과전류 차단 제어 시스템 소프트웨어의 정형적 설계 및 검증

김진현*. 최진영*. 송호엽[0]

고려 대학교 컴퓨터학과*, 한우테크[0]
{jhkim, choi}@formal.korea.ac.kr*, hys1923@hanmail.net[0]

Design and Verification of Over-Current Controlling Systems Software

Jin-Hyun Kim*, Jin-Young Choi*,Ho-Yop Song[0]
Department of Computer Science Korea University*, Hanwoo Tech[0]

요 약

전력 시스템 가운데 과전류 차단 시스템은 전력 시스템의 보호를 위해 중요한 내장형 시스템이다. 이러한 시스템은 꼭 필요한 작동을 해야 하는 mission-critical 시스템이라 볼 수 있다. 이러한 mission-critical 시스템에 내장되는 소프트웨어의 개발은 요구 사항 분석 및 설계, 개발 단계에서 시스템의 신뢰도를 높이는 것이 매우 중요하다. 본 논문에서 이러한 시스템의 소프트웨어의 설계가 요구사항과 일치하는지를 확인하고 요구된 성질을 만족하는 지를 검증하기 위해 회로 설계에 적합한 언어인 ESTEREL 과 모델 체킹 도구를 이용하고 설계 및 검증 기법을 구현함으로 설계의 안정성과 정확성 및 설계에 대한 더 정확한 이해와 분석을 가능케 하는 설계 방법을 구현하는데 목적을 두고 있다.

1. 서론

전력 시스템과 같은 기간 시설 가운데 특정 과전류를 차단하는 과전류 차단 시스템은 전력 시스템을 보호해주는 중요한 내장형 시스템이다. 이러한 시스템은 특정 조건을 만족하며 요구된 동작을 정상적으로 완료해야 하는 mission-critical 시스템이라고 볼 수 있다. 이러한 시스템에 들어가는 회로는 과거 Verilog 나 VHDL 을 설계하여 하드웨어로 직접 만들었다. 하지만 이것은 하드웨어의 재구성이나 유지보수에 대한 융통성을 제공하지 못하였다. 따라서 근래에 들어서 이를 소프트웨어를 통한 컴퓨터 제어를 통해 구현하였다. 하지만 이러한 mission-critical 시스템 소프트웨어의 개발은 요구 사항 분석 및 설계, 개발 단계에서 시스템의 신뢰도를 높이는 것이 매우 중요하다. 이러한 신뢰도가 높은 시스템을 설계하기 위해 시스템의 요구 명세를 시각적으로 구현하고 이렇게 시각적 요구 명세서를 이용자의 요구 사항과 일치한지를 확인하며 또한 요구된 성질을 지니고 있는 지를 검증하는 것이 매우 중요하다. 본 논문에서는 이러한 내장형 소프트웨어 회로의 설계 및 검증을 정형기법을 도입하여 설계함으로 시스템의 신뢰도를 높이고자 한다. 정형 기법은 설계 단계에서 특정 오류를 찾아내어 뒤 따르는 과정에서 이러한 설계를 기반으로 완제품을 만들어 냄으로, 제품을 만들고 테스트하는 시간을 대폭 단축시킬 수 있게 된다. 또한 정형 기법을 도입하여 적용한 설계는 시스템이 지닐 수 있는 모든 상태를 만들어 분석하기 때문에 테스트 케이스에 의해 테스팅 된 시스템보다 훨씬 더 나은 정확한 설계를 가능케 한다. 이 때문에 완벽한 설계도를 바탕으로 출시된 제품의 오류 발견 가능성을 최대한으로 줄 일 수 있다. 본 논문에서는 정형 명세 언어인 ESTEREL[1]을 이용하여 회로를 명세하고 ESTEREL 의 검증 도구인 XES 를 이용하여 시스템의 Validation 을 시행하고 정형 기법 중에 하나인 모델 체킹(model checking)[2] 을 ESTEREL 도구의 일부인 XEVE[3]와 모델 체킹 툴로 잘 알려진 VIS[4]를 이용하여 모델 체킹을 함으로 설계된 회로가 사용자의 요구를 시스템이 이를 수 있는 어떤 상태에서도 충족시키는가를 실험할 것이다.

본 논문의 2 장에서는 정형기법과 정형명세 언어인 ESTEREL 과 관련된 검증 도구를 설명할 것이며 3 장에서는 정형검증 방법인 모델 체킹 기법과 그 도구인 VIS 를 소개할 것이다. 4 장에서는 case study 로 실제 과전류 차단 시스템 회로의 소프트웨어를 정형기법을 도입 설계한 실례를 보일 것이다. 5 장에서는 결론 및 향후 연구 방향을 기술 할 것이다.

2. 정형기법

정형기법(Formal Method)[2]은 소프트웨어 공학의 일종으로 오류가 없는 시스템을 설계하여 시스템의 신뢰도를 높이려는데 그 목적이 있다. 정형 기법은 시스템의 구성 요소를 수학적 객체로 취급하며 이러한 객체의 성질을 동적으로 묘사하고 예측하기 위해 수학적 모델을 제공한다. 이러한 수학적 기호를 사용함으로 시스템의 명세를 작성하는데 있어서 자연어가 일으킬 수 있는 애매모호함이나 불확실성을 최소한으로 줄일 수 있고, 설계된 시스템이 사용자의 요구사항과 동일한지 수학적 성질을 이용하여 증명할 수 있다. 그러므로 설계 단계 이후에 나타날 수 있는 오류를 미연에 방지할 수 있다. 정형 기법은 크게 두 가지 범주로 나누어 정형명세(Formal Specification) 과 정형검증(Formal Verification) 이 있다. 정형명세는 시스템이 달성해야 하는 요구 조건과 그러한 요구 사항을 묘사하는데 그 목적이 있다. 여기서는 수학적 모델링 언어가 사용되기 때문에 자연어의 모호함을 최대한으로 줄일 수 있다. 또한 정형 검증은 명세가 정확한지, 그리고 그러한 설계가 사용자의 요구사항을 만족하는지 검사하는 것을 의미한다. 이 단계에서는 설계도가 묘사한 시스템이 이를 수 있는 모든 상태를 분석함으로 특정 오류나 요구사항을 만족 못 시키는 상태를 발견할 수 있다.

2.1 정형명세 언어인 ESTEREL의 소개 및 관련 검증도구

ESTEREL 은 Reactive 시스템의 동기적(synchronous) 프로그램을 작성하는데 사용되는 동기적 언어(synchronous language) 이다. ESTEREL 은 일종의 결정적 reactive 시스템을 프로그래밍하는 정형적 의미를 지닌 언어로써 동시적인 입력 set 을 기다리고 출력을 계산하고 생산하는 것으로 입력에 대한 반응을 하고 정지한 다음, 다음 새로운 입력을 기다린다. ESTEREL 은 동기적 가설(synchronous hypothesis) 을 기반으로 하기 때문에 입력 set 에 대한 모든 반응은 시간을 소모하지 않고 instantaneous 한 것으로 간주된다. ESTEREL 의 프로그램 모델은 컴퍼넌트 혹은 모듈의 명세하는 것과 이를 평행하게 실행하는 것으로 이루어져 있다. 모듈들은 시그널을 통하여 각각의 모듈끼리 혹은 외부 세계와 통신을 한다. 이러한 시그널은 임의의 타입의 값을 지닐 수 있으며 전체 모듈에게 broadcasting 된다. ESTEREL 은 결정적 행위만을 명세할 수 있게 한다. ESTEREL 컴파일러는 자동적으로 복잡하고 평행한 모듈을 인터리빙(interleaving)으로 수행한다. 이것은 finite state machine 으로 구현되어지며 또한 전이에 대한 수행 시간이 알려진다면 모든 반응에 의한 최대 시간의 양은 정해질 수 있다. ESTEREL 에서 선점(Pre-emption) 연산자는 인터럽트나 watchdog 과 같은 행위를 명세할 수 있다. ESTEREL 의 구조는

표현력이 풍부하고 프로그래머가 입력 이벤트에 대한 요구된 반응을 정확하고 모호하지 않게 명세 한다. 예를 들어. 자연어로 명세된 요구 명세는 다음과 같다.

두 입력 A 와 B 을 받을 때, 출력 O 를 발산하게 된다.
입력 R 을 받을 경우는 언제나 reset 을 하게 된다.

이것을 ESTEREL 로 나타내면

```
loop
     [ await A || await B] ;
     emit O
each R
```

위 형태는 imperative 스타일의 프로그램이다. await A 는 A 시그널을 기다리고 즉시 종료하게 된다. ";"은 일렬적 수행을 의미하고 "||" 는 병렬적 수행을 의미한다. 병렬적 수행은 모든 가지가 모두 끝나야만 다음 명령을 수행할 수 있다. "loop ...each R" 문장은 선정형 문장으로 R 시그널이 나올 때 마다 내부 문장들은 현재 어떤 상태에 있든지 재 수행하게 된다.
ESTEREL 은 갖가지 형태의 중간 코드를 가지고 있다. 모의 실험할 수 있는 C 코드를 만들며 이는 xes 라는 도구로 모의 실험하여 시스템의 validation 을 수행할 수 있다. 또한 xeve 라는 도구는 ESTEREL 로 만들어진 코드의 출력을 검사함으로 정형검증을 시행한다. 즉 시스템이 만족해야 하는 요구 사항을 시제 논리로 표현하여 ESTEREL 코드로 바꾸어 원래의 ESTEREL 설계와 병렬적으로 수행시킴으로 특정 출력을 확인함으로 시스템이 요구 조건에 만족하는지를 확인 할 수 있다.

3. 정형검증 기법인 모델 체킹 및 VIS 소개

모델 체킹(Model Checking)은 유한 상태 동시 시스템을 검증하기 위한 오토마타 기반의 기술이다. 이것은 모의실험, 테스팅 및 연역적 추리(deductive reasoning)를 기반으로 문제를 해결하는 기존의 방법보다 나은 몇 가지 이점을 가지고 있다. 이것은 회로 설계나 통신 프로토콜을 검증하는 실용적인 방법 가운데 하나이다. 모델 체킹은 시스템을 오토마타로 형태의 표현할 수 있는 언어로 표현하고, 이 시스템이 만족해야 하는 요구 사항을 시제 논리로 표현하여 이 시스템의 도달할 수 있는 모든 상태에서 요구된 시제 논리를 만족하는가를 알아내는 검증 기법이다. 특히 본 논문에서 소개하는 VIS(Verification Interacting with Synthesis) 는 유한 상태 하드웨어 시스템의 검증(verification), 모의 실험(simulation) 및 합성(synthesis)을 통합한 도구이다. VIS 는 Verilog 를 front end 언어로 사용하고 적당한 CTL 모델 체킹, language emptiness, combinational 및 sequential equivalence 체킹, cycle-based simulation 및 계층적 합성을 지원한다. VIS 는 대화형 명령어 인터페이스(interactive command interface)와 일괄 모드(batch mode) 모두를 지원한다. VIS 의 두드러진 특징들을 소개하자면 다음과 같다.

Verilog front end : VIS 는 BLIF-MV 라는 중간 포맷 위에 운용되는데, 이것은 SIS 에 의해 받아들여지는 논리 합성을 위한 중간 포맷이 BLIF 의 확장이다. VIS 는 Verilog 를 BLIF-MV 로 바꾸어 주는 독립된 컴파일러인 VL2MV 라 불리는 컴파일러를 가지고 있는데 이것은 합성 가능한 Verilog 의 부분집합을 지원한다. VL2MV 는 실험된 결과의 관점에서 정의된 Verilog 의 행위를 포함하고 있는 상호 작용하는(interacting) 유한 상태 기계의 집합을 추출한다. Verilog 에 추가된 두 가지 새로운 특징은 다음과 같다.

1. 비결정적. 비결정적 생성, $ND, 는 wire 변수에 대한 비결정성을 명세하기 위해 첨가되었다. 즉 이것은 VIS 에서 비결정성을 나타내기 위한 유일한 옳은 방법이다.
2. Symbolic variable. 때로 변수들의 값을 명시적(explicitly)하게 명세 하는 것보다는 symbolically 하게 변수들의 값을 명세하거나 검사할 수 있어야 한다. VL2MV 는 Verilog 를 C 프로그램 언어에서와 비슷하게 열거형 형태·메커니즘을 사용하는 symbolic 변수를 허용하도록 확장하였다.

Hierarchy and initialization : BLIF-MV 가 VIS 내로 읽혀지면, 이것은 하위 모듈로 차례로구성된 모듈 트리로서 계층적으로 정렬된다. 모의실험과 검증은 계층의 subtree 에서 수행될 수 있다.

Interaction with synthesis : VIS 는 SIS 와 상호 작용하면서 BLIF 포맷을 읽거나 쓰면서 현재의 논리를 최적화 할 수 있다. Synthesis 는 계층의 어느 노드 상에서도 수행할 수 있다.

Abstraction : 사람이하는 추상화는 추상화에 대한 변수의 이름을 포함한 파일을 통해 수행될 수 있다. 파일 내의 나타나는 각 변수에 대해, 이 변수에 의해 미리 유도된 모든 노드들을 유도하기 위해 새로운 원시 입력 노드(primary input node)가 만들어진다. 효과적으로 하나의 net 을 추상화하는 것은 각 clock cycle 에서 그 값의 범위 내의 모든 값을 가지도록 허용한다.

Fair CTL model checking 과 language emptiness check : VIS 는 부치(Buchi) fairness 하의 fair CTL 모델 체킹을 시행한다. 그에 더하여 VIS 는 EG true 식(formula)의 모델 체킹을 통해 language emptiness 를 수행한다. 설계 언어는 fairness constraint 를 위반하지 않는 도달 가능한 state 의 set 상의 sequence 에 의해 주어진다. Language emptiness check 은 시스템의 다른 컴포넌트로서 잘못된 행위를 표현함으로 language containment 를 수행할 수 있다.
만약 모델 체킹 또는 language emptiness 가 위배된 경우, VIS 는 반례와 함께 실패를 알려준다. 즉 시스템 내의 행위는 성질(property)를 만족하지 않는다. 이것을 debug trace 라 한다. Debug trace 들은 fair cycle 로의 path 와 CTL 식을 위반한 state 들의 집합을 나열한다.

Equivalence checking : VIS 는 두 설계의 combinational equivalence 를 검사하는 능력을 가지고 있다. Combinational equivalence 의 주요 사용은 network 의 부분들을 합성할 때, 명료한 검사를 제공한다. 또한 VIS 는 두 설계의 sequential equivalence 를 테스트하는 능력을 가지고 있다. Sequential verification 은 finite state machine 을 만들고, 두 상응하는 결과 값이 다른 state 가 product machine 의 초기 상태들의 집합으로로부터 도달 가능한지를 검사함으로 시행된다. 이것이 일어나면 debug trace 가 주어진다. Combinational 과 sequential verification 모두 BDD 기반 루틴을 사용해 구현되게 된다.

Simulation : VIS 는 BDD 테크닉을 이용하는 cycle-based simulator 행태의 전통적 design verification 을 제공한다.

4. Case Study

본 논문에서 고려하고 있는 과전류 차단 시스템은 전력 시스템의 과 전류를 흐름을 감지하고 차단하는 시스템의 소프트웨어이다. 그러한 소프트웨어로 만들어질 회로 가운데 일부는 그림 1 과 같다.

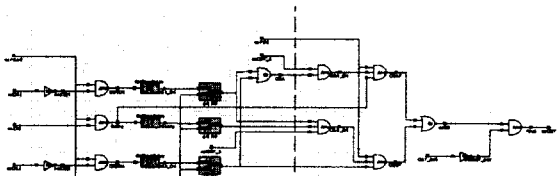

그림 1. Out of Step Tripping Logic

그리고 이 회로의 일부를 ESTEREL 언어로 표현하면 그림 2 와 같다. 이러한 ESTEREL 명세는 다음 두 가지 방법으로 모델 체킹을 통한 검증시행 할 수 있다.

1. CTL 형태의 시제 논리로 표현된 요구 사항을 ESTEREL 언어로 바꾸어 이것을 설계된 회로의 명세와 평행하게 수행시킴으로 요구 사항을 충족하는지를 출력 값을 통해 검증한다.
2. ESTEREL 로 표현된 회로 설계를 회로를 위한 확장을 거친 후, 이를 VIS 의 입력을 바꾼 후, CTL 형태의 시제 논리로 바꾸어 모델 체킹을 수행한다.[5]

첫 번째 방법은 XEVE 를 이용하여 시스템이 만족해야 하는 요구사항을 표현한 시제 논리가 회로 설계가 만족하는지를 관찰하게 된다. 이 때, 시스템이 이러한 논리를 만족하지 않는다면, 만족하지 않음을 보고하게 된다. 이것은 다음과 같은 단계를 거치게 된다.

I. ESTEREL 로 설계된 회로가 만족해야 하는 성질을 CTL 형태의 시제논리로 표현한다.
II. 이를 tl2strl 이라는 도구를 이용하여 ESTEREL 형태로 표현한다.
III. 이를 시스템 회로를 명세한 설계도와 동시에 수행시

```
module onTimeDelay_2 :
input I : boolean;
output O : boolean;
var s:=0 : integer in
loop
    present I then
        trap OTD in
            loop
                emit O(false)
            each tick
        ||
            loop
                if ?I = true then
                    s := s + 1;
                    if s = 2 then
                        emit O(true);
                        s := 0;
                        exit OTD
                    end if
                else
                    emit O(false);
                    s := 0;
                end if
            each tick
        end trap
    end present
each tick
end var
end module
```

그림 2. On Time Delay gate 의 ESTEREL 명세

킨다.

IV. 이때 검증하게 되는 것은 회로의 safety 혹은 fairness 등을 검증하게 된다.

V. 요구 사항을 만족하는지를 출력 값을 통해 관찰한다. 이 때, XEVE 를 이용하여 검증하게 된다.

두 번째 소개된 VIS 를 이용한 모델 체킹 방법은 다음과 같다.

I. 회로를 Boolean 변수를 갖는 ESTEREL 코드로 설계한다.

II. 이를 action table(프로그램 데이터의 계산을 수행)을 운용하는 circuit(프로그램 제어를 표현) 형태의 SC(Sequential circuit) 포맷으로 바꾼다.

III. 동일한 SC 형태의 확장된 형태로 바꾼다. 이것은 Boolean 변수의 data-path 를 가지고 있어, control nets 에 의해 유도되는 Boolean 연산자의 흐름을 표현하게 된다. 이것은 현재의 instance 내의 하나의 Boolean 변수에 대하여 다음 값을 계산하게 되고 action table 에 맞는 다음 assignment 를 대응시키게 된다.(scdata 를 이용)

IV. 이를 다시 blif 형태의 포맷을 바꾸고 VIS 의 입력 값으로 입력하게 된다.

V. 요구 사항을 CTL 형태의 포맷으로 바꾸어 검증을 시한다.

본 논문에서 예로 들고 있는 회로의 요구사항은 다음과 같다.

AG(!(osPSAB=1) -> !(osOSt=1))

회로 가운데 특정 신호 osPSAB 가 1 로 들어오지 않으면 어떤 경우에도 .출력 신호 osOST 는 1 이 아님을 뜻한다.그림 3 은 XEVE 로 검증을 시행하는 화면이다. 이를 VIS 를 이용하여 검증한 시행 결과는 그림 4 와 같다.

그림 3 에서는 osPSAB 의 신호가 없으면 결코 회로는 결코 osOST 를 내보내지 않는다. 또한 VIS 의 실행화면에 이 회로는 osPSAB 가 1 에 아니면 어떤 경우에도 osOST 는 1 을 내보내지 않고 0 을 내보낼 것임을 뜻한다.

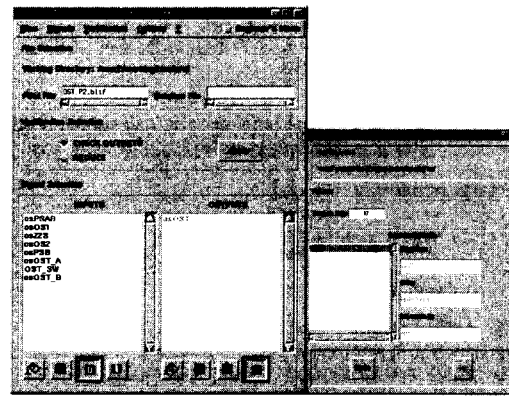

그림 3. XEVE 를 이용한 검증

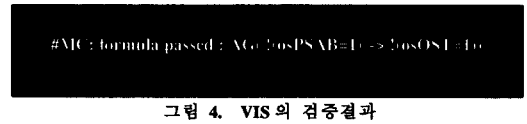

그림 4. VIS 의 검증결과

5. 결론 및 향후 연구 방향

본 논문에서는 기존의 verilog 나 VHDL 을 이용한 회로 설계가 아닌 함수형 언어인 ESTEREL 을 이용한 mission-critical 내장 시스템의 소프트에어 설계 및 검증에 대해서 논하였다. Verilog 나 VHDL 은 단지 하드웨어를 직접 만들 때 사용되나 여기의 ESTEREL 은 특히 내장형 시스템의 회로를 컴퓨터의 소프트웨어로 만들 때 사용될 수 다. 이 때 회로가 사용자의 요구사항에 따라 설계도가 올바르게 구현되었는지를 검증할 수 있게 된다. 또한 ESTEREL 은 C 나 JAVA 와 같은 언어로 바꿔어질 수 있으며 이 코드를 통해 모의 실험이나 내장형 소프트웨어를 직접 만들 수 있다는 가장 큰 장점이 있다.

또한 ESTEREL 은 모델체킹이나 equivalence checking 을 통한 검증을 시행할 수 있기 때문에 실제 회로를 소프트웨어로 만들 때 중요한 효율성 줄 수 있게 된다.

향후 연구 방향 및 과제로는 ESTEREL 을 이용한 mission-critical 시스템 회로의 검증의 다양한 기법 개발 및 실험, 사용자 위주의 용이한 검증의 기법의 개발이 연구 되어야 할 것이다.

6. 참고 문헌

[1] Gerard Berry. The Esterel v3 Language Primer V ersion 5.21. Centre de Mathematiques Appliqueses . April 6.1999

[2] Edmund M. Clake. Orna Grumberg. Doron A. Peled. Model Cheking. MIT press 19999

[3] Amar Bouali. XEVE: an Esterel Verification Environment. Technique report. Inria in France. 1997

[4] Adnan Aziz. Robert K.Rrayton. Gary D Hachtel. e *t.al.* VIS User's Manual. Univ. of California Berkeley.

[5] Alain Girault. Geread Berry. Circuit Generation for Verification of ESTEREL Programs. Technique report. Inria in France. 1997