한국시뮬레이션학회 '99춘계학술대회 논문집 1999.4.17 동의대학교

하드웨어/소프트웨어 통합시뮬레이션을 위한 HDL모델의 자동 변환

김준경, 김탁곤

한국과학기술원, 전기.전자공학과
시스템 모델링 시뮬레이션 연구실
URL: http://smsl.kaist.ac.kr
E-mail: {jkkim@smslab, tkim@ee}.kaist.ac.kr

[ABSTRACT]

Codesign방법론은 하드웨어와 소프트웨어가 공존하는 시스템을 설계할 때 이들의 설계를 각각의 특성에 맞는 방법을 사용함으로써 효율적인 디자인방법을 제공한다. 전체 시스템의 동작 및 성능을 검증하기 위해서는 다른 방법으로 개발된 하드웨어와 소프트웨어를 같이 시뮬레이션해야 하는데 이를 통합시뮬레이션(Co-simulation)이라고 한다. 하드웨어와 소프트웨어를 개발하는 방법이 다르기 때문에 야기되는 통합의 문제점을 해결하기 위하여 DEVS(Discrete Event System Specification)에 기반한 중간단계형태를 통한 변환방법론을 제시하고 이를 사용하여 C++모델과 Verilog HDL모델간의 통합시뮬레이션을 구현함으로써 효용을 보이고자 한다.

1. 서론

설계하고자 하는 시스템의 특성에 따라 모델링 및 시뮬레이션 방법을 선택적으로 사용하여 결과를 얻는 것이 효율적이다. 그러나 모든 부분들을 통합하여 전체적인 시뮬레이션이 수행되어야 할 경우 각기 다르게 모델링된 부분이 통합되어야 전체적인 시뮬레이션이 가능하다. 이 논문에서는 정확한 의미론에 입각한 중간단계형태를 통한 변환기법을 사용하여 이러한 문제를 해결해 보고자 한다.

그림 1과 같이 범용 프로세서에서 원하는 응용프로그램을 수행할 때 성능에 크게 영향을 미치는 부분들을 하드웨어로 구성함으로써 전체적인 성능을 향상시켜보고자 하는 것이 HW/SW Mixed 시스템의 목표이다. Codesign방법론[1]은 이러한 HW/SW Mixed 시스템을 개발하기 위한 방법론으로써 그림 2에서 볼 수 있는 바와 같이 세 가지 부분으로 나뉜다. 하드웨어, 소프트웨어, 그리고 이 둘 사이의 인터페이스를 고려하여 각각 다른 설계방법을 사용하고 마지막 단계인 통합시뮬레이션(co-simulation)단계에서

전체적인 시뮬레이션을 수행하여 원하는 결과를 얻어낸다. 모델링 방법이 다르기 때문에 발생하는 문제점은 이 통합시뮬레이션 단계에서 해결되어야 한다.

이러한 문제를 해결하기 위하여 DEVS (Discrete Event System Specification)에 기반한 중간 단계 형태를 정의하고 이 중간 단계 형태를 사용하여 여러 가지 하드웨어 모듈을 상호 교환할 수 있는 환경을 제안하였다.

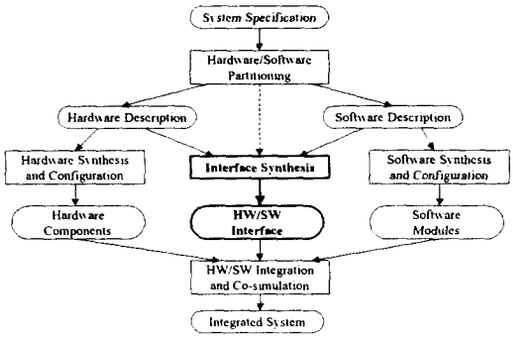

그림 2. Codesign Framework[1]

2. 이론적 배경

2.1 Hardware를 나타내기 위한 표현력

하드웨어는 모델링하는 단계에서부터 구현되는 단계까지 많은 추상화 단계의 표현력을 지원해야 한다. 그림 3에 하드웨어의 여러 추상화 단계를 표현하였다. 우선 memory가 있는지에 따라 combinational과 sequential의 구분이 되어 있다. combinational 회로를 표현하기 위해서는 boolean 방정식으로 표현된 함수를 처리할 수 있어야 한다. sequential 회로를 표현하기 위해서는 하드웨어의 상태

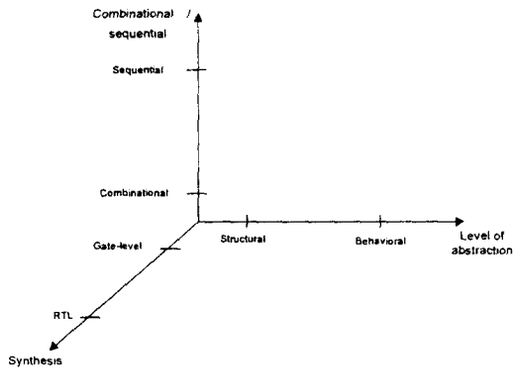

그림 3. 하드웨어의 여러 측면

를 저장할 수 있어야 하고 그 상태간의 전이를 표현할 수 있어야 한다. Synthesis 측면에서 보면 gate level과 RT-level이 있다. RT-level의 하드웨어를 표현하기 위해서는 상태를 저장하는 것은 물론, 보다 높은 레벨의 연산을 지원하여야 한다. Gate-level의 하드웨어를 표현하기 위해서는 역시 boolean 방정식의 함수처리 능력뿐만 아니라 propagation delay와 같은 상태전이함수의 시간적인 특성을 나타내어야 한다. 추상화 단계라는 측면에서 보면 하나의 모듈이 하나의 전체적인 하드웨어를 표현하는 behavioral level과 내부적으로 보다 작은 primitive들이 연결되어 전체 시스템을 이루는 structural level 모두를 표현할 수 있어야 한다.

이 모든 표현능력을 수용하는 형식론으로 DEVS가 있다. 우리는 DEVS를 이용하여 이러한 하드웨어를 기술한 형식을 변환하도록 하였다.

2.2 DEVS 형식론

Zeigler에 의해 제안된 DEVS형식론[3]은 이산사건 시스템을 modular/hierarchical하게 모델링할 수 있는 수학적인 기반을 제공한

다. DEVS모델은 atomic모델과 coupled model로 나누어진다.

Atomic 모델은 상태 집합, 입력 사건 집합, 출력 사건 집합, 전이함수, 출력함수 및 시간 전진함수로 구성되며 수학적인 정의는 다음과 같다.

Definition 1. Atomic model

$M=\langle X, Y, S, \delta_{int}, \delta_{ext}, \lambda, ta\rangle$;
X : *input events set;*
Y : *output events set;*
X : *sequential states set;*
δ_{int} : $S \rightarrow S$: *internal transition function;*
δ_{ext} : $Q \times X \rightarrow S$: *external transition function;*
$Q=\{(s, e) | s \in S, 0 \le e \le ta(s)\}$
λ : $S \rightarrow Y$: *output function;*
ta : $S \rightarrow Real$: *time advance function*

Coupled 모델은 components간의 관계를 나타낸 것으로 atomic모델과 coupled모델을 사용하면 시스템을 modular/hierarchical하게 설계할 수 있다. 수학적인 정의는 다음과 같다.

Definition 2. Coupled model

$DN=\langle X, Y, M, EIC, EOC, IC, SELECT\rangle$
X : *input events set;*
Y : *output events set;*
M : *set of all component models in DEVS;*
$EIC \subseteq X \times \bigcup_i X_i$: *external input coupling relation;*
$EOC \subseteq \bigcup_i Y_i \times Y$: *external output coupling relation;*
$IC \subseteq \bigcup_i Y_i \times \bigcup_i X_i$: *internal coupling relation;*
$SELECT$: $2^M - \phi \rightarrow M$: *tie-breaking selector;*

2.3 DEVS의 phase표현형태

DEVS로 모든 이산사건 시스템을 모델링할 수 있지만 모델링의 편의를 위하여 몇 가지 변수개념을 도입하는 것이 좋을 때가 있다. 가령 4개의 8bit길이를 갖는 레지스터가 있는 프로세서를 생각해 보자. 정의에 의하면 이 프로세서는 $2^8*2^8*2^8*2^8$개의 상태수를 갖게 된다. 이 모든 상태를 사용하여서는 프로세서를 모델링하기가 쉽지도 않는 것은 물론 모델링을 한다고 해도 지나치게 많은 상태수와 그에 따른 특성함수가 너무 복잡해지기 때문에 시뮬레이션의 성능에 문제가 생기게 된다. 그러므로 phase라는 개념을 도입하여 이 문제를 해결해 보도록 한다.

Definition 3. phase

A macro state which is equivalent to

(i) insensitive input event and

(ii) internal state transition with no output and time advance zero

즉 phase란 DEVS모델로는 별도의 상태로 구분되었어야 하는 것으로써 위의 조건에 대해 하나의 phase로 mapping되는 상태들의 집합이라고 할 수 있다. 중요한 점은 이렇게 DEVS모델을 phase에 기반한 형태로 바꾸었을 때 기존의 DEVS모델의 표현력에 변화가 생겨서는 안 된다는 것이다.

2.3.1 첫 번째 조건 : latch/condition

첫 번째 조건은 insensitive input event가 발생할 경우 phase transition을 하지 않으면서도 이 정보를 갖고 있어야 한다. 이를 위하여 각각의 input event에 대해 상태변수를 정의하여 insensitive input event의 값을 이

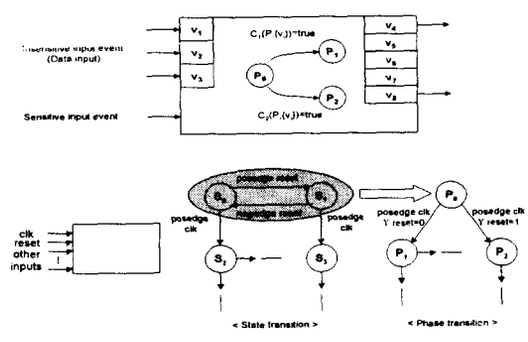

그림 3 상태전이와 phase전이

상태변수로 저장하는 방법을 사용한다. Phase정보와 상태변수들이 모두 기존 DEVS의 상태에 해당하게 된다. 그림 4는 reset이라는 insensitive input event에 대해 DEVS의 상태전이와 phase전이의 차이점을 보이고 있다. 다른 상태로 구분되었어야 할 상태들이 하나의 phase에 해당하기 때문에 다음 상태 전이시 내부에 latch된 상태변수와 그것에 정의된 condition으로 결정한다.

2.3.2 두 번째 조건 : activity

두 번째 조건은 출력을 내지 않으면서 내부 상태 전이하는 경우 여기에 관련된 상태들은 모두 하나의 phase에 해당하여야 한다는 것이다. 하드웨어적인 측면에서 얘기하면 어떤 조건에 의해 일련의 내부처리를 하는 경우이다. DEVS모델에서도 출력을 내지 않으면서 time advance가 0인 경우에도 연속적인 내부 상태 전이의 의해 상태를 바꾸게 되므로 같은 표현력을 가진다는 것을 알 수 있다.

위의 정의에 의해 phase표현형태는 다음과 같은 특성함수를 갖게 된다.

Internal transition :
(*phase*)*(*condition*)→(*phase*)*(*activity*)
External transition :
(*phase*)*(*input event*)*(*condition*)→(*phase*)*(*activity*)
Output function :
(*phase*)*(*condition*)→(*output event*)
Time advance function :
(*phase*)*(*condition*)→(*Real*)

3. Verilog HDL 모델의 변환

앞에서 설명한 DEVS의 phase표현형태를 사용하면 임의의 Verilog 모델을 DEVS의 phase 표현형태로 변환할 수 있고, 그 형태를 C++ 프로그램으로 만들어 주면 처음부터 C++로 구성한 다른 부분의 모델들과 쉽게

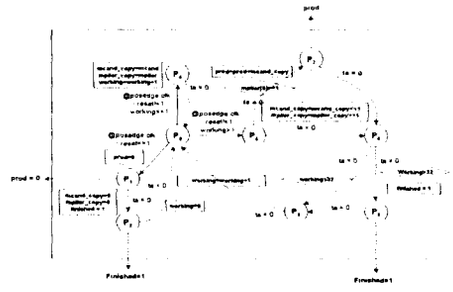

그림 5. Multiplier모델의 phase형태

통합할 수 있다. 그림 5는 일반적인 multiplier를 Verilog HDL로 모델링한 후 우리가 구성한 변환기를 거쳤을 때 구성되는 phase의 변화와 그에 따른 조건, activity를 표현한 것이다.

4. 변환기의 구성

Front-end 파서를 통하여 Verilog HDL모델을 linked-list형태의 내부 데이터구조로 바꾸어준다. 파서는 GNU Bison 1.25와 flex 2.5.4를 사용하여 구성하였다. 파서를 통하여 구성된 내부 데이터 구조는 그림 6과 같다. 하나의 하드웨어 모듈에 입력, 출력, 레지스터 정보가 저장되고 가장 중요한 phase전이가 graph형태의 linked-list로 구성된다. 각각의 노드는 phase이고 phase간의 arc에는 condition과 activity가 저장된다. 하나의 모듈을 파싱했을 때 다른 모듈과 연결된 정보는 다른 모듈을 파싱하여 정보를 얻기 전까지는 알 수 없으므로 일단 저장한다. 이 정보를 가지고 전체 하드웨어 모듈의 정보를 구성하고 이 정보들로부터 structural module을 구성하게 된다. 이렇게 구성된 내부 데이터구조를 back-end code generator

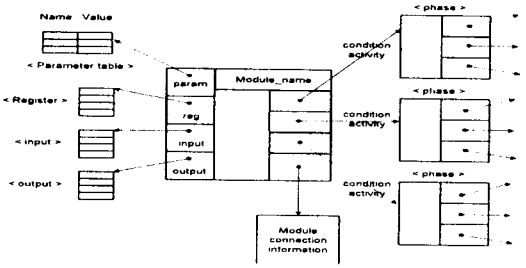

그림 6 내부 데이타구조

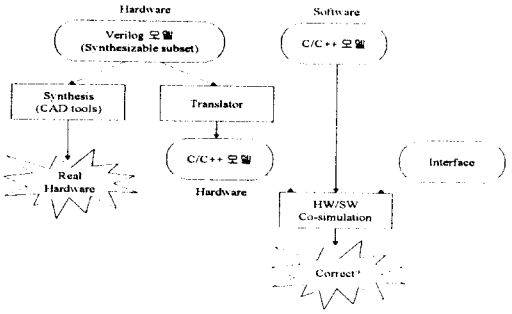

그림 6 시스템 설계 프레임웍

를 통해 C++ 모델을 구성하게 된다.

5. 결론

이 논문에서는 HW/SW 통합시뮬레이션에서 하드웨어와 소프트웨어의 표현방식이 다른 데에서 오는 문제점을 DEVS의 phase 표현방식에 기반한 변환기법을 사용하여 해결하였다. Verilog HDL로 기술된 모델은 우리가 구성한 translator를 통하여 C++모델로 변환되고 C++로 개발된 software와의 통합 시뮬레이션을 통하여 정확성을 검증하고 성능을 알아본다. 이 과정이 끝나면 기존의 CAD 툴을 사용하여 implementation level로의 synthesis를 거쳐 실제 하드웨어를 얻어내게 된다.

그 이외에 얻을 수 있는 이점으로는 정형론을 사용하였기 때문에 formal verification이 가능하다는 것이다. 기존의 하드웨어 설계에서 정확성 검증을 위해서는 테스트 방법을 사용하였다. 그러나 우리의 방법을 사용하게 되면 state reacheablity나 여러 모델간의 equivalence를 체크하는 strong/weak bi-simulation[4]방법을 도입함으로써 다른 추상 레벨에서 설계된 모델들이 결과적으로 같은 동작을 수행하게 되는지와 같은 것들을 알 수 있을 것이다.

6. Reference

[1] P.Subrahmanayam, "Hardware-Software Codesign : Cautious Optimism for the Future (Hot Topics)", *IEEE Computer*, 26(1), pp.84-85, 1993.

[2] Klaus Buchenrieder and Jerzy W.Rozenblit, "Codesign : An Overview", in *Codesign, Computer-Aided Software/ Hardware Engineering* Ed. J.Rozenblit and K.Buchenrieder, IEEE Press, pp.4, 1995.

[3] B.P Zeigler, *Multifacetted Modeling and Discrete Event Simulation*, Academic Press, London, 1984

[4] 송해상, 김대현, 김탁곤, "DEVS Bisimulation : 이산사건 모델의 계층적 검증 방법", *한국시뮬레이션 춘계학술대회 논문집*, pp.43-49, May, 1998.