property) 형태의 수많은 컴포넌트를 가지고 블록 다이어그램 형태의 설계가 가능하며, 높은 추상화를 지원한다[4]. 그러한 장점으로 인하여 Matlab/Simulink 는 시스템 모델링에 많은 부분에서 사용되고 있다. 최근의 추세에 따르면 알고리즘 검증 단계에서의 자원을 그대로 HW 설계 및 SW 설계에서 사용하고 최종적으로 검증에까지 사용되는 것이 바람직하다[5]. 이 때 검증 단계의 자원은 Simulink 모델을 뜻하며, 검증 단계에서 실제 동작가능한 다양한 테스트케이스를 모델에 적용시키는 것이 중요하다.

이 때 Simulink 에서는 다양한 테스트케이스를 적용함에 있어 다음과 같은 문제점을 가지고 있다. 첫째로 외부에서 생성된 테스트케이스를 일괄적으로 편리하게 넣을 방법이 없다. Simulink 에 입력을 넣는 모든 방법들이 외부에서 변화되는 테스트케이스에 대해서 사용자가 일일히 적용해야 하는 불편함을 가지고 있다. 둘째로는 N 개의 테스트케이스를 넣는 방식에 대해서 고려되지 않았다. 하나의 테스트케이스로 테스트를 하는 것보다는 다수의 테스트케이스로 테스트를 수행하는 것이 테스트의 신뢰성을 높일 수 있다. 이 때 Simulink 에서 특정 N 개의 테스트케이스 모음을 적용하기 위해서는 수동으로 테스트케이스를 넣어주어야 한다. 이러한 작업은 노동 집약적이고 단순 반복적인 작업이다. 위의 두 가지 작업을 편리하게 수행하는 FrameWork 를 제안한다.

2. 본 Framework 에서 사용된 입력방법

Simulink 에서 사용자로부터의 입력을 받을 수 있는 방법은 3 가지가 있다. 첫째로 Const 블록이 있다. Const 블록은 입력으로 지정된 값을 계속 내보내는 블록이다. 이 때 Const 블록의 지정된 값은 사용자가 블록에 접근해서 직접 값을 바꾸거나 혹은 command 상에서 값을 변경해 줄 수 있다. 사용자가 입력을 변경시키면 변경된 것이 Const 블록에 나타나 있는 값의 변화를 통해 바로 확인가능하고 그에 따라 시스템의 동작이 변경된다. 이 방식에서는 시스템이 입력을 기다리는 상태에 있으면서 사용자가 입력을 넣어주는 시점에 값이 변경된다. 테스트케이스에서 지정된 시간은 굉장히 짧거나 불규칙적일 수 있는 가능성이 존재한다. 그러한 다양한 경우를 사용자가 수동으로 맞추어서 모델을 시뮬레이션 하기가 힘들다. 두번째로 Signal Builder 가 있다. Signal Builde 로 사용자는 임의의 signal wave 를 만들 수 있다. Signal wave 를 테스트케이스의 시간에 따른 입력으로 설정해서 저장과 실행이 가능하다. 이 방법을 이용해서 요구사항

기반의 테스트를 수행한 사례가 있다[6]. 해당 방법은 외부에서 생성된 테스트케이스를 dynamic 하게 적용할 수 없는 단점이 있다. 외부에서 생성된 테스트케이스에 따른 signal 을 직접 설정해주어야 한다. 세번째로 Inport 를 이용해서 외부의 입력을 받아들일 수 있다. 이 방법은 모델에서 지정된 이름의 변수에 입력데이터를 넣어주고 시뮬레이션 하면 그에 따른 테스트케이스가 수행된다. 이 때 입력이 수행되는 시간값을 자유롭게 넣어줄 수 있다. Matlab 명령을 이용해서 해당 변수의 제어가 가능하며 그 변수의 값은 외부의 파일에서 읽어서 적용시킬 수 있다. 그래서 외부에서 생성된 테스트케이스 파일의 적용이 편리하게 이루어진다. 본 연구에서는 세번째 방법을 이용해서 Framework 를 구축하였다. 본 Framework 에서는 Dynamic 하게 변화되는 테스트케이스 파일을 지정된 포맷으로 받으면 특별한 작업 없이 다양한 테스트케이스를 한 번에 시뮬레이션하고 결과를 볼 수 있다.

3. Testing Framework 를 이용한 테스트

3.1 테스트케이스의 Simulink 적용

다음 <표 1>은 Testing FrameWork 에서 자동수행되는 작업의 슈도코드이다.

<표 1> Framework 의 슈도코드

```
1.  Open Selected TestcaseSet File
// Separate TestCaseSet into Testcase Files
2.  While (until Selected File end point)
    2.1 Read Current Line Data
    2.2 Make specified TestCaseFile (total N)
// Simulate model with each TestCaseFile
3.  Loop (N)
    3.1 Simulation (N Testcase)
        3.1.1 Read TestCaseFile
        3.1.2 Calculate Executing Time
        3.1.3 Read Simulink Input
        3.1.4 Save variable value relation time
        3.1.5 Mapping with simulink input
        3.1.6 Assign test vector to base workspace
        3.1.7 Set Simulation Time
        3.1.8 Simulation
    3.2 Delete simulated TestCaseFile
```

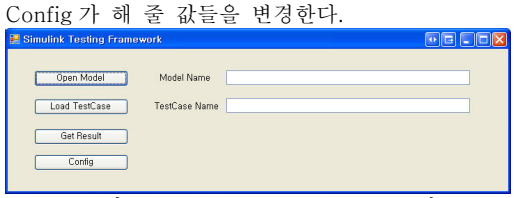
해당 Framework 가 동작되고 나서 처음에 테스트케이스를 적용시킬 모델 파일을 열어준다. 그리고 N 개의 테스트케이스가 포함된 파일을 각각 하나씩의 테스트케이스를 가진 N 개의 파일로 저장한다.

그 후에 실제적인 Simulation 작업이 이루어진다. 먼저 하나의 테스트케이스가 들어있는 파일을 열어서 Simulink 에서 적용될 수 있는 데이터 포맷으로 변경해주고 테스트를 수행해준다. 3.1.1 에서 3.1.3 까지 테스트케이스파일과 Simulink Input 에서 필요한 데이터를 얻어준다. 테스트케이스파일에는 테스트케이스의 실행 Step 값과 그 시간에 따른 입력값이 들어있다. 그리고 지정된 Sample Time 에 따라서 Step 에 따른 시간값을 계산해준다. 그 후에 시뮬레이션 될 Simulink 모델의 입력을 읽어준다. 3.1.4 와 3.1.5 는 Simulink 에서 적용될 수 있는 데이터 포맷으로 변경해준다. 계산된 시간마다의 입력값의 셋을 입력의 숫자만큼 만들고, Simulink Input 과 맵핑된 순서대로 연결해준다. 3.1.6 에서 3.1.8 은 Simulink Input 에 적용시키는 부분이다. 먼저 만들어진 Simulink 입력 데이터를 Simulink 가 시뮬레이션 시에 읽을 수 있게 base workspace 에 할당해준다. 그리고 계산된 시뮬레이션 시간을 이용해서 Simulation 의 최종시간을 구해서 설정한다. 마지막으로 시뮬레이션을 수행한다. 3.1 의 작업은 전체 테스트케이스 숫자인 N 번만큼 반복된다. N 번의 테스트케이스 수행이 끝난 후에는 3.2 에서 분리된 N 개의 테스트케이스 파일을 지워주게 된다

3.2 테스트케이스의 Simulink 적용

다양한 테스트케이스를 입력으로 하여 Simulink 모델을 실행시키는 방법은 2 가지가 있다. 첫번째 방법은 sim command 를 사용한다. sim command 를 사용하면 모델의 시뮬레이션이 끝나고 난 후에 다음 Script 가 수행된다. 해당 방법은 시뮬레이션이 시작되고 종료되기 전까지 다른 커맨드가 실행되지 않아서, 해당 테스트케이스에 대한 정확한 결과를 알 수 있다. 두번째로 set_param command 를 사용하는 방법이 있다. 이 방법은 모델을 start 시킨 후에 바로 다음 Script 를 수행한다. 하지만 이 방법을 이용할 경우에만 Outport 에 지정된 변수에 데이터가 들어간다. 기본적으로는 첫번째 방법으로 시뮬레이션이 되고, 파일을 생성하는 옵션을 설정하면, 두번째 방법을 사용하도록 하였다. 이 때 하나의 테스트케이스 시뮬레이션이 끝나기 전에 다른 테스트케이스의 시뮬레이션이 되는 것을 방지하기 위해서 그 사이에 적절한 지연시간을 설정하게 된다. 각 테스트케이스의 적용이 끝남에 따라 결과 Report 가 만들어진다. Simulation 을 이용하면, 이와 같은 report 를 Simulink 에서 자동으로 만들어준다. report 는 이전까지의 테스트케이스로 수행된 coverage 들을 누적해서 출력해준다. 마지막 N 번째 report 를 보면 N 개의 테스트케이스 전체의 coverage 를 얻을 수 있다. 마지막에 테스트케이스 수행에 따른 결과값을 저장해 줄 수 있다.

(그림 2)는 본 framework 의 UI 를 나타내었다. “ Open Model” 버튼으로 모델을 열어주고, “ Load TestCase” 버튼으로 적용시킬 테스트케이스를 선택한다. 적용시킬 테스트케이스를 선택하면 N 번의 시뮬레이션이 자동으로 수행된다. 시뮬레이션이 다 끝난 후에 “ Get Result” 버튼을 누르면 시뮬레이션 된 결과값을 저장한다. “ Config” 버튼으로 필요한 Config 가 해 줄 값들을 변경한다.



(그림 2) Simulink Testing Framework 의 UI

4. Simulation 및 결과 분석

(그림 3)의 간단한 커피자판기 모델을 대상으로 본 Framework 를 사용해 보았다. 커피자판기의 전체 요구사항 중에서 커피가 나올 수 있는지의 여부의 결정하는 부분과 잔액 계산하는 부분을 간략화해서 모델링하였다.

커피자판기 모델에는 4 개의 입력이 있다. SystemState 는 시스템이 On 되어 있는지의 여부를 나타내며, Button 은 커피를 주문하는 동작을 수행했음을 나타낸다. CurrentMoney 와 Price 는 현재 자판기에 들어있는 금액과 커피의 가격을 나타낸다. CoffeeOut 가 발생되는 조건은 2 가지이다. 첫번째로 SystemState 와 Button 이 동시에 만족해야 한다. 두번째로 CurrentMoney 가 Price 보다 크거나 같아야 한다. 2 가지 조건이 만족하면 CoffeeOut 의 값이 1 이 되고, CurrentMoney 에서 Price 를 뺀 UpdateCurrentMoney 값이 계산된다. 본 모델에서는 커피의 가격은 한 가지인 것으로 생각하였다

4.1 Coverage Setting 변경

Simulink 에서 Coverage Setting 에서 Enable Coverage Reporting 을 체크하고, 측정하고자 하는 Coverage 를 선택한다. 그리고 N 개의 테스트케이스에 의한 Coverage 의 합을 보기 위하여 Report 에서 생성되는 HTML report 를 Cumulative runs 로 해준다.

4.2 하나의 테스트케이스를 돌린 결과

2 개의 Operator 로 구성된 해당 요구사항은 총 5 개의 테스트케이스를 적용해주어야 모든 Coverage 를 만족하게 된다. <표 2>는 커피자판기 모델에 적용된 테스트케이스 하나를 보였다. 테스트케이스 파일의 형태는 먼저 테스트케이스의 명칭이 나온 후에, 입력으로 들어가는 변수의 명칭이 나온다. 그리고 Step 에 따른 변수의 값의 목록이 나타난다.

<표 2> 커피자판기 모델에 적용된 TC 1 개 예

TestCase1				
Step	SystemState	Button	CurrentMoney	Price
0	1	1	500	200
1	1	0	100	200

(그림 4) <표 2>와 같은 테스트케이스 하나를 돌렸을 때의 Coverage Report 이다. 하나의 테스트케이스를 이용해서 체크할 수 있는 Coverage 는 Decision Coverage 와 Condition Coverage 의 경우 50%가 나왔다. MC/DC 의 경우는 전혀 만족하지 못했다.

4.3 모든 테스트케이스를 돌린 결과

<표 3>은 커피자판기 모델의 모든 테스트케이스를 나타내었다. 하나의 테스트케이스에서 모두 체크할 수 없었던 Coverage 가 다수의 테스트케이스를 사용함으로 체크가능하게 된다

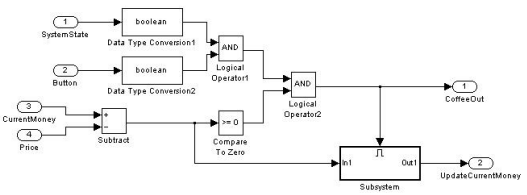
<표 3> 커피자판기 모델에 적용된 4 개의 TC

TC 1				
Step	System_State	Button	Current_Money	Price
0	1	1	500	200
1	1	0	100	200
TC 2				
Step	System_State	Button	Current_Money	Price
0	0	1	500	200
1	1	1	100	200

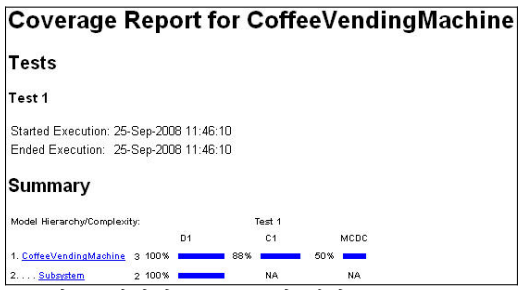
(그림 5)는 Framework 를 이용하여 <표 3>에 나온 테스트케이스 2 개를 모두 돌렸을 때의 Coverage Report 이다. 2 개의 테스트 케이스의 구동시간과 Coverage 가 나타난다. 하나의 테스트케이스를 돌렸을 때와 비교해서 여러 개의 테스트케이스를 이용한 시뮬레이션에서 높은 Coverage 를 얻을 수 있었다. 하나의 테스트케이스만을 가지고 테스트를 수행했을 때에 100%로 만족하지 못했던 Coverage 가 2 개의 모든 테스트케이스를 돌렸을 때는 Decision Coverage, Condition Coverage, MC/DC 3 개 모두 100%가 나온 것을 볼 수 있다.

5. 결론

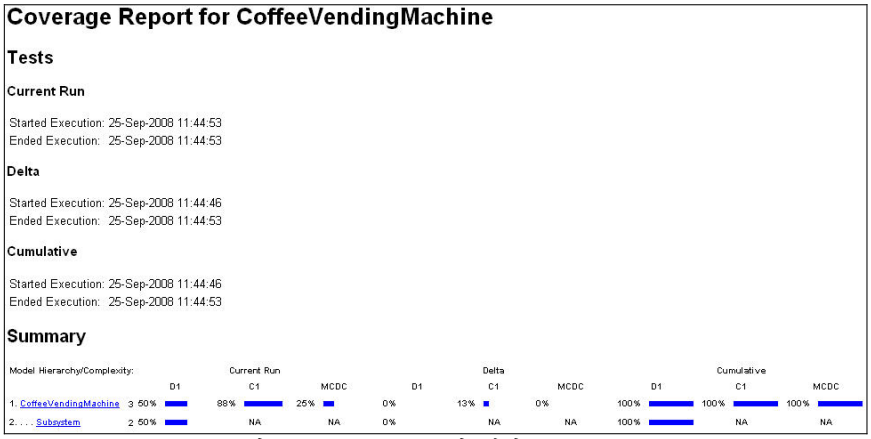
본 연구에서는 Simulink 에 적용시킬 수 있는 Testing Framework 를 제안하였다. N 개의 테스트케이스를 Simulink 모델에 적용시키는 작업을 본 Framework 를 이용해서 손쉽게 수행할 수 있었다. 그로 인하여 필요한 시점마다 테스트케이스를 다양하게 적용시킬 수가 있다. 본 Framework 로 인하여 software productivity, software quality 의 향상이 기대된다.



(그림 3) 커피자판기의 모델



(그림 4) 하나의 TC 를 돌릴 때의 Coverage Report



(그림 5) 모든 TC 를 돌릴 때의 Coverage Report

참고문헌

[1] Conformiq Test Generator (product information), Verifysoft Technology GmbH. http://www.verifysoft.com/en_conformiq_testgenerator.html

[2] ATG tool(product information), OSC – Embedded System AG, http://www.osc-es.de/products/en/atg.php.

[3] Reactis Tester (product information), Reactive Systems Inc., www.reactive-systems.com.

[4] http://www.mathwork.com

[5] 송문빈, 송태훈, 오재곤, 정연모, “효율적인 통합 시뮬레이션에 의한 스피커 연결 시스템의 SoC 설계”, 2006 년 10 월 전자공학회 논문지 제 43 권 SD 편 제 10 호

[6] J. R. Ghidella and P. J. Mosterman, “ Requirements-based testing in aircraft control design,” in *Proceedings of the AIAA Modeling and Simulation Technologies Conference and Exhibit 2005,* San Francisco, CA, Aug. 2005, CD-ROM, ID: 2005-5886.