SCADE/Lustre를 이용한 임베디드 시스템의 명세, 검증 및 구현

송관호° 심재환 안영정 최진영
고려대학교 컴퓨터학과
{ghsong°, jhsim, yjahn, choi}@formal.korea.ac.kr

Specification, verification, and implementation of an embedded system with SCADE/Lustre

Gwan-Ho Song° Jae-Hwan Sim Young-Jung Ahn Jin-Young Choi
Dept. of Computer Science and Engineering, Korea University

요 약

본 논문은 safety critical 실시간 반응형 시스템 하에서의 임베디드 소프트웨어 개발에 적합한 방법을 찾기 위해서 여러 가지 정형기법에 대해 논의하고 그 중 하나인 SCADE를 이용해 실제 임베디드 시스템을 직접 명세하고 검증한 후 구현한 내용을 서술한다. 본 논문에서는 data flow synchronous 언어인 Lustre를 소개하고, 소개된 언어가 실시간 반응형 시스템의 검증과 구현에 왜 적합한지를 논의하며, 하나의 시스템을 SCADE를 이용해서 명세하고 검증한 후 구현 한다. 수행된 실험을 통해서 data flow synchronous 언어는 실시간 반응형 시스템의 명세, 구현 그리고 검증에 적합한 언어라는 것을 언급하고, 이 언어의 사용이 복잡한 임베디드 시스템 개발에 효과적으로 사용될 수 있음을 제시한다.

1. 서 론

1996년도에 일어난 Ariane5 로켓 발사 실패, 미국 Virtual Case File (VCF) 프로젝트 실패 등과 같은 사건의 원인은 소프트웨어 오류이다 [1, 2].

특히 우주왕복선이나 의료장비 시스템의 소프트웨어 오류는 엄청난 인명피해를 초래할 수 있다. 더욱이 소프트웨어로 인한 프로젝트 실패는 천문학적인 예산 손실을 가져올 수 있다.

이러한 문제를 해결하기 위해 많은 방법론이 제시되고 있다. 본 논문은 그 중 하나인 정형기법을 통한 문제 해결을 제시하고 있다. 정형기법은 만들고자 하는 소프트웨어를 설계 단계에서부터 정형기법 언어로 명세하고 명세된 설계를 검증하는 방법을 연구하는 학문이다.

정형기법의 대표적인 언어들로는 Z, PVS, SMV 등이 있다 [3]. 이 언어들은 설계를 명세하고 검증하는 것에 중점을 두고 있다.

위와는 달리 실행가능한 명세란 개념이 있다. 이 개념은 명세 자체를 수행가능하게 하여 작성자가 그 동작을 확인함으로써, 그 명세의 정확성에 대한 확신을 가지는 것을 도울 수 있게 한 방법론이다 [4]. 또한 명세로부터 직접 구현되어질 시스템 상에서 실행 가능한 코드를 얻어내려는 연구가 활발히 진행되고 있다. 그 연구의 결과로 만들어진 언어 및 도구들로는 StateMate, Esterel, Lustre 등이 있다.

그 중 Esterel과 같은 동기화 언어를 이용한 임베디드 시스템의 개발 방법론을 제시한 여러 논문들이 있다 [5, 6, 14]. 그러나 위에 언급된 언어는 흐름 제어적인 (control flow) 부분에 초점이 맞추어져 있다. 자료 흐름적인(data flow) 특징을 가지고 있는 실시간 반응형 시스템을(real time reactive system)을 설계하고 구현할 때에 자료 흐름 동기화 언어인 Lustre를 이용하면 효율적인 코드 생성, 실행시간 예측, 강력한 일치성 검사등과 같은 이점들을 얻을 수 있다 [7].

본 논문에서는 SCADE/Lustre가 자동제어 혹은 모니터링 시스템과 같은 실시간 반응형 시스템의 개발 및 구현에 최적화 되어 있다는 점에 주목 하였다 [8]. 그리고 SCADE/Lustre를 사용하여 간단한 자동운전 시스템을 명세하고 검증한다. 그 후 자동 생성된 소스 코드를 LEGO® MINDSTORMS™ 시스템에 적용한다.

본 논문의 구성은 다음과 같다.

2장에선 본 논문이 사용하는 Lustre와 SCADE에 대해 설명을 한다. 3장에선 SCADE/Lustre를 이용한 설계, 검증 그리고 구현을 다룬다. 마지막으로 4장에선 결론과 향후 연구 방향을 다룬다.

2. 배 경 연 구

2.1 Lustre

Lustre 언어는 자동제어 혹은 모니터링 시스템과 같은 실시간 반응형 시스템의 프로그래밍을 위해 만들어진 자료 흐름 동기적 (data flow synchronous) 언어이다 [7, 8].

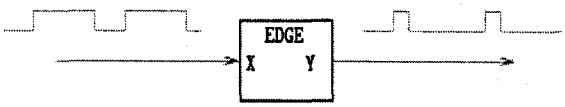
Lustre 언어에서 자료의 흐름은 시간적인 제약 안에서

엄밀하게 다루어진다 [7]. 또한 Lustre의 동기화 자료 흐름 개념은 실시간 반응형 시스템 명세에 유리하다. 그리고 이 명세로부터 효율적인 코드를 생성 수 있다 [7, 8].

Lustre 언어는 시제논리(temporal logic)의 부분집합으로 간주된다 [9, 10]. 검증 하고자 하는 safety property는 Lustre 언어로 명세된다 [7]. 따라서 Lustre 언어로 명세된 safety property는 시제논리로 변환될 수 있고 변환된 논리는 모델 체킹과 비슷한 기법으로 검증된다 [7].

위의 특징들에 의해서 Lustre 언어는 효율적인 Safety Critical 실시간 반응형 시스템의 설계, 검증 그리고 구현에 적합한 언어이다.

Lustre 프로그램은 기본적으로 node라 불리 우는 프로그램들의 집합이다. [그림 1]은 입력(X)이 false에서 true로 바뀌면 출력(Y)은 바뀐 순간부터 한 틱 동안 true를 출력하는 EDGE node의 개념도이다.



[그림 1] EDGE node

[그림 1]의 입력과 출력을 다시 표현해 보면 $X=(x_1,x_2,x_3,\ldots,x_n,\ldots)$, $Y=(y_1,y_2,y_3,\ldots,y_n,\ldots)$이 된다. 여기에서 아래첨자는 시간이다. 예를 들어 3이라는 시간일 때 X와 Y 값은 각각 X_3 와 Y_3 된다. [그림 1]의 EDGE node는 x_n 이 false 이고 x_{n+1}이 true 일 때 y_{n+1}의 값이 1틱 동안 true가 된다는 것을 표현하고 있다.

[그림 1]을 Lustre로 작성하면 [그림 2]와 같다 [11].

```
node EDGE (X: bool) returns (Y: bool);
let
  Y = false -> X and not pre(X);
tel
```

[그림 2] EDGE node의 Lustre 코드

현재 n 틱이라 가정할 때 pre(X) 연산을 쓰면 n-1 틱일 때의 X 값을 출력된다. X and not pre(X)의 의미는 '현재 X가 true 이고 이전의 X가 false이면 true가 된다.' 이다. 그리고 -> 연산은 y_0은 false로 하고 그 후의 틱은 -> 다음에 나오는 값이라는 것을 의미한다.

2.2 Safety Critical Application Development Environment (SCADE)

Esterel Technology에서 개발된 SCADE는 safety critical 임베디드 소프트웨어를 개발하기 위한 통합 개발 환경이다.

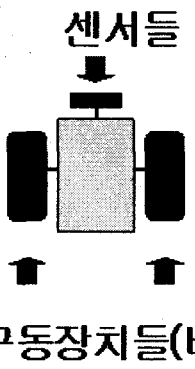
SCADE에서 Lustre 언어를 도식적으로 쓸 수 있다. 하지만 Lustre와 같은 동기화 자료 흐름기반 언어는 제어 흐름을 표현하는데 적합하지 않다. SCADE에선 이러한 Lustre 언어의 단점을 Safe State Machine (SSM)을 통해서 보완한다.

설계된 시스템의 가시적인 확인은 모의실험을 통해 가능하다. 정형적인 검증은 Design Verifier를 통해서 할 수 있다. 이 때 Design Verifier가 사용하는 방법은 다음과 같다. 일단 Lustre 언어를 이용해서 검증 속성을 명세한다. 그 후 명세된 검증속성은 미리 설계된 node들과 연결되고 자동으로 시제논리로 바뀐다. 바뀐 시제 논리는 모델 체킹과 비슷한 방법으로 검증된다 [7, 12, 13].

3. 연구 내용

3.1 실험 시스템

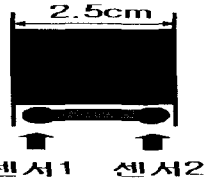
본 논문에서 실험한 시스템은 [그림 3]과 같다. [그림 3]은 전방의 상황을 센서를 통해서 감시하며 탐지된 정보에 따라서 구동장치를 조작하여 안전한 운전을 자동으로 가능하게 하는 시스템이다.



[그림 3] Target system

본 시스템의 센서는 [그림 4]에서 알 수 있듯이 검은색 라인을 감지한다. 자동 운전 시스템에서 사용될 수 있는 센서로는 GPS 센서, 이미지 센서(비디오 카메라) 등이 될 수 있다. 본 논문에서는 도로 밑에 특별한 전파 영역을 발생시키는 장치가 있고, 센서는 자동차가 전파 영역을 벗어나는 것을 감지할 수 있다고 가정했다. 특별한 전파영역은 2.5cm의 검은색 영역으로 정의 한다

([그림 4]). 또한 센서1과 센서2가 검은색 영역을 벗어나는 것은 시스템이 전파영역을 벗어나는 것으로 정의한다.



[그림 4] 센싱 개념도

제시된 시스템은 다음의 동작을 해야 한다. 첫째로, 시스템이 전파영역을 아예 탈선한다면, 시스템을 멈추어야 한다. 둘째는 뒤따라오는 시스템과의 충돌을 막기 위해서 시스템은 특정한 신호를 보내 뒤의 차들을 멈추어야 한다.

본 논문에서는 설계의 단순화를 위해 둘째 상황을 배제하였다.

3.2 Target platform

본 시스템이 구현될 platform은 LEGO사의 제품인 LEGO® MINDSTORMS™ 이다. LEGO® MINDSTORMS™ 의 컨트롤러는 Robot Command eXplorer (RCX) 이다. 이 컨트롤러에는 3개의 입력부와 3개의 출력부가 있다. 입력부에는 색 감지 센서, 버튼, 온도 센서등과 같은 입력 장치들이 연결된다. 출력부에는 모터, 전구등과 같은 장치들이 연결된다.

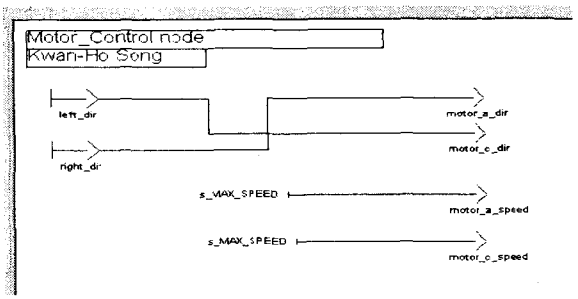
여러 가지 운영체제 및 개발도구를 RCX 에서 사용할 수 있다. 우선 기본적으로 LEGO사에서 제공하는 운영체제와 GUI 프로그래밍 도구가 있다. 그리고 BrickOS라는 C언어 기반 운영체제와 GNU gcc cross compiler라는 개발도구가 있다. 또한 LeJOS라는 자바 기반 운영체제도 있다.

본 논문에서 사용하는 운영체제는 BrickOS-0.2.6.10.6 이며 개발도구로는 GNU gcc-3.4 cross compiler를 사용한다.

3.3 설계

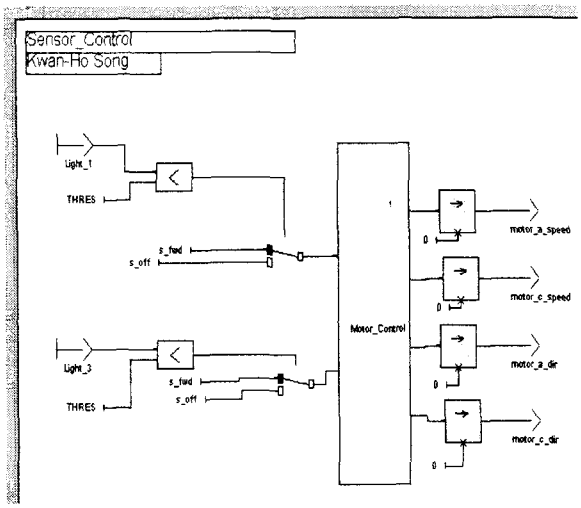
구현 시스템의 입력장치는 색 감지 센서 2개 이며, 출력장치는 바퀴 구동 모터 2개이다.

우선 첫 번째로 설계한 Motor_Control node는 입력 값에 따라서 모터제어를 하는 node이다. [그림 5]는 SCADE를 이용한 실제 설계 화면이다.



[그림 5] Motor_Control node

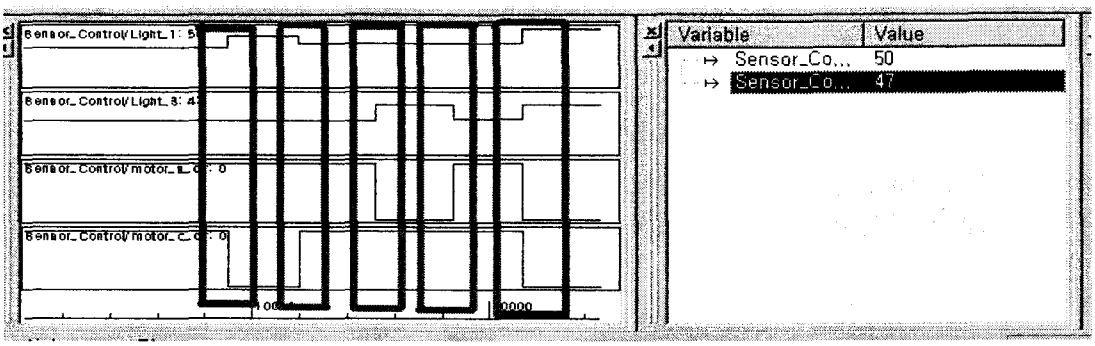
[그림 6]은 두 번째로 설계한 Sensor_Control node 이다. 이 node는 입력 받은 센서 값을 이용해서 시스템이 검은색 영역을 벗어났는가를 판단하여 모터 제어 정보를 Motor_Control node에 넘겨주는 역할을 한다.



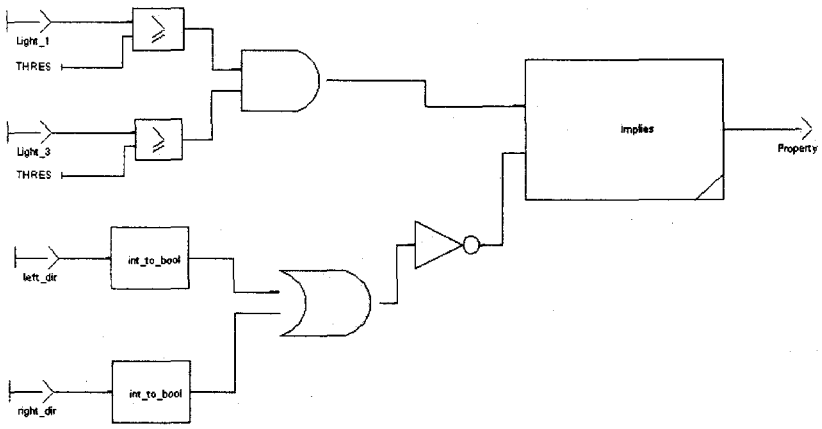
[그림 6] Sensor_Control node

3.4 모의실험 및 검증

모의실험은 SCADE에서 제공하는 [그림 7]과 같은 실험도구를 통해 진행하였다. 색 감지 센서는 검사한 값이 THRES 값(여기에선 40)보다 작으면 검은색 영역 안에 있다고 인식한다. [그림 6]의 Light_1 과 Light_3의 위치는 각각 [그림 4]의 센서1과 센서2의 위치이다. [그림 6]의 motor_a_dir 과 motor_c_dir 은 motor의 회전 방향을 결정하는 변수로써 0일 때 정지, 1일 때 직진, 2일 때 후진, 3일 때 제동의 동작을 하게 한다. motor_a_dir 과 motor_c_dir 은 각각 [그림 3]의 왼쪽과 오른쪽 바퀴의 회전을 책임진다.



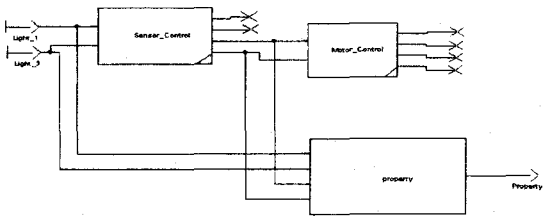
[그림 7] 모의실험 결과



[그림 8] 검증 속성

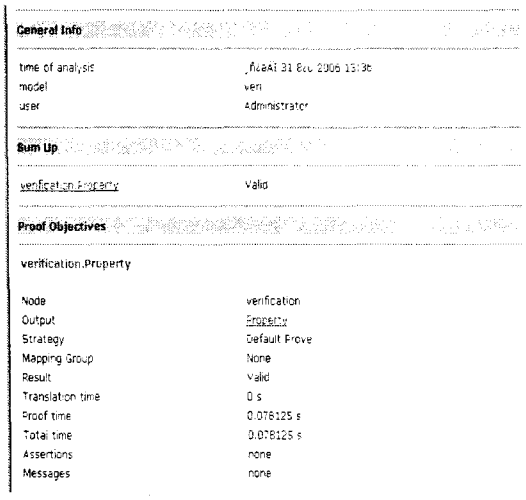
[그림 7]의 왼쪽에서 첫 번째 사각형은 Light_1이 영역에서 벗어난 상황이다 (Light1>THRES). 따라서 motor_c_dir 은 0이 되어 정지되고 motor_a_dir 은 1을 가리키기 때문에 시스템은 오른쪽으로 움직인다. 계속 오른쪽으로 움직여서 영역 안으로 이동하면 motor_c_dir 은 [그림 7]의 두 번째 사각형처럼 1이 되고 시스템은 직진을 하게 된다.

세 번째, 네 번째 사각형은 위의 상황과 정확히 반대되는 상황을 가리킨다. 마지막으로 다섯 번째 사각형은 시스템이 영역을 아예 탈선한 상황이며 motor_a_dir 과 motor_c_dir 는 정지를 가리킨다. 따라서 시스템의 움직임은 정지상태가 된다.



[그림 9] 검증 속성과 설계된 node들의 연결

시스템의 검증은 SCADE의 Design Verifier를 이용하여 검증하였다. 우선 검증해야 할 상황은 3.1 의 마지막 문단에서 언급한 바와 같이 시스템이 영역을 벗어나면 정지하는 것이다. 즉 센서1과 2의 값이 동시에 THRES 값보다 크다면((Light_1 >= THRES) AND (Light_3 >= THRES)) -명제 p라 정의- 모터의 구동이 둘 다 멈추어야 한다는 것(NOT(left_dir or right_dir)) -명제 q라 정의 - 이다. 이를 논리식으로 표현한다면 p -> q 가 된다. [그림 8]은 Design Verifier를 사용해서 논리식을 property node로 변환한 그림이다.



General Info

time of analysis	[illegible] 2006 13:36
model	veri
user	Administrator

Sum Up

verification.Property	Valid

Proof Objectives

verification.Property

Node	verification
Output	Property
Strategy	Default Prove
Mapping Group	None
Result	Valid
Translation time	0 s
Proof time	0.078125 s
Total time	0.078125 s
Assertions	none
Messages	none

[그림 10] 검증 결과

[그림 9]는 property node를 전에 설계했던 Sensor_Control node와 Motor_Control node에 연결하여 만들어낸 검증되어야 할 node이다. [그림 9]의 node를 검증한 결과는 [그림 10]에 나타나 있으며 검증속성이 Valid로써 결과가 타당하다는 것을 나타내주고 있다.

3.5 Target platform 구현

SCADE를 통해 생성된 코드는 3.2에서 언급한 개발도구를 통해 cross compile 되어 Sensor_Control.lx 란 바이너리 파일로 변환되었다.

자동으로 생성된 코드에서 수정해야 할 부분은 main 함수의 추가, 센서 인터페이스 그리고 모터 인터페이스와 SCADE에 의해 생성된 코드와의 연결이다.

[표 1]은 Target system 을 SCADE, Esterel, C 언어로 디자인하고 구현한 결과를 나타낸다.

	SCADE	Esterel v5_91	C
크기(byte)	410	1882	300

위의 결과는 바로 본 논문에서 설계한 시스템이 자료흐름 실시간 반응형 시스템의 한 종류이기 때문에 가능한 결과이다.

4. 결론 및 향후연구

SCADE는 자료 흐름 동기화 언어인 Lustre를 기본 언어로 하고 있는 실시간 반응형 시스템의 설계, 모의실험, 검증 그리고 효과적인 소스코드 생성에 중점을 둔 통합 개발 환경이다.

본 논문에서는 실시간 반응형 시스템적인 특징을 가진 시스템을 정해서 SCADE를 이용해 이를 설계, 모의실험, 검증하는 과정을 거쳤다. 그리고 실제 코드를 생성하고 이를 target platform인 LEGO® MINDSTORMS™ 에 구현하였다. 구현된 바이너리 파일은 C언어로 작성된 바이너리 파일의 크기와 크게 차이가 없었다. 더욱이 SCADE에 의해 구현된 바이너리 파일은 제어 흐름 기반 동기화 언어인 Esterel로 구현된 바이너리 파일보다 약 1/4배 작다는 것을 확인할 수 있었다.

하지만 이는 시스템이 단지 자료 흐름에 크게 의존하는 것이기에 가능한 결과다. 보다 범용적인 효율성 검사를 위해서는 실시간 반응형 시스템의 종류를 엄밀하게 정의하고 각각 정의된 시스템을 위의 언어를 이용해서 구현한다. 그 후 나온 여러 바이너리 파일을 다각도로 분석해만 엄밀한 효율성 검사가 될 것이며, 이는 향후 더 발전시켜야 할 사항이 된다.

SCADE를 이용한 서로 다른 두 node의 연결은 매우 직관적이며, SCADE는 안전한 타입 검사를 수행해 준다. 만약 두 node가 서로 다른 임베디드 시스템이고 입력값과 출력 값이 명확히 주어진다는 가정을 한다면 SCADE를 이용한 설계는 서로 다른 여러 시스템을 통합적으로 설계하고 이를 합칠 수 있는 방법론을 제공해 준다는 사실을 알 수 있다. 이는 앞서 실험한 2개의 node를 연결하는 부분에서 확인할 수 있는 내용이다.

위와 같이 자료 흐름 동기적 언어는 여러 복잡한 시스템을 설계하고 합치는 과정을 단순히 할 수 있다. 더욱이 이 언어는 자체적으로 안전한 타입 검사를 수행하기 때문에 설계할 시스템을 보다 안전하게 만들 수 있다. 따라서 이 언어는 비행기, 자동차 같은 safety critical 한 커다란 임베디드 시스템을 설계하고 검증하는데 적합한 방법론을 제공해 준다는 것을 확인할 수 있다.

5. 참고문헌

[1] Peter B. Ladkin, "The Ariane 5 Accident: A Programming Problem?", http://www.rvs.uni-bielefeld.de/publications/Reports/ariane.html#References

[2] Harry Goldstein, "Who Killed the Virtual Case File?", IEEE Spectrum September 2005.

[3] 조지호, "보안운영체제의 검증을 위한 정형기법 및 도구에 관한 연구", 2003년 한국정보과학회 가을 학술발표논문집 Vol. 30. No. 2.

[4] 조승모, 차성덕, "LUSTRE를 이용한 SCR 명세의 구

현," 정보과학회논문지(B) 제 26 권 제 2 호(1999.2).
[5] 양진석, 최진영, "ESTEREL을 이용한 RTOS Scheduler의 검증 및 구현", 2004년도 한국정보과학회 가을 학술발표논문집 Vol. 31, No.2.
[6] 양진석, 최진영, "Esterel 기반 임베디드 소프트웨어의 신뢰성 향상을 위한 개발 기법", 한국정보과학회 2005 논문집 Vol. 32, No.1(B).
[7] N. Halbwachs, P. Caspi, P.Raymond, and D. Pilaud, "The synchronous dataflow programming language Lustre," Proceedings of the IEEE, 79(9):1305-1320, September 1991.
[8] P. Caspi, D. Pilaud, N. Halbwachs, and J. Plaice, "Lustre: a declarative language for programming synchronous systems," In 14th ACM Symposium on Principles of Programming Languages, POPL'87, Munchen, January 1987.
[9] A. Bouajjani, J. C. Fernadez, and N. Halbwachs, "On the verification of safety properties," Tech. Rep. SPECTRE L12, IMAG, Grenoble, France, Mar. 1990
[10] D. Pilaud and N. Halbwachs, "From a synchronous declarative language to a temporal logic dealing with multiform time," in Symp. on Formal Techniques in Real-Time and Fault-Tolerant Systems, LNCS 331, M. Joseph, Ed. New York: Springer Verlag, Sept. 1998
[11] Nicolas HALBWACHS, Pascal RAYMOND, "A TUTORIAL OF LUSTRE", http://www.cs.chalmers.se/ComputingScience/Education/Courses/form/Papers/LustreTutorial.ps
[12] E. M. Clarke, E. A. Emerson, and A. P. Sistla, "Automatic verification of finite-state concurrent systems using temporal logic specification," TOPLAS, vol. 8, no. 2, 1986.
[13] R. L. Schwartz, P. M. Melliar-Smith, and F. H. Vogt, "An interval logic for higher-level temporal reasonning: Language definition and examples," Res. Rep. CSL-138, Computer Science Lab., SRI International, Feb. 1983.
[14] 강인혜, 양진석 공저, "초보자를 위한 Esterel 프로그래밍", 홍릉과학출판사