DOI: 10.3745/KIPSTD.2010.17D.4.283

Timing diagram의 테스트 케이스 생성 전략

이 홍 석[†] · 정 기 현[††] · 최 경 희[†††]

요 약

Timing diagram은 시스템의 사양을 작성하는 도구로 많이 사용되고 있으나, Timing diagram을 이용하여 테스트 케이스를 생성하는 연구는 존재하지 않는다. 이와 같은 문제를 해결하기 위해 본 연구에서는 Timing diagram 으로부터 테스트 케이스를 생성하기 위해 다음과 같은 과정을 거쳤다. 1) Timing diagram을 형식적으로 기술하였다. 2) 정의된 Timing diagram의 의미로부터 이와 동치 관계에 있는 Stateflow 모델로 변환하는 방법을 기술하였다. 3) 변환된 Stateflow 모델에 대해 Simulink의 플러그 인 되어 있는 도구인 SDV를 사용하여 테스트 케이스를 생성했다. 이 접근 방법이 유용함을 보이기 위해서 도난 경보 모델과 임의의 Timing diagram모델들을 대상으로 실험을 수행하였다. 실험에서 Timing diagram 모델들로부터 Stateflow모델들로 변환하고 이를 SDV를 사용하여 테스트 케이스를 생성하였으며 그 결과에 대해 분석하였다. 이 연구를 통해 얻을 수 있는 결론은 Timing diagram이 더 이상 사양서를 작성하는 도구로만 사용되는 것이 아니라 모델 기반의 테스트 케이스를 생성하고자 하는 경우에도 유용한 도구가 된다는 점이다.

키워드 : 자동 테스트 케이스 생성, Stateflow, Timing diagram, 사양서, 임베디드 시스템

Test Case Generation Strategy for Timing Diagram

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

ABSTRACT

Timing diagram is a useful tool for describing the specification of system, but there is no study for test case strategy of a timing diagram. To solve this problem, we followed the steps to generate test cases from timing diagram in this paper. 1) We defined a timing diagram formally. 2) We describe the method of transforming from a timing diagram model into a Stateflow model which has an equivalent relationship between a timing diagram model and a transformed Stateflow model. 3) We generated test cases from a transformed Stateflow model using SDV which is plugged in Simulink. To show that our approach is useful, we made an experiment with a surveillance model and arbitrary timing diagram models. In the experiment we transformed timing diagram models into Stateflow models, generated test cases from transformed Stateflow models using SDV, and analyzed the generation results. The conclusion that can be obtained from this study is that timing diagram is not only a specification tool but also a useful tool when users are trying to generate test cases based on model.

Keywords : Automatic Test Case Generation, Stateflow, Timing Diagram, Specification, Embedded System

1. 서 론

Timing diagram은 시스템의 행동을 표현하기에 유용한 도구로 주로 전자공학 분야에서 하드웨어 시스템의 행동을 표현하기 위한 용도로 유용하게 사용되고 있다. 거의 대부분의 집적 회로에 대한 데이터 시트에서 시스템의 입출력 행동을 표현하고자 할 때 Timing diagram을 사용하여 표현한다. Timing diagram은 시간의 순서대로 입력의 값과 출력의 값의 변화를 볼 수 있기 때문에 한눈에 시스템의 행동을 이해할 수 있다는 장점이 있다. 주로 시스템의 연속적인 행동들에 대한 입력과 출력의 값의 변화를 파악하고자 할 때 유용하게 사용된다. 하지만 이와 같은 장점에도 불구하고 Timing diagram으로부터 테스트 케이스를 생성하려는 시도는 없었기 때문에 Timing diagram을 이용하여 테스트 케이스를 생성하고자 할 경우에는 사람이 그 모델을 분석하고 테스트 케이스를 생성해야 한다.

위와 같은 문제를 해결하기 위해서 본 논문에서는 Timing diagram의 테스트 케이스를 생성하는 전략에 대해 기술하고자 하는데, 다음과 같은 단계를 거쳤다. 1) Timing diagram

† 준 회 원 : 아주대학교 전자공학과 박사과정
†† 정 회 원 : 아주대학교 전자공학부 교수
††† 정 회 원 : 아주대학교 정보통신전문대학원 교수
논문접수 : 2010년 3월 2일
수 정 일 : 1차 2010년 4월 9일, 2차 2010년 4월 24일
심사완료 : 2010년 4월 24일

의 문법과 의미에 대해 형식적으로 정의하였다. 2) 정의된 Timing diagram의 행동으로부터 임의의 Timing diagram 모델에 대해 이와 동일한 행동을 수행하는 Stateflow 모델로 변환한다. 3) 변환된 Stateflow 모델에 대하여 Stateflow에서 제공하는 도구를 사용하여 테스트 케이스를 생성한다. 여기서 Stateflow는 Mathworks사에서 개발한 Statecharts[15]와 유사한 기능을 가진 도구이다. 이와 같은 접근 방식을 택한 이유는 본 논문에서 제시하는 Timing diagram이 Stateflow로 변환 가능한 의미를 담고 있기 때문이며, Statecharts에서 테스트 케이스를 생성하는 연구가 이미 많은 연구가 이루어졌기 때문에 기존의 연구를 이용하기 위해서이다.

이 접근 방법의 유용함을 보이기 위해서 Timing diagram 도난 경보 모델을 Stateflow모델로 변환하고 변환된 모델에 대한 테스트 케이스 생성 결과를 보이고, 임의로 생성한 Timing diagram 모델 50개에 대해 각각 Stateflow로 변환한 후 변환된 모델에 대한 테스트 케이스 생성 결과에 대해 분석할 것이다.

본 논문의 구성은 다음과 같다. 2장에서는 Timing diagram을 이용한 관련 연구에 대해 기술하였고 본 연구와의 차이점을 설명할 것이다. 그리고 Statecharts를 대상으로 테스트 케이스를 생성하는 연구에 대해서 기술할 것이다. 3장에서는 Timing diagram의 문법과 그 의미를 설명하고 그 의미로부터 Stateflow에서 같은 행동을 보이는 모델로의 변환과정에 대해서 설명할 것이다. 4장에서는 Stateflow모델의 테스트 케이스를 생성하는 Simulink Design Verifier(SDV)에 대해서 설명하고 이 도구가 테스트 케이스를 생성하는 동작의 흐름에 대해서 설명할 것이다. 5장에서는 SDV를 사용한 도난 경보 모델과 임의의 모델에 대한 테스트 케이스 생성 결과 및 결과 분석에 대해서 기술할 것이다. 마지막으로 6장은 결론 및 향후 연구로 끝을 맺는다.

2. 관련 연구

Timing diagram은 시간에 따른 시스템의 행동을 기술하는 도구로 표현된 모델이 매우 직관적으로 알아보기 쉬워서 전자공학 분야[3]에서 시스템의 행동을 표현하는 도구로 많이 사용이 되어 왔다. 이와 같은 편리함과 엔지니어들의 폭넓은 사용으로 Timing diagram과 관련된 연구는 주로 Timing diagram을 이용하여 VHDL 코드를 생성[17]하거나 혹은 하드웨어 시스템을 검증[4, 8, 9, 14]하기 위한 목적으로 진행되어 왔다.

하드웨어의 특성상 동기 시스템, 비동기 시스템으로 나눌 수 있는데, 각 연구들에서는 시스템의 특성 및 각 연구 대상에 적합한 Timing diagram을 정형화 하였다. Timing diagram은 기본적으로 시나리오 기반으로 시스템의 행동을 설명하는 것이기 때문에 시스템 전체의 행동을 표현하는 것은 어려움이 있다. 하지만 Moeschler의 연구[7]에서는 Timing diagram을 확장하여 이벤트, 액션, 제약 조건뿐만 아니라 Timing diagram간의 계층성이나 병렬성에 대한 개념을 추가하여 시스템의 행동을 Timing diagram만으로 표현하려는 시도가 있었으며, 또는 Timing diagram을 언어와 같이 표현하도록 하는 등의 개선 작업이 있었다. 하지만 이와 같은 개선 작업은 기존의 Timing diagram이 가진 단점을 극복할 수는 있었지만, Timing diagram이 가진 본연의 간단함과 직관적 인식을 어렵게 만들기도 했다.

Timing diagram을 이용하여 검증을 수행하는 연구[4, 8, 9, 14]에서는 Timing diagram을 LTL 시제로직이나 ω오토마타, 혹은 Büchi 오토마타로 간주하고, 시제로직이나 오토마타를 이용해서 검증하는 접근 방식을 취했는데, 각 논문에서는 Timing diagram이 시제로직이나 오토마타로 변환 가능한지의 여부에 초점을 맞추어 연구를 진행했으며, 시제로직이나 오토마타를 이용하여 검증하는 것은 기존의 연구를 이용하였다.

Timing diagram을 이용한 과거의 연구는 주로 Timing diagram을 이용하여 시스템을 검증하기 위한 목적으로 사용되었던데 반해, 본 연구는 Timing diagram을 이용하여 시스템을 시뮬레이션하거나 모델 기반의 테스트 케이스를 자동으로 생성하고자 하는 목적으로 사용하였다는 점에서 과거의 연구와 차이가 있다.

비록 본 연구에서 Statecharts를 이용하여 테스트 케이스를 생성하는 연구를 수행하지는 않지만 기존의 연구들을 이용하여 테스트 케이스를 생성하는 방법을 택하였기 때문에 Statecharts를 이용한 테스트 케이스의 생성에 관련된 연구를 기술하고자 한다. Statecharts를 이용하여 테스트 케이스를 생성하는 연구는 여러 가지 접근 방법이 있었다. Yoo[2]는 Statecharts 모델을 모델 체킹 도구인 SMV가 인식할 수 있는 모델로 변환하고, 그 모델의 반례를 테스트 케이스로 정의하여 SMV이 그 반례를 찾아내어 테스트 케이스를 생성하는 접근 방식을 택하였다. 하지만 Statecharts를 SMV 모델로 변환하는 과정이 자동적으로 이루어지지 않으며, 시스템이 복잡할 경우 SMV가 처리할 수 있는 상태의 범위를 초과할 수 있기 때문에 이 점이 문제가 된다. 위 접근 방법에 대한 관련 연구로는 Rayadurgam[16]가 있으며, 본 연구에서 사용하는 도구인 SDV가 이와 비슷한 방법으로 Statecharts모델에 대한 테스트 케이스를 생성한다.

Hong[5]은 Statecharts를 프로그래밍 언어 연구에서 많이 사용되는 제어 및 자료 흐름 그래프로 변환하고 이 그래프를 이용하여 기존의 데이터 흐름 분석 기법을 이용하여 테스트 케이스를 생성하였다. 또한 Bogdanov[1]는 계층적이고 병렬적인 모델을 가진 일반적인 Statecharts 모델에 대해서 테스트 케이스를 생성하는 방법에 대하여 연구하였는데, 계층적이지 않고 병렬적이지 않으면서 원 모델과 동치관계를 가진 모델로 변환하고 이 모델에 대해 테스트 케이스를 생성하는 전략을 택하였다. 이 연구의 핵심은 일반적으로 계층적이고 병렬적인 속성을 가지는 Statecharts모델을 그와 동치이면서 계층적이지도 않으며 병렬적이지 않은 모델로 변환시키는 연구에 집중되어 있다. 이 연구의 단점은 테스트 입력이 어떤 값이 되어야 하는지를 결정하지 않고 단순

히 변환된 모델에 대해서 각 상태 전이가 커버되어야 하는지에 대한 결과만 나옴으로써 실제적으로 상태전이가 되기 위해 필요한 입력 값이 어떤 값이어야 하는지에 대한 정보를 알 수 없는 한계가 있다. Oh[6]는 Simulink에서 사용되는 Statecharts모델인 Stateflow모델에 대해서 테스트 케이스를 생성하는 연구를 수행했는데, Stateflow모델을 분석하여 MC/DC[18]관점의 테스트 케이스를 생성하고 각각의 테스트 케이스를 커버하는 테스트 입력을 생성하기 위해서 입출력 연관관계 분석 및 피드백 입력에 대해 분석하고 모델의 상태를 탐색하는 접근방법을 통해 테스트 입력을 생성했다.

3. Timing diagram

본 장에서는 본 논문의 주제인 Timing diagram으로부터 테스트 케이스를 생성하기 위한 첫 번째 단계인 Timing diagram의 문법과 의미에 3.1절과 3.2절에서 수학적으로 설명하고, 본 연구에서 제시하는 Timing diagram의 특징에 대해서 설명할 것이다. 그리고 두 번째 단계인 Timing diagram의 의미로부터 Timing diagram 모델의 행동과 같은 행동을 하는 Stateflow 모델로의 변환을 3.3절에서 설명할 것이다.

3.1 Timing diagram의 Syntax

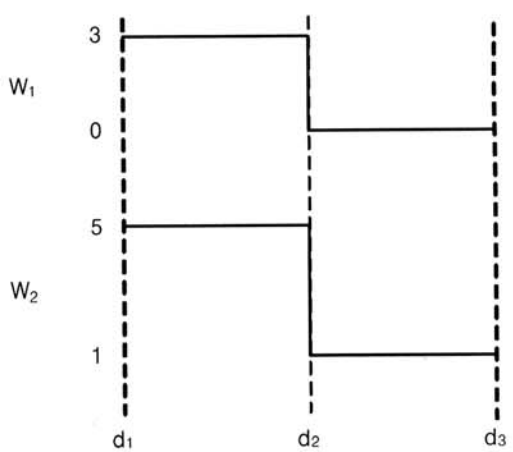
Timing diagram은 $<X, Y, \Sigma, D, C, \rho, \delta>$로 구성되어 있는데, X는 입력 waveform의 집합이고, Y는 출력 waveform의 집합이다. Σ = {Signal, Event}는 wavefor 타입의 집합을 의미하며, D는 시점의 집합, C는 시간 제약의 집합이고 $\rho : (X \cup Y) \rightarrow \Sigma$는 waveform에 대한 타입을 정의하는 함수를 의미하며, $\delta : (X \cup Y) \times D \rightarrow VALUE$는 waveform과 시점에 대한 값을 정의하는 함수를 의미하며, VALUE는 값의 집합을 의미한다. 이벤트 타입의 경우 waveform이 가질 수 있는 값의 종류는 0, 1로 두 가지가 있으며, 0은 이벤트가 발생하지 않음을 의미하고, 1은 이벤트가 발생했음을 의미한다. Timing diagram에서 Waveform은 시간에 따른 값의 변화를 표현할 수 있는데, 어떤 경우 어떠한 값이 되어도 상관없는 경우를 표현할 때가 있는데, 그 경우에는 그 구간을 빗금으로 표시하여 이를 표현할 수 있다. 시점은 waveform들의 값이 변하는 시점 혹은 빗금이 표시되지 않은 영역과 표시된 영역의 경계를 의미한다. 대부분의 Timing diagram이 그렇듯이 본 연구에서 정의한 Timing diagram도 연속적으로 값이 변화하는 시스템을 대상으로 하지 않는다. 예를 들어 Sin(x) 함수와 같이 시간에 따라 값이 연속적으로 표현하는 경우는 본 연구의 대상에 포함되지 않는다.

시간 제약 C는 $<d_s, d_e, t, v>$로 구성되어 있으며 $d_s \in D$, $d_e \in D$는 각각 시작 시점, 끝 시점을 의미하고, 시간 제약의 타입에 대한 집합 T = {Duration, In, After}일 때, $t \in T$는 시간 제약의 타입을 의미하고, $v \in N$(N은 자연수의 집합)는 시간 제약의 값을 의미한다. 시간 제약 타입 Duration은 시작 시점에서 끝 시점 기간의 시간이 시간 제약의 값이어야 함을 의미하며, In은 시간 제약의 값보다 구간 내의 시간이 작아야 함을 의미하며, After는 시간 제약의 값보다 구간내의 시간이 커야 함을 의미한다.

3.2 Timing diagram의 Semantics

본 연구에서 제시하는 Timing diagram의 특징은 동기적이다. 예를 들어 (그림 1)과 같은 Timing diagram이 있다고 할 때, d_1시점에 w_1과 w_2의 값이 3, 5이어야 함을 의미하고, d_2시점에는 각각 0, 1이어야 함을 의미한다. 만약 시스템의 입력 (w_1, w_2)가 시간의 순서대로 (3, 5), (0, 5), (0, 1)이 들어왔다고 가정하자. 이 때, (3, 5)는 Timing diagram의 d_1시점에 표현된 값이고, (0, 1)은 Timing diagram의 d_2시점에 표현된 값이다. 그리고 (0, 5)는 입력 값이 (3, 5)에서 (0, 1)로 가는 도중의 입력이라 생각할 수 있다. 하지만 Timing diagram은 동기적이기 때문에 (0, 5)와 같은 입력은 어느 시점에도 표현된 입력이 아니므로 올바르지 않은 입력이라 할 수 있다. 그러므로 Timing diagram이 초기화 되어 있지 않을 때, 첫 번째 입력 (3, 5)가 들어오면 Timing diagram은 시점이 d_1으로 이동할 수 있지만, 두 번째 입력 (0, 5)가 들어오면 시점이 d_2으로 움직이지 않고 Timing diagram이 종료된다. Timing diagram은 두 번째 입력에 의해 종료된 상태이고, 재 초기화 되어야 할 필요가 있기 때문에 세 번째 입력인 (0, 1)이 들어와도 Timing diagram은 아무 행동도 보이지 않는다.

Timing diagram의 전이 행동은 초기화, 다음 시점으로의 전이, 현재 시점의 유지, 비 정상적 종료, 정상적 종료의 5가지로 나눌 수 있다. 이 중에서 비 정상적 종료는 Timing diagram에 기술된 대로 입력이 들어오지 않는 경우의 행동을 의미하는데, 현재 시점을 커버하고 있는 시간 제약을 위반하거나 현재 시점 또는 다음 시점을 만족시키는 입력이 들어오지 않을 경우에 대한 행동이다. 그 외의 4가지는 Timing diagram에 기술된 대로 입력이 들어왔을 경우에 대한 행동을 의미한다. 임의의 Timing diagram TD가 N개의 시점(d_1, d_2, … ,d_N)을가지고 있다고 할 때, TD에 대한 유효 시점은



(그림 1) 두 개의 Waveform을 가진 Timing diagram의 예

N-2개로 d_1 ~ d_{N-2}까지 이다. TD의 전이 행동은 아래와 같이 정의된다.

(1) 초기화 - TD가 동작 중이지 않을 때, TD는 초기화 되어야 할 필요가 있다. 초기화 조건은 d_1에 표현된 값과 입력 waveform의 값이 같아야 한다.

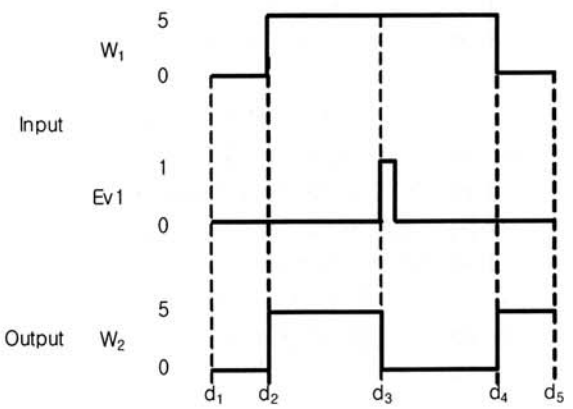
(2) 다음 시점으로의 전이 - i의 값이 1과 N-3범위 내에 있고 현재 시점을 d_i라고할 때, 현재 시점에서 다음 시점으로 이동하기 위한 조건은 현재 시점을 커버하고 있는 시간 제약들이 모두 시간 제약을 위반하지 않으면서 다음 시점에서 끝나는 시간 제약들을 모두 만족하고, 입력 waveform의 값이 d_{i+1}에 표현된 값과 같으면 시점은 d_i에서 d_{i+1}로 이동한다.

(3) 현재 시점의 유지 - i의 값이 1과 N-2범위 내에 있고 현재 시점을 d_i라고 할 때 현재 시점을 커버하고 있는 시간 제약들을 위반하지 않고, 다음 시점으로 이동하기 위한 조건에 맞지 않으며, 입력 waveform이 현재 시점에 표현된 대로 값이 들어오게 되면 시점은 d_i를 유지한다. 단 이벤트 타입의 waveform은 현재 시점을 유지하기 위해서는 이벤트가 발생이 되면 안된다. 그 이유는 이벤트 타입의 입력의 경우 현재 시점을 유지시키기 위해 이벤트를 주기적으로 발생시켜야 하는 시스템의 사양을 고려하지 않았기 때문이다.

(4) 비정상적 종료 - i의 값이 1과 N-2범위 내에 있고 현재 시점을 d_i라고 할 때 현재 시점을 커버하고 있는 시간 제약들 중 적어도 하나가 제약을 위반하면 Timing diagram은 비 정상적으로 종료된다. 혹은 현재 시점을 유지한 조건도 만족하지 못하고 다음 시점으로 이동하기 위한 조건도 만족하지 못하는 경우에도 Timing diagram은 비 정상적으로 종료된다. 예를 들어 시간 제약이 (d_s, d_e, t, 3)인 시간 제약이 있고, 현재 시점 d_i가 $d_s \le d_i < d_e$의 관계를 만족할 때, 시간 제약의 타입 t가 After인 경우에는 시간 제약을 위반하는 조건은 존재하지 않으며, In혹은 During인 경우 3보다 크게 되면 시간 제약을 위반한 것이 된다.

(5) 정상적 종료 - 현재 시점이 d_{N-2}일 때, 현재 시점을 커버하고 있는 시간 제약들이 모두 시간 제약을 위반하지 않으면서 다음 시점 d_{N-1}에서 끝나는 시간 제약들을 모두 만족하고, 입력 waveform의 값이 d_{N-1}에 표현된 값과 같으면 Timing diagram은 정상적으로 종료된다.

Timing diagram의 전이의 결과에 대한 출력은 위의 전이 종류에서 초기화, 다음 시점으로의 전이, 정상적 종료일 때에만 출력이 발생하며, 그 외의 행동에 대해서는 출력 결과를 내보내지 않는다. 전이에 의해 이동하게 될 시점이 d_{i+1}일 경우, 구간 [d_{i+1}, d_{i+2}]에 대한 출력 waveform의 결과가 출력으로 나가게 되며, 시점 d_{i+1}에서 새로 시작하는 시간 제약이 있을 경우 그 시간 제약을 초기화 시킨다.

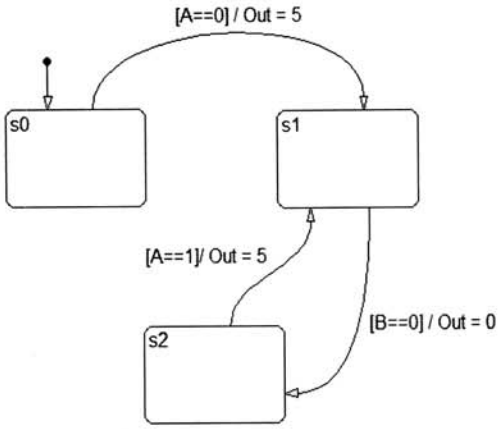


(그림 2) Timing diagram의 예

(그림 2)의 Timing diagram모델을 예로 들어서 Timing diagram의 행동을 설명해 보도록 하겠다. 아래의 Timing diagram의 입력 Waveform은 W_1, Ev1이고 출력 Waveform은 W_2이다. Timing diagram에 들어오는 입력 Waveform W_1, Ev1의 값을 각각 Value(W_1), Value(Ev1)이라 할 때, 편의상 Timing diagram에 들어오는 입력 Waveform들의 값을 (Value(W_1), Value(Ev1))으로 표현하도록 한다. (그림 2)의 Timing diagram이 초기화 되지 않았다고 가정할 때, Timing diagram이 초기화 되기 위해서는 (0, 0)의 값이 들어와야 한다. (0, 0)이 들어오면 시점은 d_1으로 초기화되며, W_2의 값은 0으로 정의된다. 시점이 d_1에 있을 때, d_2로 이동하기 위해서는 (1, 0)의 값이 들어와야 하며 이때의 출력 W_2는 5로 정의된다. 하지만 d_1에서 d_1을 유지하기 위해서는 (0, 0)의 값이 들어와야 한다. 만약 d_1에서 (5, 0)이란 값이 들어왔다고 가정해 보자. (5, 0)은 현재 시점을 유지하는 조건을 만족시키지 못하면서 다음 시점인 d_2로 이동하기 위한 조건도 만족하지 못한다. 그러므로 이 경우 Timing diagram은 비정상적으로 종료하게 되며, 다시 동작하기 위해서는 초기화 입력인 (0, 0)이 입력으로 들어와야 한다. d_2에서 d_3로 이동하기 위해서는 (5, 1)의 값이 들어와야 하며, 이때의 W_2의 값은 0이 된다. 현재 시점이 d_3일 때, 현재 시점을 유지시키기 위해서는 (5, 0)의 값이 들어와야 한다. d_3에서 d_4로 이동하기 위해서는 (0, 0)의 입력이 들어와야 한다. 이 때 Timing diagram은 d_4가 종료하는 시점이기 때문에 출력 W_2를 5로 정의하고, Timing diagram은 종료된다.

3.3 Timing diagram의 Stateflow로의 변환

Stateflow는 Harel[15]이 정의한 Statecharts를 Mathworks에서 Simulink와 연동하여 사용할 수 있도록 한 도구로 기본 문법과 의미는 Statecharts와 매우 유사하다. Stateflow는 <S, T, L>로 정의할 수 있는데, S는 상태 집합을 의미하고, T는 상태 전이 집합을 의미하며 S x L x S로 정의된다. 여기서 L는 상태 전이의 레이블을 의미한다. (그림 3)에 Stateflow모델에 대한 예를 나타내었다. 이 모델의 상태 집



(그림 3) Stateflow 모델의 예

합 S는 {s0, s1, s2}이며, 상태 전이 집합 T는 {(s_ε, ε, s0), (s0, "[A==0] / Out = 5", s1), (s1, "[B==0] / Out = 0", s2), (s2, "[A==1]/ Out = 5", s1)}이며, 여기서 s_ε는 초기 전이를 위한 null 상태를 의미하며, ε는 null 문자열을 의미한다. 초기 전이란 Stateflow가 활성화 될 때 시작을 정의하기 위한 전이로, (그림 3)에서는 s0가 초기 상태가 된다. 상태 전이 레이블에서 조건은 "["과 "]"사이에 조건을 기술하며, 전이의 행동은 "/" 뒤에 기술한다.

Stateflow의 행동과 Timing diagram의 행동을 비교하자면, Stateflow는 상태의 전이에 의하여 행동이 일어나고, Timing diagram은 시점의 전이에 의하여 행동이 일어난다. 그러므로 Stateflow에서의 상태는 Timing diagram의 시점과 매우 비슷한 의미를 지님을 알 수 있다. 또한 Stateflow에서의 전이는 Timing diagram에서의 전이와 매우 비슷한 의미를 지니고 있으나, Timing diagram에서는 근접 시점의 한 방향으로만 전이가 될 수 있다는 점이 Stateflow와 Timing diagram의 차이가 있다. 이런 유사한 점을 바탕으로 임의의 Timing diagram모델로부터 동치 관계에 있는 Stateflow 모델로 변환하는 것이 가능한데, 이를 설명하기 전에 3.2절에서 설명한 Timing diagram의 전이 조건을 수식으로 표현하고자 한다. k개의 입력 waveform을 가지고 N개의 시점을 가지고 있는 Timing diagram의 전이 조건들은 <표 1>과 같이 정의된다.

여기서 $C_{stay}(d_i)$, $C_{center}(d_{i+1})$는 각각 시점 d_i, d_{i+1}에 표현된 Waveform의 값을 만족시키기 위한 조건인데, $C_{stay}(d_i)$는 현재 시점을 유지하기 위한 waveform의 조건이고, $C_{center}(d_{i+1})$은 다음 시점으로 이동하기 위한 waveform의 조건이다. Value(w_i)를 Timing diagram에 들어오는 waveform W_i에 대한 입력값이라고 정의할 때, $C_{stay}(d_i)$, $C_{center}(d_{i+1})$는 각각 다음과 같이 정의할 수 있다.

$$C_{stay}(d_i) \equiv \bigwedge_{i=1}^{k} \begin{array}{l} \text{Value}(w_i) = \delta(w_i, d_i), \text{ if } \rho(w_i) = \text{Event} \\ \text{Value}(w_i) = 0, \text{ if } \rho(w_i) = \text{Signal} \end{array}$$

$$C_{center}(d_i) \equiv \bigwedge_{i=1}^{k} \text{Value}(w_i) = \delta(w_i, d_i)$$

$TC_{meet}(d_{i+1})$는 시점 d_{i+1}에서 끝나는 시간 제약의 조건이 만족되기 위한 조건을 의미하며, $TC_{violation}(d_i)$는 시점 d_i를 커버하고 있는 시간 제약들 중 적어도 하나가 제약위반을 할 조건을 의미한다. $TC_{meet}(d_{i+1})$, $TC_{violation}(d_i)$는 다음과 같이 정의할 수 있다.

$$TC_{meet}(d_i) \equiv \bigwedge_{tc \in TC(i)} \text{Meet}(tc)$$

여기서, TC(i)는 d_i에서 끝나는 시간 제약의 집합

tc = <d_s, d_e, t, v>일 때,

$$\text{Meet}(tc) \equiv \begin{array}{l} tc.timer > tc.value, \text{ if } t = \text{After} \\ tc.timer = tc.value, \text{ if } t = \text{Duration} \\ tc.timer < tc.value, \text{ If } t = \text{In} \end{array}$$

$$TC_{violation}(d_i) \equiv \bigvee_{tc \in Cov(i)} \text{Violation}(tc),$$

여기서 Cov(i)는 d_i를 커버하는 시간 제약의 집합

$$\text{Violation}(tc) \equiv \begin{array}{l} \text{false, if } t = \text{After} \\ tc.timer > tc.value, \text{ if } t = \text{Duration} \\ tc.timer \geq tc.value, \text{ If } t = \text{In} \end{array}$$

지금까지 설명한 Timing diagram의 의미와, Stateflow의 의미로부터 Timing diagram 모델을 Stateflow모델로 변환하는 방법에 대해서 설명한다. 임의의 Timing diagram 모델 T는 하나의 Stateflow의 상태인 s_T로 변환이 되며, T의 행동은 s_T의 내부 상태와 내부 전이에 의해 표현이 된다. 그리고 Stateflow의 상태 s_T가 활성화 되면 그 s_T의 하위 상

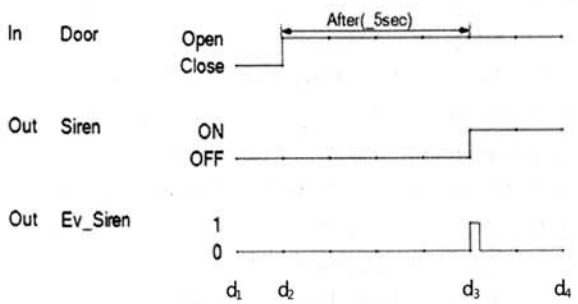
〈표 1〉 Timing diagram의 전이 종류에 따른 조건

전이 종류		조건식
초기화		$C_{center}(d_1)$
d_i에서 d_{i+1}로의 전이	$(1 \leq i \leq N-3)$	$C_{center}(d_{i+1}) \wedge TC_{meet}(d_{i+1}) \wedge \neg C_{violation}(d_i)$
d_i의 유지	$(1 \leq i \leq N-2)$	$C_{stay}(d_i) \wedge \neg C_{violation}(d_i) \wedge (\neg C_{center}(d_{i+1}) \vee (C_{center}(d_{i+1}) \wedge \neg C_{meet}(d_{i+1})))$
d_i에서의 비정상적 종료	$(1 \leq i \leq N-2)$	$TC_{violation}(d_i) \vee (\neg C_{stay}(d_i) \wedge (\neg C_{center}(d_{i+1}) \vee \neg C_{meet}(d_{i+1})))$
d_{N-2}에서의 정상적 종료		$C_{center}(d_{N-1}) \wedge TC_{meet}(d_{N-1}) \wedge \neg C_{violation}(d_{N-2})$

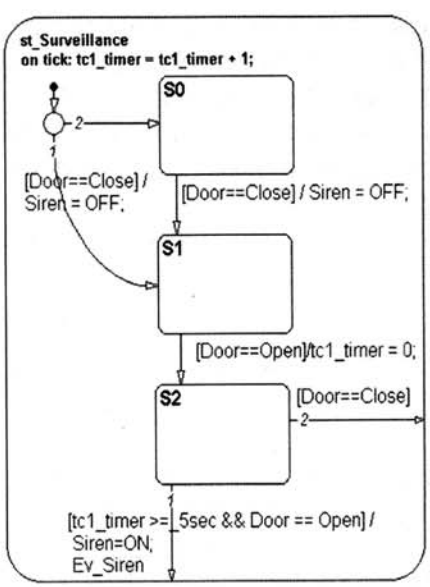
태도 활성화 되어야 한다. 이를 초기 전이라고 하는데, Stateflow는 초기전이가 반드시 필요하기 때문에 Timing diagram 모델에서 Stateflow 모델로 변환할 때에 이 점을 고려해야 할 필요가 있다. 이를 위해 Timing diagram이 초기화 되는 경우와 초기화 되지 않는 경우로 나누고 s_T가 활성화 될 때, Timing diagram이 초기화 조건을 만족시키면 초기 전이를 s_1으로 발생시키고 그렇지 않은 경우에는 초기 전이를 s_0로 발생시키는 방식으로 Timing diagram의 의미와 Stateflow의 의미를 같도록 변환할 수 있다. 그리고 임의의 시점 d_i에서 다른 시점 d_{i+1} 혹은 Timing diagram의 종료는 상태 s_i에서 s_{i+1}혹은 s_T로의 전이로 변환시킬 수 있으며, Timing diagram의 전이조건과 행동은 Stateflow의 전이 레이블 규칙에 맞추어서 변환시킬 수 있다. 위의 내용을 바탕으로 N개의 시점을 가지고 있는 Timing diagram T로부터 Stateflow로 변환하는 함수를 다음과 같이 정의할 수 있다.

(1) Timing diagram T에 대해 Stateflow에서의 상태 s_T로 변환한다.
(2) T의 초기화 되지 않은 상태에 대해 s_0로 변환한다.
(3) T의 시점 $d_i(1 \le i \le N-2)$는 Stateflow에서의 상태 s_i로 변환한다.
(4) T의 시점 이동은 Stateflow에서의 상태 전이로 변환한다.
(4-1) T의 초기화는 Stateflow에서의 상태 s_0에서 s_1로의 전이 혹은 s_ε에서 s_1로의 전이인 초기 전이로 변환된다. 두 전이들의 조건은 모두 $C_{center}(d_1)$이다. 그리고 s_ε에서 s_0으로의 전이인 초기 전이 조건은 $\neg C_{center}(d_1)$와 같다.
(4-2) i가 $1 \le i \le N-3$일 때, T에서의 시점 d_i에서 d_{i+1}로의 이동은 Stateflow에서의 상태 s_i에서 s_{i+1}로의 전이로 변환된다. 전이 조건은 $TC_{violation}(d_i) \vee (\neg C_{stay}(d_i) \wedge (\neg C_{center}(d_{i+1}) \vee \neg C_{meet}(d_{i+1})))$와 같다.
(4-3) T에서의 d_{N-2}에서의 정상적 종료는 Stateflow에서의 상태 s_{N-2}에서 상태 s_T로의 전이로 변환된다. 전이 조건은 $C_{center}(d_{N-1}) \wedge TC_{meet}(d_{N-1}) \wedge \neg C_{violation}(d_{N-2})$와 같다.
(4-4) i가 $1 \le i \le N-2$일 때, T에서의 비정상적 종료는 Stateflow에서의 상태 s_i에서 s_T로의 전이로 변환된다. 전이 조건은 $TC_{violation}(d_i) \vee (\neg C_{stay}(d_i) \wedge (\neg C_{center}(d_{i+1}) \vee \neg C_{meet}(d_{i+1})))$와 같다.

위 변환 과정에 대한 설명을 위해 도난 경보 예제를 이용하도록 하겠다. (그림 4)는 도난 경보 시스템의 Timing diagram모델이고 (그림 5)는 Timing diagram 모델을 Stateflow 모델로 변환한 그림이다. (그림 4)의 Timing diagram모델은 Door가 Close에서 Open상태가 되고 _5sec가 지나게 되면 Siren이 OFF에서 ON이 되고 이벤트 Ev_Siren을 출력하는 사양을 기술한 모델이다. 이 Timing diagram 모델의 이름을 st_Surveillance라고 할 때, 이 Timing diagram모델은 Stateflow의 상태 st_Surveillance로 변환이 되며, Timing diagram 모델의 행동은 Stateflow에서 상태 st_Surveillance의 내부 상태와 상태 전이로 기술된다. (그림 4)에서 시점은 d_1, d_2, d_3, d_4로 네 개가 존재하는데, Stateflow로의 상태 변환은 모든 시점에서 뒤의 시점 두 개를 제외한 것과 s_0를 포함하여 s_0, s_1, s_2가 생성된다. 그리고 초기 전이는 (s_ε, "[Door=Close]/Siren=OFF", s_1)과 (s_ε, "", s_0)이며, s_ε에서 s_1으로의 전이는 상태 st_Surveillance가 활성화 될 때, Timing diagram의 모델이 초기화가 될 때의 행동을 의미하며, s_ε에서 s_0으로의 전이는 Timing diagram의 모델이 초기화가 되지 않을 때의 행동을 의미한다. 그리고 Stateflow에서의 상태전이 (s_ε, "", s_0)는 (그림 5)에서 s_ε에서 발생하는 두 번째 전이에 해당하며, 이 전이가 일어나기 위한 조건은 "[Door!=Close]"를 의미한다. 그리고 시점 d_2에서 시간제약 하나가 시작되므로, 이를 Stateflow모델에서는 s_1에서 s_2로의 전이의 출력 행동에서 tc1_timer를 0으로 초기화 시켰다. 그리고 Stateflow의 st_Surveillance 모델에서 on tick은 시스템이 동작하는 매 시점마다 발생하는 이벤트로 "on tick:



(그림 4) Timing diagram의 예



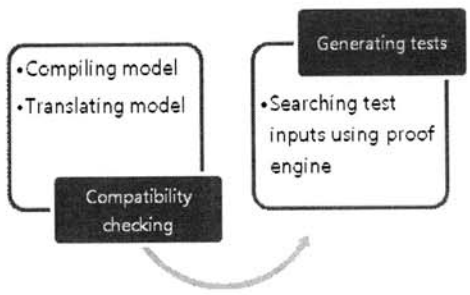
(그림 5) (그림 4)로부터 변환된 Stateflow 모델

tc1_timer = tc1_timer + 1;"의 의미는 시스템이 동작하는 시점마다 시간 제약의 타이머 카운트를 증가시키라는 의미가 된다.

4. Simulink Design Verifier

Simulink Design Verifier(SDV)[12]는 Mathwork사의 Matlab/Simulink의 툴 셋으로 본 논문의 Timing diagram으로부터 테스트 케이스를 생성 하는 전략의 세 번째 단계에서 사용하는 도구이다. SDV는 시스템을 위한 테스트 케이스 생성뿐만 아니라 시스템의 검증을 위한 도구로 Simulink 모델의 주어진 속성을 만족하는지에 대해 검사하기 위해서 모든 경우에 대한 형식적 분석을 수행한다. SDV는 이를 위해 Prover라는 프로그램을 사용하는데, Prover[13]는 조합 디자인 자동화 문제를 해결하기 위해 Prover Technology에서 개발한 프로그램으로 BDD[11] 기반 혹은 DPLL[10]과 같은 SAT solver와 같은 기술들을 통합한 증명 엔진 접근 방식의 프로그램이며, Matlab/Simulink에 플러그 인 되어 있다.

SDV에서 테스트 케이스를 생성하는 방식은 다음과 같은 절차에 따라 진행된다. SDV는 첫 번째 단계로 호환성 체크를 하는데, 이 과정은 Prover가 모델을 처리할 수 있는지의 여부를 검사하는 과정이다. 호환성을 체크하는 방법은 일차적으로 Simulink 모델을 컴파일 하고 모델에 문제가 없으면 이를 Prover가 인식할 수 있는 모델로 변환시킨다. 호환성 체크가 끝나면 두 번째 과정으로 변환된 모델을 이용하여 테스트 입력들을 찾아낸다. 이를 위해 Prover는 증명 엔진을 사용하여 Simulink모델의 커버리지 목적에 따라 테스트 입력들을 찾아낸다. 위의 설명에 대한 내용이 (그림 6)에 표현되어 있다.



(그림 6) Simulink Design Verifier의 동작 흐름

5. 실 험

본 장에서는 Timing diagram으로부터 Stateflow로 변환하여 변환된 모델을 이용하여 테스트 케이스를 생성하는 접근 방법이 적합한지를 확인하기 위해 두 가지 방법으로 실험을 수행하였다. 첫 번째는 3장에서 제시한 도난 경보 모델을 대상으로 수행하였고, 두 번째는 임의의 Timing diagram 모델 50개를 대상으로 수행하였다. 첫 번째와 두 번째 모두 Timing diagram으로부터 Stateflow로 변환하고, 변환된 Stateflow모델에 대해 SDV를 이용하여 MC/DC커버리지 범주로 테스트 케이스를 생성하였으며, 모델에서 생성할 수 있는 총 MC/DC 케이스 목표와 SDV에서 생성한 MC/DC 케이스 개수를 비교하여 변환된 모델에 대해 테스트 케이스 생성 효율을 평가하였다.

5.1 도난 경보 모델에 대한 테스트 케이스 생성 실험

본 연구의 3장에서 언급한 (그림 4)의 도난 경보 Timing diagram 모델에 대해 (그림 5)와 같이 Stateflow모델로 변환하고 SDV를 이용하여 MC/DC관점의 테스트 케이스를 생성하였다. 수행 결과 21개의 테스트 목표가 생성되었고, 21개의 테스트 케이스 목표를 만족시키는 테스트 입력을 생성했으며, 테스트 입력을 생성하는데 걸리는 시간은 0.01초였다. (그림 7)은 SDV의 수행 결과 만족된 테스트 케이스 목표에 대한 보고서이며, (그림 8)은 SDV가 생성한 테스트 케이스 입력을 나타낸다.

(그림 7)의 만족된 테스트 케이스 목표가 실제 Timing diagram 모델에 대해 어떤 의미를 갖는지에 대해서 설명해 보도록 하겠다. (그림 7)에서 tc_n을 n번째 테스트 케이스 목표라고 표현할 때, 첫 번째와 두 번째 테스트 케이스인 tc_1, tc_2는 Timing diagram 모델의 시점 d_2에서 비 정상적 종료에 대한 전이행동을 확인하기 위한 테스트 케이스 목표가 된다. 즉 Timing diagram의 시점이 d_2일 때, Door의 값이 Close이면 Timing diagram은 비정상적으로 종료되며, Close가 아니면 비 정상적으로 종료되지 않는다. 그러므로 이 테스트 케이스들은 Timing diagram의 모델의 시점이 d_2일 때의 행동을 테스트 하는 것이므로 의미가 있다고 할 수 있다.

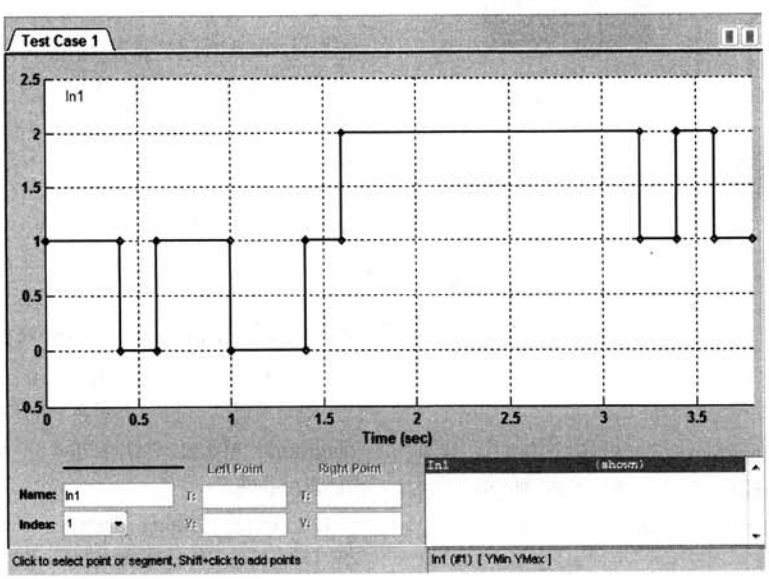
tc_3부터 tc_{12}는 Timing diagram의 모델의 시점이 d_2에서 정상적으로 종료하는 상황에 대한 테스트 케이스를 나타내는데, 조건(Condition), 판단(Decision), MC/DC 커버리지 범주에 의해 생성이 되었으며 중복되는 테스트 케이스들이 존재한다. tc_3과 tc_4는 조건 커버리지 관점에서 "tc1_timer >= _5sec"의 값이 참 또는 거짓이 되기 위한 테스트 케이스 목표이고, tc_5와 tc_6은 조건 커버리지 관점에서 "Door == Open"의 값이 참 또는 거짓이 되기 위한 테스트 케이스 목표이다. 그리고 tc_7, tc_8은 판단 커버리지 관점에서의 테스트 케이스 목표이고, tc_9~tc_{12}는 MC/DC커버리지 관점에서의 테스트 케이스 목표이다. 이 중에서 tc_9와 tc_{11}은 "tc1_timer >=_5sec"의 값만 변화시키고 다른 조건은 변화시키지 않을 때, Timing diagram의 행동에 변화가 있음을 보이기 위한 테스트 목표를 의미하며, tc_{10}과 tc_{12}는 "Door == Open"의 값만 변화시키고 다른 조건은 변화시키지 않을 때, Timing diagram의 행동에 변화가 있음을 보이기 위한 테스트 목표를 의미한다.

tc_{13}, tc_{14}는 Timing diagram 모델의 시점이 d_1에서 d_2로 전이되는지를 확인하기 위한 목표이다. 그리고 tc_{15}, tc_{16}, tc_{20}, tc_{21}은 Timing diagram이 초기화되지 않았을 때의 행동

#	Type	Model Item	Description
1	Decision	Chart.st_Surveillance."[Door==Close]"	Transition: Transition trigger expression F
2	Decision	Chart.st_Surveillance."[Door==Close]"	Transition: Transition trigger expression T
3	Condition	Chart.st_Surveillance."[tc1_timer >=_5sec && Door ..."	Transition: Condition 1, "tc1_timer >=_5sec" T
4	Condition	Chart.st_Surveillance."[tc1_timer >=_5sec && Door ..."	Transition: Condition 1, "tc1_timer >=_5sec" F
5	Condition	Chart.st_Surveillance."[tc1_timer >=_5sec && Door ..."	Transition: Condition 2, "Door == Open" T
6	Condition	Chart.st_Surveillance."[tc1_timer >=_5sec && Door ..."	Transition: Condition 2, "Door == Open" F
7	Decision	Chart.st_Surveillance."[tc1_timer >=_5sec && Door ..."	Transition: Transition trigger expression F
8	Decision	Chart.st_Surveillance."[tc1_timer >=_5sec && Door ..."	Transition: Transition trigger expression T
9	Mcdc	Chart.st_Surveillance."[tc1_timer >=_5sec && Door ..."	Transition: MCDC Transition trigger expression with Condition 1, "tc1_timer >=_5sec" T
10	Mcdc	Chart.st_Surveillance."[tc1_timer >=_5sec && Door ..."	Transition: MCDC Transition trigger expression with Condition 2, "Door == Open" T
11	Mcdc	Chart.st_Surveillance."[tc1_timer >=_5sec && Door ..."	Transition: MCDC Transition trigger expression with Condition 1, "tc1_timer >=_5sec" F
12	Mcdc	Chart.st_Surveillance."[tc1_timer >=_5sec && Door ..."	Transition: MCDC Transition trigger expression with Condition 2, "Door == Open" F
13	Decision	Chart.st_Surveillance."[Door==Open]/tc1_timer = 0;"	Transition: Transition trigger expression F
14	Decision	Chart.st_Surveillance."[Door==Open]/tc1_timer = 0;"	Transition: Transition trigger expression T
15	Decision	Chart.st_Surveillance."[Door==Close] / Siren = OFF;"	Transition: Transition trigger expression F
16	Decision	Chart.st_Surveillance."[Door==Close] / Siren = OFF;"	Transition: Transition trigger expression T
17	Decision	Chart.st_Surveillance	State: Substate executed State "S0"
18	Decision	Chart.st_Surveillance	State: Substate executed State "S1"
19	Decision	Chart.st_Surveillance	State: Substate executed State "S2"
20	Decision	Chart.st_Surveillance."[Door==Close] / Siren = OFF;"	Transition: Transition trigger expression F
21	Decision	Chart.st_Surveillance."[Door==Close] / Siren = OFF;"	Transition: Transition trigger expression T

(그림 7) 만족된 테스트 케이스 목표

을 확인하기 위한 목표이다. 즉 Door의 값이 Close이면 Timing diagram모델은 초기화되며, 시점은 d_1으로 이동하게 된다. 반면 Door의 값이 Close가 아니면 Timing diagram모델은 초기화되지 않게 된다. 그러므로 초기화되지 않은 상태에서 Door의 값이 Close인지 아닌지의 여부를 검사하는 것은 Timing diagram의 모델이 올바르게 동작하는지의 여부를 확인할 수 있으므로, 의미가 있다고 할 수 있다. 그리고 마지막으로 tc_{17}~tc_{19}는 각각 Timing diagram모델이 초기화 되지 않는지의 여부, 시점 d_1, d_2에 도달 가능한지의 여부를 확인하기 위한 테스트 케이스 목표가 된다.



(그림 8) (그림 5)의 테스트 입력

SDV를 이용하여 MC/DC커버리지 범주로 테스트 케이스를 생성하면 조건(Condition), 판단(Decision), MC/DC 범주를 모두 생성하는데, 조건과 판단 범주가 MC/DC범주에 포함되므로 생성된 테스트 케이스 목표들이 중복되는 것들이 존재한다. 하지만, 생성된 테스트 케이스 목표들은 Timing diagram의 행동이 올바른지의 여부를 검사하는 테스트 케이스들임을 알 수 있다.

5.2 임의의 Timing diagram 모델에 대한 테스트 케이스 생성 실험

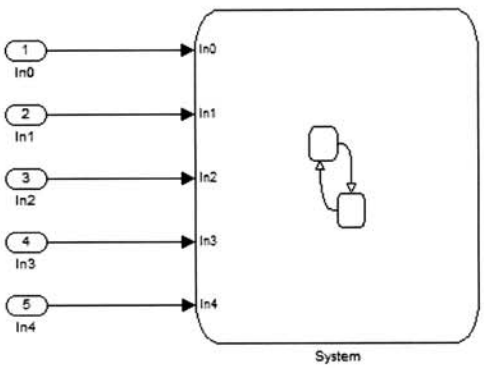
Timing diagram의 복잡도는 Waveform의 개수, 시간 제약의 개수, 시점의 개수에 영향을 받는다. 변환된 Stateflow의 관점에서 봤을 때, Waveform의 개수는 상태 전이에 대한 조건식의 복잡도에 영향을 미친다. Waveform의 개수가 적으면 조건식이 비교적 간단하지만, 개수가 많으면 조건식이 복잡해진다. 시간 제약의 개수는 상태 전이에 대한 조건식의 복잡도 및 상태전이의 개수에 영향을 미친다. 마지막으로 시점의 개수는 상태의 개수에 영향을 미친다.

임의의 Timing diagram모델을 생성하기 위해서 Waveform의 개수와 시간 제약의 개수와 시점의 개수를 변화시켜서 50개의 Timing diagram 모델을 생성하였으며, 각각의 Timing diagram모델을 Stateflow모델로 변화시켜서 변화된 Stateflow 모델로부터 SDV를 이용하여 MC/DC관점의 테스트 케이스 목표와 테스트 케이스 생성 효율을 평가하고자 하였다. 이를 위해 임의의 Timing diagram 모델 5개를 Stateflow모델로 만들어서 하나의 세트로 구성하여 각각의 timing diagram 모델이 병렬적으로 동작하도록 모델을 구축하였으며, 각각의 timing diagram은 다른 timing diagram에게 영향을 미치지 못한다.

하지만 timing diagram이 복잡한 경우 변환된 stateflow도 복잡하게 되며, 이는 SDV가 테스트 케이스를 생성하는데 부담으로 작용하기 때문에 복잡한 모델에 대해서는 하나의 세트에서 따로 뽑아서 새로운 모델로 구성하였다. 예를 들어 <표 2>는 다섯 개의 timing diagram모델을 한 세트로 묶어서 테스트 케이스를 생성한 결과에 대한 정리를 나타낸 표인데, <표 2>의 Set1의 경우 네 개의 모델을 한 세트로 묶어서 하나의 Stateflow모델로 생성하고, 나머지 하나의 모델을 따로 Stateflow모델로 생성했다. Set2의 경우에는 다섯 개의 모델을 하나의 세트로 묶어서 하나의 Stateflow모델로 생성하였다. 이와 같이 함으로써 SDV가 효율적으로 테스트 케이스를 생성하도록 하였다.

(그림 9)는 Simulink의 최상위 모델로 각 세트의 최 상위 모델은 (그림 9)와 동일한 구조로 되어 있다. 시스템의 입력은 In0에서 In4까지 5개가 있으며, System블록 내의 Stateflow모델들은 In0, In1, …, In4를 입력으로 받아 행동을 하게 된다. 본 실험에서는 출력 포트를 정의하지 않았는데, 그 이유는 시스템의 상태 전이를 Stateflow에서의 상태로 판별이 가능하기 때문이다.

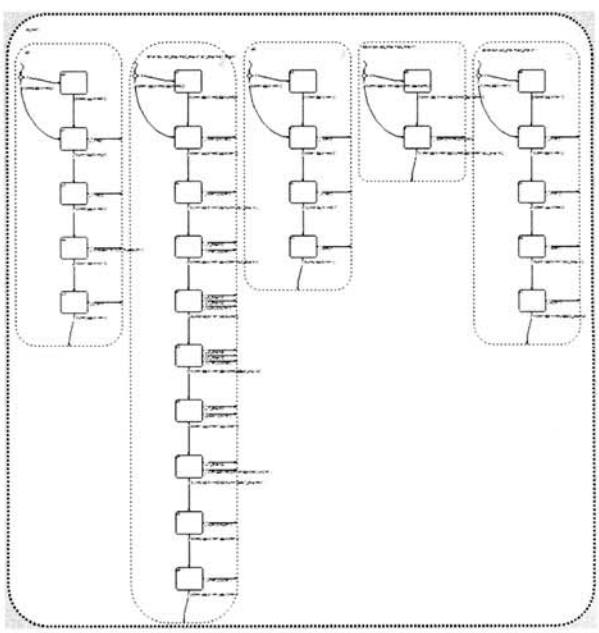
System블록의 하위 블록에는 (그림 10)과 같은 변환된



(그림 9) Simulink의 최상위 모델

Stateflow모델들이 들어있는데, 이 그림은 두 번째 세트의 실험에 대한 변환된 Stateflow모델을 나타낸 것이다. 그림에서 다섯 개의 그룹화된 박스들은 그 각각이 임의로 생성된 Timing diagram모델로부터 변환된 것을 의미한다. 다시 말하면 (그림 10)에서 가장 왼쪽에 있는 첫 번째 사각형은 두 번째 세트(Set2)의 첫 번째 Timing diagram모델로, Timing diagram 모델의 상태의 개수는 5개이고 상태 전이의 개수는 11개이다.

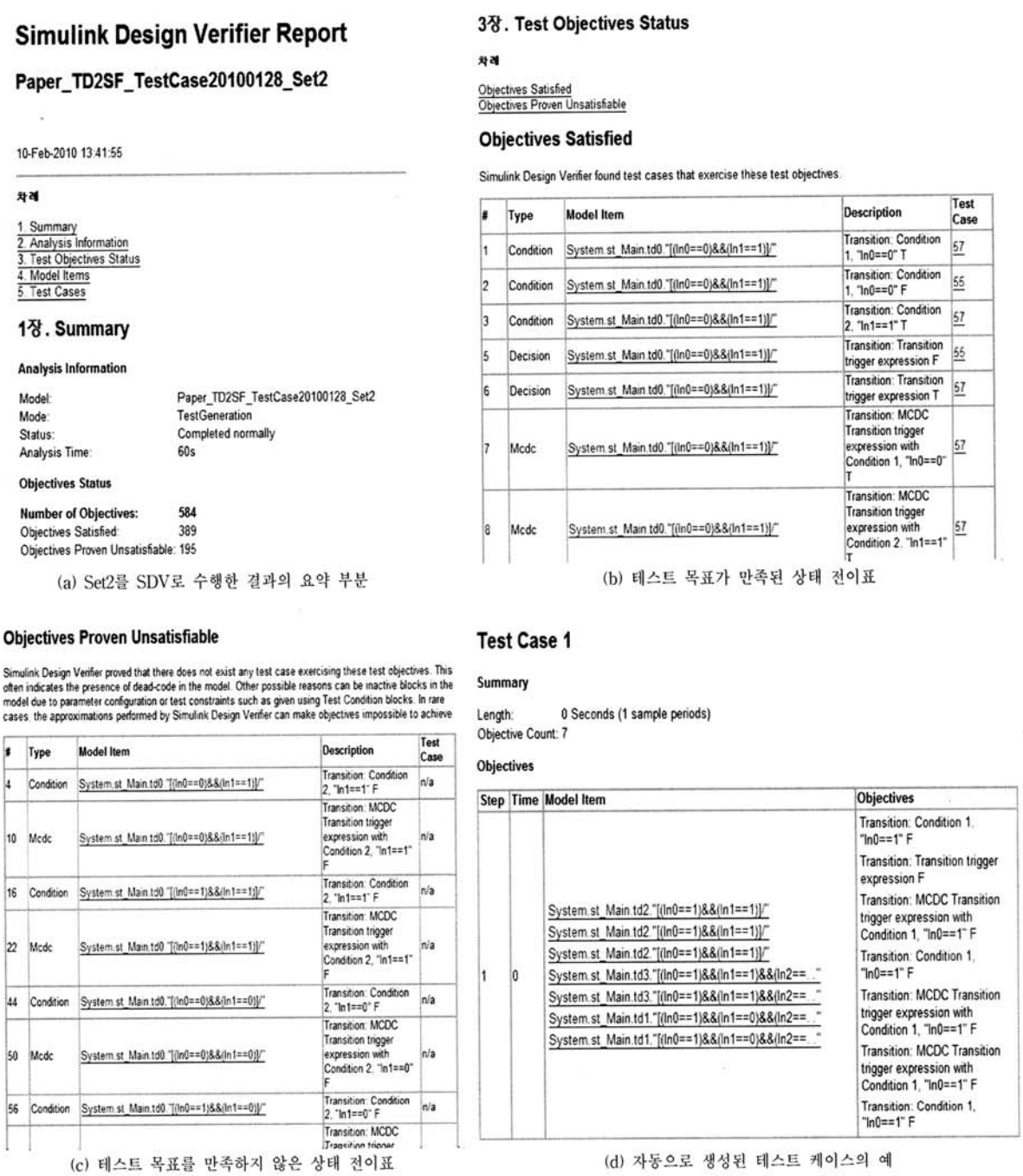
(그림 10)과 같이 변환된 Stateflow모델에 대해 SDV를 수행하면 (그림 11)과 같은 테스트 케이스 결과 보고서가 생성된다. 이 보고서를 통해 (그림 11)의 (a)와 같이 모델에 대한 테스트 케이스 생성과 관련된 통계 정보를 알 수 있으며, (b)와 (c)와 같이 테스트 커버리지 범주에 대한 테스트 케이스 목표 중에서 만족된 목표 및 만족되지 않은 목표에 대한 정보를 알 수 있고, (d)와 같이 자동으로 생성된 테스트 케이스의 정보를 확인할 수 있다. 이와 같이 변환된 Stateflow모델 50개에 대해 SDV를 이용하여 테스트 케이스



(그림 10) <표 2>의 Set2에서 변환된 Stateflow모델

〈표 2〉 임의의 Timing diagram모델에 대한Test case의 생성 결과

	# State	# Tran-sition	# Wave-form	# Timing Constraint	# Obj-ective	# Sat	# Unsat	# Undecided	Analysis Time	Coverage Ratio
Set1	11	25	4	1	574	482	70	22	1:12	84.0%
	3	7	2	1						
	2	5	3	0						
	2	5	2	1						
	12	36	4	7	516	433	78	5	11:07	83.9%
Set2	5	11	2	0	584	389	195	0	3:34	66.6%
	10	28	3	2						
	4	9	2	0						
	2	5	4	1						
	5	11	2	1						
Set3	4	9	2	1	602	535	68	2	3:46	88.9%
	9	21	3	5						
	8	18	3	1						
	2	5	2	0						
	14	44	3	6	486	146	340	0	0:50	30.0%
Set4	4	10	4	1	350	306	44	0	0:15	87.4%
	2	5	3	1						
	6	13	3	1						
	8	20	2	1	296	216	80	0	5:05	73.0%
	9	20	2	1						
Set5	4	12	2	2	630	547	83	0	1:15	86.8%
	6	14	2	1						
	6	14	4	1						
	3	7	4	0						
	5	11	3	0						
Set6	8	19	2	1	948	558	389	1	7:03	58.9%
	7	15	2	0						
	6	13	2	1						
	9	20	2	2						
	11	35	4	5						
Set7	9	23	4	5	463	378	85	0	6:55	81.6%
	9	21	2	2	143	116	27	0	1:50	81.1%
	12	29	4	4	518	423	78	17	8:30	81.7%
	11	35	3	6	393	242	151	0	0:48	61.6%
	5	13	4	1	171	147	24	0	0:05	86.0%
Set8	6	15	2	1	285	228	57	0	0:25	80.0%
	4	10	2	1						
	5	13	2	2						
	8	23	2	4	146	100	43	3	2:26	68.5%
	10	25	4	3	460	410	50	0	1:00	89.1%
Set9	5	12	2	2	801	708	93	0	5:08	88.4%
	7	18	2	2						
	10	23	3	1						
	7	16	2	1						
	4	9	4	1						
Set10	4	9	2	0	637	481	156	0	1:51	75.5%
	6	13	4	1						
	6	13	4	1						
	9	21	2	3						
	12	28	4	5	552	366	94	92	9:01	66.3%



Simulink Design Verifier Report

Paper_TD2SF_TestCase20100128_Set2

10-Feb-2010 13:41:55

차례

1 Summary
2 Analysis Information
3 Test Objectives Status
4 Model Items
5 Test Cases

1장. Summary

Analysis Information

Model:	Paper_TD2SF_TestCase20100128_Set2
Mode:	TestGeneration
Status:	Completed normally
Analysis Time:	60s

Objectives Status

Number of Objectives:	**584**
Objectives Satisfied:	389
Objectives Proven Unsatisfiable:	195

(a) Set2를 SDV로 수행한 결과의 요약 부분

3장. Test Objectives Status

차례

Objectives Satisfied
Objectives Proven Unsatisfiable

Objectives Satisfied

Simulink Design Verifier found test cases that exercise these test objectives.

#	Type	Model Item	Description	Test Case
1	Condition	System.st_Main.td0."[(In0==0)&&(In1==1)]/"	Transition: Condition 1, "In0==0" T	57
2	Condition	System.st_Main.td0."[(In0==0)&&(In1==1)]/"	Transition: Condition 1, "In0==0" F	55
3	Condition	System.st_Main.td0."[(In0==0)&&(In1==1)]/"	Transition: Condition 2, "In1==1" T	57
5	Decision	System.st_Main.td0."[(In0==0)&&(In1==1)]/"	Transition: Transition trigger expression F	55
6	Decision	System.st_Main.td0."[(In0==0)&&(In1==1)]/"	Transition: Transition trigger expression T	57
7	Mcdc	System.st_Main.td0."[(In0==0)&&(In1==1)]/"	Transition: MCDC Transition trigger expression with Condition 1, "In0==0" T	57
8	Mcdc	System.st_Main.td0."[(In0==0)&&(In1==1)]/"	Transition: MCDC Transition trigger expression with Condition 2, "In1==1" T	57

(b) 테스트 목표가 만족된 상태 전이표

Objectives Proven Unsatisfiable

Simulink Design Verifier proved that there does not exist any test case exercising these test objectives. This often indicates the presence of dead-code in the model. Other possible reasons can be inactive blocks in the model due to parameter configuration or test constraints such as given using Test Condition blocks. In rare cases, the approximations performed by Simulink Design Verifier can make objectives impossible to achieve

#	Type	Model Item	Description	Test Case
4	Condition	System.st_Main.td0."[(In0==0)&&(In1==1)]/"	Transition: Condition 2, "In1==1" F	n/a
10	Mcdc	System.st_Main.td0."[(In0==0)&&(In1==1)]/"	Transition: MCDC Transition trigger expression with Condition 2, "In1==1" F	n/a
16	Condition	System.st_Main.td0."[(In0==1)&&(In1==1)]/"	Transition: Condition 2, "In1==1" F	n/a
22	Mcdc	System.st_Main.td0."[(In0==1)&&(In1==1)]/"	Transition: MCDC Transition trigger expression with Condition 2, "In1==1" F	n/a
44	Condition	System.st_Main.td0."[(In0==0)&&(In1==0)]/"	Transition: Condition 2, "In1==0" F	n/a
50	Mcdc	System.st_Main.td0."[(In0==0)&&(In1==0)]/"	Transition: MCDC Transition trigger expression with Condition 2, "In1==0" F	n/a
56	Condition	System.st_Main.td0."[(In0==1)&&(In1==0)]/"	Transition: Condition 2, "In1==0" F	n/a
			Transition: MCDC	

(c) 테스트 목표를 만족하지 않은 상태 전이표

Test Case 1

Summary

Length: 0 Seconds (1 sample periods)
Objective Count: 7

Objectives

Step	Time	Model Item	Objectives
1	0	System.st_Main.td2."[(In0==1)&&(In1==1)]/"	Transition: Condition 1, "In0==1" F
		System.st_Main.td2."[(In0==1)&&(In1==1)]/"	Transition: Transition trigger expression F
		System.st_Main.td2."[(In0==1)&&(In1==1)]/"	Transition: MCDC Transition trigger expression with Condition 1, "In0==1" F
		System.st_Main.td3."[(In0==1)&&(In1==1)&&(In2==...)"	Transition: Condition 1, "In0==1" F
		System.st_Main.td3."[(In0==1)&&(In1==1)&&(In2==...)"	Transition: MCDC Transition trigger expression with Condition 1, "In0==1" F
		System.st_Main.td1."[(In0==1)&&(In1==0)&&(In2==...)"	Transition: MCDC Transition trigger expression with Condition 1, "In0==1" F
		System.st_Main.td1."[(In0==1)&&(In1==0)&&(In2==...)"	Transition: Condition 1, "In0==1" F

(d) 자동으로 생성된 테스트 케이스의 예

(그림 11) <표 2>의 Set2를 SDV로 수행하여 얻은 테스트 케이스 생성 보고서

생성을 수행하여 얻은 결과물들을 정리한 통계는 <표 2>와 같다.

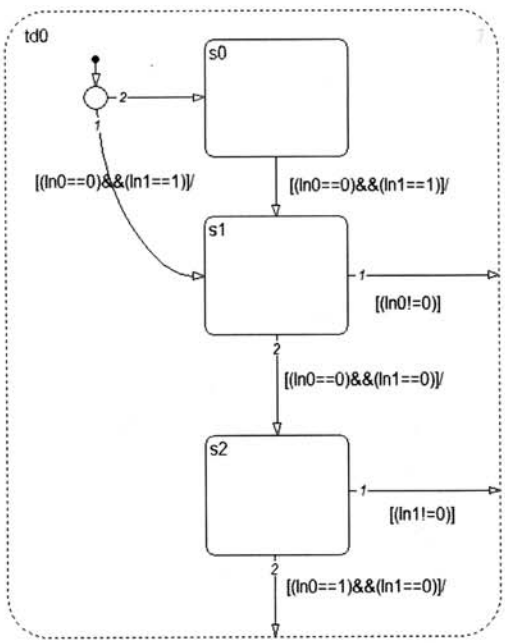
<표 2>에서 #State, #Transition, #Waveform, #Timing Constraint는 Timing diagram모델이 가지고 있는 정보로 각각 Stateflow모델의 상태의 개수, Stateflow모델의 전이의 개수, 원 Timing diagram 모델의 Waveform의 개수, 원 Timing diagram모델의 시간 제약의 개수를 의미한다. 그리고 # Objective, #Sat, #Unsat, #Undecided는 SDV에서 변환된 Stateflow모델로부터 생성한 데이터로 각각 MC/DC관점에서 생성해야 하는 테스트 케이스의 목표 개수, SDV에서 MC/DC관점에서 생성해야 하는 목표를 만족한 테스트 케이스의 개수, MC/DC관점의 목표를 만족하지 못한 테스트 케이스의 개수, SDV가 테스트 케이스를 생성할 수 있는지의 여부를 결정하지 못한 개수를 나타낸다. SDV가 테스트 케이스의 생성 여부를 결정하지 못하게 되는 이유는 모델이 복잡하거나 SDV가 목표를 달성하기에 충분치 않은 시간을 수행할 경우 발생하게 되는데, 실험 결과 충분히 많은 시간을 주어도 일부 모델에서 테스트 케이스를 생성할 수 있는

지의 여부를 판단 못한 것들이 존재했다.

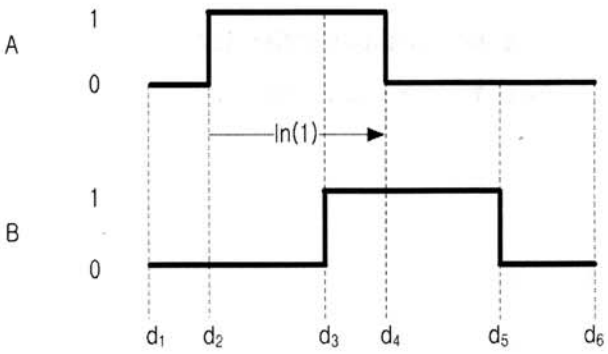
그리고 Analysis Time, Coverage Ratio는 각각 SDV가 변환된 Stateflow모델 시스템을 분석한 시간, SDV에서 생성한 테스트 케이스가 만족시킨 테스트 케이스 목표 대비 총 테스트 케이스 목표 개수를 의미한다. Coverage Ratio가 높으면 SDV가 MC/DC관점의 테스트 케이스 목표를 잘 생성함을 의미하고, Coverage Ratio가 낮으면 SDV가 테스트 케이스를 잘 생성하지 못함을 의미한다.

<표 2>의 결과를 보면 대체로 테스트 케이스의 목표 대비 만족한 개수의 비율이 80% 이상으로 본 논문의 접근 방법이 대체로 유효함을 확인할 수 있었다. 하지만 일부 모델의 경우 낮은 수치를 기록했는데, 이를 분석하여 그 원인을 발견했다. 1) Stateflow의 우선순위가 낮은 상태 전이의 일부 테스트 케이스 목표는 우선 순위가 높은 상태 전이에 의해 생성이 불가능한 경우가 존재한다. 2) 생성된 Timing diagram중 임의의 시점이 도달 불가능한 모델이 존재하였으며, 이 모델로부터 변환된 Stateflow의 임의의 상태가 도달 불가능하다.

첫 번째 원인에 대해 (그림 12)와 같은 stateflow로 변환된 모델을 예로 들어 설명하도록 하겠다. 이 모델에서 s1으로부터 발생하는 전이에는 (s1, "[(In0==0) && (In1 ==0)]", s2)과 (s1, "", td0)가 있는데, (s1, "", td0)의 우선순위가 1이고 (s1, "[(In0==0) && (In1 ==0)]", s2)는 우선순위가 2이다. Stateflow는 같은 상태에서 시작하는 두 개 이상의 전이가 있을 때, 우선순위가 낮은 것이 먼저 수행된다. s1에서 s2로의 전이 조건에서 첫 번째 우선순위를 가진 전이의 테스트 목표는 (In0!=0)이 참이거나 거짓으로 두 개이다. 그리고 두 번째 우선순위를 가진 전이의 테스트 목표는 ((In0 ==0), (In1==0))이 (거짓, 거짓), (참, 거짓), (거짓, 참)의 세 개이다. 여기서 두 번째 우선순위의 전이의 (In0==0)가 거짓이 되는 테스트 입력은 만들어 낼 수 없다. 그 이유는 (In0==0)이 거짓이 되면 (In0!=0)이 참이 되어 첫 번째 우선순위가 먼저 수행이 되어 버리기 때문에 두 번째 우선순위를 가진 전이의 조건에서 (In0==0)가 거짓이 되도록 생성하는 것이 불가능하다.



(그림 12) Timing diagram으로부터 생성된 Stateflow 모델의 예



(그림 13) 시점 d_4로의 이동이 불가능한 Timing diagram

커버리지 비율이 낮은 두 번째 원인은 다음과 같다. 모든 Timing diagram 모델은 시점을 이동하기 위해서는 1 tick을 필요로 한다. (그림 13)과 같은 Timing diagram에서 시간 제약 (d_2, d_4, In, 1)에 의해 1 tick이내에 시점 d_2에서 d_4까지 도달해야 하지만 이는 불가능하다. <표 2>의 Set3의 두 번째 모델의 커버리지 비율이 30.5%인 이유가 바로 모든 시점에 도달 가능하지 못한 것이 그 원인이었다.

6. 결론 및 향후 연구

본 논문에서는 Timing diagram으로부터 테스트 케이스를 생성하기 위해 1) Timing diagram을 수학적으로 정의하여 문법과 의미를 정의하였고, 2) Timing diagram의 의미로부터 임의의 Timing diagram모델에 대해 그와 동일한 행동을 할 수 있는 Stateflow모델로 변환하는 방법에 대해 정의하였으며, 3) Stateflow모델로부터 테스트 케이스를 생성하는 도구인 SDV를 사용하여 두 번째 단계에서 변환된 Stateflow모델로부터 테스트 케이스를 생성하는 전략에 대해서 설명하였다. 위 접근 방법이 타당한지를 확인하기 위해서 Timing diagram모델을 Stateflow로 변환하여 테스트 케이스 목표 및 테스트 입력 시퀀스를 생성하고 그 결과를 분석하였다. 이 연구를 통해 얻을 수 있는 효과는 다음과 같다. 과거에는 임베디드 시스템의 사양을 표현할 때 Timing diagram을 이용하여 표현하게 되면 비록 시스템의 행동을 직관적으로 표현할 수 있는 장점은 있었지만 Timing diagram 모델을 바탕으로 테스트 케이스를 생성하고자 할 때에는 테스트 케이스를 수동으로 생성해야만 했다. 하지만, 본 연구를 통해 앞으로는 Timing diagram으로 작성된 시스템의 사양도 자동으로 테스트 케이스를 생성할 수 있게 될 것이다. 향후 연구로 Timing diagram의 테스트 커버리지 범주

나 Timing diagram으로부터 테스트 입력 시퀀스를 생성하는 연구를 수행할 계획이다.

참 고 문 헌

[1] Kirill Bogdanov, "Automated testing of Harel's statecharts," PhD thesis, The University of Sheffield, Jan., 2000.

[2] Jee-Eun Yoo, "Using Model Checking to Generate Data-Flow Oriented Test Case from Statecharts," Master thesis, KAIST, 2002.

[3] John F. Wakerly, "Digital Design-Principles and Practices," Prentice Hall, 4th ed, pp.682-686, 2005.

[4] E. K. Ogoubi, Eduard Cerny "Synthesis of checker EFSMs from Timing diagram specifications", ISCAS (1), pp.13-18, 1999.

[5] Hyoungseok Hong, "Verification and Testing Methods for Statecharts," PhD thesis, KAIST, 2000.

[6] Jungsup Oh, "Automatic Generation of Test Cases based on Requirement Models," PhD thesis, AJOU University, 2009

[7] Moeschler, P.; Amann, H.P.; Pellandini, F., "High-level modeling using extended Timing diagrams - A formalism for the behavioral specification of digital hardware," Design Automation Conference, EURO-VHDL '93. Proceedings EURO-DAC '93. European, Vol., No., pp.494-499, 20-24 Sep., 1993.

[8] Stefan Lenk, "Extended Timing diagrams as a specification language," European Design Automation Conference, Proceedings of the conference on European design automation, pp.28-33, 1994.

[9] Nina Amla, "Model Checking Synchronous Timing Diagrams," LNCS Proceedings of the Third International Conference on Formal Methods in Computer-Aided Design, Vol.1954, pp.283-298, 2000.

[10] Davis, M., Logemann, G., and Loveland, D. "A machine program for theorem-proving," Commun. ACM Vol.5, issue 7, pp.394-397, 1962.

[11] R.E. Bryant, "Graph-Based Algorithms for Boolean Function Manipulation," IEEE Transactions on Computers, pp.677-691, August, 1986.

[12] The MathWorks Inc, "Simulink design verifier™ 1 User's Guide", Available from URL http://www.mathworks.com/access/helpdesk/help/pdf doc/sldv/sldv ug.pdf, 2008.

[13] Andersson, G., Bjesse, P., Cook, B., Hanna, Z., "A proof engine approach to solving combinational design automation problems," In: Proc. 39th Design Automation Conference (DAC'02), IEEE Computer Society Press, pp.725-730, 2002.

[14] C. Chen, T. Lin, and H. Yen, "Modelling and Analysis of Asynchronous Circuits and Timing Diagrams Using Parametric Timed Automata," in Proc. of the 23rd IASTED Int'l Conf. on Modelling, Identification and Control (MIC 2004), ACTA press, 2004.

[15] David Harel, "Statecharts: A visual formalism for complex systems," Science of Computer Programming, Vol.8, Issue 3, pp.231-274, 1987.

[16] S. Rayadurgam; M.P.E. Heimdahl, "Coverage based test-case generation using model checkers," Engineering of Computer Based Systems, 2001. ECBS 2001. Proceedings. Eighth Annual IEEE International Conferenceand Workshop on the , Vol., No., pp.83-91, 2001.

[17] Grass, W.; Grobe, C.; Lenk, S.; Tiedemann, W.-D.; Kloos, C.D.; Marin, A.; Robles, T., "Transformation of timing diagram specifications into VHDL code," Design Automation Conference, 1995. Proceedings of the ASP-DAC '95/CHDL '95/VLSI '95., IFIP International Conference on Hardware Description Languages; IFIP International Conference on Very Large Scale Integration., Asian and South Pacific, Vol., No., pp.659-668, 1995.

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

이 홍 석

e-mail : hsyi98@naver.com

2003년 아주대학교 전자공학부(공학사)

2003년 아주대학교 정보 및 컴퓨터 공학부(공학사)

2005년 아주대학교 전자공학과(공학석사)

2005년~현 재 아주대학교 전자공학과 박사과정

관심분야 : 명세 기술, 정형 검증, 소프트웨어 공학, 임베디드 시스템

정 기 현

e-mail : khchung@ajou.ac.kr

1984년 서강대학교 전자공학과(학사)

1988년 미국 Ilinois주립대 EECS(석사)

1990년 미국 Purdue대학 전기전자공학부(박사)

1991년~1992년 현대반도체 연구소

1993년~현 재 아주대학교 전자공학부 교수

관심분야 : 컴퓨터구조, VLSI 설계, 멀티미디어 및 실시간 시스템 등

최 경 희

e-mail : khchoi@madang.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