구문중심적 변환을 통한 C언어의 비동기회로 합성기법

곽상훈, 이정근, 이동익
광주과학기술원 정보통신공학과

Synthesis of Asynchronous Circuits from C Language Using Syntax Directed Translation

Sang-Hoon, Kwak, Jeong-Gun Lee, and Dong-Ik, Lee
Dept. of Info. & Comm. K-JIST
E-mail : {shkwak, eulia, dilee}@kjist.ac.kr

Abstract
Due to the increased complexity and size of digital system and the need of the H/W-S/W co-design, C/C++ based system design methodology gains more interests than ever in EDA field. This paper suggests the methodology in which handshake module corresponding to each basic statement of C is provided of the form of STG(Signal Transition Graph) and then, C statements is synthesized into asynchronous circuit through syntax-oriented translation. The 4-phase handshaking protocol is used for the communications between modules, and the modules are synthesized by the Petrify which is asynchronous logic synthesis CAD tool.

1. 서 론

현재 반도체 설계자동화 분야는 소자기술의 발달에 따른 고집적화, 온칩(on-chip)화가 두드러지고 있다. 수백만 게이트급의 복잡도를 가진 시스템을 설계해야 하는 현 상황에서 보다 높은 수준에서의 추상화를 지원하는 설계언어와 도구를 이용하는 것은 필수적이다. 또한 마이크로프로세서, 마이크로 콘트롤러의 성능이 점점 향상됨에 따라 앞으로 시스템에서 내장 소프트웨어의 비중과 중요성이 더욱 증대되고 있다. C언어를 이용하여 시스템을 설계할 경우 이러한 행위수준에서의 기술을 통한 높은 추상화 수준에서의 설계가 가능하고, 하드웨어-소프트웨어 통합설계를 위한 초기 사양으로 적합하다. 이러한 장점 때문에 현재 C언어 기반 하드웨어 설계에 관한 관심이 고조되고 있으며, 많은 연구그룹들이 관련연구를 수행하고 있다.

비동기회로는 회로자체가 가지는 내재적인 저전력성, 평균수행시간의 이용가능성, 타이밍에 대한 비민감성등의 장점 등으로 인해 많은 연구가 되어 왔으며 앞으로 공정기술의 발전에 따라 부각되는 동기회로의 문제점을 상당부분 해결해 줄 것으로 기대되고 있다. 그러나 비동기회로 설계를 위한 상위수준의 설계도구 중 하드웨어 설계자들에 익숙한 설계언어를 제공하는 툴은 미미한 실정이며, C언어로부터 직접 비동기회로를 생성하는 방법에 대한 연구는 존재하지 않았다. 따라서 본 논문에서는 C언어를 이용한 비동기회로 설계환경에 대한 기본 방법론과 환경을 제공함으로서 좀 더 용이한 비동기회로설계 환경을 구축하고 나아가 시스템수준의 설계방법, 비동기회로를 이용한 하드웨어-소프트웨어 통합설계를 위한 기본 도구를 제공하고자 한다.

2. 관련연구

비동기회로의 상위수준 합성에 관한 기존의 연구로는 크게 두 가지 접근방법이 있다. 첫째는 하나의 기술언어로부터 구문중심의 변환을 이용하여 각 구문에 대응되는 모듈로 매핑하여 핸드셰이크회로(handshake circuit)를 생성하는 방법이고 [1],[2] 둘째는 CDFG와 같이 이미 변환된 중간형태로부터 변환(transformation)을 이용하여 논리합성을 위한 STG(Signal Transition Graph), AFSM(Asyncronous

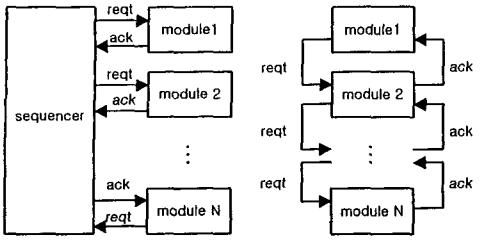

그림 1 기존의 핸드셰이크 회로와 연결된 구조를 사용한 핸드셰이크 회로

Finite State Machine)을 생성하는 방법이다.[3] [4] 본 논문에서는 합성 가능한 C언어 구문을 정의하고 각각의 C언어 구문을 기정의된 매크로모듈(macro module)로 변환하는 방법을 제시하고자 한다. 이 매크로 모듈의 내부는 적절한 지연시간을 포함하여 동작하는 bundled data constraint를 가정하고 있으며, 각 매크로 모듈은 핸드셰이크를 이용한 통신으로 데이터를 주고 받는다. 이 때 핸드셰이크의 구현은 효율적이고 작은 회로를 생성할 수 있는 4-phase 핸드셰이크 프로토콜을 이용한다.

3. C언어의 하드웨어 매핑을 위한 아키텍처

C언어 요소들에 대해서 다음과 같은 요소들을 비동기회로로 합성한다.

①자료형: 정수, 실수, 문자, 배열등을 지원한다.

②연산자: +,-,*,/ 이외에 &&, ||와 같은 논리연산 &,|,~같은 비트연산을 지원한다.

③선택구문: if-else, if-else if - else, switch 와 같은 선택구문을 지원한다.

④반복구문: do .. while, while, for와 같은 반복구문을 지원한다.

이와 같은 방식으로 매크로모듈의 네트워크로 구성된 비동기회로를 구현하고 이 매크로 모듈을 게이트수준에서 구현한 라이브러리를 제공하는 것이 본 연구의 목적이다. C언어를 이용할 경우 STG, 혹은 AFSM을 사양기술도구로 선택한 신호수준에서 회로를 기술하는 것보다 한층 상위에서 회로를 기술할 수 있고, 또한 대부분의 시스템설계자들이 C언어에 익숙하므로 앞서 언급한 설계생산성이 증대되고, 더욱 용이한 비동기회로설계 환경을 구축할 수 있다.

• 핸드셰이크 회로의 구성

C언어의 if, while, for문과 같은 제어구문을 위한 매크로 모듈을 STG 형태로 정의하고 그 이외의 데이터 연산부분도 매크로 모듈로 정의하여 C언어로부터 핸드셰이크 회로 기반의 비동기회로를 생성하는 방법을 구축한다.

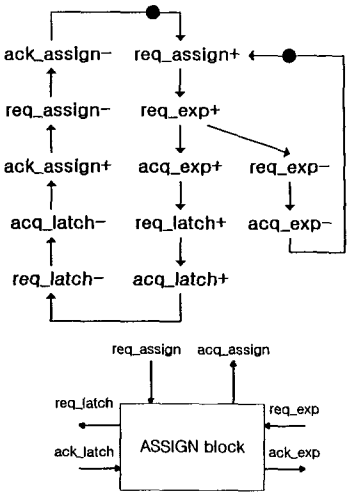

그림 2 할당문에 대한 핸드셰이크모듈의 행동과 인터페이스

순차구문의 경우 그림 1과 같이 별도의 시퀀서(sequencer)회로를 사용하지 않고 각 모듈 행동의 종결시 다음수행 모듈의 request 신호로 입력되게 함으로서 구조적인 수준의 순차구문 실행을 지원한다. 그리고 모듈간의 핸드셰이크는 각 신호의 상승전이만을 인식하는 4-phase handshake 프로토콜을 사용한다.

4. C언어 각 구문요소의 매크로모듈 구현

C언어의 구문요소중 대표적인 할당문, 이항연산자, 반복문, 선택문에 대한 매크로모듈 내부 행동을 STG로 기술하였다. 모든 매크로모듈들은 request 신호를 외부로부터 입력받아 자신의 행동을 시작하며 acknowledge 신호를 외부로 출력하여, 내부행동이 종료되었음을 알린다. 각 구문요소의 매크로모듈구현에 대한 상세한 설명은 다음과 같다.

할당문 : 할당문에 대한 모듈인터페이스와 행위기술이 그림 2에 나타나 있다. 본 합성방법에서 사용되는 변수는 모두 4-phase 핸드셰이킹 프로토콜을 만족하는 latch로 구현된다고 가정한다.

이항연산자 : 각 이항연산자(+, *, /, 등)를 위한 매크로모듈은 다음과 같이 설계한다. 실제데이터 연산부분은 동기식회로의 덧셈기, 곱셈기, 제산기등을 그대로 사용하고 4-phase 핸드셰이크 프로토콜을 따르는 제어부분과 최악의 지연시간을 고려한 지연시간요소 삽입을 통해 C언어의 이항연산자에 매크로 모듈을 제공한다

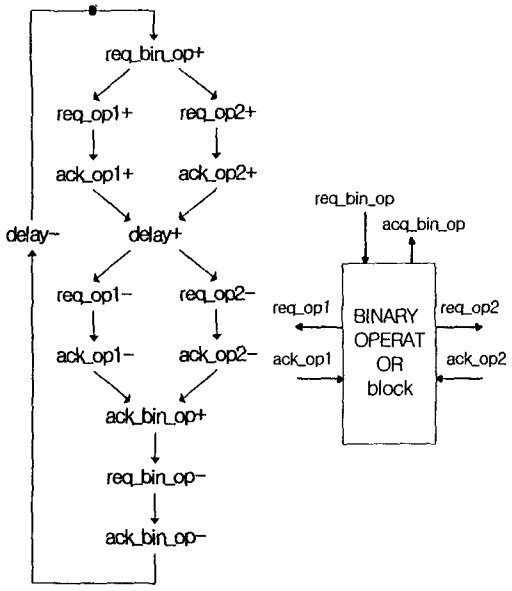

그림 3 이항연산자의 제어부에 대한 행동기술과 인터페이스

. 그림 3에 이항연산자의 제어부에 대한 행동이 STG로 기술되어 있다. 데이터 연산과 핸드셰이크 신호간의 타이밍가정(bundled timing constraints)을 만족시켜주기 위해 delay 신호가 삽입되었다. delay 신호가 상승한 후 해당연산자의 최악연산시간보다 더 큰 지연시간이 경과하여 req_op1-, req_op2-와 같은 이후의 제어부 행동이 계속된다.

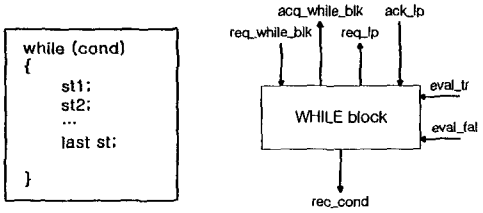

그림 6 WHILE 구문에 대한 매크로 모듈의 인터페이스

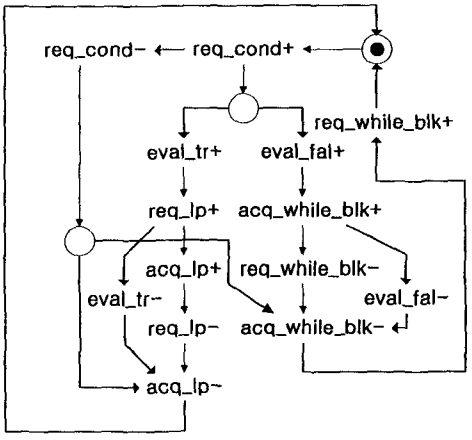

그림 7 WHILE 구문에 대한 매크로모듈의 행동기술

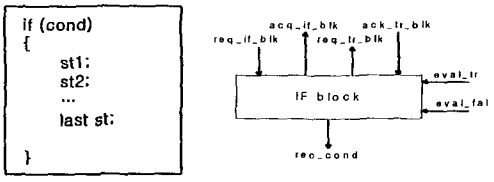

그림 4 IF 구문에 대한 매크로 모듈의 인터페이스

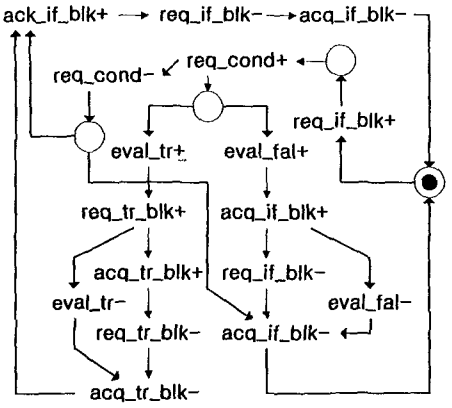

그림 5 IF구문에 대한 핸드셰이크 모듈의 행동기술

선택문 : if 구문에 대한 매크로 모듈의 인터페이스가 그림 4에 나타나 있고 이 모듈에 대한 행동(behavior)는 그림 5에 STG 형태로 표현하였다. req_if_blk, ack_if_blk은 각각 이 if 구문의 매크로 모듈에 대한 request, acknowledge 신호이며, eval_tr, eval_fal 는 조건식을 평가한 모듈에서 이 if 모듈로 들어오는 입력으로 조건식의 값에 따라 각각 배타적으로 동작하며, 이 모듈안에서 핸드셰이크 신호의 역할을 하게 된다. req_tr_blk, ack_tr_blk 신호는 조건식의 값에 따라 참인 경우 수행될 블록을 위한 request, acknowledge 신호이다. 그 밖의 if - else, switch - case 구문 또한 이와 유사한 방식으로 STG 수준의 행위기술을 통하여 핸드셰이크 회로를 구성할 수 있다

반복문: 그림 6에서 while 구문에 대한 매크로 모듈의 인터페이스를 나타내었고. 매크로 모듈의 행동을 나타낸 STG는 그림 7에 나타내었다. while 구문에 대한 행위는 앞서 보여준 if 구문에 대한 구조와 매우 유사하나 조건 평가 후 반복해서 수행할 수 있게 해주는 부분이 if 구문에 대한 모듈과 다른 점이다.

```
void diffeq(int x, int y, int u, int dx, int a)
{
    while (xl < a);
    {
        xl = x + dx;
        ul = u - (3*x*u*dx) - (3*y*dx);
        yl = y + u*dx;
        x = xl; u = ul; y = yl;
    }
}

void main()
{
    diffeq(X, Y, U, d, a);
}
```

그림 8 C 언어로 기술된 예제 알고리즘

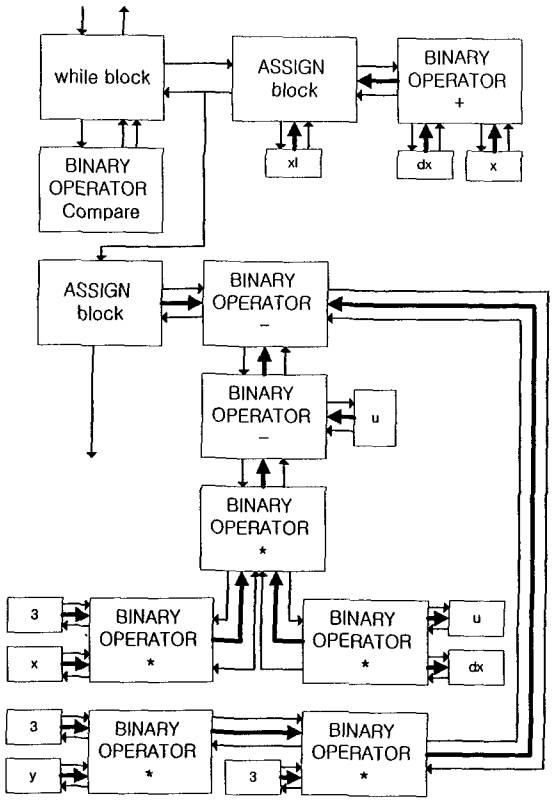

그림 9 예제회로의 음영부분에 대한 합성 결과

5. 예제회로

예제로는 그림 8에 나타낸 미분방정식 y"+3xy'+3y = 0에 대한 해답을 구하는 알고리즘의 C기술을 사용하였다. 그림 9에서 예제 알고리즘 중 음영부분이 매크로 모듈로 변환된 블록수준의 구현을 보여주고 있다. 가는 실선은 기정의 된 각 모듈의 request, acknowledge 핸드셰이크 신호를 나타내며 굵은 실선은 데이터 패스를 나타낸다. 화살표는 데이터의 이동방향을 의미한다.

6. 결론 및 향후과제

본 논문에서는 C언어의 각 구문들에 대한 매크로 모듈을 STG 형태로 제시하고 구문중심적(syntax directed)인 컴파일 과정을 통하여 C언어로 기술된 사양을 빠른시간에 비동기회로로 합성하는 방법을 제시하였다. 표현된 STG는 Petrify[5]와 같은 비동기회로 논리합성 CAD툴에 의해 게이트수준 논리회로로 합성가능함이 검증되었다.

구문중심적 비동기회로 합성방법은 빠른 시간에 회로를 합성할 수 있으나, 구문수준의 최적화, 간략화 과정만을 거친다. 구문중심적인 맵핑에 의한 합성과 부분적인 상위수준합성을 통한 최적화기능을 확장한 비동기회로의 설계도구의 개발이 향후 연구과제이다.

Acknowledge

본 연구는 한국과학재단의 한일국제공동연구과제(2000 6-302-01-2)및 광주과학기술원 초고속광네트워크연구센터를 통한 한국과학재단 우수연구센터 지원금에 의한 것입니다.

6. 참고문헌

1. A.M. G. Peeters. "Single-Rail Handshake Circuits", PhD thesis, Eindhoven University of Technology, 1996

2. Andrew Bardsley, "Implementing Balsa Handshake Circuits", PhD thesis, University of Manchester, 2000

3. Michael Theobald, Steven M. Nowick, "Transformations for the Synthesis and Optimization of Asynchronous Distributed Control", Proc. DAC , 2001

4. E. Kim, J. -G. Lee, and D. -I Lee Automatic process-oriented control circuit generation for asynchronous high-level synthesis. International Symposium on Advanced Research on Asynchronous Circuits & System, 2000.

5. J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno and A. Yakovlev. "Petrify: a tool for manipulating concurrent specifications and synthesis of asynchronous controllers", *IEICE Transactions on Information and Systems*, Vol. E80-D, No. 3, March 1997, pages 315-325.