시각적 컴포넌트 기반의 Embedded Application Software 개발 도구

김진현*, 김지영**, 이상한[0], 표창우** 김지인[0], 최진영*, 송호엽[00], 손혁수[00]
*고려대학교 컴퓨터학과,**홍익대학교 전자계산학과
[0]건국대학교 컴퓨터학과, [00]한우테크(주) 부설 기술연구소
e-mail : jhkim@formal.korea.ac.kr*, jykim@cs,hongik.ac.kr**,
leesh@cse.konkuk.ac.kr[0], pyo@cse.hongik.ac.kr**
jnkm5052@hitel.net[0], choi@formal.korea.ac.kr* ,
hys1923@hanmail.net, neoson@hitel.net[00]

Embedded Application Software Development Tool Based on Visual Component

Jin-hyun Kim*, ,Ji-Young Kim**, Sang-Han Lee[0]
Chang-Woo Pyo,** Ji-In Kim[0], Jin-Young Choi*, Ho-Yop Song[00]
*Dept. of Computer Science, Korea University,
**Dept. of Computer Engineering, Hongik University,
[0]Dept. of Computer Science, Konkuk University, [00]Hanwoo Tech

요 약

단위 자동화 시스템에 내장되어 운용되는 내장형 시스템의 응용 소프트웨어의 제작은 대부분 다년간 실무 경험을 갖고 있는 전문가들이 타겟 시스템의 목적과 기능 및 성능 명세를 작성하고 이를 구현함으로써, 개발 생산성이나 타겟 시스템의 신뢰성 향상이 곤란하다. 또한 개발의 마지막 단계에서 테스트를 통해 오류를 발견하게 되고 이로 인해 대부분 앞 단계로 돌아가 전체 설계를 재 검토해야만 하는 실정이다. 본 논문에서는 이러한 내장형 응용 소프트웨어 개발을 위해 편의성과 신뢰성을 향상시킬 수 있기 위해 소프트웨어 자동 검증을 위한 정형기법, 컴파일러 및 사용자 편의를 위한 GUI, 재사용 및 유지보수가 용이한 목적코드의 발생 등 전반적인 소프트웨어 공학적인 기술들을 이용하여 시각적 명세에 의한 실시간 내장형 응용 소프트웨어 개발 자동화 도구의 개발을 위한 기술의 개발을 목표로 삼고 기술되었다.

1. 서론

타겟 시스템 개발은 단위 자동화 기기라고 하는 하드웨어를 설계 제작하고, 이를 제어하는 응용 소프트웨어를 개발하여 실시간 운영체제(RTOS)와 결합시켜 제작된 하드웨어에 실행 파일을 내장한 후, 독자적인 실행으로 사용자가 요구한 타겟 시스템의 기능과 성능이 높은 신뢰성과 안전성을 갖고 동작하게 하는 것이다. 현재, 이러한 도구들은 개발된 응용 소프트웨어와 RTOS 를 결합시키는 과정과 결합된 실행 파일을 하드웨어에 내장하는 과정 및 내장된 실행 파일(즉, 응용 소프트웨어)의 동작 오류를 테스트할 수 있는 도구들이 대부분이라 할 수 있다. 즉, 현재 공급되고 있는 타겟 시스템 개발 도구들은 Unix 나 Windows 환경 등 독립적인 호스트 시스템에서 동작되며, 도구들의 일반적인 기능은 타겟 시스템에서 실행 가능한 코드를 생성해 주는 Cross-Compiler, 호스트에서 타겟을 실시간으로 제어하는 각종 쉘과 응용 소프트웨어(Embedded Application)의 소스를 디버깅해 주는 디버거 등이 포함되며, 이러한 도구들은 대부분 GUI 환경을 지원하여 개발자에게 편의성을 제공하고 있다. 그러나, 타겟 시스템 개발과정에서 발생되는 첫 번째 문제점은 개발하고자하는 타겟 시스템에 대하여, 관련 분야에서 다년간의 실무경험을 갖고 있는 전문가들이 타겟 시스템의 목적과 기능 및 성능 명세를 작성하고, 하드웨어와 응용 소프트웨어에 대한 분석 및 설계를 하여, 경험적인 기술을 기반으로 구현 개발함으로서, 개발 생산성이나 타겟 시스템의 신뢰성 향상이 곤란할뿐 아니라 현실적으로 소프트웨어 공학적으로 체계적인 접근이 불가능한 실정이다. 두 번째 문제점은 엔지니어링 과정에서 응용 소프트웨어와 하드웨어에 대한 분석·설계를 하여, 프로그램을 작성하고 현재 제공되고 있는 개발 도구들을 활용하여 타겟 시스템을 테스트해야만 응용 소프트웨어나 시스템의 오류를 발견하게 된다는 점이다. 경우에 따라서는 프로그램의 일부만을 간단히 수정하여 개발을 완료할 수도 있으나, 대부분은 앞 단계로 되돌아와 엔지니어링이나 분석·설계 등을 재검토해야만 하는 실정이다. 따라서, 본 연구개발과제에서는 소프트웨어 자동검증을 위한 정형기법, 컴파일러 및 사용자 편의를 위한 GUI, 재사용 및 유지보수가 용이한 목적코드의 발생 등 소프트웨어 공학적인 기반 기술들을 응용하여, <그림 1>과 같이 타겟 시스템 개발을 하고자 하는 실무 전문가가 호스트 컴퓨터의 화면에서 개발 단계별로 목적, 기능 및 성능에 대한

정확성과 신뢰성을 검증하면서, 타겟 시스템 개발을 수행하도록 지원하는 "시각적 명세에 의한 Real-Time Embedded Application 개발 자동화 도구"대해 설명하고자 한다. 본 논문의 2 장에서는 이러한 자동화 도구의 전체 전략에 대해 기술하며 3 장에서는 자동화 도구의 각 단계의 세부적인 점들을 단계별로 기술한다. 4 장에서는 국내외 관련 연구를 통해서 본 연구의 특징을 기술하며, 5 장에서는 결론 및 향후 연구 방향을 제시한다.

2. 시각적 명세에 의한 실시간 내장형 응용 소프트웨어

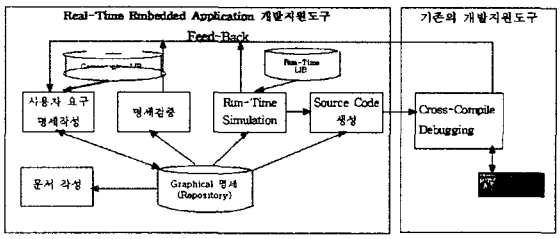

그림 1. 내장형 응용 소프트웨어 개발 도구

본 논문에서 목표로 삼은 실시간 내장형 응용 소프트웨어는 주로 전력 산업과 연관되어 있는 내장형 시스템의 회로 소프트웨어에 초점이 맞추어질 것이다. 전체 개발 도구의 흐름은 다음 <그림 1>과 같다. 개발도구의 전체 과정은 먼저 개발자의 요구사항을 도식화 명세 도구로 명세한 다음 이를 특정 형태의 중간 포맷으로 만든 다음 이를 검증 언어로 바꾸어 시스템의 다양한 특성을 검증하여 설계된 시스템이 요구자의 요구사항과 일치하는가를 검증한 다음 이를 C 코드와 같은 내장형 소프트웨어 언어로 변환되어 내장형 시스템으로 내장되게 된다. 따라서 시각적 도구가 사용자와 개발자에게 원활한 의사소통과 이해를 제공하게 되면 검증 도구를 통해 전체 시스템이 요구 사항에 맞게 설계 되었는지를 검증하게 된다. 이렇게 검증된 설계는 바로 목적코드로 변환되어 시스템에 내장될 수 있게 된다.

3. 내장형 응용 소프트웨어 개발 도구 개발의 세부단계

3.1 시각적 명세 및 사용자 인터페이스

본 논문의 내장형 응용 소프트웨어의 도구 이름은 DrawLogic 이다. DrawLogic 의 전체 화면은 <그림 2>와 같다 DrawLogic 은 회로 설계를 위한 프로그램으로 직관적인 사용자 인터페이스를 제공한다. 사용자에게 익숙하도록 널리 알려진 Windows 애플리케이션 환경을 채택하였으며, 회로의 종류를 쉽게 알아볼 수 있도록 모두 아이콘화 하였다. 뿐만 아니라 각 컴포넌트 간의 계층적인 구조를 한 눈에 파악할 수 있도록 인터페이스 왼쪽에는 워크스페이스 창을 두었다. <그림 3>과 같이 본 시스템은 필요한 모든 전자 회로를 아이콘화하여 하나의 도구상자로 묶어 놓았다. 따라서 사용자는 설계할 때 필요한 회로를 단지 마우스로 드래그해서 작업창에 끌어다 설계하면 된다. 다

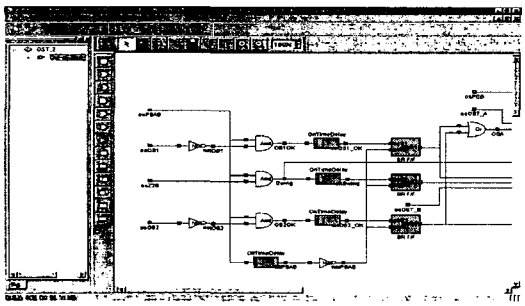

그림 2. DrawLogic 사용자 인터페이스

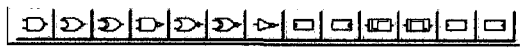

그림 3. 아이콘화된 회로

른 시스템과는 달리 누구나 쉽게 알아 볼 수 있도록 회로를 아이콘화 함으로써 작업의 효율성을 높였다. 하나의 회로에 해당하는 각 아이콘들은 내부적으로 전부 클래스화 시켰다. 이는 프로그램의 유지보수와 생산성 측면을 고려한 것이다. DrawLogic 은 회로 설계 시 벡터 드로잉 방식을 이용한다. 회로의 아이콘을 비트맵 방식으로 작업 창에 뿌려주는 것이 아니라, 모든 선분을 벡터 방식으로 표현함으로써 사용자가 필요한 부분을 축소 확대 시 전혀 이미지의 찌그러짐이나 계단현상과 같은 것이 나타나지 않는다.

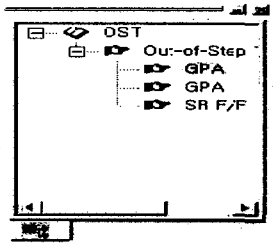

그림 4. Workspace 창

그리고 자주 사용되는 모듈을 컴포넌트 형태로 관리를 할 수 있다. 특정 회로를 컴포넌트로 등록한 후 나중에 언제든지 재사용이 가능함으로써 반복적인 작업을 할 필요가 없다. 또한 <그림 4>과 같이 컴포넌트를 관리할 수 있는 워크스페이스라는 창을 둠으로써 사용자들은 회로를 일목요연하게 설계할 수 있다. 본 시스템은 Named Buffer 라는 회로를 둠으로써 분량이 큰 회로를 복잡하게 선을 연결하지 않아도 간단하게 표현할 수 있다. Named Buffer 는 특정 부분의 회로를 입력으로 받고 이 Buffer 와 같은 이름을 사용하는 모든 Buffer 는 같은 값을 가지고 있기 때문에 회로의 설계를 쉽고 간단하게 할 수 있다. 그리고 DrawLogic 은 디자인 시 사용자가 범할 수 있는 심각한 오류를 자동으로 방지한다. 이는 회로의 설계가 끝난 후 회로의 타당성 검사를 하지 않고, 디자인과 동시에 회로의 오류를 검사함으로써 작업의 효율과 회로 설계의 검증을 한층 더 높일 수 있다.

3.2 설계의 검증

설계자와 사용자의 요구에 따라 회로가 설계되면 이 회로가 사용자의 요구에 따라 올바르게 설계되었는가를 검증할 필요가 있다. 본 논문에서는 시각적으로 명세한 회로를 중간 포맷으로 바꾸고 이를 검증도구에 맞는 언어로 바꾸어 이를 검증 기법 중 하나인 모델 체킹[1] 방법을 사용하여 검증할 것이다.

3.2.1 모델 체킹

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 몇가지 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. 본 논문에서는 검증언어로 Reactive 시스템 및 회로를 설계할 수 있는 ESTEREL[2] 언어를 사용할 것이며 이를 ESTEREL 의 검증 도구와 모델 체커인 VIS[3]를 사용하여 검증할 것이다. 시스템을 정형 명세한 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.[3]

첫 번째 방법은 ESTEREL 의 검증 도구인 XEVE[4]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 수행시킨다.
IV. 이때 검증하게 되는 것은 회로의 safety 혹은 fairness 등을 검증하게 된다.
V. 요구 사항을 만족하는지를 출력 값을 통해 판찰한다. 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 내 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 실시한다.

회로의 요구사항은 다음과 같을 수 있다.

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 5>은 XEVE 로 검증을 시행하는 화면이다. 이를 VIS 를 이용하여 검증한 시행 결과는 <그림 6>과 같다. <그림 5>에서는 osPSAB 의 신호가 없으면 회로는 결코 osOST 를 내보내지 않는다. 또한 VIS 의 실행화면에 이 회로는 osPSAB 가 1 이 아니면 어떤 경우에도 osOST 는 1 을 내보내지 않고 0 을 내보낼 것임을 뜻한다. 이렇게 검증된 설계는 ESTEREL 에서 바로 C 나 JAVA 의 병렬적이 프로세스의 제어 프로그램을 전화되어 사용될 수 있다. 이러한 방법으로 보다 신뢰성있는 안정된 시스템 제어의 내장형 소프트웨어를 설계할 수 있다

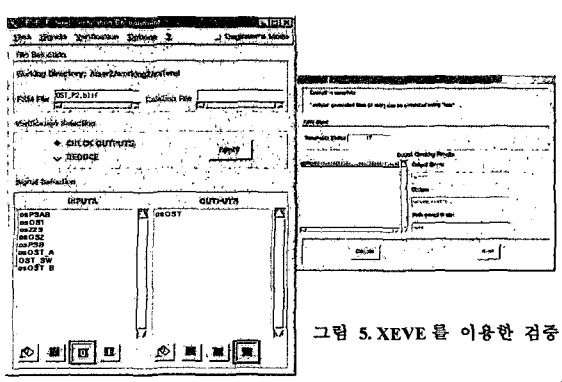

그림 5. XEVE 를 이용한 검증

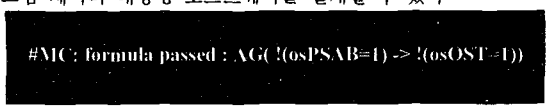

그림 6. VIS 의 검증결과

3.3 목적 코드 생성

C 코드 자동 발생기는 특정 목적의 회로를 논리 회로로 구현하는 대신 프로그램으로 구현하여 내장형 프로세서에서 실행되도록 하므로서 회로의 수정과 재사용을 용이하게 할 수 있다.

3.3.1 C 코드 자동 발생기 구현의 과정

회로도에서부터 C 코드 생성까지의 단계는 <그림 7>과 같다.

1 단계: 중간코드로부터 필요한 정보를 추출하는 단계

중간코드에는 C 코드 생성, 모의 실험, 검증 등의 모든 단계에서 필요한 정보들이 나타나게 된다. 이 중 C 코드 생성에 필요한 자료들만을 추출하여 추상구문트리(AST)를 구성하고 개체들의 번호를 키로 하여 심볼테이블에 개체들을 등록시킨다.

2 단계: 개체들의 연결관계를 그래프로 구성하는 단계

개체들의 실행순서를 결정하는데에 필요한 준비작업으로 추상구문트리에서 노드들의 연결관계에 대한 정보를 이용하여 그래프를 구성한다. 여기에서 노드는 개체번호가 되고 에지는 개체들간의 연결관계가 된다.

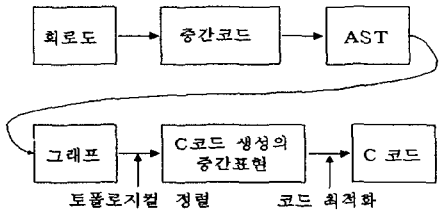

그림 7. 코드 자동 발생기 구현 과정

3 단계: 토폴로지컬 정렬 단계

개체들의 연결관계에 대한 정보만으로는 데이터의 흐름에 맞추어 개체들을 실행시킬 수 없으므로 실행순서에 대한 정보를 얻기 위한 작업이 필요하다. 이 정보는 토폴로지컬 정렬 방법으로 생성할 수 있다. 토폴로지컬 정렬은 회로도 말단의 입력이 되는 포트부터 시작하여 연결된 개체순서로 진행된다. 개체의 정렬은 자신의 선행자(predecessor)들의 정렬 후에 수행된다.

4 단계: C 코드 생성과정의 중간표현 작성 단계

개체들의 연결관계가 어떤 형태인지에 대한 정보를 구성한다. 구성되는 중간표현의 종류는 개체가 포트에서 포트로 연결되는 '포트 타입'과 게이트에서 포트로 연결되는 '컴포넌트 타입'으로 나눌 수 있다. 포트 타입인 경우, C 코드는 변수에서 변수로 값을 할당하는 형태가 되고, 컴포넌트 타입인 경우, 호출된 함수의 결과를 변수에 할당하는 형태가 된다. 개체가 포트에서 컴포넌트로 연결되는 경우는 중간표현의 한 타입으로 생성하지 않고 컴포넌트의 인자에 대한 정보를 입력하는 작업을 하게 된다.

5 단계: 코드 최적화 단계

개체들의 연결 관계는 포트에서 포트로만 값을 전달하고 다음 단계에서 전달되어진 값을 사용하는 경우가 있을 수 있다. 여

기에서 포트간의 값의 전달 부분은 <그림 8>과 같이 제거될 수 있다. 이러한 최적화 기법을 복사전파, 데드코드 삭제라고 한다. 중간표현에서는 복사전파, 데드코드삭제의 최적화를 적용하여 불필요한 코드를 삭제 시킨다.

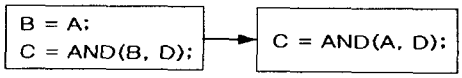

그림 8. 코드 최적화

6 단계: C 코드 생성 단계

최적화가 끝난 중간표현으로부터 C 코드를 생성해낸다.

회로도 전체를 하나의 함수로 정의하고 입출력 포트를 외부변수로(extern) 선언하여 함수 사용에 이용할 수 있도록 한다. 포트들은 변수로 선언하여 사용하는데 회로도에서 전달되는 데이타는 0 과 1 만의 값을 갖게 되므로 변수를 struct 내에서 비트 타입을 갖는 멤버로 정의하여 메모리의 사용을 줄이도록 한다. 게이트에 대해서는 매크로 함수를 정의하여 사용한다. 중간표현이 포트 타입인 경우 '변수 = 변수' 형태의 코드를 생성하고 컴포넌트 타입인 경우는 '변수 = 매크로함수(인자)' 형태의 코드를 생성한다.

3.3.2 C 코드 생성의 예

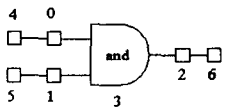

그림 9. 회로도

<그림 9>에 대해 함수 COM_and()를 생성하였고 입출력 변수들은 and.h 에서 외부변수(extern)로 선언하였다. 포트들은 and.c 에서 struct 안의 비트 타입을 갖는 멤버로 정의하였고 AND 게이트에 대해서는 매크로 함수를 정의하여 사용하였다. 생성된 프로그램은 <그림 10>과 같다.

```
#include  "and.h"
#define  COM_AND2(X, Y)  ((X) && (Y))

/* External Input Variables */
struct  S_and_I_tag  S_and_I;

/* External Output Variables */
struct  S_and_O_tag  S_and_O;

/* Local Port Variables */
struct {
  unsigned  int  Port_2 : 1;
} S_and_LP;

void  COM_and() {
  S_and_LP.PORT_2
    = COM_AND2(S_and_I.Port_5, S_and_I.Port_4);
  S_and_O.Port_6 = S_and_LP.Port_2;
}
```

그림 10. 그림 9 에 대해 생성된 C 코드

4. 관련 연구

국내에서는 타겟 시스템의 프로세서 또는 마이크로 제어 유닛과 관련된 하드웨어 연구는 LG 반도체 회사와 KAIST 등 산.학에서 연구되고 있다. 그러나 타겟 시스템의 하드웨어에 내장되는 실시간 RTOS 와 응용 소프트웨어 개발 환경 등에 대한 연구는 미미하다. 선진 외국에서 타겟 시스템에 내장을 위한 RTOS 와 구현개발 도구들은 WindRiver 사의 Tornado[1], Green Hills Software 사의 MULTI, Microware Systems 사의 FasTrak, Mentor Graphics 사의 Spectra, Integrated Systems 사의 pRISM+ 등이 있다. WindRiver 사의 Tornado 는 VxWorks 의 실시간 OS 를 지원하는 내장형 시스템의 대표적인 개발 도구이며, 호스트에서 수행하는 개발 지원 도구들은 쉘, 컴파일러, 디버거 등을 포함한다.

본 논문에서 제시하고 구현된 도구는 이러한 도구들의 장점을 수용하고 소프트웨어 공학의 개발 주기를 응용하여 개발되었다. 특히 시각적 설계 도구를 통해서 개발자와 사용자와의 원활한 의사 소통 및 이해를 가질 수 있게 한다. 또한 정형 기법의 검증기법은 시스템이 가질 수 있는 모든 상태를 파악하여 생길 수 있는 오류를 검증하고 이를 직접 사람의 손을 거치는 것이 아니라 컴퓨터가 내장형 언어로 바꿈으로써, 사람이 일으킬 수 있는 오류를 최소화하게 되고, 또한 설계에서의 보완이나 수정이 바로 내장형 소프트웨어로 바꾸어 질 수 있어 개발 및 유지보수의 시간을 단축시킬 수 있게 된다. 따라서 전체 소프트웨어 개발에 효용성을 증대시키게 된다. 또한 각 단계 별로 피드백이 용이함으로 개발 및 유지 보수 기능을 강화할 수 있으며 이러한 이유로 인해 전체 내장형 응용 소프트웨어의 개발 주기의 보다 효과적인 운용이 가능하다. 또한 독자 실행 가능한 C 코드는 다양한 플랫폼에서의 실행이 가능하게 함으로 특정 시스템에 독립적으로 실험 및 운용될 수 있기 때문에 코드의 재사용 및 실용성이 증대된다. 또한 중간 코드의 운용은 다른 JAVA 나 혹은 하드웨어 설계 언어인 Verilog 나 VHDL 로도 변형할 수 있게 만들 수 있기 때문에 실제 하드웨어를 만들 때에도 사용 가능하다는 장점을 가지고 있다.

5. 결론 및 향후 연구 방향

본 연구 과제를 통해 시각적 실시간 내장형 응용 소프트웨어를 만드는 개발 방법 및 도구를 구현하였다.

본 논문의 도구 개발에서 시각적 명세 도구는 먼저 사용자 편의 인터페이스 개발이 우선되어야 할 것이다. 또한 사용자의 요구를 명세할 수 있는 다양한 도구의 개발이 연구되어야 할 것이다. 본 도구 개발에서의 검증 방법은 사용자가 좀더 쉽게 사용할 수 있는 방법으로 제공되어야 할 것이다. 특히 컴포넌트 별 검증 뿐 아니라 전체 시스템의 검증 역시 중요한 연구 방향이 될 것이며, 더 많은 실험 예제 및 편리한 검증기법의 개발이 연구되어야 할 것이다. 코드 생성 과정에서는 코드의 재사용 및 유지보수가 용이해져야 한다. 현재의 시스템은 회로도 전체를 하나의 함수로 구현하였다. 이 방법으로는 회로도 내에서 동일한 컴포넌트를 여러번 사용하는 경우, 각각의 내부를 구성하는 포트들과 게이트 등의 코드를 다시 생성해야 한다. 이러한 작업은 중복되는 코드들을 발생시키게 되고 전체 회로의 재사용은 용이하지만 내부를 구성하는 일부 컴포넌트의 재사용은 어렵게 한다. 이러한 문제에 대한 개선 방안으로 중간코드의 정보 중 개체가 속하는 레벨과 컴포넌트 이름에 대한 정보를 사용한다. 개체들을 레벨별, 컴포넌트별로 구분하고 각각에 대해서 함수를 구성하여 코드를 생성해내는 방법을 사용할 수 있다. 그러나 이러한 방법으로 함수를 생성했을 경우 여러 레벨에서 사용되는 똑같은 컴포넌트들이 각각 함수로 발생되어 코드의 중복은 여전히 존재하게 된다. 이러한 중복되는 컴포넌트를 생성하지 않기 위한 다양한 기법의 연구가 진행되어 질 것이다.

참고문헌

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

[2] Gerard Berry. The Esterel v3 Language Primer Version 5.21. Centre de Mathematiques Appliqueses . April 6.1999

[3] Adnan Aziz. Robert K.Rrayton. Gary D Hachtel. et.al. VIS User' s Manual. Univ. of California Berkeley

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

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