$Stateflow_{TD}$ - Stateflow와 Timing diagram의 통합 모델링 방법론

정회원 이 홍 석*, 정 기 현*, 최 경 희**

$Stateflow_{TD}$ - A unified modeling Formalism of Stateflow and Timing Diagram

Hongseok Lee*, Kihyun Chung*, Kyunghee Choi** *Regular Members*

요 약

Stateflow는 임베디드 시스템을 모델링 하는 유용한 도구로 산업 현장에서 많이 사용되고 있다. 하지만 Stateflow는 연속적인 동작이나 시간에 따른 시스템의 행동을 표현하기에 불편한 점이 있다. 한편 Timing diagram은 전자공학의 하드웨어 분야에서 시스템의 동작을 표현하기 위한 도구로 널리 사용되고 있는 도구이다. 이 도구는 연속적인 동작이나 시간에 따른 시스템의 행동을 표현하기에 편리한 장점이 있다. 그러므로 본 논문에서는 시스템의 행동을 Stateflow로 모델링 할 때 기존의 불편했던 점들을 개선하기 위해서 Stateflow 도메인에서 Stateflow 표현방식과 Timing diagram 표현방식을 통합하여 모델링 하는 방법론인 $Stateflow_{TD}$를 제안하고자 한다. 본 논문은 Stateflow 모델과 Timing diagram 모델들을 통합하는 방법과 $Stateflow_{TD}$이 가지는 장점에 대해서 설명하였을 뿐만 아니라, 두 가지 예제 시스템을 사용하여 $Stateflow_{TD}$이 유용함을 보였다.

Key Words : integrated modeling methodology, Stateflow, Timing diagram, Specification, Embedded system

ABSTRACT

Stateflow is a useful system modeling tool, which is frequently used in industry. But Stateflow has defects when users are trying to describe consecutive behaviors or a system's temporal behaviors. Timing diagram, on the other hand, is a widely used tool of explaining behaviors of a hardware system in electronics. This tool has a merit when specifying consecutive behaviors and temporal behaviors of a system. Thus, this paper suggests $Stateflow_{TD}$, the unified modeling methodology in Stateflow domain integrating Stateflow with Timing diagram in order to improve the shortcomings of Stateflow. This paper not only explains a technique of integrating both a Stateflow model and Timing diagram models, and describes advantages of what $Stateflow_{TD}$ formalism has, but also shows its convenience using two example systems.

Ⅰ. 서 론

Stateflow는 시스템을 모델링하고 시뮬레이션 하는 도구로 항공, 자동차, 통신, 전자공학 등 여러 분야에서 사용된다. 비록 Stateflow가 강력한 도구이지만, 시간과 관련된 시스템의 행동을 기술할 때 불편한 점이 있다. Stateflow에서는 시간 제약과 상태 전이를 이용하여 시스템의 시간에 따른 행동을 표현할 수 있지만 표현된 그림은 직관적이지 않으며 쉽게 이해할 수 없다. 반면에 Timing diagram은 시간에 따른 시그널의 변화의 추이를 나타내는 도구로 전자공학 분야에서 시스템의 입력과 출력의 행동을 기술하는 도구로 많

* 아주대학교 전자공학부 (hsyi98@naver.com khchung@ajou.ac.kr), ** 아주대학교 정보 및 컴퓨터 공학부 (khchoi@ajou.ac.kr)
논문번호 : KICS2010-05-225, 접수일자: 2010년 5월 25일, 최종논문접수일자 : 2010년 11월 26일

이 사용되어 왔다. Timing diagram의 직관적인 표현과 익히기 쉬운 점 때문에 엔지니어들이 많이 사용한다. 특히 Timing diagram은 칩의 데이터 시트에 시스템의 동작을 표현하고자 할 때 많이 사용된다.

시스템의 연속적인 동작을 표현하는 부분에 대해서 Stateflow와 Timing diagram을 비교했을 때 Stateflow는 직관적이지 않으며 Stateflow로 그리는 것도 쉽지 않은 반면 Timing diagram은 시간과 관련된 시스템의 행동을 기술하는데 편리하며, 시스템의 연속적인 동작을 표현하는 것도 쉽고, Timing diagram으로 표현된 그림도 매우 직관적이다. Timing diagram의 단점은 시스템의 모든 영역을 Timing diagram으로만 그리는 것은 매우 어려우며 Timing diagram으로만 그리도록 하기 위해서는 전통적으로 많이 사용하는 Timing diagram을 보완해야 할 필요가 있다. Timing diagram의 이와 같은 문제점을 해결하기 위해 Timing diagram을 확장한 연구[12]가 있었지만, Timing diagram이 가진 편리성과 직관적인 특성을 잃어버리게 되었다.

본 연구에서는 Stateflow를 이용하여 모델링 할 때의 불편한 점들을 해결하기 위해서 임베디드 시스템의 사양서 작성을 위한 모델로 Stateflow와 Timing diagram을 Stateflow도메인에서 통합하여 그리는 방법론을 제시하고자 한다. 이 방법론의 목적은 비록 Stateflow가 강력한 도구이기는 하지만, Stateflow는 위에서 언급한 두 가지에 대해 취약점이 있으며, 이를 Timing diagram이 보완해 줄 수 있기 때문에 사양서를 기술하거나 모델을 표현하고자 할 때 훨씬 유용한 방법이 되기 때문이다. Timing diagram을 이용한 여러 연구들[4, 9-15, 18, 19, 24]중에서 일부의 연구[18,19]를 제외한 모든 연구들에서 정의한 Timing diagram의 의미들은 Stateflow와 Timing diagram과의 통합 모델링에 부합하는 정의나 표현은 존재하지 않으며, 주로 시스템을 검증하기 위한 목적으로 Timing diagram에 대한 연구가 이루어졌기 때문에 이홍석[18, 19]에서 정의한 Timing diagram을 본 연구에서 사용하였다.

본 논문의 구성은 다음과 같다. 2장은 통합된 모델링 표현 방식에 대해서 설명하였는데, Stateflow와 Timing diagram에 대해 간략하게 소개하였으며 3장에서는 본 논문에서 제시하는 통합 방법론인 Stateflow_{TD}에 대해 소개하였다. 그리고 Stateflow_{TD}에서 Stateflow와 Timing diagram의 통합 절차와 Stateflow_{TD}가 가지는 장점들에 대해서도 설명하였으며 또한 통합 방법론에서 부가적으로 필요한 일에 대해서도 설명 하였다. 4장에서는 두 가지 예제 시스템으로부터 통합 모델링 방법론으로 모델링 하는 사례를 통해 Stateflow_{TD}로 모델링 하는 것이 직관적이고 유용함을 보였다. 5장에서는 통합 표현방법론 및 Timing diagram과 관련된 연구에 대해서 기술하였으며 마지막으로 6장에서 결론으로 끝을 맺는다.

Ⅱ. 배 경

이 장에서는 본 논문에서 통합 모델링 방법론에 사용되는 모델링 언어인 Timing diagram과 Stateflow에 대해서 간략하게 소개하도록 하겠다.

2.1 Timing diagram

Timing diagram은 시간에 따른 시스템의 행동을 표현할 때 유용한 도구로 오래 전부터 전자공학에서 디지털 회로의 행동을 표현하기 위해 많이 사용되었다[23]. Timing diagram은 여러 변이가 있지만, 본 논문에서는 이홍석[18, 19]의 Timing diagram에 대한 형식적인 정의를 소개하도록 하겠다. Timing diagram TD는 7개의 요소인 $<X, Y, \Sigma, D, C, \rho, \delta>$로 구성된다. 여기서 X는 입력 waveform의 집합이고, Y는 출력 waveform의 집합, Σ는 waveform의 타입의 집합, D는 입력 waveform의 값이 변화되는 시점에 대한 집합, C는 시간 제약의 집합, ρ: $(X \cup Y) \rightarrow \Sigma$는 waveform들의 타입을 정의하는 함수의 집합, δ: $(X \cup Y) \times D \rightarrow VALUE$는 waveform의 시점에 대한 값을 정의하는 함수의 집합을 의미하며, VALUE는 값의 집합을 의미한다. Σ는 두 개의 원소를 가지며, $\Sigma = \{Signal, Event\}$이다. 시간 제약은 4개의 요소인 $<d_s,d_e,t,v>$로 구성되며 $d_s \in D$는 시간 제약의 시작 시점을 의미하고, $d_e \in D$는 시간 제약의 끝 시점을 의미하고, t는 시간 제약의 타입이고, v는 값을 의미한다. 시간 제약의 타입에는 세 종류가 있는데, 'Duration', 'In', 'After'가 존재한다.

2.2 Stateflow

Stateflow[21]는 Harel[6]의 Statecharts의 변종으로 Mathworks[20]에서 개발한 언어이다. Stateflow는 상태들이 전이로 연결되어 있는 그래프의 형태를 띤다. 또한 확장된 버전의 유한 상태 기계(EFSM)[22]로 상태 전이의 레이블에 전이 조건 및 행동을 기술할 수 있기 때문에 유한 상태 기계(FSM)보다 더 높은 추상성을 가지며 기술하기 쉽다. 또한 Stateflow는 계층성 및 병렬성을 제공하기 때문에 시스템을 표현하고자 할 때 복잡도를 낮춰서 기술할 수 있는 장점을 가진다.

Stateflow SF는 7개의 요소인 <S, s_0,T,E,C,A>로 이루어져 있다. 여기서 S는 상태의 집합, s_0는 초기상태, $T \subseteq S \times E \times C \times 2^A \times S$는 상태 전이의 집합으로 2^A는 A의 멱집합을 의미한다. 그리고 C는 조건의 집합이고, A는 행동의 집합이다.

Ⅲ. 통합된 모델링 방법론 - $Stateflow_{TD}$

이 장에서는 본 논문에서 제시하고자 하는 방법론인 $Stateflow_{TD}$에 대해서 소개하고, $Stateflow_{TD}$의 타당성, $Stateflow_{TD}$를 적용하기 위한 과정, 그리고 $Stateflow_{TD}$가 가지는 장점과 $Stateflow_{TD}$방법론에서 부가적으로 필요한 일에 대해서 설명하도록 한다.

3.1 $Stateflow_{TD}$의 체계

본 논문에서 제시하고자 하는 통합된 모델링의 표현 방법론인 $Stateflow_{TD}$는 Stateflow와 Timing diagram을 통합하여 Stateflow 도메인에서 통합되어 표현하는 방법론이다. Stateflow 도메인에서 통합을 한 이유는 각 표현방법론의 표현력에서 차이가 있기 때문이다. 다시 말하면 Timing diagram은 Stateflow가 기술할 수 있는 모든 행위를 기술할 수가 없지만, Stateflow는 Timing diagram이 기술할 수 있는 모든 행위를 기술할 수 있다. 그렇다면 왜 Timing diagram을 굳이 사용해야 하는지에 대해서 의문이 들 수 있다. 그것은 서론에서도 언급했듯이 Stateflow로 표현했을 때보다 Timing diagram으로 시스템의 행동을 표현할 때 직관적이고 쉬운 경우가 존재하기 때문이다.

$Stateflow_{TD}$방법론의 타당성을 보이기 위해서는 다음과 같은 조건을 만족해야 한다. 1) 두 표현 방법론이 행위에 대해 의미론적으로 서로 상충되지 않아야 한다. 2) 두 표현 방법론이 통합되기 위한 자세한 과정을 설명할 수 있어야 한다. 첫 번째 조건에 대해서 설명하면, Timing diagram은 Stateflow와 문법과 표현 방법은 다르지만, 행동상으로 동일한 의미를 가질 수 있다. 즉 이 말의 의미는 임의의 어떤 Timing diagram 모델에 대해서 이와 동치 관계를 갖는 Stateflow 모델이 존재한다[18,19]. 그러므로 첫 번째 조건은 만족한다. 두 번째 조건도 역시 만족하는데, 이에 대해서는 다음 절에서 설명하도록 하겠다.

3.2 Timing diagram 모델의 Stateflow 모델로의 통합 절차

Timing diagram은 통신 매커니즘, 계층성, 병렬성이 존재하지 않지만, Stateflow로 변환이 가능하기 때문에 Stateflow의 통신 매커니즘, 계층성, 병렬성을 이용할 수 있다. 이를 설명하기 위해서 Timing diagram 모델의 Stateflow 모델로의 변환에 대해서 간략하게 설명하도록 하겠다. 하나의 Timing diagram 모델 T를 Stateflow로 변환하면 하나의 상태 s_T로 변환이 되며 s_T의 하위에 Timing diagram의 행동이 표현된다. 변환된 상태 s_T는 다른 Stateflow 모델 M에 병합이 가능한데, 이를 위해서 s_T는 M에 속한 상태 s의 하위 상태로 병합 과정이 필요하다.

s_T가 s에 병합되기 위해서 s의 하위 상태는 존재하지 않아야 하며 이 조건을 만족시키면 s_T는 s의 하위 상태로 병합 될 수 있다. T와 M이 통합된 모델을 M´라고 할 때, M´에서의 Timing diagram 모델 T의 행위는 M에 속한 상태 s에 의존적이 된다. 즉 s가 활성화가 되면 Timing diagram 모델 T도 활성화 되며, s가 비활성화가 되면 T도 비활성화 된다. 그리고 두 개 이상의 Timing diagram 모델들이 같은 상태에 병합되며 병렬성을 표현하는 것도 가능하다. 즉 Timing diagram 모델 T, T´에 대해 Stateflow로 변환된 모델을 각각 s_T,s_T´라고 가정하고 이 들을 Stateflow 모델 M에 속한 상태 s의 하위상태로 병합한다고 가정하자. 이 경우 s_T,s_T´는 s의 하위 상태로 병합이 가능한데, 병렬의 의미를 가지는 형태로 병합이 된다. 물론 이 경우에도 모델 M의 상태 s는 하위 상태를 가지지 않아야 한다. 두 개의 Timing diagram모델과 하나의 Stateflow 모델이 통합된 모델에서 상태 s가 활성화가 되면 s_T와 s_T´는 동시에 활성화 된다. 하지만 두 개 이상의 Timing diagram 모델을 동일한 상태 s에 병합하기 위해서 Timing diagram 모델 간의 우선순위를 정의해 주어야 한다. 이것은 Stateflow에서 병렬성을 표현하고자 할 때 상태간의 우선순위를 정의하기 때문에 필요한 개념[21]으로 이는 Stateflow가 순차적 기법[26]으로 병렬성을 구현했기 때문이다.

그리고 두 개 이상의 Timing diagram 모델을 동일한 상태 s에 병합하기 위해서는 다음과 같은 사항을 확인해야 한다. 그것은 임의의 서로 다른 Timing diagram 모델 T, T´에 대해 각 모델에서 정의한 출력들이 서로 겹치면 안 되는 것이다. 이를 수학적으로 표현하면 다음과 같다. Stateflow 모델 M = <S, s_0,T,E,C,A>에서 임의의 상태 $s \in S$에 대해 leaf(s)를 s가 하위 상태를 포함하고 있는지의 여부를 나타내는 함수로 정의하자. 즉 leaf(s)가 참이면 s는 하위 상태를 포함하고 있지 않으며, leaf(s)가 거짓이면 s는 하위 상태를 포함한다고 정의한다. 그리고 임의의 Timing diagram 모델 T = <X, Y, Σ, D, C, ρ, δ>, T´ = <X´,

Y′, Σ′, D′, C′, ρ′, δ′>에 대해 모델 T와 T′이 s에 병렬로 병합되기 위한 조건은 다음의 두 가지 조건을 모두 만족해야 한다.

(1) leaf(s) = True,

(2) Y ∩ Y′ = Ø

이와 같은 조건이 필요한 이유는 자원 경쟁[31]이 발생하는 것을 체크하기 위함인데, Stateflow에서는 병렬로 동작하는 상태들간의 우선순위가 존재하여 항상 순차적으로 동작하기 때문에 우선순위가 높은 상태의 행동을 우선순위가 낮은 상태가 변경시켜버릴 수 있는 문제가 발생하기 때문이다. 그러므로 이와 같은 상황은 모델링이 적절하게 이루어진 것이 아니기 때문에 사양서 혹은 모델의 문제가 되며, Stateflow_{TD}방법론으로 모델링을 하게 될 경우 체크가 가능한데, 이는 Stateflow_{TD}방법론의 장점 중 하나이다.

3.3 Stateflow_{TD}방법론의 유용성

Stateflow방법론의 장점은 계층성, 병렬성을 제공하며 상태전이에 조건과 행동을 기술할 수 있으며 변수를 정의하여 사용할 수 있기 때문에 복잡한 시스템의 기능을 추상화 시켜서 단순하고 직관적으로 시스템을 모델링 할 수 있다. 하지만 전기 신호의 행동과 같은 연속적인 행동 또는 시간 제약이 많이 포함된 시스템의 행동을 표현하고자 할 때에는 Stateflow방법론으로 표현하고자 할 때 오히려 직관성이 떨어지며, 그런 행동들은 추상화 시키는 것이 매우 어렵다. 반면에 Timing diagram 방법론의 장점은 시간에 따른 시스템의 행동을 직관적으로 표현하는 것에 유용하며 특히 연속적인 시스템의 행동을 표현하고자 할 때에 다른 어느 표현 방법론보다 쉽고 편리하며 매우 직관적이다. 그리고 Timing diagram으로 표현된 모델을 기반으로 테스트를 수행하고자 할 때 자동으로 완벽하게 테스트 스크립트를 생성하는 알고리즘이 존재한다[19]. 이는 Stateflow 방법론과 비교했을 때 큰 장점이다. Stateflow도메인에서는 모델로부터 테스트 케이스를 자동으로 생성하는 알고리즘[27-30]이 여러 가지가 있으나 아직 완벽한 것은 없기 때문이다. 하지만 Timing diagram 방법론의 단점은 여러 가지 종류의 복잡한 기능을 수행하는 행동을 표현하는 데는 Timing diagram으로 표현하는 것이 어려우며 계층성, 병렬성을 지원하지 않기 때문에 Timing diagram만으로 시스템의 모든 행동을 표현하는 것은 매우 어렵다.

Stateflow_{TD}방법론으로 시스템을 모델링 하면 다음과 같은 장점이 있다. 첫째, 3.2절에서 언급했듯이 모델의 의미상으로 문제가 있는 부분을 정적 분석을 통한 체크[25]를 통해 확인이 가능하다. 둘째, Stateflow로만 기술했을 때 발생할 수 있는 문제점들을 Timing diagram방법론을 이용하여 기술함으로써 모델을 간단하고 더 직관적으로 기술할 수 있는 장점이 있다. 셋째, Timing diagram으로 기술된 모델들의 경우 100% 테스트 케이스 및 테스트 스크립트 생성이 가능하기 때문에 모델 기반의 테스트를 수행하고자 할 경우에도 테스트 케이스를 생성하고자 할 때 효율을 높일 수 있다. 넷째, Stateflow_{TD}방법론으로 표현한 모델은 Stateflow 모델로 변환이 가능하기 때문에 Mathworks[20]에서 제공한 도구를 이용하여 모델의 실행, 검증, 시뮬레이션 및 분석이 가능하다.

3.4 Stateflow_{TD}방법론에서 필요한 부가적인 일

Stateflow_{TD}가 위 절에서 언급한 장점만 있는 것은 아니다. 시스템을 Stateflow_{TD}로 표현하기 위해서는 시스템을 분해해서 각각의 모듈 혹은 서브 시스템에 대해 어떤 방법론으로 표현해야 하는지를 결정해야 하는 작업이 필요하다. 이는 분명 Stateflow혹은 Timing diagram방법론을 단독으로 표현하고자 할 때에는 필요하지 않은 작업일 수 있다. 하지만 3.2절에서 Stateflow및 Timing diagram방법론의 장점과 단점을 언급했듯이 임의의 모듈이 가져야 할 기능이 명확히 정의된 상태에서 어떤 표현방법론으로 모델링을 해야 하는지 결정하는 것은 어려운 일은 아니다.

Ⅳ. 예 제

이 장에서는 예제 시스템을 이용하여 Timing diagram과 Stateflow표현 방식의 특징과 예제 시스템을 이용하여 모델링을 하는 방법에 대해서 설명하고자 한다. 첫 번째 예는 Door Lock 시스템으로 이 예제의 목적은 Door Lock 시스템을 Stateflow으로만 모델링 한 경우와 통합 방법론으로 모델링 한 경우의 예를 제시하며 이로부터 Stateflow_{TD}방법론으로 표현하는 것이 더 직관적이고 편리함을 보였다. 두 번째 예는 Actuator의 위치 조절 시스템으로 이를 Stateflow와 Timing diagram로 표현하여 이 둘을 비교하여 Timing diagram이 기술하기 편리하고 모델이 직관적임을 나타냈다. 비록 Timing diagram이 Stateflow보다 표현력이 떨어지지만, 일부 영역에서 Stateflow로 표현하는 것보다 Timing diagram으로 표현하는 것이 더 직관적임을 보이고자 했으며, 이를 통해 Stateflow_{TD}가 유용한 방법론임을 간접적으로 입증했다.

4.1 Door Lock System

도어락 시스템의 입력은 [Start]버튼, [End]버튼, [Lock]버튼, 그리고 [0]부터 [9]까지 버튼이 있다고 가정하자. 출력은 출입문의 잠금 또는 풀림이라고 가정하자. 만약 도어가 잠겨있을 때 도어를 열기 위한 시퀀스가 [Start]-[4]-[2]-[1]-[9]-[5]-[3]-[End]라고 가정했을 때, 이 사양을 Stateflow로 기술한 모델과 Stateflow$_{TD}$로 기술한 모델이 각각 그림 1, 그림 2와 같다. 그림 1은 그림 2와 비교했을 때, 훨씬 더 복잡함을 알 수 있다. 그 이유는 그림 1의 Stateflow에서는 매 상태에서 모든 입력에 대한 처리를 상태 전이로 표현해야 하기 때문에 사양서 에서 직접적으로 표현하지 않은 입력에 대한 처리를 했기 때문이다. 반면 그림 2에서의 Timing diagram모델은 매 상태에서 원하지 않는 입력에 대한 처리를 직접 기술해주지 않아도 되기 때문에 그림이 더 간단하고 사람이 인식하기 쉬운 장점을 갖는다.

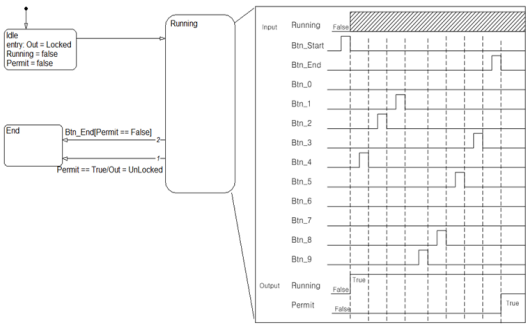

그림 1. Door Lock system에 대한 Stateflow 모델

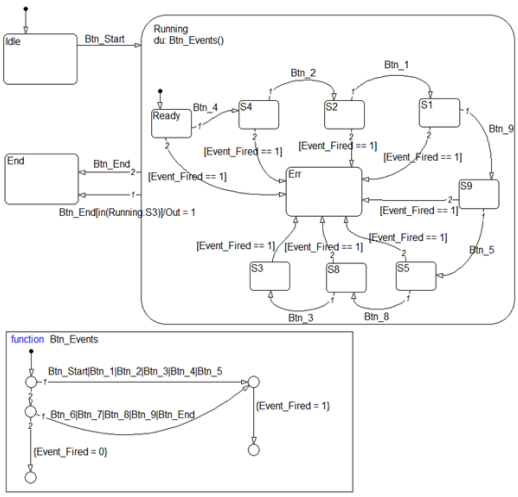

그림 2. Door Lock system에 대한 통합 모델

4.2 Actuator 조절 제어 시스템

이 시스템은 actuator의 위치를 제어하는 시스템인데, actuator의 방향이 시계 방향(CW)인 경우에 한해서만 기술을 했다. 입력은 Mode_Direction과 Right_Position이다.

Mode_Direction이 가질 수 있는 값은 CW, CCW, Stop이 있는데, 이 값이 의미하는 것은 CW는 Mode actuator가 시계 방향으로 움직여야 한다는 것을 의미하는 것이고, CCW는 반시계 방향으로 움직여야 한다는 것을 의미한다. Stop은 동작을 멈추는 것을 의미한다. Right_Position의 의미는 Mode actuator의 목표 위치와 실제 위치를 비교해서 오차범위 이내에 도달하면 Yes가 되며, 그렇지 않으면 No가 된다. 시스템의 사양은 다음과 같다.

Mode_Direction이 Stop에서 CW가 되면 20초 이내에 올바른 위치에 도달해야만 하며, 올바른 위치에 도달할 경우 Right_Position은 Yes가 된다. actuator의 방향이 CW가 된지 20초가 지나도 계속 actuator의 방향은 CW이고 Right_Position의 값이 No가 되면 Mode actuator를 3초 동안 휴식하고 2초 동안 CW방향으로 출력을 내보내는 행동을 2회하고 멈춘다.

위 시스템의 사양을 각각 Timing diagram과 Stateflow로 그리면 그림 3, 그림 4와 같다. 두 그림으로부터 Timing diagram으로 그린 그림이 훨씬 직관적이고 간단하다는 것을 알 수 있다. 시스템의 출력이 시퀀스로 되어 있는 경우 Stateflow로 표현하는 것은 쉽지 않은 반면 Timing diagram으로 기술하면 간단하게 표현이 가능함을 알 수 있다. 그림 4에서 빗금 친 박스의 의미는 입력이 어떤 값이 들어와도 상관하지 않음을 나타낸다[18,19]. Stateflow모델에 대해서 설명하면 Idle상태가 초기상태이고, 초기 상태에서 Model_ Direction이 CW가 되면 Running 상태로 천이한다. Running상태에 진입하면 타이머를 초기화하고 타이머가 동작되는데, Right_Position이 Yes가 되거나 혹은 Mode_Direction이 CW상태를 유지 하지 않으면 Idle상태로 전이하게 된다. Running상태의 하위 초기 상태는 Not_Stuck상태인데, 이 상태에서는 Mode_CW가 12, Mode_CCW가 0이 되게 출력을 내보낸다. 이 상태에서 20초가 지나면 First_Step상태로 천이하게

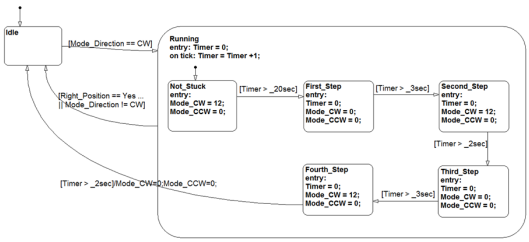

그림 3. Actuator의 위치 제어 시스템을 Stateflow로 표현한 모델

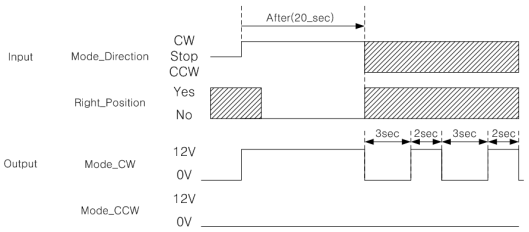

그림 4. Actuator의 위치 제어 시스템을 Timing diagram으로 표현한 모델

되는데, 여기서는 두 출력 변수의 값을 0으로 만들고 3초의 시간을 보낸다. 3초가 지나면 Second_Step상태로 천이하는데 이 상태에서는 Mode_CW가 12, Mode_CCW가 0이 되게 출력을 내보내며, 2초 동안 머무른다. 2초가 지나면 Thrid_Step상태로 천이하게 되며, 3초간 출력을 0으로 내보내고, 그 뒤에 Fourth_Step상태로 천이하는데, 이 상태에서는 다시 출력 값을 Mode_CW가 12, Mode_CCW가 0로 내보내며, 2초 후에 Idle상태로 천이한다.

V. 관련연구

본 연구에서와 같이 시스템의 사양을 표현하는 여러 표현 도구들을 통합한 연구는 Ptolemy[1, 3], VHDL/S[4,5], PSM[7-9]과 같은 연구들이 있다. Ptolemy는 1990년대에 Berkeley대학에서 개발된 통합된 계산 모델로, 유한 상태 기계 영역에서 다른 이질적인 특징을 갖는 동기화 데이터흐름(synchronous dataflow), 이산 이벤트 병렬(discrete event concurrency)모델을 통합한 모델이다. 또한 Ptolemy는 유한 상태 기계의 표현의 유용성을 제공하고 표현된 그림이 인식하기 쉽도록 하기 위해 계층성과 병렬성을 도입하였다. Ptolemy 는 주로 시스템을 디자인하고 프로토타이핑 및 시뮬레이션 하기 위한 목적으로 사용되는데, 멀티미디어 네트워크, 실시간 임베디드 시스템, 하드웨어 및 소프트웨어 합동 디자인, 혼합 모드의 하드웨어 시뮬레이션 등[3]의 다양한 분야에서 많이 사용되고 있으며, 현재는 Ptolemy II[2]에 대한 연구가 진행 중이다. VHDL/S는 VHDL과 Statecharts[6], symbolic timing diagram[4], 시제 로직을 VHDL영역에서 통합한 모델이다. VHDL/S가 위와 같은 모델들을 통합한 이유는 병렬 프로세스와 시퀀스 프로세스 사이의 상호작용에 대해 VHDL코드로 기술하는 것은 복잡하기 때문에 오류를 일으킬 가능성이 높은 반면, Statecharts는 그래픽 기반으로 계층성 및 병렬성을 지원하고 FSM을 확장하여 사용성을 높였기 때문에 VHDL의 약점을 보완할 수 있기 때문이다. 또한 Timing diagram은 VHDL이나 Statecharts와는 다르게 입력에 따른 출력의 행동의 시나리오에 대해서 기술하고자 할 때, 다른 도구들보다 직관적이고 이해하기 쉬운 장점을 가지고 있다. 또한 Timing diagram으로 기술된 사양은 검증을 위한 목적으로 시제 로직으로 변환하는 것도 가능하다는 장점도 가지고 있다. 이와 같은 장점을 바탕으로 VHDL/S는 시스템의 사양을 기술하는 것을 지원하는 것 외에도 symbolic model checking을 제공한다. 하지만, 현재까지 연구가 계속 진행 중이지는 않다.

PSM은 임베디드 시스템의 사양을 기술하기 위해 필요한 5가지 특징을 모두 지원하기 위해 개발된 모델로 5가지 속성은 1) 순차적이고 병렬적인 행동의 분해 2) 상태 전이 3) 예외 4) 순차적 알고리즘 5) 행동의 완성 등이 있다[7]. PSM의 특징은 Ptolemy나 VHDL/S와 비슷하게 여러 이질적인 표현 방법론을 하나로 통합한 점이다. PSM이 통합한 표현 방법론들은 유한 상태 기계(finite state machine, FSM), 계층적 병렬적인 유한 상태 기계(hierarchical concurrent finite state machine, HCFSM), 데이터 플로우 그래프(dataflow graph, DFG), 데이터 패스를 가지고 있는 유한 상태 기계(finite-state machines with datapath, FSMD), 프로그래밍 언어이다. PSM이 가진 유용성 때문에 SystemC를 이용하여 PSM을 모델링 하는 연구[8]도 있으며, VHDL과 유사하지만 추가 기능이 덧붙여진 SpecCharts라는 도구에 대한 연구[7]도 존재하는데, 이들은 모두 PSM을 지원하기 위한 연구들이다.

본 연구와 통합 계산 모델간의 비슷한 점은 본 연구는 Stateflow와 Timing diagram을 통합한 모델이라는 점에서 위 연구들과 비슷하다. 차이점은 통합된 표현 도구가 차이가 있다는 점과, PSM의 경우 시스템을 디자인하고 프로토타이핑(prototyping) 하는데 주 목적이 있고, VHDL/S는 시스템을 디자인하고 검증하는데 주 목적이 있다면, 본 연구는 시스템을 디자인하고, 모델기반의 테스트 케이스를 생성하는데 주 목적이

있다.

Timing diagram과 관련된 연구는 하드웨어 분야와 소프트웨어 분야로 나눌 수 있는데, 하드웨어 분야의 Timing diagram에 대한 연구는 대상 시스템을 동기(Synchronous) 시스템[10,11,15], 비동기(Asynchronous) 시스템[12,14] 각각에 대해 연구가 진행되었으며, 대부분 하드웨어의 검증을 하기 위한 용도로 Timing diagram을 이용하는 연구를 진행했다.

Lenk의 논문[12]에서는 비동기 통신 시스템의 사양을 기술하기 위한 Timing diagram을 제안하였는데, 그 이유는 시제 논리나 Process calculi로 기술하는 것은 매우 복잡하기 때문이다. 이를 위해 비형식적인 Timing diagram을 형식화하여 process calculus의 용어로 정의하였고 비 동기성의 특성에 부합한 Timing diagram을 정의하여 교통 신호등과 같은 시스템에 적용하여 연구의 유용성을 보였다. 이 연구의 특징은 하나의 Timing diagram으로 시스템 전체를 표현하는 것은 매우 어려운 일이기 때문에 시스템을 쪼개서 Timing diagram을 그리고 그것을 나중에 병합하는 접근 방법을 택하였다. 하지만, 이 연구는 계층성에 대한 개념은 포함되어 있지 않아 분산 시스템의 모델에서만 사용 가능한 한계를 갖는다.

Ogoubi의 연구[10]에서는 PCI와 같은 동기화된 버스 프로토콜의 구현을 검증하기 위해 Timing diagram 사양서를 동기화되는 상태 기계로 변환시켜 이를 시뮬레이션, 에뮬레이션 또는 정형 검증하는 동안 버스의 행위를 관찰하고 프로토콜을 어기는 행동이 관찰될 때 에러 시그널을 발생시키는 체크 도구로 사용하는 연구를 수행했다. 그리고 다른 연구들[11, 15]에서는 하드웨어 디자인에 자주 사용되는 Timing diagram을 검증하기 위해서 Timing diagram을 LTL 시제 논리나 ω 오토마타, 혹은 Büchi 오토마타로 변환시키는 연구를 수행했다. Amla의 연구[15]에서 모델 체킹을 위해 Timing diagram을 모델을 기술하기 위한 형식으로 택한 이유도 역시 Timing diagram이 엔지니어가 많이 사용하고 있으며, 엔지니어가 직접 시제 언어를 사용하여 모델 체킹을 기술하는 것은 어렵기 때문이다. 이 연구의 목적은 엔지니어가 모델 체킹을 Timing diagram을 이용하여 쉽게 사용할 수 있도록 하는 것이 목적이다. 이 논문에서는 동기적인 특성을 가진 Timing diagram을 정의하고 이 모델로부터 ω 오토마타나 시제 언어로 변환시켜서 모델 체킹을 수행했으며, Lucent의 PCI synthesizable core에 적용하여 그 유용성을 보였다. Chen은 그의 논문[14]에서 비 동기 회로의 지연을 모델링 하기 위해서 PTA를 적용하였는데, Timing diagram으로부터 PTA를 변환시키는 연구를 수행하였다. PTA는 Timed Automata의 일종인데 미지수의 파라미터가 추가된 형태이다.

프로토콜에 검증을 위한 모델을 기술하기 위한 사양을 조사하는 연구에서[17] Timing diagram을 모델을 기술하는데 적합하다고 판단하였는데, 그 이유는 Timing diagram은 간단하며, 모듈화 된 Timing diagram은 검증 도구를 만드는 단체에서 유래했으며, 또한 Timing diagram이 엔지니어들이 많이 사용하고 있기 때문이라고 하였다.

소프트웨어분야에서 Timing diagram을 이용하여 모델링 하려는 시도는 아직까지는 많지 않다. UML2에서 Timing diagram을 이용하여 시스템을 모델링 하기 위한 도구로 사용할 수는 있는데, sequence diagram와 비슷한 정도로 인식하고 있으며, 그만큼 활용 빈도는 아직 적은 편이다[16]. Joochim는 그의 연구[13]에서 Event-B[17]를 위한 Timing diagram을 제안하고, Timing diagram을 Event-B로 변환시키는 연구를 수행했다.

본 논문이 Timing diagram과 관련된 연구와의 공통점은 각각에서 정의한 Timing diagram이 1) 시그널의 변화의 연속체로 이루어져 있다는 것과 2) 시그널의 변화들 간의 선후 관계에 대한 정보를 포함 3) 시그널에 대한 시간 제약에 대한 정보를 포함하고 있다는 점이다. 반면 차이점은 기존의 연구들은 주로 시스템을 검증하기 위한 목적으로 연구되었던 것에 반해 본 연구는 Stateflow에 통합되기 위한 모델링 방법론을 개발하기 위한 목적으로 연구한 점이다.

Ⅵ. 결 론

본 연구에서는 임베디드 시스템의 소프트웨어를 기술하기 위해 기존의 Stateflow가 가진 문제를 해결하기 위해서 Stateflow와 Timing diagram을 통합하여 표현하는 방법론인 $Stateflow_{TD}$에 대해서 제안하였는데, Stateflow와 Timing diagram을 Stateflow 도메인에서 통합하는 방법, 그리고 $Stateflow_{TD}$가 가지는 장점들과 부가적 작업에 대해 설명하였다. 두 가지 예제를 통해 $Stateflow_{TD}$의 필요성을 입증하였으며, 실제적으로 $Stateflow_{TD}$를 적용하기 위한 방법에 대한 사례를 보였다.

이 논문의 결론은 Timing diagram은 행동의 시퀀스와 시간이 포함된 행동을 표현하는데 Stateflow보다 훨씬 직관적이고 편리하기 때문에 위와 같은 사양을 가진 시스템을 표현하고자 할 때는 Stateflow만으로

표현하는 것보다 본 논문에서 제시한 방법론인 $Stateflow_{TD}$로 표현하는 것이 훨씬 유용하다는 점이다. Timing diagram은 전자공학에서는 매우 익숙한 표현 방식이며, 본 논문에서 제시하는 Timing diagram역시 기존의 Timing diagram과 매우 유사하기 때문에 본 연구에서 제시한 Timing diagram을 어려움 없이 사용할 수 있을 것이고 Stateflow와 혼합하여 사용함으로써 산업 현장에서 훨씬 더 편리하고 직관적으로 사양서를 표현할 수 있을 것이라고 기대한다.

참 고 문 헌

〔1〕 Bilung Lee; Lee, E.A.;, "Hierarchical concurrent finite state machines in Ptolemy," *Application of Concurrency to System Design*, 1998. Proceedings., 1998 International Conference on , Vol., No., pp.34-40, 23-26 Mar 1998

〔2〕 Edward A. Lee, "Finite State Machines and Modal Models in Ptolemy II," *EECS Department, University of California, Berkeley, Tech. Rep.* UCB/EECS-2009-151, Nov. 2009.

〔3〕 T. Buck, S. Ha, E. A. Lee, and D. G. Messerschmitt, "Ptolemy: A Framework for Simulating and Prototyping Heterogeneous Systems," *Int. Journal of Computer Simulation, special issue on "Simulation Software Development,"* Vol.4, pp.155-182, April, 1994.

〔4〕 R. Schlor, "Symbolic timing diagrams: A visual formalism for model verification," *Ph.D. dissertation, Fachbereich Informatik, Carl-von-Ossietzky Universit¨at Oldenburg,* 2001.

〔5〕 Helbig, J.; , "Extending VHDL for state based specifications," *Design Automation Conference*, 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.675-684, 29 Aug-1 Sep 1995

〔6〕 David Harel, "Stateflow: A visual formalism for complex systems", *Science of Computer Programming*, Vol.8, Issue3, pp.231-274, 1987

〔7〕 Vahid, F.; Narayan, S.; Gajski, D.D.;, "SpecCharts: a VHDL front-end for embedded systems," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* , Vol.14, No.6, pp.694-706, Jun 1995

〔8〕 Kim Gruttner, Wolfgang Nebel, "Modelling Program-State Machines in SystemCTM," *Specification, Verification and Design Languages, FDL 2008. Forum on* , Vol., No., pp.7-12, 23-25 Sept. 2008

〔9〕 Frank Vahid; Tony Givargis, "*Embedded System Design: A Unified Hardware/Software Introduction*", John Wiley & Sons, 2002

〔10〕 E. K. Ogoubi, Eduard Cerny "Synthesis of checker EFSMs from Timing diagram specifications", *ISCAS* (1), p13-18, 1999

〔11〕 Kathi Fisler, "On Tableau Constructions for Timing Diagrams", *NASA Langley Formal Methods Workshop*, 2000

〔12〕 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

〔13〕 Joochim, T. "Timing diagrams add Requirements Engineering capability to Event-B Formal Development". *Technical Report UNSPECIFIED, DSSE Group, Electronics and Computer Science, Souhtampton University*, 2008.

〔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〕 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

〔16〕 Scott W. Ambler, "*The Object Primer: Agile Model Driven Development with UML 2*", Cambridge University Press, 2004

〔17〕 Event-B: http://www.event-b.org/

〔18〕 이홍석, 정기현, 최경희, "Timing_diagram의 테스트 케이스 생성 전략", *정보처리학회 논문지 D*,

v.17D, No.4, pp.283-296, 2010
〔19〕 이홍석, 정기현, 최경희, “선형 계획법을 이용한 Timing Diagram의 테스트 입력 시퀀스 자동 생성 전략”, *정보처리학회 논문지 D*, v17D, No.5, pp.337-346, 2010
〔20〕 Mathworks, http://mathworks.com
〔21〕 The MathWorks Inc, “Stateflow User’s Guide”, Available from URL www.mathworks.com/access/helpdesk/help/pdf.../stateflow/sf_ug.pdf, 2008
〔22〕 Cheng, K-T; Krishnakumar, A.S., “Automatic Functional Test Generation Using The Extended Finite State Machine Model”. *International Design Automation Conference (DAC)*. ACM. pp.86–91, 1993
〔23〕 John F. Wakerly, “*Digital Design – Principles and Practices*”, Prentice Hall, 4th ed, pp.682-686, 2005
〔24〕 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
〔25〕 Flemming Nielson, “*Principles of Program Analysis*”, Springer, 2004
〔26〕 Lee, Edward; Alberto Sangiovanni-Vincentelli, “A Framework for Comparing Models of Computation”, *IEEE Transactions on CAD*, Vol 17, issue12, pp.1217–1229, 1998
〔27〕 Kirill Bogdanov, “Automated testing of Harel’s statecharts,” *PhD thesis, The University of Sheffield*, Jan. 2000
〔28〕 Jee-Eun Yoo, “Using Model Checking to Generate Data-Flow Oriented Test Case from Statecharts,” *Master thesis, KAIST*, 2002
〔29〕 Jungsup Oh, “Automatic Generation of Test Cases based on Requirement Models,” *PhD thesis, AJOU University*, 2009
〔30〕 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 Conference and Workshop* on the , Vol., No., pp.83-91, 2001
〔31〕 Abraham Silberschatz, “*Operating System Concepts*”, Wiley, 2008

이 홍 석 (Hongseok Lee) 정회원

2003년 아주대학교 전자공학부 (공학사)
2003년 아주대학교 정보 및 컴퓨터 공학부(공학사)
2005년 아주대학교 전자공학과 (공학석사)
2005년~현재 아주대학교 대학원 전자공학과 박사과정
<관심분야> 명세 기술, 정형 검증, 소프트웨어 공학, 임베디드 시스템

정 기 현 (Kihyun Chung) 정회원

1984년 서강대학교 전자공학과 학사
1988년 미국 Ilinois주립대 EECS 석사
1990년 미국 Purdue대학 전기전자공학부 박사
1991~1992년 현대반도체 연구소
1993년~현재 아주대학교 전자공학부 교수
<관심분야> 컴퓨터구조, VLSI 설계, 멀티미디어 및 실시간 시스템 등

최 경 희 (Kyunghee Choi) 정회원

1976년 서울대학교 수학교육과 학사
1979년 프랑스 그랑데꼴 Enseeiht 대학 석사
1982년 프랑스 Paul Sabatier 대학 정보공학부 박사
1982년~현재 아주대학교 정보통신전문대학원 교수
<관심분야> 운영 체제, 분산시스템, 실시간 및 멀티미디어시스템 등