論文94-31A-2-15

비동기 인터페이스를 지원하는 상위 수준 합성 시스템의 설계

(Design of a High-Level Synthesis System Supporting Asynchronous Interfaces)

李 炯 鍾*, 李 鍾 和**, 黃 善 泳**

(Hyung Jong Lee, Jong Hwa Lee and Sun Young Hwang)

要 約

이 논문에서는 인터페이스를 위한 타이밍 제약을 만족시키는 하드웨어를 생성하는 상위 수준 합성 시스템인 ISyn(Interface Synthesis System for ISPS-A)에 대해 기술한다. 기존의 ISPS를 인터페이스 구문과 타이밍 제약을 기술할 수 있도록 확장한 ISPS-A가 입력으로 주어지며 이 제약을 만족시키는 합성을 위하여 스케쥴링은 pre-scheduling과 post-scheduling의 두 단계로 수행되어지고 clique-partitioning 알고리듬을 이용한 allocation 과정을 통하여 입출력 포트를 포함한 하드웨어 모듈을 할당한다. 실험을 통하여 ISyn은 기존의 내부 연산에 대한 합성뿐만 아니라 입출력 인터페이스의 기술에 대해서도 효율적인 하드웨어를 생성해냄을 확인하게 하였다.

Abstract

This paper describes the design of a high-level synthesis system, ISyn: Interface Synthesis System for ISPS-A, which generates hardware satisfying timing constraints. The original version of ISPS is extended to be used for the description/capture of interface operations and timing constraints in the ISPS-A. To generate the schedule satisfying interface constraints the scheduling process is divided into two steps: pre-scheduling and post-scheduling. ISyn allocates hardware modules with I/O ports by the clique partitioning algorithm. Experimental results show that ISyn is capable of synthesizing hardware modules effectively for internal and/or interactive operations.

I. 서 론

디지탈 시스템 기술은 시스템이 수행해야 하는 내부 연산과 외부와의 통신을 위한 인터페이스의 두 주요 부분으로 구성된다. 그동안 상위 수준의 기술로부터 레지스터 전송 (RT: register transfer) 수준의 하드웨어를 자동으로 생성하는 상위 수준 합성 시스템의 개발에 대한 많은 연구가 있었으나 [1, 2, 3, 4, 5] 이들은 효과적인 하드웨어 동작을 위한 내부 연산에 대한 합성에만 주의를 기울인 것이 사실이다. 즉 내부 연산의 알고리듬 기술을 받아들여 그에 대한 RT 수준의 데이타 패스를 자동 생성하는 것이 그동

* 準會員, ** 正會員, 西江大學校 電子工學科
(Dept. of Elec. Eng., Sogang Univ.)

接受日字 : 1993年 1月 21日

안의 상위 수준 합성 시스템들의 주된 경향이었다. 그러나 실제로 대부분의 칩들이 다른 여러 칩이나 외부 환경과의 인터페이스를 통하여 동작해야 하기 때문에 사실상 이러한 합성 시스템들의 실제 응용 부문은 극히 제한될 수 밖에 없고, 따라서 앞으로의 상위 수준 합성 시스템은 외부와의 인터페이스를 위한 여러 제약 조건을 만족시킬 수 있어야 한다.

상위 수준 합성 시스템이 인터페이스 제약을 지원하기 위해서는 우선 이 제약을 표현할 수 있는 일반화된 문법이나 포맷이 있어야 한다. Janus/Suture 시스템에서 인터페이스 사양은 타이밍 다이어그램으로 표현되는데[6] 이것은 그 사양이 그래픽 터미널 상의 waveform으로 표현되어 명확히 이해할 수 있고 synchronous와 asynchronous 인터페이스 사양을 표현할 수 있으나, 시스템이 그래픽 정보만을 받아들인다는 단점이 있을뿐만 아니라 타이밍 제약을 내부 연산과 함께 표현하는데 한계가 있다. 인터페이스 사양은 신호들의 타이밍 제약을 표현하기 위하여 이벤트 그래프나 STG(signal transition graph), 내부 연산은 data flow 그래프의 중간 형태로 주로 표현이 되는데 상위 수준 기술에 인터페이스 사양을 표현할 수 있도록 이 두 중간 형태를 혼합한 unified behavior graph가 제안되었으나[7] 이는 내부 연산과 외부 인터페이스 기술이 명확히 분리된 기술에만 효율적 적용이 가능하므로 상위 수준 합성 시스템으로의 응용은 적당하지 않다.

인터페이스 사양을 상위 수준 합성 시스템의 입력으로 받아들이는 CMU 합성 시스템[8]과 Hercules[9]는 포트 입출력 연산을 HDL에 포함시키고 인터페이스 사양을 표현하기 위해 각 문장들 간에 타이밍 제약을 기술할 수 있게 하고 있으며, 타이밍 제약은 보통 두 문장들 간의 최대/최소 시간 간격을 의미한다. 그런데 CMU 합성 시스템은 스케쥴링 과정에서 최대 타이밍 제약 조건의 만족 여부를 검색하지 않으며 모듈 할당 과정에서 포트 레지스터는 다른 내부 레지스터와 공유되지 않아 생성된 하드웨어의 면적면에서 비효율적이다. Hercules는 relative 스케쥴링 알고리듬을 통해 최대/최소 타이밍 제약과 지연 시간이 정해지지 않은 연산에 대한 스케쥴링을 행하나 resource 제약이 연산 모듈에만 국한되어 있어 레지스터와 연결 구조에 대한 공유 가능성은 고려하지 않아 역시 면적 측면에서 비효율적일 수 있다.

ISyn(Interface Synthesis System for ISPS-A)은 CMU의 ISPS[10]를 확장시켜 인터페이스 타이밍 제약 조건의 표현이 가능하도록 한 ISPS-A를 입력으로 하여 인터페이스 사양을 만족시키는 RT 수준의 데이타 패스와 콘트롤러를 자동으로 생성하는 상위 수준 합성 시스템으로, ISPS-A 파서와 시뮬레이터, 의존성 분석기와 스케쥴러 그리고 모듈 할당기를 포함하고 있다. 모듈 할당 과정에서는 연산 모듈 뿐만아니라 conflict가 일어나지 않는 레지스터와 연결 구조들을 공유하여 면적을 최소화시킨다.

이 논문은 ISyn 합성 시스템의 설계에 대하여 기술하며 Ⅱ장에서는 전체적인 합성 과정을 설명하고 Ⅲ, Ⅳ장에서는 각각 스케쥴링과 모듈 할당에 대해 설명하며 Ⅴ장에 실험결과와 Ⅵ장에 결론을 보인다.

Ⅱ. 합성 과정

그림 1은 ISyn의 시스템 흐름도이다. 행위 기술 언어로는 ISPS를 확장한 ISPS-A를 사용한다. ISyn은 입력으로부터 AST(abstract syntax tree)[11]를 구성하며, 이는 ISPS-A 시뮬레이터[12]를 사용하여 행위 레벨에서 해석을 하는 입력으로 사용된다. 오류없이 파싱된 후 생성된 AST로부터 기본 블럭 인식과 의존성 분석 과정을 거쳐 문장들 간의 병렬성을 추출하며 이로부터 중간 형태인 C/DFG(control data flow graph)[4]를 생성한다. ISPS-A는 포트 입출력 연산 구문을 기술할 수 있으며, 타이밍 제약 기술은 두 문장의 최대, 최소 시간 간격을 label을 통하여 지정한다. 이 최대, 최소 타이밍 제약을 만족시키는 하드웨어를 생성해내어 외부 모듈과의 인터페이스를 가능하게 하는 것이 ISyn 합성 시스템의 목적이다.

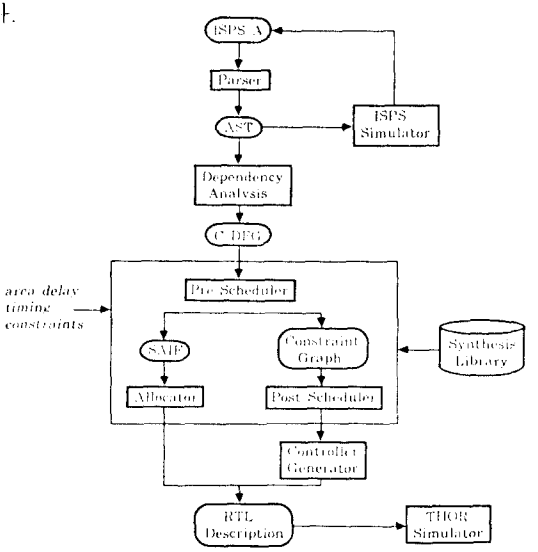


그림 1. ISyn 시스템 구성
Fig. 1. Overall Configuration of ISyn.

상위 수준 합성 과정은 스케쥴링과 모듈 할당의 과정으로 이루어지며 합성 라이브러리를 참조하게 되는데 이 라이브러리에는 연산 모듈의 이름과 그 입출력 포트, 지연 시간, 면적 등의 정보가 기록되어 있다. 스케쥴링은 인터페이스 연산과 그에 대한 타이밍 제약을 만족시키기 위하여 pre-scheduling과 post-scheduling의 두 과정으로써 이루어진다. 모듈 할당 과정에서는 스케쥴링 후 실행 시간이 결정된 연산들을 최적의 하드웨어 모듈을 사용하여 데이타 패스를 생성하고, 콘트롤러 생성 단계에서는 그 모듈들에 대한 제어 신호를 발생시키는 콘트롤 로직을 생성한다.

Ⅲ. 스케쥴링

일반 상위 수준 합성 시스템에서의 스케쥴링은 연산들을 특정 제어구간에 할당하는 과정으로, 이는 스케쥴링하고자 하는 모든 연산들이 유한한 지연 시간을 가지고 그들 간에 타이밍 제약이 없는 경우에 가능한 방법이다. 각 연산들은 선택된 스케쥴링 알고리듬과 주어진 지연 시간이나 면적에 대한 제약에 따라 다른 제어구간에 할당 될 수 있기 때문에, inport() (입력 포트에서 값을 읽어들이는 연산) 문에 대해서는 그것이 수행되는 제어구간 상의 입력 포트에 항상 올바른 값이 실려 있다는 가정이 따르게 된다. 이로 인하여 특정한 타이밍 제약을 따른 입출력을 행하는 모듈과 연결하는 하드웨어의 경우 입출력 포트를 통해 전달하고자 하는 올바른 값이 특정 시간에 실려 있는가의 여부를 확인할 수 없으므로, 이러한 하드웨어의 합성은 불가능하다. 그리고 다른 상위 수준 합성 시스템에서 한 연산을 하나 또는 몇 개의 제어구간에 할당할 수 있다는 것은 모든 연산이 유한한 지연 시간을 갖고 있고 또 그 값이 합성시에 알려져 있다는 가정 하에서 가능한 것이다. 대부분 연산들의 지연 시간은 합성 라이브러리 정보에 포함되어 있으나 비동기 시멘틱스를 갖는 wait 문이나 send/receive 문 등의 경우 그 지연 시간을 알 수 없으므로 기존의 상위 수준 합성 시스템으로는 이에 대해 스케쥴링을 행할 수 없다. 인터페이스 사양을 만족시키는 합성을 위해서는 이러한 지연 시간을 알 수 없는 연산들을 처리할 수 있기 위한 다른 스케쥴링 알고리듬이 필요하다.

ISyn은 입출력 인터페이스를 위한 타이밍 제약을 포함한 연산에 대해 스케쥴링을 행하고 연산 모듈과 레지스터, 연결 구조의 효율적인 공유를 위하여 스케쥴링 과정이 pre-scheduling과 post-scheduling의 두 부분으로 나뉘어져 있다. Pre-scheduling은 주어진 면적과 지연 시간을 제약으로 하여 지연 시간이 알려진 일반 연산들 간의 resource conflict를 제거하고 연산들의 상대적인 실행 시간을 결정하며 여기에 타이밍 제약을 지원할 수 있도록 수정된 EBS (entropy-based scheduling) 알고리듬[13] 을 사용한다. Post-scheduler는 pre-scheduling이 끝난 C/DFG를 constraint graph로 전환하여 relative 스케쥴링[14] 을 행함으로써 연산들 간의 최대/최소 타이밍 제약을 만족시키는 콘트롤러 정보를 생성한다.

1. Pre-scheduling

Pre-scheduling은 주어진 지연 시간이나 면적 조건 하에서 연산 모듈에 대한 resource conflict를 제거하고 최소 지연 시간에 각 연산들의 상대적 순서를 결정하며 결정된 순서를 rc-step (relative control-step)이라 부르는 제어 구간에 할당한다. 주어진 면적 제약 조건 하에서 최소의 하드웨어 모듈을 사용하기 위하여 연산들을 rc-step에 균등히 배분하는 것이 목적 함수이며 EBS 알고리듬[13] 을 수정하여 적용하였다.

EBS는 연산자들의 지연시간이 미리 알려진 경우에 가능한 방법이므로 C/DFG 상에서 지연 시간이 알려진 연산자들에 대해서만 스케쥴링을 하며 이때 지연 시간 제약 조건은 이러한 연산자들에 대한 지연 시간만을 고려한다. 여기서 EBS의 목적은 연산들을 제어구간에 고르게 분포시켜 하드웨어 모듈의 공유를 최대화하는 데 있다. 그러나 입출력 포트는 포트 연산에 대한 모듈 할당을 통해 합성하지 않고 설계자의 ISPS-Λ 기술에 나타난 포트 선언문에 따라 생성하므로 스케쥴링을 행할 때 EBS 알고리듬에서 보는 바와 같이 포트 연산들을 제어 구간 내에 균등히 배분할 경우 포트의 면적을 줄이지 않고 오히려 지연 시간이 늘어나 비효율적일 수 있다. 그러므로 ASAP, ALAP 스케쥴링 알고리듬에 대한 수정이 필요하다.

그림 2 (a)는 rc-step이 4일 때 세 개의 inport() 문에 대한 ASAP 스케쥴링 결과이고 (b)는 이에 대한 EBS 결과이다. 세 inport() 문들은 각각 다른 입력 포트에 대한 것이며 이들 간에 의존성이 없다. 만약 입력 포트가 하나인 경우에는 (b)와 같이 고르게 할당하는 것이 resource conflict가 일어나지 않도록 고르게 분포시킨 경우이지만, 설계자가 기술한 입력 포트가 세 개이므로 (a)가 더 빨리 수행될 수 있는 스케쥴이다. 즉, 입출력 연산에 대해서는 엔트로피의 개념을 적용시키면 비효율적인 결과를 얻으므로 기존의 스케쥴링 알고리듬을 변형하여 지연 시간 측면에 대해서 효율적인 결과를 얻도록 하였으며 이는 면적

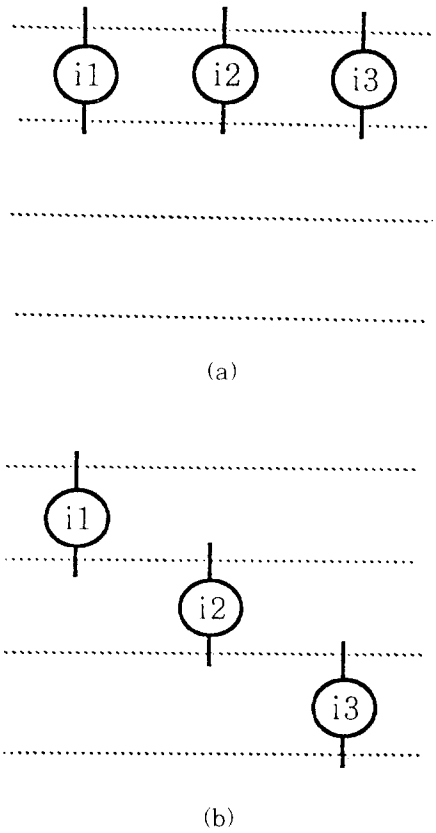


그림 2. 입력 연산에 대한 스케쥴링
(a) ASAP 스케쥴링
(b) Entropy-based 스케쥴링
Fig. 2. Scheduling for inport() operations.
(a) ASAP scheduling.
(b) Entropy-based scheduling.

제약 조건에 아무런 영향을 미치지 않는다.

면적 제약 조건하의 스케쥴링 알고리듬은 일종의 리스트 스케쥴링으로 각 제어 구간마다 면적 제약을 만족시킬 수 있는 최대의 연산들을 선택하는데 가장 작은 엔트로피를 갖는 연산에 높은 우선 순위를 두어 선택한다. 입출력 문에 대한 면적 제약 조건은 내부 연산들에 대한 면적 제약 조건과 별도로 고려한다. 즉, 포트의 면적 제약 조건을 ISPS-A 기술에 선언된 포트의 수와 같게 하여 스케쥴링 알고리듬을 적용하면 다른 연산들과의 의존성을 만족시키는 범위 내에서 그림 2 (a)와 같이 가능한 가장 앞의 rc-step에 할당되어진다. 시간 제약 조건하의 EBS 알고리듬은 주어진 지연 시간 제약 조건 하에서 최소의 모듈을 사용하는 것이 목적으로 엔트로피의 값이 가장 큰 연산을 우선적으로 할당하는데 입출력 문에 대해서는 그 우선 순위 함수를 모두 0으로 하여 같은 값을 갖게 하면 ASAP 스케쥴을 한 것과 같은 결과를 얻을 수 있다.

2. Post-scheduling

기존의 합성 시스템은 제어 구간 정보를 받아들여 콘트롤러를 생성하나 relative 스케쥴링 알고리듬은 의존성을 나타내는 간선에 그 지연시간을 weight로 한 constraint graph를 입력으로 하므로[14] pre-scheduling 후 rc-step에 할당된 연산 노드들로부터 constraint graph를 생성한다.

의존성이 없는 두 연산이 resource conflict 등의 합성 제약으로 다른 rc-step에 할당되었을 경우 이는 두 연산 간에 새로운 콘트롤 의존성이 생겼음을 의미한다. 따라서 이웃하는 rc-step의 연산 노드들 간에 콘트롤 간선을 연결하여 pre-schedule 된 연산 들의 실행 순서가 relative 스케쥴링 단계에서도 유지되도록 하고 constraint graph의 간선의 weight는 그 source 연산자의 지연 시간으로 지정한다.

기존의 스케쥴링 알고리듬은 각 연산들이 유한하고 일정한 지연 시간을 갖는다는 가정하에서 제안되어왔다. 그러나 앞에서 언급된 바와 같이 외부와의 인터페이스 신호와 그 이벤트의 처리를 고려하는 합성 시스템에서는 지연 시간이 일정하지 않는 연산도 고려되어야 한다. 기존의 스케쥴링 알고리듬은 이러한 연산은 제외시킨 스케쥴링의 결과이므로 relative 스케쥴링 알고리듬을 적용하여 이 결과로부터 콘트롤 로직을 생성한다.

Ⅳ. 모듈 할당

모듈 할당은 스케쥴링의 결과로부터 RT 수준의 데이타 패스를 구성하는 과정이며, 이 때 사용되는 하드웨어 모듈의 수를 최소화하는 데 그 목적이 있다. 모듈 할당은 보통 연산 모듈, 메모리 소자, 버스나 MUX 등의 연결 구조로써 이루어진다. 행위 기술 상의 변수는 레지스터나 메모리에, 각 연산은 ALU나 곱셈기 등의 연산 모듈로 할당되며 이들 간의 입출력 구조로부터 연결 구조을 할당하게 된다.

모듈 할당 알고리듬에는 iterative/constructive 알고리듬과 global 알고리듬 등이 있다.[15] Iterative/constructive 알고리듬은 레지스터, 연산자, 연결 구조를 한번에 하나씩 할당하고 이 과정을 반복적으로 수행하면서 최적화를 해 나간다. 이 알고리듬은 탐색 공간의 수는 작으므로 실행 시간은 짧으나 지엽

적인 고려로 인하여 최적의 결과는 얻기 힘들다. Global 알고리듬은 한 번에 여러 개의 할당을 하기 위해 전체 공간을 탐색하므로 시간적인 면에서 효율적이지 못하지만 보다 최적화된 결과를 얻을 수 있다.

HAL 시스템[4]은 스케줄링의 결과로부터 각 연산이 사용할 모듈을 선택하기 위해 rule-based expert system을 사용하였고, iterative 알고리듬을 사용하여 모듈 할당을 수행한다. MIMOLA 시스템[16]은 연산 모듈의 모듈 할당에는 global 알고리듬을 사용하고 나머지에 대해서는 iterative 알고리듬을 사용하며, DAA[15]는 rule-based expert system으로 iterative 알고리듬을 이용한다. Emerald 시스템에서는 clique partitioning 알고리듬[1]을 제안하였으나 최적화 과정을 하나의 기본 블럭 내에 국한하였고 모듈 할당 과정에서 실제 구현에 필요한 연산자 지연 시간, 면적, 기능의 차이를 전혀 고려하지 않았다.

ISyn에서는 스케줄링된 C/DFG를 모듈 할당에 적합하도록 SAIF (scheduling allocation intermediate format)[11]로 전환하고, 연산자의 지연 시간, 면적, 기능을 목적 함수로 한 similarity measure[1]를 이용하여 효율적인 모듈 할당이 이루어지도록 하는 global 알고리듬을 사용하였다. Similarity는 공유 가능한 하드웨어 모듈의 조합을 찾아내는 데 있어 그 기능, 면적, 지연 시간 면에서 유사한 모듈을 우선적으로 merge하기 위한 측정 함수이다.

모듈 할당의 흐름은 레지스터 할당, 연산자 할당, 연결 구조 할당의 순으로 이루어진다. 연산자 할당의 과정에서 연결 구조의 면적을 줄이기 위하여 연산의 source나 destination이 같은 연산자를 공유하려 하므로 레지스터 할당을 먼저 하면 이러한 정보를 얻을 수 있고, 이를 통해 연결 구조의 비용을 줄이는 방향으로 연산자 할당을 하게 한다.

레지스터 할당은 기술 상에 사용된 변수들에 대해 최적의 공유할 수 있는 조합들을 찾아내어 레지스터 모듈을 구성하는 과정이다. 변수들의 사용 구간이 서로 겹치지 않아야 공유가 가능하므로 lifetime analysis를 수행하여 공유가 가능한 변수의 쌍들을 찾아내고 이로부터 compatible graph를 생성한 후 그에 대해 clique partitioning을 적용하여 공유가 가능한 최대의 조합을 찾아낸다. 이때 각 변수가 사용된 연산의 similarity를 고려하여 우선 순위를 정하여 면적 면에서 효율적인 조합을 찾아낸다.

출력 포트는 변수가 아니지만 구분의 의미상 출력 포트와 하나의 레지스터가 같이 할당되어야 하기 때문에 포트로 선언된 entity도 하나의 변수로 취급하여 그 포트 레지스터를 다른 내부 변수와 공유하게 할 수 있다. 일반적인 연산에 대해서는 변수가 마지막으로 사용된 시점까지를 live 구간으로 설정하나 출력 포트에 대해서는 다른 기준을 적용해야 한다. 즉, outport()(출력 포트로 값을 출력시키는 연산)에 의해 한 번 출력된 값은 그 출력 포트에 대해 free (port value를 tri-state로 하는 연산) 문이 실행되기 전까지 레지스터에 유지되고 있어야 하기 때문에 출력 포트 변수의 live 구간은 처음 지정된 시점부터 이에 대한 마지막 free( ) 문까지이다. 그림 3에 출력 포트의 lifetime 예를 보였다. free( ) 문이 없는 경우에는 마지막 제어 구간까지가 live 구간이 된다.

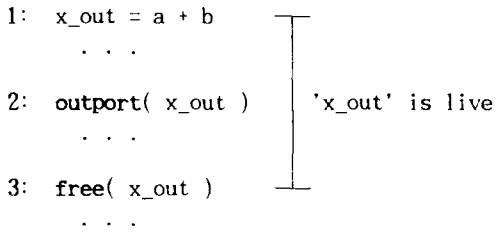


그림 3. 출력 포트의 lifetime
Fig. 3. Lifetime of an output port.

입출력 포트는 설계자가 하드웨어 기술에 선언한 포트만 생성하므로 inport( )와 outport( ) 연산에 대해서는 공유 가능성을 찾지 않는다.

앞의 두 할당 과정에서 변수들은 레지스터로 연산자들은 연산 모듈로 매핑되는데 연결 구조 할당은 이들 레지스터와 연산 모듈을 연결함에 있어 그 면적을 최소화하는데 그 목적이 있다. 스케줄된 C/DFG 상의 연결 정보를 노드로 그들간의 연결이 겹치지 않는 경우를 에지로 나타내는 compatible graph를 구성한후 이에 대해 clique partitioning 알고리듬을 적용하여 최대의 공유 가능한 조합을 찾아내며 여기서 source나 destination이 같은 연결 구조에 우선 순위를 줌으로써 생성되는 MUX의 비용을 줄이며 하나의 버스로 연결 구조을 생성한다.

일반적인 연산 모듈간의 연결 구조에 대해서는 operand의 값을 읽거나 쓰는 구간에서 active이지만, 출력 포트와 출력 레지스터에 대해서는 출력값이 유지되는 동안 그 값을 계속 유지하고 있어야 하므로 포트 레지스터의 lifetime동안 active이다.

V. 실험 결과

ISyn은 SUN 워크스테이션의 UNIX 프로그래밍 환경 하에서 C 언어로 구현되었다. 시스템의 성능 평

가를 위하여 미분방정식 연산기[3]와 체크섬 발생기[7]의 합성 결과를 5.1과 5.2절에 보인다. 미분방정식 연산기는 산술 연산을 위주로하여 그 앞과 뒷부분에 각각 입출력 구문이 있는 하드웨어 모델링이며, 체크섬 발생기는 내부 연산과 함께 입출력이 동시에 이루어지며 입출력 타이밍을 위주로한 하드웨어의 모델링이다.

1. 미분방정식 연산기

그림 4은 2차 미분방정식 y" + 3xy' + 3y = 0의 해를 수치해석적으로 구하는 하드웨어의 행위 기술이다. 여기서 입력 데이타는 입력 신호인 ready_in이 HIGH로 되고 다음 클럭 주기에 x_in, y_in, u_in의 값이 동시에 입력 포트에 실리고 그 다음 클럭 주기에 dx_in, a_in이 입력될 때의 타이밍 제약 기술을 보였다. 계산 결과의 출력은 먼저 출력 신호 ready를 HIGH로 한 후 다음 주기에 y_out 포트로 그 결과를 출력하고 다시 다음 주기에 ready 출력 포트를 high-impedance 상태로 한다.

표 1에 while 문 안의 기본 블럭에 대해 시간 제약 조건을 6으로, 한 클럭 주기를 50 nsec로 하였을 때의 ISyn과 HAL[3]의 합성 결과를 비교하였으며 그림 5은 ISyn 시스템의 합성 결과 생성된 데이타 패스이다. 여기서 HAL의 결과는 while 문 내의 연산에 대한 합성 결과이고 ISyn의 결과는 전체 기술에 대한 결과이기 때문에 변수의 lifetime이 길어져 공유할 수 있는 변수의 조합이 줄어 생성된 레지스터의 수가 많이 소요된다. 그러나 이 미분방정식 연산기의 기술에서처럼 입출력 부분과 연산 부분을 구분

```
Differential_Equation( )
   input port  : ready_in<>, a_in<15:0>, dx_in<15:0>,
                 x_in<15:0>, y_in<15:0>, u_in<15:0>
   output port : ready<>, y_out<15:0>
begin
   variable:  a, dx, x, u, y, y1, u1, u2, u3, u4, u5, u6;
   L1: wait(ready_in) ;
   L2: x  = inport(x_in) ;
   L3: y  = inport(y_in) ;
   L4: u  = inport(u_in) ;
   L5: dx = inport(dx_in) ;
   L6: a  = inport(a_in) ;
   while( x < a ) begin
      u1 = u * dx  ; u2 = x * 5 ;
      u3 = y * 3   ; y1 = u * dx ;
      x  = x + dx  ; u4 = u1 * u2 ;
      u5 = dx * u3 ; y  = y + y1 ;
      u6 = u - u4  ; u  = u6 - u5
   end
   readyt = 1 ; y_out = y ;
   L7: outport( ready ) ;
   L8: outport( y_out ) ;
   L9: free( ready )

   minimum timing constraint
      ( L1, L2 : 50 nsec ), ( L1, L3 : 50 nsec ),
      ( L1, L4 : 50 nsec ), ( L1, L5 : 100 nsec ),
      ( L1, L6 : 100 nsec ), ( L7, L8 : 50 nsec ),
      ( L7, L9 : 150 nsec )
   maximum timing constraint
      ( L7, L8 : 100 nsec )
end
```

그림 4. 미분방정식 연산기의 행위 기술

Fig. 4. Behavioral description of differential equation.

하여 기술하면 출력 포트 레지스터와 내부 레지스터와의 공유 가능성이 커져 출력 구문으로 인한 레지스터 수의 증가는 상대적으로 작아짐을 알 수 있다. 그림 5에서 모든 출력 포트 레지스터는 다른 내부 레지스터와 공유되었음을 볼 수 있다. 연결 구조에 대해 HAL은 버스를 지원하지 않고 MUX를 사용하여 그

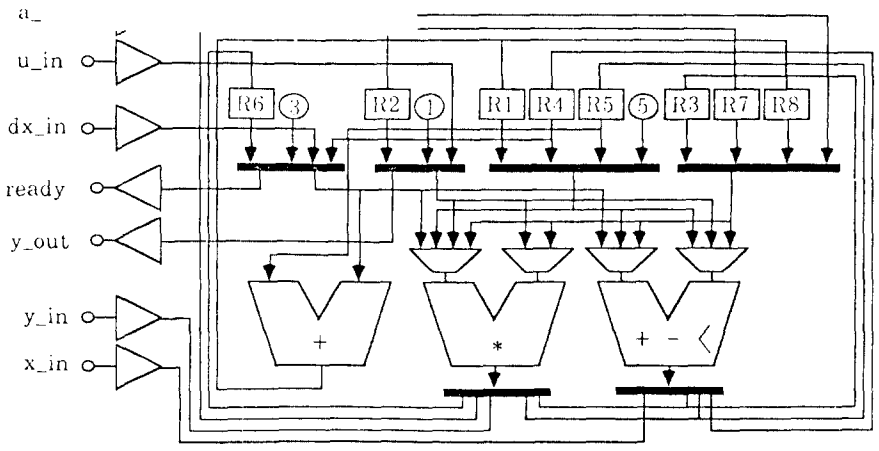


그림 5. 미분방정식 연산기의 데이타패스

Fig. 5. Data path of differential equation.

입력 갯수가 많으나 ISyn은 버스도 동시에 고려를 하므로 연결 구조의 비용이 감소되었다.

표 1. 시간 제약 하의 합성 결과
Table 1. Synthesis results under delay constraint.

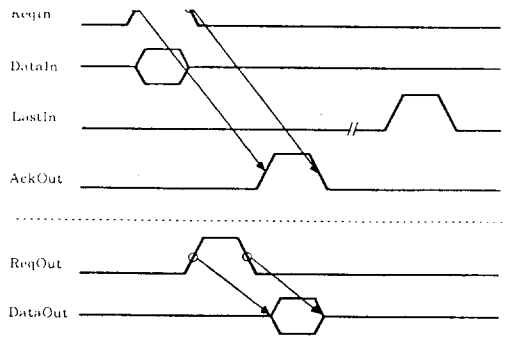
	ISyn	HAL
# of Reg.	8	6
# of FUs	3	3
# of MUX inputs	4	17
# of BUSes	6	0
# of Interconn.	31	35
# of I/O ports	6	n/a

2. 체크섬 발생기

그림 6 (a), (b)는 체크섬 발생기의 타이밍 다이어그램과 그 행위 기술이다. 이 모듈에 대한 기술은 내부 연산보다는 외부와의 입출력 구문이 주를 이룬다. 시스템 클럭의 주기는 20 nsec이다.

그림 6 (a)에서 체크섬 발생기는 LastIn 신호가 HIGH인 동안 DataIn의 값을 연속적으로 입력 받아 그를 출력시키면서 그 값들의 합을 구한다. 점선의 위는 입력 타이밍, 아래는 출력 타이밍으로 여기서 보듯이 입력 인터페이스는 asynchronous이며 출력은 synchronous이다.

타이밍 제약으로 인하여 while 문 내의 문장들에



(a)

```
CheckSumGenerator( )
   input port  : ReqIn<>, DataIn<7:0>, LastIn<>
   output port : AckOut<>, ReqOut<>, DataOut<7:0>, LastOut<>,
                 ChkSumOut<7:0>
begin
   variable: Byte<7:0>, ChkSum<7:0>
   while( Not LastIn ) begin
         /* Receive phase. */
      L1: wait( ReqIn ) ;
      L2: Byte = inport( DataIn ) ;
          AckOut = 1 ;
      L3: outport( AckOut ) ;
      L4: wait( not ReqIn ) ;
          AckOut = 1 ;
      L5: outport( AckOut ) ;
         /* Send phase. */
          ChkSum = Byte + ChkSum ;
          ReqOut = 1 ;
      L6: outport( ReqOut ) ;
           DataOut = Byte ;
      L7: outport( DataOut ) ;
           ReqOut = 0 ;
      L8: outport( ReqOut ) ;
           free( DataOut )
   end
        LastOut = 1 ; ChkSumOut = ChkSum ;
   L9:  outport( LastOut ) ;
   L10: outport( ChkSumOut )

   minimum timing constraint
      ( L1, L2 : 20 nsec ), ( L2, L3,  20 nsec ),
      ( L4, L5 : 20 nsec ), ( L6, L7,  20 nsec ),
      ( L7, L8 : 20 nsec ), ( L9, L10, 20 nsec )
   maximum timing constraint
      ( L6, L7 : 20 nsec ), ( L7, L8 : 20 nsec )
end
```

(b)

그림 6. 체크섬 발생기
(a) 타이밍 다이어그램
(b) 체크섬 발생기 행위 기술
Fig. 6. Checksum generator.
(a) Timing diagram.
(b) Behavioral description.

서 최소 지연 시간은 클럭 주기 5가 된다. 체크섬 발생기에 대한 합성의 결과를 표 2에 보였으며 그림 7은 생성된 데이타 패스이다. 데이타 패스는 모듈 할당 과정에서의 similarity를 고려한 우선 순위로 인하여 데이타 연산과 콘트롤 시그날의 두 부분으로 나뉘어져 있음을 볼 수 있다. 표 2의 레지스터의 수는 입출력 포트의 수와 같으며 이는 내부 연산에 필요한 레지스터와 입출력 레지스터가 모두 공유되었음을 보여준다.

표 2. 체크섬 발생기 합성 결과
Table 2. Synthesis results of checksum generator.

Reg	FU	MUX input	BUS	InterConn.	I/O Port
5	1	2	2	11	5

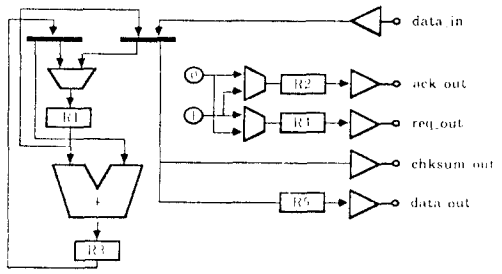


그림 7. 체크섬 발생기의 데이타 패스
Fig. 7. Data path of checksum generator.

Ⅵ. 결 론

이 논문에서는 포트 입출력 연산과 인터페이스 제약을 만족시키는 상위 수준 합성 시스템 ISyn에 대해 설명하였다. ISPS-A는 기존의 ISPS를 확장하여 포트 입출력 연산과 타이밍 제약을 기술할 수 있으며 이 기술은 ISPS-A 시뮬레이터를 통하여 검증이 가능하다.

면적 제약과 인터페이스의 타이밍 제약을 만족시키도록 하기위하여 스케쥴링은 pre-scheduling과 post-scheduling의 두 부분으로 나누어 각각의 제약을 만족시킬 수 있게 하였다. Pre-scheduling에서는 연산들의 상대적 순서를 결정하는 데에 엔트로피의 개념을 써서 연산자들이 고르게 분포될 수 있도록 하여 사용되는 하드웨어의 수를 줄일 수 있도록 하고 입력 연산에의 스케쥴링 알고리듬 적용을 보였으며, post-scheduling 과정에서는 최대/최소 타이밍 제약을 만족시키기 위하여 relative 스케쥴링 알고리듬을 이용하였다. 모듈 할당 과정에서는 기능이나 면적 또는 지연 시간 면에서 유사한 연산자끼리 하드웨어를 공유할 수 있도록 similarity의 개념을 이용하였으며, 출력 포트 레지스터와 내부 연산 레지스터간 공유할 수 있도록 하는 과정을 보였다.

시스템의 합성 결과에서 ISyn은 내부 연산 위주의 행위 기술은 물론 외부 인터페이스 제약 기술에 대해서도 효율적인 합성이 가능함을 보였다.

ISyn은 앞으로 다중 프로세스 지원을 위한 병렬문에 대한 합성과, 하위 레벨의 합성 시스템과의 효율적인 연결 작업 그리고 사용자 인터페이스에 관한 연구가 진행되어져야 할 것이다.

參 考 文 獻

[1] C. Tseng, D. P. Siewiorek, "Automated Synthesis of Data Paths in Digital Systems," *IEEE Trans. CAD*, vol. 5, no. 3, July 1986, pp. 379-395.

[2] M. C. McFarland, "The High Level Synthesis of Digital Systems," *IEEE Proc.*, vol. 78, no. 2, Feb. 1990, pp. 301-318.

[3] P. G. Paulin, "HAL: A Multi-paradigm Approach to Automatic Data Path Synthesis," in Proc. *23th ACM/IEEE Design Automation Conference*, June 1886, pp. 263-270.

[4] H. D. Lee, H. S. Jun, S. Y. Hwang, "Datapath Synthesis in Sogang Silicon Compiler," in Proc. *JTC-CSCC*, Dec. 1990, pp. 430-435.

[5] K. S. Hwang, A. E. Casavant, "Scheduling and Hardware Sharing in Pipelined Data Paths," in Proc. *ICCAD*, Nov. 1989, pp. 24-27.

[6] G. Borriello, R. H. Katz, "Synthesis and Optimization of Interface Transducer Logic," in Proc. *ICCAD*, Nov. 1987, pp. 274-277.

[7] G. Borriello, "Combining Event and Data-flow Graphs in Behavioral Synthesis," in Proc. *ICCAD*, Nov. 1988, pp. 56-59.

[8] J. A. Nestor, D. E. Thomas, "Behavioral Synthesis with Interfaces," in Proc. *ICCAD*, Nov. 1986, pp. 112-115.

[9] G. De Micheli, D. Ku, "HERCULES - A System for High-Level Synthesis," in Proc. *25th Design Automation Conference*, June 1988, pp. 203-209.

[10] M. Barbacci, "Instruction Set Processor Specifications (ISPS): The Notation and its Applications," *IEEE Trans. Computers*, vol. c-30, no. 1, Jan. 1981, pp. 24-40.

[11] A. V. Aho, R. Sethi, J. I. Ullman, *Compilers : Principles, Techniques, and Tools*, Addison-Wesley Pub. : Reading, MA., 1988.

[12] 이 형종, 이 종화, 황 선영, "ISPS-A : 확장

된 ISPS 시뮬레이터의 설계," 한국정보과학회 가을 학술대회 논문집, vol. 18, no. 2, Oct. 1991, pp. 679-682.

[13] 전 홍신, 황 신영, "엔트로피에 바탕을 둔 새로운 스케쥴링 알고리듬," 한국정보과학회 봄 학술발표 논문집, vol. 18, no. 1, 1991년 4월, pp. 427-430.

[14] D. Ku, G. De Micheli, "Relative Scheduling under Timing Constraints: Algorithms for High-Level Synthesis of Digital Circuits," *IEEE Trans. CAD*, vol. 11, no. 6, June 1992, pp. 696-717.

[15] A. C. Parker, M. C. McFarland, R. Camposano, "Tutorial on High-Level Synthesis," in Proc. *ACM/IEEE Design Automation Conference*, June 1988, pp. 330-336.

[16] P. Marwedel, "A New Synthesis Algorithm for the MIMOLA Software System," in Proc. *ACM/IEEE Design Automation Conference*, June 1986, pp. 271-277.

[17] M. McFarland, "Using Bottom-Up Design Techniques in the Synthesis of Digital Hardware from Abstract Behavioral Description," in Proc. *ACM/IEEE Design Automation Conference*, 1986, pp. 474-480.

著者紹介

李 炯 鍾(準會員)

1968年 6月生. 1991年 서강대학교 전자공학과 졸업. 1993年 서강대학교 전자공학과 공학석사 학위 취득. 1993年 2월 ~ 현재 삼성전자 반도체부문 연구원. 주관심 분야는 CAD 시스템, Computer Architecture 및 System Design 등임.

李 鍾 和(正會員)

1991年 2月 서강대학교 전자공학과 졸업. 1993年 2月 서강대학교 전자공학과 공학 석사 취득. 1993年 3月 ~ 현재 서강대학교 전자공학과 대학원 박사과정 재학중. 주관심 분야는 VLSI 설계, Computer Architecture 및 Systems Design, CAD 등임.

黃 善 泳(正會員)

1976年 2月 서울대학교 전자공학과 졸업. 1978年 2月 한국 과학원 전기 및 전자 공학과 공학 석사 취득. 1986年 10月 미국 Stanford 대학 공학 박사 학위 취득. 1976年 ~ 1981年 삼성 반도체 주식회사 연구원. 1986年 ~ 1989年 Stanford 대학 Center for Integrated Systeme 연구소 연구원, Fairchild Semiconductor Palo Alto Research Center 기술자문. 1989年 3月 ~ 현재 서강 대학교 전자공학과 부교수. 주관심 분야는 CAD 시스템, Computer Architecture 및 Systems Design, VLSI 설계 등임.