임베디드 시스템을 위한 동기적 언어 기반 하드웨어/소프트웨어 통합 설계 및 검증

이수영° 김진현 최진영
고려대학교 컴퓨터학과
{sylee°, jhkim, choi}@formal.korea.ac.kr

Hardware/Software Co-design and Verification by Synchronous language for Embedded System

Suyoung Lee° Jinhyun Kim Jinyoung Choi
Dept. of Computer Science and Engineering, Korea University

요 약

전통적인 임베디드 시스템 개발은 하드웨어와 소프트웨어가 독립적으로 개발된다. 그러나 시스템 개발 후 오류 발생 시, 하드웨어와 소프트웨어 둘 중 어디에서 발생했는지 알아내기 어려웠다. 따라서 임베디드 시스템 개발을 위해 하드웨어/소프트웨어의 통합 설계 방법이 연구기관들에 의해 제시되어 왔다. 본 논문은 현실적으로 많이 사용되고 있는 일반 임베디드 시스템 개발 방법으로부터 접근하는 HW/SW 통합 개발 방법을 제안하였다. 즉, 이미 만들어진 하드웨어를 설계 단계로 끌어올려 정형 기법을 통해 하드웨어를 설계 및 정형 검증하여 견고한 하드웨어를 만들고, 이를 기반으로 소프트웨어를 정형 명세 및 검증하는 하드웨어/소프트웨어 통합 개발을 수행하였다. 따라서 개발 후 하드웨어 또는 소프트웨어에서 발생할 수 있는 오류를 최소화하고 오류가 발생하였다고 해도 개발 전에 설계상에서 오류를 수정할 수 있어 임베디드 시스템의 신뢰성을 보장하였다. 또한 설계 과정의 어떤 시점에서 개발 중인 가상의 하드웨어가 아닌 개발 완료된 하드웨어의 실제 코드를 테스트할 수 있으므로, 현실적인 임베디드 시스템 개발에 더 효과적인 하드웨어/소프트웨어 통합 개발 방법론을 제시하여 그 효율성을 높였다.

1. 서 론

임베디드 시스템이란 전용 동작을 수행하거나 또는 특정 임베디드 소프트웨어 응용 프로그램과 함께 사용되도록 디자인된 특정 컴퓨터 시스템 또는 컴퓨팅 장치를 말한다. 전통적인 임베디드 시스템의 설계과정에서 하드웨어와 소프트웨어는 대부분 따로 개발되어지며 보통의 경우 하드웨어가 완성된 이후에 소프트웨어가 개발되어 통합되어진다. 많은 경우, 소프트웨어의 실제적인 검증이 하드웨어가 완성된 뒤에 시작되기 때문에 전체 시스템 검증과 동시에 소프트웨어를 검증하는 시간은 전체 검증 시간의 절반 이상을 차지할 수 밖에 없으며 상당히 비효율적인 과정을 가지고 있을 수 밖에 없다. 이외에도 소프트웨어 엔지니어는 하드웨어가 릴리스 버전에 가깝게 견고한 구성을 드러낼 때까지 대부분의 운영체제 개발을 완료할 수 없었다. 한편, 하드에어 엔지니어는 소프트웨어를 개발할 때까지 하드웨어 개발 중에 생긴 오류를 발견하지 못하면 다시 원점으로 돌아가서 수정해야 했다. 수주 또는 수개월의 귀중한 시간을 허공으로 날릴 가능성이 큰 것이다. 또한 설계 후반부에서의 수정 작업은 더욱 복잡하다. 디자이너는 개발할 제품의 요건에 맞춰 하드웨어를 제작하기 때문에, 최종 제품의 실행 속도, 전력 소모 및 제조비용 목표를 맞추기 위해 새로운 컴퓨팅 엔진을 만들고 대부분의 주변 장치 컨트롤러를 새로 설계하는 경우가 흔하다. 그런데 새 하드웨어 아키텍처에는 운영체제에서부터 장치 드라이버 및 버스 프로토콜까지 새로운 한 벌의 소프트웨어를 사용해야 한다. 따라서 하드웨어와 소프트웨어 구성 요소의 동시 개발은 선택이 아니라 필수가 되었고, 임베디드 시스템의 설계단계에서부터 임베디드 소프트웨어의 신뢰성을 보장할 수 있도록 엄격한 개발과정이 요구되었다. 기존 임베디드 시스템 개발은 하드웨어가 만들어지고 난 다음에 소프트웨어가 개발되어 그 후 하드웨어와 통합, 이후 오류 발생 시 하드웨어에서 발생했는지 아님 소프트웨어에서 발생했는지 알 수가 없었다. 이에 하드웨어와 소프트웨어를 통합 설계하고 검증하는 다양한 방법들이 많은 연구기관들에 의해 제시되어 왔다. 본 논문에서는 임베디드 시스템 개발을 위해 동기적 언어인 Esterel을 이용하여 하드웨어/소프트웨어 통합 설계 및 통합 검증을 제안하고자 한다. 보통 하드웨어/소프트웨어 통합 개발을 할 경우, 시스템 레벨에서 하드웨어/소프트웨어를 명세하고 검증 한 뒤 하드웨어/소프트웨어를 분할하여 각각 하드웨어와 소프트웨어로 개발한다. 따라서 소프트웨어를 실현하는 부분과 하드웨어로 실현하는 부분의 분할이 어렵다. 본 논문은 하드웨어와 소프트웨어가 처음부터 쿨리어하게 분할되어 있는 상태에서 설계를 한다. 즉, 이미 만들어져 있는 하드웨어 스펙으로부터 정형 명세 언어인 Esterel[1]을 이용하여 하드웨어를 설계하고 검증한 후, 마찬가지로 소프트웨어를 Esterel로 설계, 이를 하드웨어와 통합한다. 그 후 임베디드 시스템의 하드웨어와 소프트웨어가 가질 수 있는 오류 가능성을 고려하여 구현한 모델을

통합 검증을 하도록 정형기법을 도입한 임베디드 시스템 개발 방법론을 제시한다.
본 논문의 구성은 다음과 같이 구성되어 있다. 2장은 본 연구와 관련된 기존 연구들에 대해 간단하게 소개하고, 3장에서는 본 논문에서 제안한 하드웨어/소프트웨어 통합 설계 및 통합 검증과정에 대해 설명한다. 4장은 Esterel 설계 명세로부터 실행 가능한 코드를 자동 생성하여 타겟 보드에 포팅 및 실행해 보았다. 마지막으로 5장에서 결론을 맺고 향후 연구 계획에 대해 소개한다.

2. 배 경 연 구

현재 하드웨어/소프트웨어 통합설계는 임베디드 시스템 개발을 위한 방법으로 많은 방법들이 제시되고 있다. 그 중 임베디드 시스템들을 둘러싼 환경변화에 순응할 수 있는 설계 절차를 효율적으로 실현할 수 있는 기술로 SpecC와 SystemC가 제시되었다. SpecC[2]는 대학에서 하드웨어/소프트웨어 혼재 시스템의 시스템 모델링을 목적으로 한 연구과정에서 탄생한 반면, SystemC는 하드웨어 설계의 추상도를 시스템 수준으로 끌어올리고자 하는 EDA 벤더의 노력으로 탄생했다. SystemC[3]는 RTL과 시스템 레벨에서 시스템들의 명세와 모델링을 위해 사용되고 있다. 이 SystemC는 제공된 특별한 C++라이브러리들과 명세된 RTL을 기반으로 하는 시뮬레이터와 시스템 레벨 설계들을 기반으로 한다. C++를 기반으로 하기 때문에 C++에 익숙한 프로그래머들이 사용하기에 편리한 하드웨어와 소프트웨어를 통합 설계하는 방법이다. SpecC는 C의 확장으로서 시스템 레벨 설계와 수작업을 통해 수정하는 것을 목적으로 한다. 둘 다 시스템 레벨 개발을 위한 생산성의 중요한 요소가 되는 단일 프레임워크와 통합 시뮬레이션 내에서 소프트웨어와 하드웨어 구성 요소들을 기술한다. 그러나 완전한 하드웨어/소프트웨어의 정형검증이 어렵다. 하드웨어/소프트웨어 통합 개발을 위한 방법으로 Polis[4]가 사용되고 있다. Polis는 시스템 수준에서 하드웨어/소프트웨어의 정형적인 통합 설계 및 정형검증을 하고 나서 하드웨어와 소프트웨어를 분할 및 구현한다. 하지만 하드웨어와 소프트웨어의 분할이 어렵다는 문제점이 있다. 본 논문은 처음부터 하드웨어와 소프트웨어가 클리어하게 분할되어 있는 상태에서 하드웨어를 설계 및 검증하고 그 후 소프트웨어를 통합 설계하여 하드웨어/소프트웨어의 통합 검증을 한다. 즉, 본 논문은 만들어진 하드웨어에서 하드웨어와 interact하는 소프트웨어를 개발하는 방법으로 접근하였다.

3. 제안된 하드웨어/소프트웨어 통합 개발 방법

본 장에서는 임베디드 시스템을 위한 하드웨어/소프트웨어 통합 설계에 대해 설명하고자 한다. 그림 1은 본 논문에서 제안한 하드웨어/소프트웨어 통합 설계 및 통합 검증 과정을 나타내고 있다. 먼저, 시스템 요구사항으로부터 하드웨어 요구사항을 분석하여 하드웨어를 정형 명세 한다. 정형 명세한 하드웨어 디자인을 하드웨어가 요구하는 특성을 검증 특성으로 주어 정형 검증한다. 이렇게 하드웨어를 완전하게 구현하는데, 이것은 Esterel이 RTL 레벨의 하드웨어 언어이기 때문에 VHDL로 코드제너레이션하여 바로 하드웨어를 만들 수가 있다. 그리고 소프트웨어 요구사항을 분석하여 이를 구현된 하드웨어 디자인에 소프트웨어를 concurrent 하게 통합 설계한다. 이 통합 설계는 시뮬레이션과 통합 정형검증을 수행한다. 통합 정형검증에 대해서는 3.5장에서 설명하겠다.

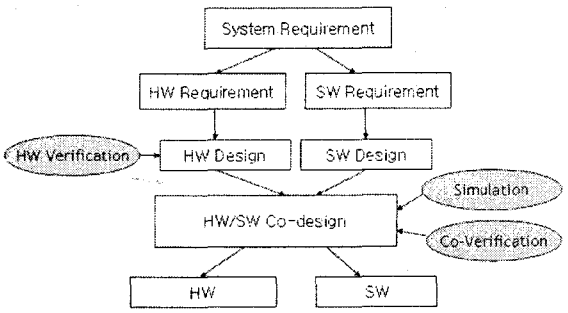

그림 1 제안된 하드웨어/소프트웨어 통합 설계 및 검증

3.1 하드웨어 설계

본 절에서는 UART 하드웨어를 선택하여 Esterel로 정형 명세하였다. UART 하드웨어는 RX, TX, CTS, RTS 등의 Wire를 가진 RS232 인터페이스를 통해 호스트와 타겟 간의 통신을 수행한다. 소프트웨어로부터 직접적으로 RX와 TX Wire를 통한 읽기와 제어기능을 UART 서킷의 RS232 serial port를 통해 읽고 쓰도록 한다. RS232 인터페이스의 전송과 수신부분은 Esterel의 Safe State Machine 부호를 이용하여 그래픽하게 기술되도록 Esterel의 그래픽컬한 표현언어인 SyncCharts[5]를 이용하여 기술하였다. RS232 전송자를 위한 모델은 그림 2와 같다. 이 상태 기계는 전송을 초기화하는 제어 신호를 기다리고 그 후 data 비트들과 시작 비트를 보낸다.

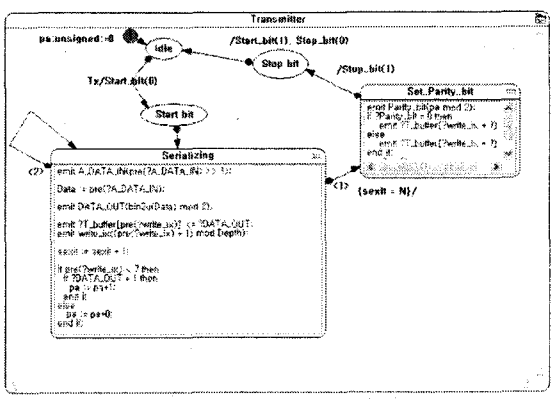

그림 2 UART 하드웨어 전송부

그림 3은 수신 부분의 명세를 보여주고 있으며 이 명세

는 하드웨어 흐름 제어는 사용하지 않는다. RS232 수신자는 RX 입력을 기다리는 상태에서 시작한다. 이 동기적 시스템은 보오드 레이트는 고려하지 않았다. RX 입력이 샘플링 주기 중에 0이 되도록 결정되었을 때 시스템은 캐릭터를 인식하고 패리티 비트를 체크하는 연속적인 전이들을 만든다. 그런 몇몇 수행들은 비록 서술형 매크로 스테이트를 이용한 버퍼에 읽기 비트 안에 쉬프팅 과정을 표현했지만 그래픽하게 명세되어 있다. 우리가 본 논문에서 사용하고 있는 실제 RS232 인터페이스는 시작 비트를 받았을 때 결정을 위해 더 정교한 정책과 읽기, 쓰기 위한 FIFO를 포함한다. UART의 전송과 수신 부분은 그림 4와 같이 완전한 UART 설계 형태를 갖추는데 함께 구성될 수 있다.

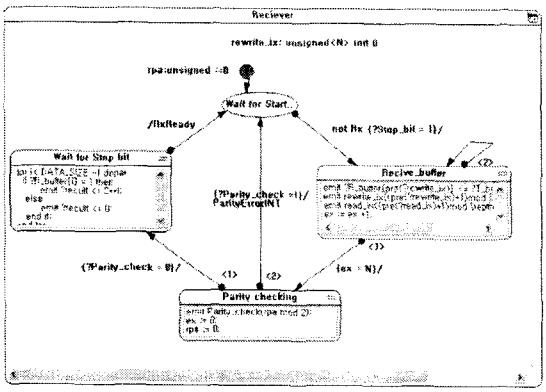

그림 3 UART 하드웨어 수신부

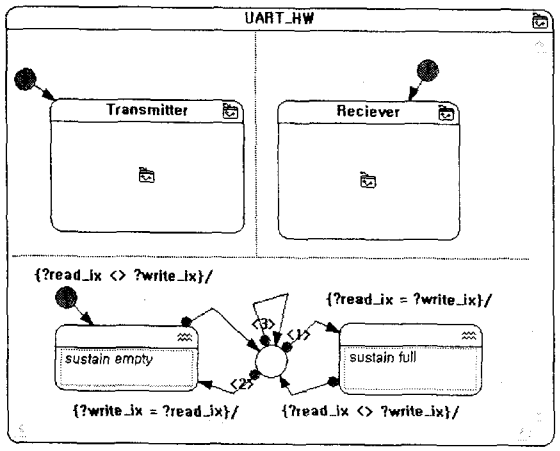

그림 4 RS232 인터페이스의 UART 하드웨어

3.2 하드웨어 정형 검증

본 절에서는 설계한 하드웨어를 검증하기 위하여 다음과 같이 주어진 검증특성을 가지고 Esterel studio[6] 검증기를 통해 정형 검증하였다. 하드웨어에서 버퍼 레지스터는 버퍼가 full인 경우에만 Serial interrupt 신호를 발생하여 버퍼로부터 데이터를 읽을 수 있도록 해야 한다. 이를 검증 특성으로 다음과 같이 LTL[7]을 이용하여 기호화하고 그림 5와 같이 Esterel에서 assertion으로 정의하여 정형 명세하였다.

$$G((full \rightarrow SERIAL\_INT) \wedge (SERIAL\_INT \rightarrow full))$$

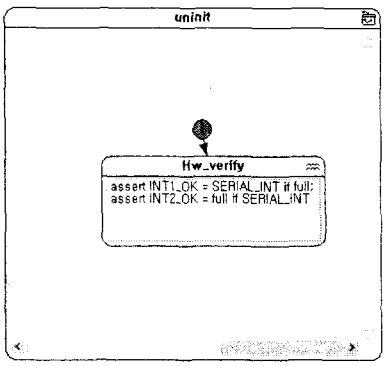

그림 5 Safety Property of HW

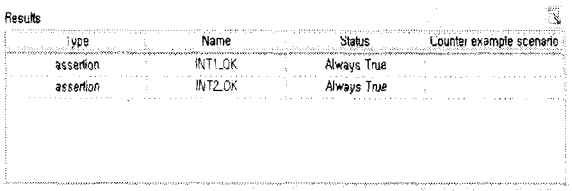

그림 6 Result of Verification

그리고 이를 Esterel studio의 검증기를 통해 정형 검증하였다. 그림 6은 검증결과로 버퍼가 full인 경우 SERIAL_INT 가 발생한다는 가정이 항상 true임을 확인할 수 있다.

3.3 하드웨어/소프트웨어 통합 설계

본 절에서는 하드웨어와 소프트웨어를 통합 검증하기 위해 정형 명세 및 검증이 완료된 하드웨어 설계와 소프트웨어를 통합 설계한다. 소프트웨어는 UART 디바이스 드라이버 역할을 하는 모듈과 application 태스크이다. 본 논문에서 제안하는 UART 디바이스 드라이버는 비동기 시리얼 통신의 하위 레벨 작업을 핸들링하는 통합된 서킷인 UART의 드라이버이다. UART 디바이스 드라이버는 데이터를 단순히 UART로부터 한 바이트의 데이터를 읽거나 쓰는 동작을 하며, Parity bit check error시 재전송 인터럽트를 발생시켜 데이터를 재전송한다. 하드웨어에 대한 소프트웨어의 동작 모델은 다음 그림 7과 같다. 임베디드 시스템의 소프트웨어 모델은 실제 운영체제에서 실행되는 Application task 모듈과 전송 모듈, 수신 모듈, 재전송 모듈로 이루어져 있다. 동작 모델의 입력, 출력 시그널은 다음 표 1과 같다.

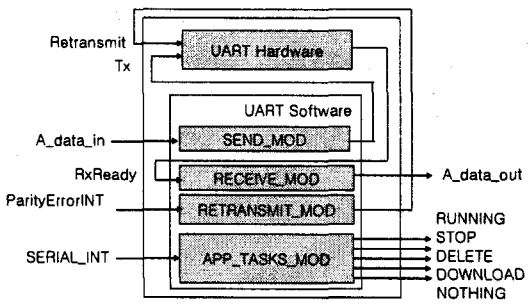

그림 7 Software의 동작 모델

표 1 동작 모델의 입력, 출력 시그널

Input signals	A_data_in ParityErrorINT SERIAL_INT	-Data value to transmit -Interrupt signal occur when parity check is error -Interrupt signal start to receive when buffer register is full
In realation to Output signals	Tx RxReady Retransmit	-Start signal for data to transmit -Wait signal for Rx -Retransmit signal
Output signals	A_data_out STOP, DELETE, DOWNLOAD, NOTHING	-Received data value -Command of task mode

이렇게 하드웨어/소프트웨어 통합 검증을 위해 Esterel로 UART 하드웨어와 소프트웨어를 다음 그림 8과 같이 메인 모듈에서 패러렐하게 동작하도록 구현하였다. 그림 8과 같이 정형 명세한 임베디드 시스템의 하드웨어/소프트웨어 통합 설계는 모듈들 간에 브로드캐스팅(Broadcasting)되는 시그널들을 통해서 동기화 되며 주어진 작업을 수행한다.

3.4 시뮬레이션

UART 하드웨어와 소프트웨어 통합 설계 명세를 Esterel Studio의 검증 시뮬레이션 도구를 이용하여 통합 설계의 타당성 검증을 수행하였다. Esterel studio 의 시뮬레이션은 시스템에 어떤 입력값을 주고 그 입력 값에 따른 상태 다이어그램의 상태 변화와 출력되는 시그널을 통해서 결과가 처음에 의도한대로 올바르게 나오는지를 확인해보는 방법이다. 본 논문에서는 그림 9와 같이 하드웨어/소프트웨어 통합 설계가 올바르게 수행함을 확인하였다. 이렇게 명세한 임베디드 시스템의 하드웨어/소프트웨어의 통합 설계를 가지고 3.5장에서 정형 검증해 보았다.

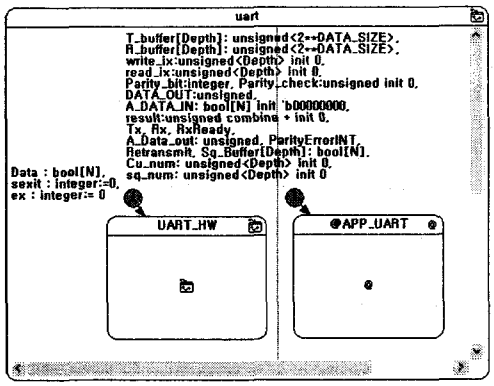

그림 8 HW/SW Co-design

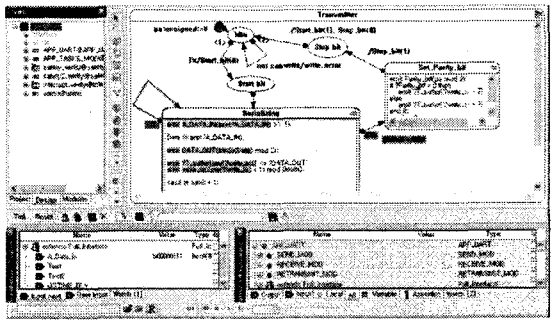

그림 9 Validation of HW/SW Co-design

3.5 하드웨어/소프트웨어 통합 검증

본 장에서는 3.3장에서 명세한 하드웨어/소프트웨어 통합 모델을 가지고 정형 검증하고자 한다. Esterel에서 사용하는 정형 검증 기법은 모델체킹 기법으로, 명세한 시스템을 시스템 사용시 발생 가능한 모든 상태를 표현하는 유한 상태 기계로 바꾼다. 등가관계 기법과 BDD기법, SAT 기법을 이용하여 유한 상태를 최소화 시켜 유한 상태 기계의 모든 상태를 찾아다니며, 출력 시그널의 상태를 확인하는 상태 탐색 기법을 이용하여 검증한다. Esterel에서 검증 방법론은 동기적 observer들의 개념을 기반으로 한다. 특별한 수학적 언어 안에서 작성한 정형 특성보다 어떤 특별한 언어 내에서 assertion으로서 일반적인 상태 기계 기술들을 개발한다. 이들은 수신자를 같은 환경적인 입력 값들을 테스트용 시스템으로서 기술하고 관심 있는 특성들을 따른 시스템 체크 과정을 감시한다. 검증 특성이 시뮬레이션에 의해 비정형적으로 타당한지, static analysis 수행에 의해 정형적으로 타당한지를 보이도록 체크할 수 있다. 본 논문에서는 시뮬레이션을 통해 타당성을 검증 받은 통합 설계의 정형 검증을 위해 검증 특성을 주어 올바른 행위를 수행하고 있음을 검증해보고자 한다. 우선 설계한 시스템이 만족해야할 요구 조건을 Esterel 언어에서 assertion로서 기술하고 이를 검증기를 통해 조건이 참인지 거짓인지를 관찰한다. 만족하는 경우에는 시스템이 본래 의도대로 제대로

설계되었다는 것을 의미하고, 위반하는 경우는 검증기 분석에 의한 해당 검증 조건의 반례(counter example)를 발생한다. 분석된 반례는 다시 시뮬레이터를 통해 그 발생 경로를 추적 후 수정하게 되고, 시스템 모델의 오류 수정 후 시스템의 재검사 작업이 이루어진다. 본 장에서는 명세한 UART HW/SW의 통합 설계에 대한 동작의 신뢰성을 확인하기 위해 시뮬레이션과 검증 기법을 이용해서 설계의 오류가 없는지를 확인해본다. 검증에 대한 확인 과정은 검증 특성을 가지고 만족하는 결과를 보이는지를 확인해본다. 검증 특성이 요구하는 결과를 만족한다면 시스템이 알고리즘의 원칙대로 동작한다는 사실을 알 수 있고 이를 토대로 시스템이 올바르게 설계되었음을 확인해 볼 수 있다.

3.5.1 검증 특성

이번 절에서는 UART HW/SW 통합 설계가 갖추어야 할 검증 특성으로 하드웨어와 소프트웨어간의 Interaction을 검증하기 위해 하드웨어 레벨에서 발생하는 시리얼 인터럽트가 소프트웨어 레벨에서 제대로 인터럽트를 처리하는지를 검증하기 위해 다음과 같은 검증 조건을 두었다. 인터럽트에 의해서만 소프트웨어 태스크가 동기화가 되어야 한다를 만족하기 위해서는 IF and Only If 조건을 만족해야 한다. 즉, 소프트웨어 태스크가 동기화 되면 시리얼 인터럽트 신호는 발생해야 하며 이는 하드웨어/소프트웨어 설계의 모든 스테이트에서 만족되어야 한다. 검증 조건은 LTL(Linear Temporal logic)을 이용하여 기호화하고 이를 그림 10과 같이 Esterel 내에서 assertion을 적용하여 명세하였다.

Safety Property 1> 시리얼 인터럽트 신호 발생에 의해서만 소프트웨어 태스크는 동기화가 되어야 한다.

$$G((SERIAL\_INT \rightarrow Q\_SYNC) \wedge (SYNC \rightarrow SERIAL\_INT))$$

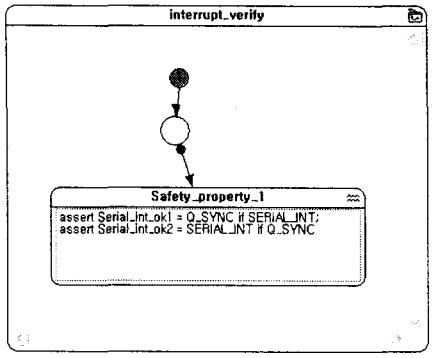

그림 10 Safety Property by Esterel

본 논문은 다른 검증 특성으로 시리얼 통신의 parity bit check시 오류가 발생하면 이를 컨트롤하는지를 정형 검증해보았다. 데이터를 수신 중 수신기는 Parity bit check를 한다. 이를 통해 데이터가 유실되었는지를 확인한다. 만약 유실되었다면 유실된 데이터를 다시 재전송하도록 application task, 즉, 소프트웨어 레벨에서 전송기에 요구한다. 이를 위해 가상의 parity bit check 오류를 발생시킨다. 즉, test 신호는 가상의 parity bit check 오류 신호이다. 이 test 신호가 발생하면 재전송 신호가 발생해야 한다. 이 역시 모든 스테이트에서 만족되어야 한다. Safety proeprty 1과 마찬가지로 하드웨어와 소프트웨어간의 interaction을 검증해본 것이다.

Safety Property 2> 언젠가 ParityErrorINT 신호가 발생하면 데이터의 재전송 신호를 발생한다.

$$G(F\ ParityErrorINT \rightarrow Retransmit)$$

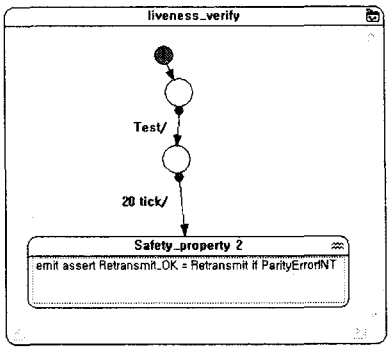

그림 11 Safety property 2 by Esterel

3.5.2 검증 결과

Esterel studio의 verifier를 통해 assertion으로 정의한 검증 조건이 그림 12와 같이 항상 참임을 볼 수 있었다. 따라서, 검증 조건 1에 의한 검증 결과는 Interrupt 신호가 발생하지 않으면 동기화 되지 않으며, Interrupt 신호가 발생하면 항상 동기화된다는 사실을 확인 할 수 있었다.

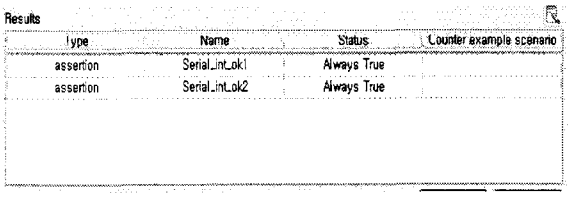

그림 12 Result of Safety Property 1

검증 조건 2 역시 그림 13과 같이 Esterel studio의 verifier를 통해 assertion으로 정의한 검증 조건이 항상 참임을 확인할 수 있었다. 따라서 검증 조건 2에 의한 검증 결과는 Parity bit check 오류시 재전송이 이루어짐을 확인할 수 있었다. 이를 통해 error control을 보장함을 확인했다.

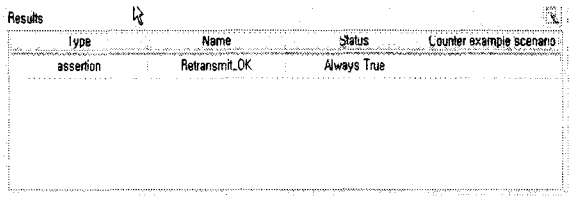

그림 13 Result of Safety Property 2

따라서 UART HW/SW의 모든 조건들의 타당성을 Esterel Studio의 시뮬레이션을 이용하여 모델링 한 후에 수행되어진 단계라 할 수 있다. 3.5.1 절에서 설명한 검증 특성들을 위배하는 이벤트가 결코 발생하지 않으므로 설명한 특성들이 모두 참의 결과 값을 갖는다는 것을 알 수 있고, 만약 검증 특성들 중 거짓(Always false)값을 갖는 특성이 있었다면, 그에 따른 counter example을 시뮬레이션 해줄 것이다.

4. 구현 및 실행

제안된 하드웨어/소프트웨어 통합 설계 및 통합 검증 flow chart에 따르면 정형 검증된 co-design 으로부터 하드웨어와 소프트웨어 코드를 생성할 수 있다. Esterel은 각각 하드웨어를 위한 RTL 레벨의 VHDL 코드와 소프트웨어를 위한 C 코드를 생성한다. 본 논문에서는 하드웨어는 이미 구현되어 있는 것으로부터 명세한 설계이기 때문에 소프트웨어 설계만을 추출하여 C코드를 자동 생성하였다. 그리고 이를 타겟 보드인 Intel 80c196kc 마이크로프로세서가 탑재된 보드에 포팅 및 실행해보았다.

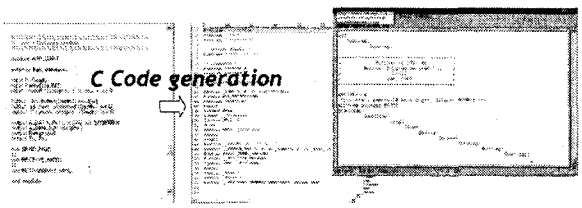

그림 14 Implementation and Execution

5. 결론

본 논문에서 제안한 하드웨어/소프트웨어 통합 설계 및 검증은 일반적인 임베디드 시스템 개발 방법론으로는 하드웨어와 소프트웨어를 각각 개발 시 복잡한 하드웨어와 소프트웨어의 구조로 인한 인터페이스상의 문제점이나 시뮬레이션의 문제점들을 해결하고자 제안하였다. 기존의 Polis나 SystemC를 이용한 하드웨어/소프트웨어 통합 개발은 하드웨어/소프트웨어의 분할이 어렵다거나 하드웨어/소프트웨어의 정형검증이 어려웠다. 따라서 본 논문은 일반적인 임베디드 시스템 개발과정에서 하드웨어와 소프트웨어를 각각 정형명세 언어인 Esterel을 통해 하드웨어/소프트웨어 설계를 정형적으로 명세하고 Esterel의 검증도구를 이용하여 정형검증을 함으로써 하드웨어/소프트웨어 통합 개발을 위한 새로운 방법론을 제시하였다. 결론적으로, 본 논문은 설계 과정의 어떤 시점에서 개발 중의 가상 하드웨어가 아닌 실제 임베디드 시스템 개발과정에서 개발 완료된 하드웨어의 코드를 정형명세 언어로 설계하여 이를 기반으로 하는 소프트웨어를 개발하는, 현실적인 임베디드 시스템 개발을 위해 더 효과적인 하드웨어/소프트웨어 통합 개발 방법론을 제시하여 그 효율성을 높였다. 그러나 개발 대상을 이해하고 모델링을 거쳐 명세 및 검증하는 과정에서 실제 임베디드 시스템에 대해 시스템 모델을 정하기 위한 완벽한 이해가 필요하다. 향후 연구 과제로는 SoC와 같은 복잡한 임베디드 시스템을 통합 설계 및 통합 검증하기 위한 개발 방법론을 연구할 계획이다.

6. 참고 문헌

[1] G. Berry, The Foundations of Esterel, in Proof, Language and Interaction: Essays in Honour of Robin Milner, ser. Foundations of Computing Series, G. Plotkin, C. Stirling, and M. Tofte, Eds. MIT Press, Aug. 2000

[2] UCI, SpecC, 2002, website at http://ics.uci.edu/specc, 2002

[3] SystmeC, 2002, website at http://www.systemc.org, 2002

[4] Polis, website at http://www-cad.eecs.berkely.edu/polis, 2000

[5] SyncCharts, website at http://www.i3s.unice.fr/sports/SyncCharts, 2005

[6] Esterel Studio, website at http://www.esterel-technologies.com/products/esterel-studio/overview.html, 2005

[7] Z. Manna, A. Pnueli : The Temporal Logic of Reactive and Concurrent Systems Specification, Springer-Verlag, 1996

[8] J J. Labrosse, "Embedded Systems Building Blocks Second Edition. Complete and Ready-to-Use Modules in C", CMP Books, 2002