Synchronous SpecCharts로부터 Synchronous VHDL 코드 생성기 설계

윤성조, 안성용, 이정아
조선대학교 전자계산학과
tcop92@rain.chosun.ac.kr, dis@rain.chosun.ac.kr, jeong@rain.chosun.ac.kr

Design of synchronous VHDL Code Generator from Synchronous SpecCharts

Seong-Jo Yun, Seong-Yong Ahn, Jeong-A Lee
Dept. Computer Science, Chosun University

요 약

현재 많은 내장형 시스템을 구현하기 위한 방법론으로 가상 프로토타입(VP)을 이용하고 있다. 본 논문에서는 가상 프로토타입을 이용하여 내장형 시스템의 설계 및 구현을 위해 사용되는 시스템 명세 언어인 SpecCharts로 명세된 시스템을 동기적 의미론에 만족하는 SpecCharts의 Subset을 규명하여 동기화 형태로 해당 명세를 변환시키고 이로부터 synchronous VHDL 코드로 생성 할 수 있는 방법을 설계하였다. 동기적 의미론을 만족시키기 위하여 비결정적인 추상적인 모델(NDAM)을 이용하여 SpecCharts로부터 VHDL 코드로 변환하는 방법을 제시하고, 변환된 VHDL 코드를 동기적 VHDL 코드로 변환하기 위하여 W. Baker에 의해 규명된 동기적 VHDL subset 적용하여 synchronous VHDL 코드를 생성하는 방법을 제안한다..

1. 개 요

현재 많은 내장형 시스템을 구현하기 위한 방법론으로 Virtual Prototyping(VP)을 이용하고 있다. VP로 설계된 시스템 요소들을 하드웨어 코드나 소프트웨어 코드로의 변환 부분이 필요하게 된다. 이 과정은 가상 현실로부터 해당 시스템의 정형명세로 이루어지는 시뮬레이션을 거쳐 요구사항에 관한 정형 명세를 검증 후 하드웨어 코드나 소프트웨어 코드 변환단계를 거쳐 합성 과정을 통해 실제 시스템과 동일한 기능을 수행하는 제품을 생성하게 해준다. VP은 특히 마이크로프로세서나 주문형 프로세스가 포함된 내장형 시스템의 설계 및 구현에 용이하다는 장점으로 생명 주기를 단축시키고 제품에 신뢰성을 높이고 있다.

본 논문에서는 시스템의 요구사항을 검증을 통하여 시스템 명세 언어인 SpecCharts로 명세된 형태를 동기적 의미론(semantics)을 만족하는 SpecCharts의 subset을 규명하는 것은 cycle-level 추상화가 가능한 모델이 존재함을 의미하고, 동기적 VHDL subset을 정의할 수 있게 한다. SpecCharts 명세를 동기적인 형태로 변환시키고, 이로부터 동기적 VHDL 코드를 생성할 수 있는 방법을 설계 한다.

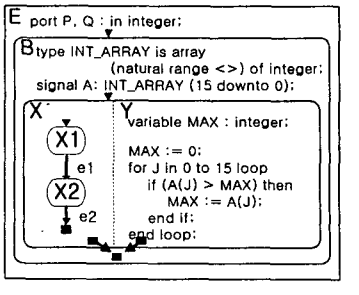
앞으로 2장에서는 SpecCharts를 고찰하고, 3장에서는 동기적 VHDL의 subset을, 4장에서는 변환된 VHDL 코드를 동기적 VHDL로 변환에 대하여, 5장에서는 결론 및 향후 방향에 대해서 언급한다.

2. SpecCharts

SpecChart는 내장형 시스템 명세를 위해서 D. Gajski가 제창한 것으로 프로그램-상태 기계(*Program-state machine;PSM*)와 VHDL를 결합한 형태로 PSM을 이용한 시스템 명세와 각 상태들에 대한 행위 기술을 VHDL로 표현할 수 있게 해준다.

SpecCharts가 지원하는 특징들은 다음과 같다.

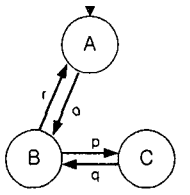
- 첫째, D.Harel이 주창한 StateCharts의 시각적인 효과와 VHDL처럼 textual하게 표현이 가능하다.
- 둘째, 행위의 계층성을 지원하여 순차적(sequential)이거나 병행적(Concurrent)한 행위를 기술할 수 있다.
- 셋째, 상태간의 전이를 TOC(transition on completion) arcs를 통해 지원한다. 수행 중 예외 상황을 TI(transition immediately) arcs를 통하여 지원한다.
- 넷째, 상태(프로세스)간의 통신을 공유 메모리(shared memory), 메시지 전달(message passing)을 통하여 지원한다.



<그림 1> SpecCharts의 표현법

또한, VHDL과 유사성을 가지고 있어서 다음과 같은 특성들을 지닌다. 프로그래밍 구성체(programming constructs)이고, 구조적인 계층성(Structural hierarchy)을 갖으며 동기화와 타이밍을 지원한다.

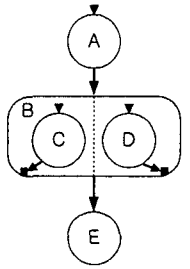
SpecCharts의 상태전이는 TOC와 TI arcs 두 행위(behavior)로 나타난다. TOC는 조건에 관계된 상태에 연결되어 있는 프로그램이 완결되었을 때 상태전이를 발생하게 하고, TI는 해당 조건이 만족하는 그 순간 상태전이를 발행하게 하는 차이를 갖는다.




```
behavior TOC_EX type sequential subbehaviors is
begin
  A : (TOC, o, B);
  B : (TOC, r, A), (TOC, p, C);
  C : (TOC, q, B);

  behavior A .....;
  behavior B .....;
  bdhavior C .....;
end TOC_EX;
```

(a) TOC arc를 이용한 상태 전이




```
behavior TI_EX type sequential subbehaviors is
begin
  A : (TOC, ture, B);
  B : (TOC, true, E);
  E : .....;

  behavior A .....;
  behavior B type concurrent subbehavior is
  begin
    C : (TI, true, stop);
    D : (TI, true, stop);

    behavior C .....;
    behavior D .....;
  end B;
  bdhavior E .....;
end TI_EX;
```

(b) TI arc를 이용한 상태 전이

<그림 2> SpecCharts의 상태전이와 behavioral 계층성

SpecCharts에서 계층성은 내포된 행위들(nested behaviors)을 통해 표현되어 지며, 각각의 행위들은 순차적(sequential)과 병행적(concurrent) 하위행위들(subbehaviors)로 분해할 수 있다. 그러므로, SpecCharts의 구조는 층계 구조 형태로 이루어져 있으며 각 층계들은 두 가지의 형태로 구성된다. 그 중 하나는 OR-level의 성격을 지닌 순차적 하부행위의 타입이 되고, 나머지 하나는 And-level의 성격인 병행적 내부행위의 타입을 갖는다. 층계 구조의 터미널 노드의 성격을 지닌 잎노드는 code라는 타입으로써 실제 수행되는 행위들이 기술되어 진다.

SpecCharts의 원래의 의미론(semantics)은 Statecharts의 discrete-semanics와 VHDL의 discrete-event semantics의 조합으로 이루어진다. VHDL-1076과 Statecharts의 semantics는 전달 이산 사건(propagation discrete event)에 근거하여 형성된다. 즉, Speccharts의 표현의 편리함은 시스템 표현 시에 언어의 문법적인 측면을 고려하였다고 말할 수 있다.

3. Synchronous VHDL subset 설정

synchronous VHDL은 기존의 full VHDL에서 동기적 의미론(synchronous semantics)를 만족시킬 목적으로 Wendell Baker에 의해 VHDL에 제한성을 부여하여 규명되었다. 그 이유로는 VHDL이나 Verilog 언어의 경우에는 discrete event 의미론으로 해석되는 경우에 동기적 의미론 성질을 표현하기가 어렵기 때문이다. Baker가 주장한 바에 의하면 유한 오토마타 VHDL 시뮬레이터를 통해 생성된 VHDL 코드와 실제의 전체의 VHDL 시뮬레이터를 통해 나온 결과를 동등하게 할 수 있다라고 하였다. 여기서 유한 오토마타 VHDL 시뮬레이터는 추상적(abstract) 시뮬레이터로 그 기능은 유한적인 메모리 공간에서 정해진 수행 시간 내에 동작하도록 하였다. synchronous VHDL subset은 유한 상태의 의미론을 기초로 생성된 의미있는 방법론이다.

synchronous VHDL subset을 규명하기 위하여 전체의 VHDL에 대한 4가지 제한 사항을 적용하여야 한다.

- 첫째, 이벤트들을 관리하는 시그널 큐들의 길이를 하나의 크기로 제한한다. 즉, VHDL상에서 waveform 대입문의 사용이나 전달(transport) 또는 관성(inertial) 지연(delay)이 임의의 상태들의 수를 적용하기 때문에 제약을 두어야 한다.
- 둘째, 프로세스 안의 순차적 코드는 스택을 사용하지 않아야 한다. 즉, 중첩된 함수(프로시저)나 재귀 호출, 동적 크기의 객체 타입(array type), 지역적인 정적(static) 객체들의 사용을 제한하여야 한다.
- 셋째, 동적 저장장소 할당 사용을 제한하여야 한다. 즉, 힙(heap) 영역을 사용하지 않아야 한다. VHDL 구문상의 access type과 new operator 같은 문장은 제한해야 한다.
- 넷째, 시그널 전달(propagation) 패스들을 반드시 인과관계(causal)로 구성하여야 한다. 즉, 무한적인 발산(oscillation)들을 허용하지 않아야 한다.

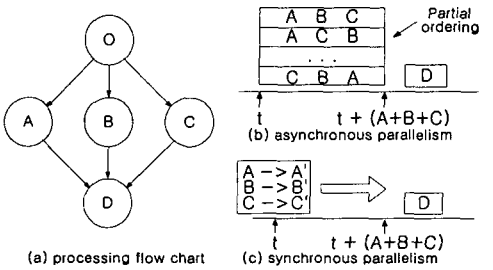
위 4가지 제한 사항은 크게 유한적인 메모리 요구 측면과 유한적인 반응(reaction) 계산(computation) 분야로 정리할 수 있다. 이 4가지 제한 사항을 통해 전체의 VHDL 구문에서 위에 제약들에 문제되는 구문들을 배제하고 synchronous VHDL subset을 설정하도록 한다.

4. synchronous VHDL로 변환기 설계

VHDL에서 지원하고 있는 병행처리 방법은 비동기적인 형태를 가진다. 그러하기 때문에 동기적 VHDL 코드를 구성하기 위해서는 VHDL의 discrete event 의미론을 만족할 수 있는 시뮬레이터를 구성해야 한다.

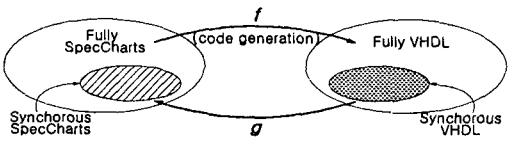
비동기적 병렬처리 방법론은 대기된 값들과 프로세스들의 리스트를 큐로 유지하여 순차적으로 처리해 나가는 방법을 사용한다. VHDL에서 병행처리는 비동기적 방법에 의해 처리되어 진다. 이는 event partial ordering을 통하지 않고서는 연계된 상태전이를 지원할 방법이 없기 때문이다.

동기적 병행처리 방법은 비결정성(non-determinism)을 허용하거나 제공하지 않는다. 이는 가능한 사건들이 단지 유한개의 수이고 각 상태들은 단지 명시적인 지연만이 존재하고 모든 계산은 즉각적으로 수행된다는 동기적 가설(synchrony hypothesis)에 기반한 방법론이다.



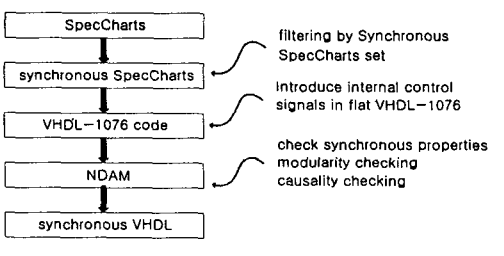
<그림 3> 비동기적과 동기적 병행처리 방법 비교

SpecCharts는 시스템 모델링을 할 수 있도록 구성되어 있으며 또한 VHDL 코드로 변환될 수 있도록 제작되어 있다. SpecCharts editor를 통하여 SpecCharts 코드를 생성하는데 Specsyn 프로그램을 이용하여 해당 SpecCharts로 구성한 시스템을 VHDL 코드로 변환시킬수 있다. 이렇게 변환된 코드는 일반적인 VHDL 구문과 동일한 형태를 취하고 있어 기존의 시스템 의미론을 맞추어 동기적인 VHDL 코드로 변환작업을 거쳐 생성된 중간 코드를 생성한다.



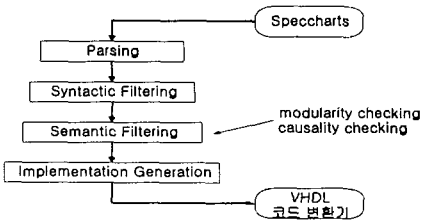
<그림 4> SpecCharts와 VHDL과의 관계

먼저 전체의 SpecCharts의 구문에서 동기적 VHDL로 생성될 수 있는 구문을 규명하여 동기적 SpecCharts subset을 규정하여야 한다. SpecCharts 상의 구문은 동기적 형태로 변환시키면 효율적으로 동기적 VHDL 구문으로 변환시킬 수 있다. 이렇게 생성된 동기적 구문을 올바르게 변환되었는지 검사하는 부분을 거쳐서 동기적 VHDL 구문을 생성할 수 있다. <그림6>에서 SpecCharts로부터 VHDL 코드가 자동 생성됨으로 원치 않는 코드 생성시에 해당 코드에 대한 검증작업이 필요로 하게 되기 때문에 중복적인 필터링을 수행한다.



<그림 5> 코드 생성기 수행기능도

그리고 동기적 VHDL로 변환하는데 구문(syntactic) 필터에는 W. Baker가 규명한 동기적 VHDL subset을 위한 제한 사항을 적용하여 동기적 VHDL 코드를 생성하여야 한다. 이렇게 생성된 VHDL 코드에서 동기적 VHDL 코드로 생성하는 과정을 NDAM(Non-deterministic Abstract Model)을 통하여 처리되어 진다.



<그림 6> synchronous SpecCharts filer 구조

5. 결론 및 향후 연구 방향

본 논문에서는 SpecCharts로 표기된 시스템을 Specsyn 도구를 통해 생성된 VHDL 코드를 의미론에 따라서 synchronous VHDL 코드로 변환하였다. SpecCharts로부터 동기적 SpecCharts subset을 찾기위하여 <그림4> 에서와 같이 SpecCharts와 VHDL의 동기적인 영역을 규명하였다. 전체적인 코드 생성기 구조는 최초의 SpecCharts로부터 동기적 SpecCharts 구하는 필터링을 하고 이를 VHDL 코드로 변환한 다음 NDAM을 이용하여 동기적 VHDL 코드로 변환하게 하였다. 또한 SpecCharts 상에서의 중복 필터링을 통해 동기적 SpecCharts를 생성하고 VHDL로 변환 후 다시 동기성 위배 여부 및 인과성, modularity를 재검사하여 동기적 변환을 한층 강화하도록 하였다. 이렇게 하여 정리되어진 동기적 SpecCharts 표기법이 시스템 디자이너들의 관점에서 일반화되면 시스템 설계단계부터 동기적 의미론에 부합되는 시스템을 설계할 수 있고, 동기적 VHDL 코드로의 변환 또한 더 효율적으로 수행될 수 있다.

향후 연구 방향으로는 SpecCharts를 통해 생성된 VHDL 코드가 시뮬레이션이나 합성 가능한 코드라고 증명할 수 없으므로 합성가능한 코드로 변환하는 작업이 필요하다. 이를 위하여 합성 가능한 VHDL 구문을 분석하여 의미론에 위배되지 않게 제한시켜 합성가능한 코드로 변환하는 문제를 해결한다면 동기적으로 수행하는 시스템을 설계하여 실제 합성까지의 공정을 가능하게 한다.

6. 참고 문헌

[1] D. D. Gajski, F. Vahid, S. Narayan and J. Gong, Specification and Design of Embedded Systems, Prentice Hall, 1994.

[2] W. Baker, An Application of a Synchronous/Reactive Semantics to the VHDL Language, M.S. Report, Department of Electrical Engineering and Computer Science, University of California, Berkeley, January 1993, UCB/ERL M93/10.

[3] IEEE Standartd VHDL Language Reference Maual, IEEE Std 1076-1987, Instituete of Electrical and Electronics Engineers Inc.,345 East 47th., New York, NY 10017 USA.

[4] D. Harel, "STATECHARTS: a Visual Formalism for Complex Systems", In Science of Computer Programming Vol. 8, north Holland, 1987, pages 231-274.