신호전이그래프에 기반한 비동기식 논리합성의 고유한 특성을 고려한 신호전이그래프의 자동생성

김 의 석[†] · 이 정 근[††] · 이 동 익[†††]

요 약

신호전이그래프는 비동기식 유한상태기와 더불어 신호수준에서 비동기식 제어회로의 사양을 기술하기 위하여 사용되는 가장 대표적인 사양기술언어이다. 그러나 신호전이그래프는 설계자에게 친숙한 사양기술언어가 아니며, 결과적으로 비동기식 시스템의 설계자가 목적시스템의 비동기식 제어부를 구성하는 수~수십개의 비동기식 제어회로에 대한 신호전이그래프를 일일이 고안하고 기술하는 것은 매우 힘들고 시간소모적인 일이다. 본 논문에서는 최근에 제안된 프로세스 중심방식을 이용하여 신호전이그래프를 자동으로 생성하는 방법을 제안하고자 한다. 특히, 제안된 방법은 신호전이그래프의 자동생성 과정에서 신호전이그래프에 기반한 비동기식 논리합성의 고유한 특성들을 주의 깊게 고려하여 준다. 결과적으로 자동 생성된 신호전이그래프로부터 합성된 비동기식 제어회로는 면적, 합성시간, 성능, 구현성의 측면에서 매우 우수하다.

Automatic STG Derivation with Consideration of Special Properties of STG-Based Asynchronous Logic Synthesis

Euiseok Kim[†] · Jeong-Gun Lee[††] · Dong-Ik Lee[†††]

ABSTRACT

Along with an asynchronous finite state machine, in short AFSM, a signal transition graph, in short STG, is one of the most widely used behavioral description languages for asynchronous controllers. Unfortunately, STGs are not user-friendly, and thus it is very unwieldy and time consuming for system designers to conceive and describe manually the behaviors of a number of asynchronous controllers which constitute an asynchronous control unit for a target system in the form of STGs. In this paper, we suggest an automatic STG derivation method through a process-oriented method. Since the suggested method considers special properties of STG-based asynchronous logic synthesis very carefully, asynchronous controllers which are synthesized from STGs derived through the suggested method are superior in aspects of area, synthesis time, performance and implementability compared to those obtained through previous methods.

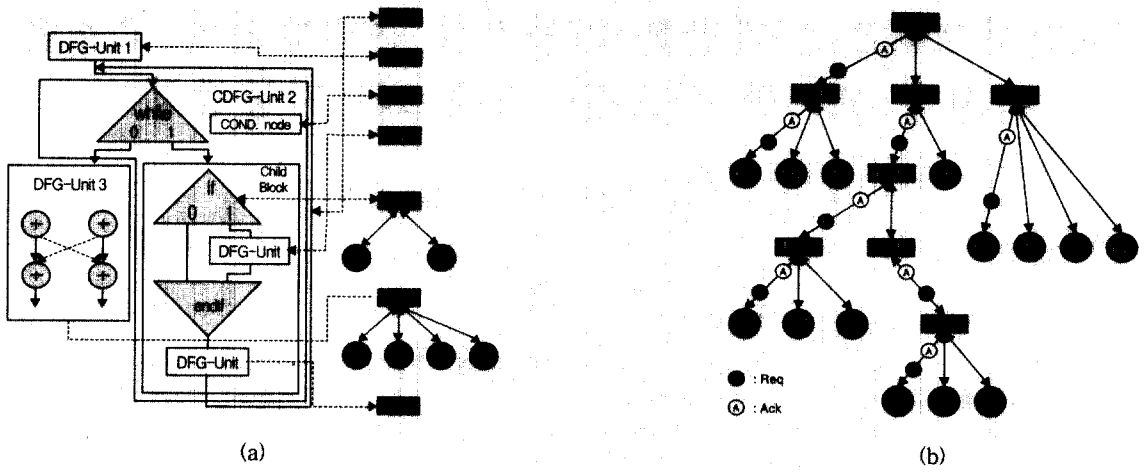
키워드 : 신호전이그래프(Signal Transition Graph), 비동기식 논리합성(Asynchronous Logic Synthesis), 제어 데이터 흐름 그래프(Control-Data Flow Graph), 자동생성(Automatic Derivation)

1. 서 론

신호전이그래프(Signal Transition Graph, STG)는 T. A. Chu[1]에 의하여 제안된 이래로 비동기식 유한상태기(Asynchronous Finite State Machine, AFSM)와 더불어 신호수준에서 비동기식 제어회로의 사양을 기술하기 위하여 사용되는 가장 대표적인 사양기술언어이다. 또한, 다수의 성공적인 신호전이그래프에 기반한 논리합성 방법들과 비동기식 논리합성기들이 존재한다[2-4]. 그러나 비동기식 유한상태기와 마찬가지로 신호전이그래프는 설계자에게 친숙한 사양기술언어가 아니며, 결과적으로 비동기식 시스템의 설계자가 목적시스템의 비동기식 제어부를 구성하는 수~수십개의 비동기식 제어회로에 대한 신호전이그래프를 일일이 고안하고 기술하는 것은 매우 힘들고 시간소모적인 일이다.

다수의 비동기식 제어회로들에 대한 신호수준의 사양을 기술하는 작업의 어려움과 소요시간을 효과적으로 단축하기 위하여, 상위수준합성 과정의 일환으로 비동기식 제어회로에 대한 신호수준의 사양을 목적시스템의 행위기술로부터 유도하는 연구가 수행되어져 오고 있다[5-8]. [5]와 [6]은 각각 데이터 흐름 그래프와 Verilog로부터 비동기식 제어회로에 대한 사양기술인 신호전이그래프를 생성한다. 그러나 이들의 방법은 신호전이그래프에 기반한 논리합성의 고유한 특성들

※ 본 연구는 광주과학기술원 초고속 광 네트워크 연구센터를 통한 한국과학재단 우수연구센터 지원금에 의한 것입니다.
† 정 회 원 : 광주과학기술원 초고속 광 네트워크 연구센터 연구교수
†† 준 회 원 : 광주과학기술원 대학원 정보통신공학과
††† 종신회원 : 광주과학기술원 정보통신공학과 교수
논문접수 : 2002년 3월 27일, 심사완료 : 2002년 9월 11일



(그림 1) (a) 제어 데이터 흐름 그래프와 프로세스 중심 제어기들 (b) 프로세스 중심 제어기들의 계층도

을 전혀 고려하지 않았으며, 결과적으로 자동생성된 신호전이그래프로부터 합성된 비동기식 제어회로는 면적, 성능, 합성시간, 구현성등의 측면에서 많은 문제점들을 야기한다. [7, 8]에서 제안한 프로세스 중심 방식은 목적시스템의 행위기술로써 주어진 제어 데이터 흐름 그래프(Control-Data Flow Graph, 이후 CDFG)로부터 계층적 분할을 통하여 분산된 비동기식 제어부에 대한 비동기식 유한상태기들의 집합을 자동생성하는 방법이다. 프로세스 중심 방식은 전체 비동기식 제어부를 계층적으로 분할·유도함으로써 효율적인 제어부의 구축을 가능하게 한다.

본 논문에서는 프로세스 중심방식의 효용성과 적용범위를 보다 높이고, 기존의 신호전이그래프 자동생성방법들[5, 6]이 야기하였던 문제점들을 효과적으로 해결하기 위하여, 비동기식 유한상태기와 더불어 대표적인 사양기술 언어인 신호전이그래프를 프로세스 중심방식을 이용하여 자동으로 생성하는 방법을 제안하고자 한다. 특히, 제안된 방법은 신호전이그래프의 자동생성 과정에서 신호전이그래프에 기반한 비동기식 논리합성의 고유한 특성들을 주의깊게 고려하여 준다. 결과적으로 자동생성된 신호전이그래프로부터 현존하는 비동기식 논리합성기를 이용하여 합성된 비동기식 제어회로들은 면적, 합성시간, 성능, 구현성의 측면에서 매우 우수하다. 아울러, 본 논문에서는 자동생성된 신호전이그래프의 합성을 통하여 획득한 비동기식 제어회로들과 기존의 프로세스 중심방식을 통하여 자동생성된 비동기식 유한상태기의 합성을 통하여 획득한 비동기식 제어회로들[7, 8]의 비교를 수행한다.

본 논문의 구성은 다음과 같다. 2절에서는 본 논문을 이해하는데 필요한 기본적인 지식들을 설명한다. 본 논문의 핵심인 3절에서는 프로세스 중심방식을 이용한 신호전이그래프의 자동생성에 관한 방법을 설명한다. 4절에서는 프로세스 중심방식을 통하여 자동생성된 신호전이그래프들과 비동기식 유한상태기들의 비교·분석을 수행한다. 5절에서는 제안된 방법의 장점을 보여주기 위하여 수행된 실험들의 결과와 본 논문의 결론을 제시한다.

2. 예비 지식

2.1 제어 데이터 흐름 그래프

본 논문에서는 [7, 8]에서와 마찬가지로 목적시스템에 대한 사양기술로써 제어 데이터 흐름 그래프(Control-Data Flow Graph, 이후 CDFG)를 가정한다. CDFG는 다음과 같이 정의되는 DFG-유닛들과 CDFG-유닛들의 순차적인 연결로 정의된다.

정의 1 : DFG - 유닛은 $\Omega = (N, E, O)$의 3 - 튜플로써 정의한다. N은 연산노드들의 집합이며, E는 연산노드들 사이의 연결들의 집합이다. O는 각각의 연산노드에서 수행되는 연산들의 집합이다.

정의 2 : CDFG - 유닛은 $\Gamma = (X, Y, Z, E)$의 4-튜플로써 정의한다. X는 제어노드들의 집합이며, Y는 조건노드들의 집합이다. Z는 제어노드들의 제어 하에 있는 Child-블록으로 X와 Y에 속한 노드들을 제외한 모든 노드들의 집합이다. E는 X, Y, Z에 속한 노드들 및 블록 사이의 연결을 나타내는 연결들의 집합이다.

정의 1에서 정의된 DFG-유닛의 연산노드들은 사실상 "연산자 획득", "연산 수행", "연산 결과의 저장"으로 이루어지는 일련의 작업들로 구성된 일종의 프로세스이다. 본 논문에서는 이후로 연산노드와 프로세스를 동일한 의미로 사용한다. 사실상 "프로세스 중심방식"이라고 하는 명칭은 DFG - 유닛의 프로세스로부터 기인한다[8]. 정의 2에서 정의된 CDFG - 유닛의 Child-블록은 일종의 CDFG이다. 그

러므로 DFG-유닛들과 CDFG-유닛들의 순차적인 연결로 정의된 CDFG는 목적시스템의 행위를 계층적으로 표현할 수 있다. 상기한 CDFG로부터 비동기식 제어회로들에 대한 올바른 사양을 추출하기 위해서는 스케줄링, 자원할당, 자원결합등과 같은 일련의 과정을 통하여 획득할 수 있는 하드웨어 구현과 관련한 정보들이 필요하다. 본 논문에서는 [7, 8]에서와 마찬가지로 이러한 정보가 이미 이용가능하다고 가정한다. CDFG에 관한 보다 자세한 설명은 [7, 8]을 참조한다. (그림 1)의 (a)는 CDFG의 예를 보여준다.

2.2 프로세스 중심방식을 이용한 비동기식 제어부의 자동생성

목적시스템에 대한 사양기술로써 주어진 제어 데이터 흐름 그래프로부터 분할된 비동기식 제어부를 유도하는 방법인 프로세스 중심방식은 제어 데이터흐름 그래프로부터 일련의 체계적이며 계층적인 분석 과정을 통하여 4가지 형태의 비동기식 제어기들, 즉, 프로세스 제어기(Process Controller, PC), 프로세스 순서 제어기(Process Sequencing Controller, PSC), 제어 노드 제어기(Control Node Controller, CNC) 및 유닛 순서 제어기(Unit Sequencing Controller, USC)들로 구성된 비동기식 제어부를 유도한다. 프로세스 제어기(PC)와 프로세스 순서 제어기(PSC)는 제어 데이터 흐름 그래프의 구성 요소인 DFG-유닛으로부터 유도하는 제어기로써, 각각 프로세스(연산노드)의 수행 및 프로세스들의 수행순서를 제어한다. 또한 제어 노드 제어기(CNC)와 유닛 순서 제어기(USC)는 제어 데이터 흐름 그래프의 또 다른 구성 요소인 CDFG-유닛과 제어 데이터 흐름 그래프로부터 유도하는 것으로 유닛들의 수행 순서를 제어한다. 이와 같은 4가지 종류의 제어기들은 목적 시스템에 대한 제어 데이터 흐름 그래프의 계층적 분석을 통하여 계층적으로 분할된 비동기식 제어회로들에 대한 신호수준의 사양기술들을 유도한 후, 이들을 기존의 비동기식 제어회로 합성기들을 이용하여 합성함으로써 획득할 수 있다. (그림 1)(a)는 목적 시스템의 사양기술인 제어 데이터 흐름 그래프의 구성요소들과 프로세스를 중심으로 계층적으로 분할된 4가지 종류의 제어기들의 대응관계를 보여주며, (그림 1)(b)는 4가지 종류의 프로세스 중심의 제어기들의 일반적인 계층도를 보여준다. 프로세스 중심방식을 통하여 유도된 비동기식 제어부의 주요한 특징은 다음과 같다.

- 비동기식 제어회로들은 "수행 제어부"와 "수행순서 제어부"로 완전히 분할된다.
- 계층적이며 균일하게 분할된 비동기식 제어부를 유도한다.
- 체계적인 분할과정을 통하여 자동 생성된다.
- 면적, 성능, 합성시간 및 구현성의 측면에서 우수하다.

프로세스 중심방식을 이용한 비동기식 제어부의 자동생성에 관한 보다 자세한 설명은 [7, 8]을 참조한다.

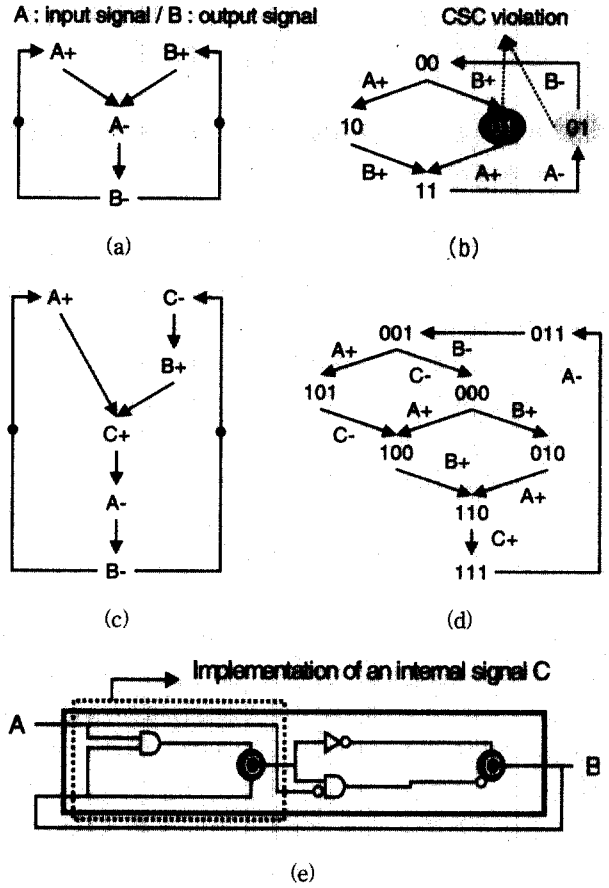
3. 프로세스 중심방식을 이용한 신호전이그래프의 자동생성

3.1 신호전이그래프와 비동기식 논리합성

신호전이그래프(Signal Transition Graph, STG)는 T. A. Chu[1]에 의하여 비동기식 제어회로의 사양기술언어로써 제안된 이래로 비동기식 유한상태기(Asynchronous Finite State Machine, AFSM)와 더불어 가장 널리 사용되는 대표적인 신호수준의 사양기술언어이다. 일종의 해석 페트리 넷(interpreted petri net)인 신호전이그래프는 비동기식 제어회로의 입·출력 신호들의 순차적 혹은 병행적 변화를 페트리 넷의 전이들의 순차적 혹은 병행적 발생을 통하여 표현한다. (그림 2)(a)는 간단한 신호전이그래프를 보여준다.

모든 종류의 신호전이그래프가 언제나 합성가능한 것은 아니며, 다음의 필요조건들, 유한성(boundedness), 지속성(persistency), 교차성(switchover), 완전상태코딩(complete state coding, CSC)성들을 만족시키고 있을때만 합성가능하다. 본 논문에서는 완전상태코딩성이 특히 중요한 의미를 가지므로 상세히 설명한다. 다른 성질들에 관한 보다 자세한 설명은 [1, 9]를 참조한다. 완전상태코딩성은 신호전이그래프를 구성신호들의 이진값을 상태로 가지는 상태그래프(State Graph, SG)로 변형하였을 때, 동일한 상태값을 가지는 상태들이 동일한 발생가능 출력신호들의 집합을 가져야 함을 의미한다. 예를 들어, (그림 2)(b)의 상태그래프는 "01"의 상태값을 가지는 2개의 상태를 가지며 이중 1개의 상태에서만 출력신호 'B-'가 발생가능하므로 완전상태코딩성을 위배한다. 일반적으로 완전상태코딩성을 만족시키지 못하고 있는 신호전이그래프는 내부신호를 삽입하여 주는 것에 의하여 인위적으로 완전상태코딩성을 만족하도록 변형하여 주어야 한다. (그림 2)(c)는 내부신호 'C'를 삽입하는 것에 의하여 (그림 2)(a)의 신호전이그래프가 완전상태코딩성을 만족하도록 변형시켜준 것이다.

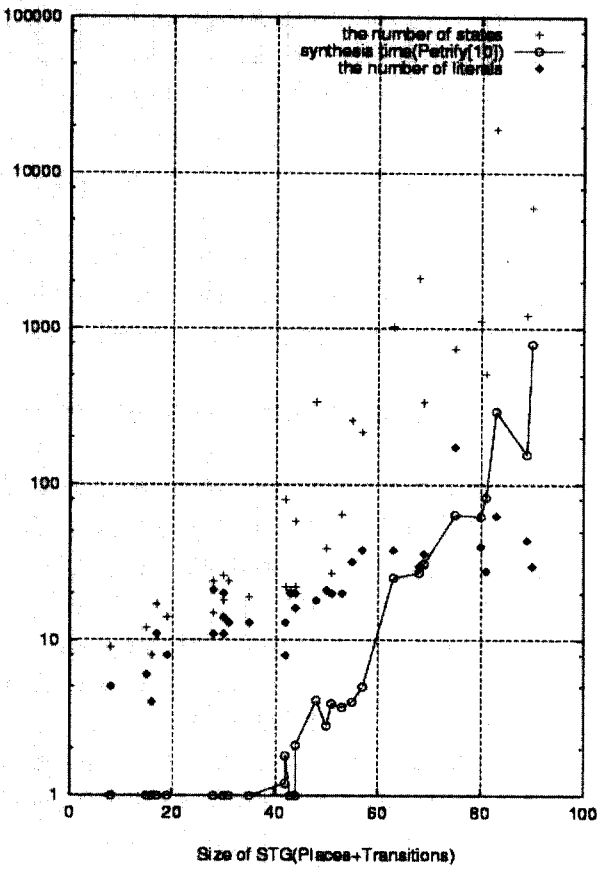
본 저자들은 기존의 신호전이그래프 자동생성 방법들이[5, 6] 신호전이그래프 기반의 비동기식 논리합성 과정에서 야기할 수 있는 두 가지 문제점들을 지적하고자 한다. 첫 번째 문제점은 자동생성된 신호전이그래프가 완전상태코딩성의 원천적 만족을 보장하지 않는다는 것이다. 완전상태코딩성은 신호전이그래프의 합성을 위한 필요조건들중의 하나이며, 만족시키지 못할 경우에는 내부신호의 삽입을 통하여 완전상태코딩성을 만족하도록 반드시 변형시켜 주어야 한다. 그러나 완전상태코딩성의 만족을 위한 내부신호의 삽입은 결코 쉬운 일이 아니다. 내부신호들의 최적 삽입을 수행하여 주는 다수의 방법론과 이들에 기반한 설계 자동화 도



(그림 2) (a) CSC-성질 위배를 가진 신호전이그래프 (b) (a)의 신호전이그래프에 대한 상태그래프 (c) CSC-성질을 만족시키는 신호전이그래프 (d) (c)의 신호전이그래프에 대한 상태그래프 (e) (c)의 신호전이그래프에 대한 비동기식 제어회로

구들이 있음에도 불구하고 현재까지는 많은 경우, 상당한 수행시간을 요구하며, 완전상태코딩성의 만족을 보장하지도 못한다. 게다가 완전상태코딩성의 만족을 위해 삽입된 내부신호는 내부신호의 구현을 위한 부가적인 회로와 이에 기인한 제어회로의 지연시간 증가라는 두 가지 심각한 문제를 야기한다. (그림 2)(e)는 새로이 삽입된 내부신호가 야기하는 두 가지 문제점을 명확히 보여준다.

두 번째 문제점은 상태공간폭발 문제이다. 상태공간폭발 문제는 주어진 문제의 해결을 위하여 탐색하여야 할 해의 공간이나 혹은 분석해야 할 중간모델의 규모가 너무 커서 적절한 길이의 시간과 적절한 규모의 컴퓨팅 자원을 이용하여 문제의 해결을 수행할 수 없는 경우를 의미한다. 비동기식 제어회로의 입·출력 신호들의 병행적인 변화를 기술하는 신호전이그래프의 합성을 위해서는 신호들의 이진값으로 구성된 상태들의 집합을 획득해야 한다. 이때, 상태들의 개수는 신호전이그래프의 크기에 대하여 지수적으로 증가할 수 있으며, 상태개수의 지수적 증가는 제어회로의 합성에 소요되는 시간을 급속히 증가시켜 회로의 합성을 어



(그림 3) 신호전이그래프의 크기 증가에 따른 합성시간과 상태개수의 증가

렵거나 혹은 불가능하게 한다. (그림 3)은 신호전이그래프의 대표적인 벤치마크인 SIS[3] 벤치마크에 속한 신호전이그래프들에 대하여 신호전이그래프의 크기 증가에 따른 상태개수의 증가와 합성시간의 변화를 보여준다. 참고로 합성에 사용된 논리합성기는 가장 대표적인 비동기식 제어회로의 논리합성기인 Petrify[10]를 사용하였다. Petrify[10]는 상태공간폭발 문제에 효과적으로 대처하기 위하여 이진결정그래프(Binary Decision Diagram, BDD)를 사용한다. 그럼에도 불구하고 (그림 3)이 보여주는 것처럼 합성시간과 상태개수는 신호전이그래프의 크기 증가와 더불어 급속도로 증가한다.

본 논문에서 제안하고자 하는 프로세스 중심방식에 기반한 신호전이그래프의 자동생성은 "완전상태코딩성"과 "상태공간폭발 문제"를 주의깊게 고려하였으며, 이는 제안된 방법의 매우 주요한 특징이다. 기존의 신호전이그래프의 자동생성 방법들은 두 가지 문제에 대한 고려없이 이들의 해결을 비동기식 제어회로의 논리합성기에게 전적으로 의존한다. 그러나 본 부절에서 설명한 바와 같이 상기한 두 문제는 면적, 성능, 합성시간의 측면에서 심각한 문제를 야기할 수 있으며 현존하는 비동기식 논리합성기를 통하여 완벽한 해결을 보장할 수도 없다. 본 논문에서 제안된 방법은 상기

한 두 가지 문제점들을 주의깊게 고려한다. 완전상태코딩성에 대하여, 제안된 방법을 통하여 자동생성된 신호전이그래프는 원천적으로 완전상태코딩성을 만족시킨다. 게다가 비동기식 제어부에 대하여 단일 신호전이그래프가 아닌 다수의 적절한 크기를 가진 분산된 신호전이그래프들의 집합을 유도함으로써 상태공간폭발 문제를 근본적으로 해결한다.

3.2 프로세스 제어기에 대한 신호전이그래프의 자동생성

프로세스 제어기(process controller, PC)는 DFG-유닛의 한 프로세스를 수행하는데 필요한 일련의 제어신호들을 입·출력하는 비동기식 제어회로이다. 2.1에서 설명했던 바와 같이, 연산노드에 대응하는 프로세스의 수행은 "연산자 획득", "연산 수행", "연산 결과의 저장"의 세 가지 과정으로 이루어지며, 프로세스 제어기는 이들의 수행에 필요한 제어신호들의 입·출력을 담당한다. 프로세스 제어기의 동작 및 이에 관련한 입·출력 신호들은 다음과 같다.

[단계 1] 프로세스 제어기의 활성화 : 프로세스 제어기는 해당 프로세스가 속한 DFG-유닛에 대응하는 프로세스 순서 제어기로부터 요구신호 'Req+'를 받음으로써 활성화 된다.

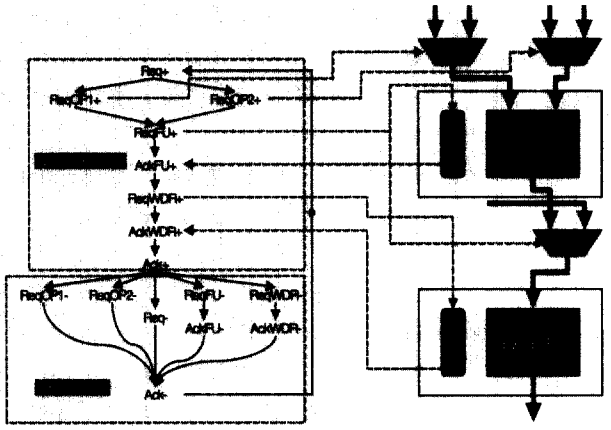
[단계 2] 피연산자의 획득 : 프로세스 제어기는 연산의 수행에 필요한 피연산자들을 취하기 위하여 '$ReqOP_1$+', '$ReqOP_2$+', …, '$ReqOP_n$+'를 발생시키며, 이들은 MUX에 연결되어 입력선택을 위한 제어신호로써 작용한다.

[단계 3] 연산의 수행 : 프로세스 제어기는 연산의 수행을 담당할 연산모듈을 활성화시키기 위하여 비동기식 연산모듈의 활성화 신호인 'ReqFU+'를 발생시킨다.

[단계 4] 연산 결과의 저장 : 연산의 수행이 끝나면, 프로세스 제어기는 연산의 수행 결과를 저장하기 위하여 ReqWDR+를 발생시키며, 이는 목적 레지스터의 활성화 신호로써 사용된다. 연산 결과를 저장한 후에, 프로세스 제어기는 프로세스 순서 제어기에 대한 응답신호 Ack+를 발생시킨다.

[단계 5] 제어신호들의 리셋 : 단계 1~4의 과정을 통하여 연산의 수행을 완료한 후, 프로세스 제어기는 모든 출력신호들을 '0'의 값으로 리셋 하여준다. 이 과정에서, 프로세스 순서 제어기로부터 요구신호 'Req-'를 받으며 이에 대한 응답신호 'Ack-'를 발생하여 준다.

(그림 4)는 프로세스 제어기의 동작에 대응하는 신호전이그래프와 프로세스의 수행을 위한 데이터경로(datapath)를 보여준다. 본 논문에서 가정하고 있는 데이터경로는 입·출력부와 연산부로 이루어진다. 입·출력부는 양의 모서리 작동 레지스터(positive edge-triggered register)와 레지스터의 입·출력단에 연결된 MUX로 구성된다. 연산부는 연산의 수행을 담당하는 비동기식 연산모듈들로 구성된다. 일반적으로 비동기식 연산모듈의 제작방식에는 묶음데이터(bundled data) 방식과 이중회선(dual-rail)[11] 방식이 존재한다. 기존 회로의 1비트 데이터를 2비트의 코딩된 데이터로 표현하는 이중회선 방식은 평균동작시간에 동작하는 비동기식 연산모듈의 제작을 가능하게 하지만 최소 2배이상의 면적을 요구하는 치명적인 단점을 가진다. 그러므로 본 논문에서는 묶음데이터 방식을 가정한다. 참고로, 묶음데이터 방식에 기반한 비동기식 연산모듈은 동기식 연산모듈과 최악지연시간(worst case delay)에 대응하는 지연소자로 구성된 연산모듈로써, 입력데이터와 연산요구 신호가 함께 가해지면 지연소자의 지연시간에 대응하는 시간이 경과한 후에 연산의 결과값과 연산완료 신호를 생성한다.



(그림 4) 프로세스 제어기에 대한 신호전이그래프와 데이터경로

(그림 4)의 신호전이그래프가 올바른 프로세스 제어기로 합성되기 위해서는 다음의 4가지 성질들, 유한성(boundedness), 지속성(persistency), 교차성(switchover), 완전상태코딩성(complete state coding, CSC)을 반드시 만족시켜주어야 한다[1, 9]. 다음의 명제는 프로세스 제어기에 대한 신호전이그래프가 상기한 4가지 성질을 근본적으로 만족시킴을 보여주며, 결과적으로 프로세스 제어기에 대한 신호전이그래프는 올바른 비동기식 제어회로로 합성가능하다.

명제 1 : 프로세스 제어기에 대한 신호전이그래프는 유한성, 지속성, 교차성, 완전상태코딩성을 만족시킨다.

증명 : 유한성 - 프로세스 제어기에 대한 신호전이그래프는 일종의 밀결합(strongly connected) 마크드 그래프이다. 그러므로 해당 신호전이그래프는 구조적으로 유한성을 만족시킨다.[12]

증명 : 지속성 - 프로세스 제어기에 대한 신호전이그래프에서 임의의 전이 발생(firing)도 발생가능한 전이를 발

생가능하지 않도록 만들지 않는다. 그러므로 해당 신호전이그래프는 본질적으로 지속성을 만족시킨다.

증명 : 교차성 - 프로세스 제어기에 대한 신호전이그래프는 각각의 입·출력 신호들에 대하여 한번씩의 상승전이와 하강전이를 가지며, 이들은 언제나 작업구간(working phase)과 휴식구간(idling phase)으로 나뉘어 교차적으로 발생한다. 그러므로 해당 신호전이그래프는 교차성을 만족시킨다.

증명 : 완전상태코딩성 - 프로세스 제어기에 대한 신호전이그래프는 (그림 4)가 보여주는 것처럼 작업구간(working phase)과 휴식구간(idling phase)으로 구성된다. 이때, 신호전이그래프의 모든 상승전이들은 작업구간에서 발생하며, 모든 하강전이들은 휴식구간에서 발생한다. 작업구간과 휴식구간의 마지막 상승전이와 하강전이가 각각 'Ack+'와 'Ack-'이므로 서로 다른 구간에서 생성된 상태들은 언제나 다른 이진 상태값을 가진다. 또한 같은 구간에서 생성된 모든 상태들은 서로 다른 이진 상태값을 가진다. 그러므로 해당 신호전이그래프로부터 생성된 모든 상태들은 언제나 다른 이진 상태값을 가지며, 완전상태코딩성을 만족시킨다.

3.3 프로세스 순서 제어기에 대한 신호전이그래프의 자동생성

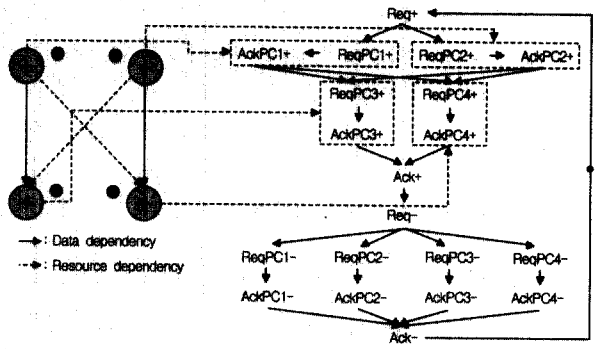
프로세스 순서 제어기(process sequencing controller, PSC)는 DFG-유닛에 대응하는 비동기식 제어기로, DFG-유닛에 속한 프로세스들에 대응하는 프로세스 제어기들을 프로세스 사이의 데이터 혹은 자원 의존관계에 기반한 순서에서 활성화시키는 기능을 수행한다. DFG-유닛으로부터 프로세스 순서 제어기에 대한 신호전이그래프의 유도는 4 단계 핸드셰이크 프로토콜에 기반한 핸드셰이크 확장을 통하여 수행된다. 다음은 DFG-유닛으로부터 프로세스 순서 제어기에 대한 신호전이그래프를 자동으로 생성하는 알고리즘이다.

알고리즘 1 : DFG-유닛으로부터 프로세스 순서 제어기에 대한 신호전이그래프의 자동생성

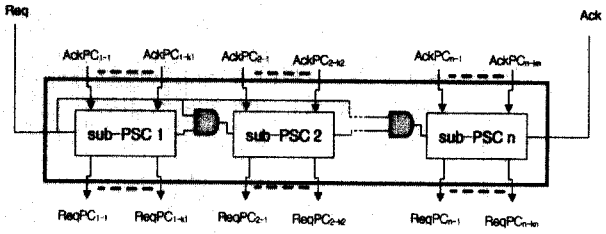
[단계 1] 프로세스 순서 제어기에 대한 활성화 요구신호 및 응답신호인 'Req+'와 'Ack+'를 각각 생성한다.

[단계 2] DFG-유닛에 속한 각각의 프로세스 P_i에 대응하는 프로세스 제어기 PC_i에 대하여 $ReqPC_i+ \rightarrow AckPC_i+$의 일련의 전이들의 연결을 생성한다.

[단계 3] DFG-유닛에 속한 두 개의 프로세스 P_i와 P_j가 P_i로부터 P_j로 데이터 혹은 자원의존 관계로 연결되어져 있다면, '$AckPC_i$+'로부터 '$ReqPC_j$+'로의 연결을 만들어 준다.



(그림 5) DFG-유닛으로부터 프로세스 순서 제어기에 대한 신호전이그래프의 자동생성



(그림 6) 분할된 프로세스 순서 제어기의 구조[8]

[단계 4] 프로세스 P_i에 대하여 데이터 혹은 자원의존 관계의 측면에서 선행하는 프로세스가 존재하지 않는다면 'Req+'로부터 '$ReqPC_i$+'로의 연결을 만들어 준다.

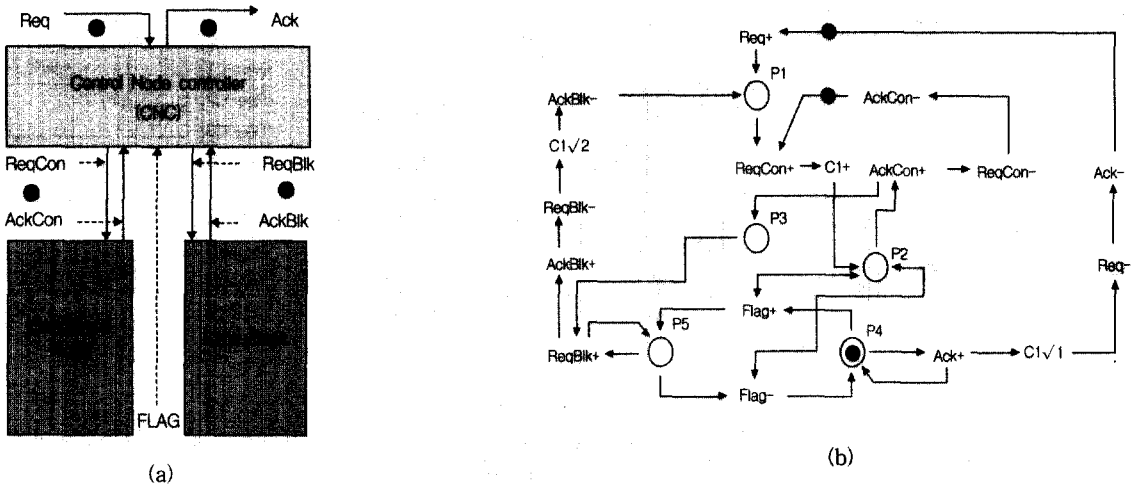
[단계 5] 프로세스 P_i에 대하여 데이터 혹은 자원의존 관계의 측면에서 후행하는 프로세스가 존재하지 않는다면 '$AckPC_i$+'로부터 'Ack+'로의 연결을 만들어 준다.

[단계 6] 프로세스 순서 제어기에 대한 활성화 요구신호 및 응답신호에 대한 리셋 신호인 'Req-'와 'Ack-'를 각각 생성한다.

[단계 7] 모든 프로세스 P_i에 대하여 $ReqPC_i- \rightarrow AckPC_i-$를 생성한다. 생성된 모든 '$ReqPC_i$-'들에 대하여 'Req-'로부터의 연결 $Req- \rightarrow ReqPC_i-$를 생성한다. 또한 생성된 모든 '$AckPC_i$-'들로부터 'Ack-'로의 연결들을 생성한다.

[단계 8] 'Ack+'와 'Req-', 'Ack-'와 'Req+'의 사이에 연결을 생성한다. 마지막으로 'Ack-'와 'Req+'의 연결에 토큰을 생성한다.

(알고리즘 1)을 통하여 DFG-유닛으로부터 대응하는 프로세스 순서 제어기에 대한 신호전이그래프를 자동으로 생성할 수 있으며, (그림 5)는 (알고리즘 1)의 적용 예를 보여준다. (알고리즘 1)을 통하여 유도된 신호전이그래프는 완전상태코딩성을 포함하여 논리합성을 위해 필요한 4가지 조건을 모두 만족시키며, 이는 명제 1의 증명과 유사한 방식으로 증명가능하다. (알고리즘 1)이 체계적인 과정을 통



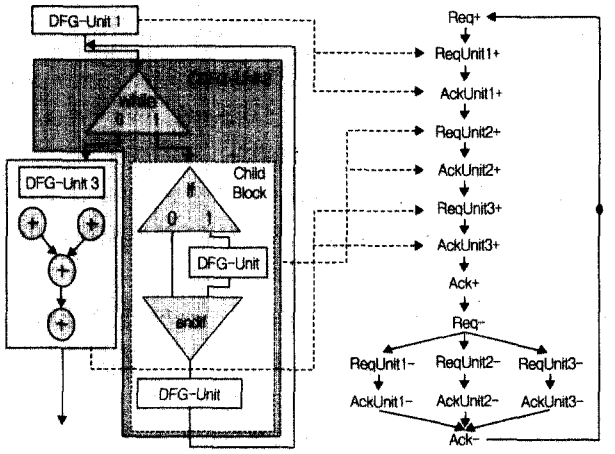
(a) (b)

(그림 7) (a) 제어 노드 제어기의 일반적인 동작 (b) WHILE-CNC에 대한 신호전이그래프

하여 프로세스 순서 제어기에 대한 신호전이그래프를 생성함에도 불구하고 생성된 신호전이그래프는 상태공간폭발 문제를 야기할 수 있다. 즉, (알고리즘 1)을 통하여 생성된 신호전이그래프는 DFG-유닛의 크기에 비례하므로 CDFG로부터 분할된 DFG-유닛의 크기가 클 경우에, 대응하는 신호전이그래프의 크기도 함께 증가하며 결과적으로 상태공간폭발 문제를 야기할 수 있다. 본 논문에서는 이러한 문제를 해결하기 위하여 [8]에서 제안된 프로세스 순서 제어기의 분할방법을 동일하게 사용하며 (그림 6)은 분할된 프로세스 순서 제어기의 구조를 보여준다. 프로세스 순서 제어기를 구성하는 서브 프로세스 순서 제어기들은 상대적으로 적은 크기이므로, 대응하는 신호전이그래프의 크기도 작으며 상태공간폭발 문제를 효과적으로 피할 수 있다. (그림 6)의 분할된 프로세스 순서 제어기는 원래의 프로세스 순서 제어기와 동일하게 동작한다.

3.4 제어 노드 제어기에 대한 신호전이그래프의 자동생성

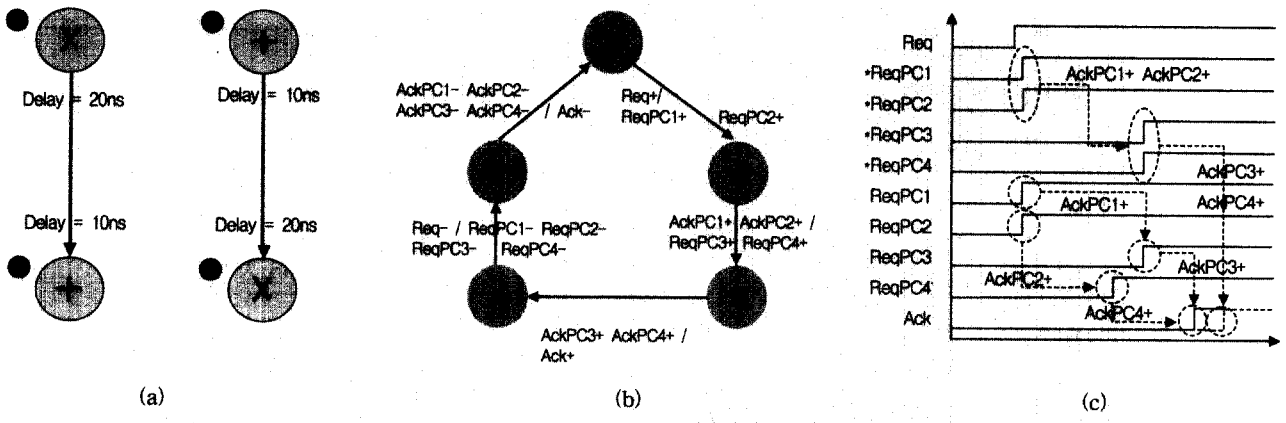
제어 노드 제어기(control node controller, CNC)는 CDFG의 제어노드에 대응하는 제어기로 사전에 정해진 순서에 의거하여 Child-블록의 수행을 제어하여 주는 비동기식 제어회로이다. 제어 노드 제어기의 일반적인 동작은 ①요구신호의 입력을 통한 활성화, ②조건노드의 수행을 통한 조건의 만족여부의 검사, ③조건의 만족여부에 따른 Child-블록의 수행, ④응답신호의 생성, ⑤입·출력 신호의 리셋으로 구성된다. (그림 7)(a)는 제어 노드 제어기의 일반적인 동작을 보여준다. (그림 7)(b)는 다양한 제어 노드 제어기들 가운데 일반적인 'while'문의 동작을 수행하는 'WHILE-CNC'에 대한 신호전이그래프를 보여준다. 제어 노드 제어기는 사실상 일종의 매크로 모듈이므로 사용자의 요구에 따라 쉽게 확장가능하다. (그림 7)(b)가 보여주는 것처럼 해당 신호전이그래프는 논리합성을 위한 4가지 조건들을 모두 만족시키도록 원천적으로 기술되었으므로, 올바른 비동기식 제어회로로 합성가능하다.



(그림 8) CDFG로부터 유닛 순서 제어기에 대한 신호전이그래프의 생성

3.5 유닛 순서 제어기에 대한 신호전이그래프의 자동생성

유닛 순서 제어기(unit sequencing controller, USC)는 프로세스 순서 제어기가 프로세스 제어기들의 순서를 제어하는 것처럼, CDFG를 구성하는 CDFG-유닛과 DFG-유닛에 대한 제어기들인 제어 노드 제어기와 프로세스 순서 제어기들의 순서를 제어하는 비동기식 제어회로이다. 2장에서 설명했던 바와 같이 CDFG는 CDFG-유닛과 DFG-유닛들의 순차적 연결이다. 그러므로 CDFG에 대응하는 유닛 순서 제어기에 대한 신호전이그래프는 제어 노드 제어기들과 프로세스 순서 제어기들을 순차적으로 활성화시키는 동작을 기술한다. (알고리즘 2)는 CDFG로부터 유닛 순서 제어기에 대한 신호전이그래프를 자동으로 생성하는 알고리즘이다.



(그림 9) (a) DFG-유닛 (b) (a)의 DFG-유닛에 대응하는 프로세스 순서 제어기에 대한 비동기식 유한상태기 (c) 비동기식 유한상태기로부터 유도된 프로세스 순서 제어기(*)와 신호전이그래프로부터 유도된 프로세스 순서 제어기의 타이밍 다이어그램

알고리즘 2 : CDFG로부터 유닛 순서 제어기에 대한 신호전이그래프의 자동생성

[단계 1] CDFG를 동일한 계층적 레벨에 속한 CDFG-유닛과 DFG-유닛들의 집합으로 분할한다. 분할 후, 유닛 사이의 순차적 연결관계에 의거하여 1, 2, …, n과 같이 순번을 매긴다.

[단계 2] 활성화 요구신호 'Req+'와 응답신호 'Ack+'를 각각 생성한다. 그리고 분할된 유닛들에 대하여 Req $Unit_i$+ → $AckUnit_i$+를 생성한다.

[단계 3] Req+ → $ReqUnit_1$+, $AckUnit_n$+ → Ack+, $AckUnit_i$+ → $ReqUnit_{i+1}$+ (i = 1 … n-1)들의 일련의 연결을 생성한다.

[단계 4] 활성화 요구신호 및 응답신호에 대한 리셋 신호인 'Req-'와 'Ack-'를 각각 생성한다.

[단계 5] 모든 유닛들에 대하여 $ReqUnit_i$- → $AckUnit_i$-를 생성한다. 생성된 모든 $ReqUnit_i$-들에 대하여 'Req-'로부터의 연결 Req- → $ReqUnit_i$-를 생성한다. 또한 생성된 모든 '$AckUnit_i$-'들로부터 'Ack-'로의 연결들을 생성한다.

[단계 6] 'Ack+'로부터 'Req-'로의, 'Ack-'로부터 'Req+'로의 연결을 생성한다. 마지막으로 'Ack-'와 'Req+'의 연결에 토큰을 생성한다.

알고리즘 2를 통하여 획득한 신호전이그래프는 합성을 위해 필요한 4가지 조건들을 모두 만족시키며 이는 명제 1과 유사한 방식을 통하여 증명할 수 있다. (그림 8)은 알고리즘 2를 통하여 CDFG로부터 유닛 순서 제어기에 대한 신호전이그래프를 생성하는 예를 보여준다.

3절에서는 프로세스 중심방식을 이용하여 제어 데이터 흐름 그래프로부터 목적 시스템의 비동기식 제어부를 구성하는 비동기식 제어기들에 대한 신호전이그래프들을 유도하는 방법을 설명하였다. 자동 생성된 신호전이그래프들은 Petrify[10]등과 같은 현존하는 비동기식 논리합성기들을 이용하여 비동기식 제어기들로 합성될 수 있다. 비동기식 제어기들은 임의의 게이트 지연가정하에서 상호간에 4단계 핸드셰이크 규약에 기반하여 입·출력 신호들을 주고받으면서 비동기식 제어부의 역할을 수행한다.

4. 프로세스 중심방식을 통하여 자동생성된 신호전이그래프와 비동기식 유한상태기의 비교

본 논문에서 신호전이그래프를 유도하기 위하여 사용한 프로세스 중심방식은 비동기식 유한상태기의 자동생성을 위한 방법으로 [8]에서 제안되었다. 본 절에서는 프로세스 중심방식을 통하여 자동생성된 신호전이그래프들과 비동기식 유한상태기들을 합성하여 획득한 비동기식 제어기들을 병행성의 측면에서 분석 비교하고자 하며, 이를 통하여 [8]의 문제점을 지적하고 적절한 해결책을 제시하고자 한다. 양자의 합성방법이 근본적으로 다르므로 합성된 회로를 면적, 합성시간등의 측면에서 비교하는 것은 사실상 무의미함을 명심한다. (그림 9)(a)와 (b)는 DFG-유닛과 [8]에서 제안된 방법을 통하여 획득한 (a)의 DFG-유닛에 대응하는 프로세스 순서 제어기에 대한 비동기식 유한상태기를 보여준다. 비동기식 유한상태기는 버스트모드(Burst Mode)에 기반하고 있다. 버스트모드에서는 신호들 사이의 병행적 동작은 오직 동일한 버스트에 속한 신호들로 한정되며, 다른 버스트에 속한 신호들은 언제나 순차적으로 발생한다. 예를 들어, (그림 9)(b)에서 'ReqPC1+'와 'ReqPC2+'는 동일한 버스트에 속하여 있으므로 서로 병행적으로 발생할 수 있으며, 'ReqPC1+'와 'AckPC1+'는 서로 다른 버스트에 속하여 있으므로 순차적으로 발생한다. 버스트모드의 이러한 특징은 프로세스 순서 제어기에 대한 비동기식 유한상태기가, 대응하는 DFG-유닛의 병행성을 손실없이 나타내는 것을 어렵게 한다. 예를 들어, (그림 9)(a)에서 프로세스 1은 프로세스 2,

〈표 1〉 하드웨어 중심 방식에 기반한 비동기식 제어회로들에 대한 신호전이그래프들과 합성결과

STG 이름	장소+전이	Literal들의 개수	CSC Signals	CSC Literals	Fanin		합성시간
					6, 7	≥ 8	
CP_{fu1}	47	28	3	15	1	0	17.70 sec.
CP_{fu2}	70	60	6	23	2	0	204.60 sec.
CP_{fu3}	89	75	9	43	3	0	664.07 sec.
CP_{fu4}	112	185	12	80	4	2	2273.45 sec.
CP_{fu5}	128	325	14	87	13	5	6262.50 sec.
CP_{reg1}	52	25	4	15	1	0	28.3 sec.
CP_{reg2}	81	70	5	28	0	2	213.87 sec.
CP_{reg3}	105	201	8	59	2	5	1397.9 sec.
CP_{reg4}	140	187	13	76	1	4	4502.00 sec.
CP_{reg5}	175	354	15	88	6	7	9140.30 sec.

- CSC Signals은 완전상태코딩 성질의 해결을 위하여 삽입된 내부신호들의 개수를 의미한다.
- CSC Literals은 삽입된 내부신호들의 구현을 위해 필요한 Literal들의 개수를 의미한다.
- CP는 하드웨어 중심방식에 기반한 비동기식 제어회로들을 나타낸다.

4와 병행관계에 있다. 이때, 프로세스 2와 4는 서로 순차관계에 있으므로 서로 다른 버스트에 속하여야 한다. 결과적으로, 프로세스 2만이 프로세스 1과 동일한 버스트에 속하게 되며 프로세스 4는 프로세스 1과 불필요하게 순차관계에 놓이게 된다. 프로세스 2와 3의 관계도 마찬가지이다. 이러한 현상은 합성된 회로의 성능감소와 직접적으로 연결된다. 신호전이그래프의 경우 구조적으로 모든 종류의 병행관계를 표현할 수 있으므로 원천적으로 이러한 문제를 해결할 수 있다. 예를 들어, (그림 9)(a)의 DFG-유닛에서 프로세스 1과 4가 지연시간이 20ns인 곱셈기들에 의하여 수행되고, 프로세스 2와 3이 지연시간이 10ns인 가산기들에 의하여 수행된다고 가정하였을 때, 비동기식 유한상태기로부터 합성된 프로세스 순서 제어기는 1, 2와 3, 4를 함께 처리하여 줌으로써 사실상 40ns의 critical path를 가지게 되는 반면, 신호전이그래프로부터 합성된 프로세스 순서 제어기는 DFG-유닛이 가진 원래의 병행성을 손실없이 보전하여 줌으로써 30ns의 critical path를 가지게 된다. (그림 9)(c)의 타이밍 다이어그램은 이러한 모습을 명확히 보여준다.

[8]에서 제안된 방법의 상기한 것과 같은 문제는 프로세스 순서 제어기에 한정하여 비동기식 유한상태기 대신에 본 논문에서 제안된 방법을 통하여 유도된 신호전이그래프로부터 합성된 비동기식 회로를 사용함으로써 해결할 수 있다.

5. 실험결과 및 결론

본 논문에서는 프로세스 중심 방식을 이용하여 제어 데이

〈표 2〉 본 논문에서 제안된 방식을 통하여 유도된 비동기식 제어회로들에 대한 신호전이그래프들과 합성결과

STG 이름	장소+전이	Literal들의 개수	CSC Signals	CSC Literals	Fanin		합성시간
					6, 7	≥ 8	
PC_c	37	15	0	0	0	0	1.25 sec.
PC_a	17	5	0	0	0	0	0.23 sec.
PSC_2	26	4	0	0	0	0	0.51 sec.
PSC_4	46	14	0	0	0	0	2.82 sec.
PSC_8	86	30	0	0	0	1	58.59 sec.
$DPSC_8$	92	28	0	0	0	0	5.64 sec.
WHILE-CNC	34	27	0	0	0	0	1.21 sec.
USC_2	25	6	0	0	0	0	0.45 sec.
USC_4	43	12	0	0	0	0	2.14 sec.
USC_8	79	24	0	0	0	1	54.84 sec.
$DUSC_8$	86	24	0	0	0	0	4.28 sec.

- PC_c는 덧셈, 곱셈등과 같은 일반적인 연산의 수행을 위한 프로세스 제어기의 신호전이그래프를 나타내며, PC_a은 저장연산의 수행을 위한 프로세스 제어기의 신호전이그래프를 나타낸다.
- $DPSC_8$과 $DUSC_8$은 각각 PSC_8과 USC_8의 분할된 구현에 대응하는 신호전이그래프들의 집합을 나타낸다.

〈표 3〉 하드웨어 중심 방식(↑)과 본 논문에서 제안된 방식(↟)을 통하여 유도된 비동기식 제어부들에 대한 합성결과

STG 이름	Literal들의 개수	면적	평균 입·출력 반응시간	하드웨어 할당			합성시간
				가산기	곱셈기	레지스터	
↑3FIR	240	638.93	2.46 ns	1	2	3	278.3 sec.
↟3FIR	85	129.72	0.68 ns				5.38 sec.
↑5FIR	500	1216.34	2.59 ns	2	2	5	1051.16 sec.
↟5FIR	155	228.53	0.68 ns				48.42 sec.
↑2IIR	410	997.28	2.38 ns	2	2	4	289.2 sec.
↟2IIR	137	204.24	0.69 ns				28.8 sec.
↑3IIR	636	1429.06	2.62 ns	2	2	6	711.2 sec.
↟3IIR	184	298.34	0.68 ns				69.92 sec.

- 면적의 기본단위는 2-입력 NAND 게이트의 면적이다.

터 흐름 그래프로부터 목적 시스템의 비동기식 제어부를 구성하는 비동기식 제어기들에 대한 신호전이그래프들을 유도하는 방법을 제안하였으며, 주요한 특징은 다음과 같다.

- 자동 생성된 신호전이그래프는 완전상태코딩 성질을 비롯하여 신호전이그래프의 비동기식 논리합성의 필요조건들을 원천적으로 만족시켜준다.
- 자동 생성된 신호전이그래프는 결코 상태공간폭발 문제를 야기하지 않는다.

제안된 방법이 신호전이그래프에 기반한 비동기식 논리합성의 고유한 특징들을 주의깊게 고려하고 있음은 본 방

법의 매우 중요한 장점이다. 본 절에서는 두 가지 실험들을 통하여 기존의 방법을 통해 획득한 비동기식 제어회로들에 대한 신호전이그래프들과 본 논문에서 제안된 방법을 통하여 획득한 신호전이그래프들과의 비교를 수행하며, 이를 통하여 비동기식 논리합성의 고유한 특징들에 대한 고려가 합성의 결과에 어떠한 영향을 미치는지 상세히 살펴보도록 한다.

첫 번째 실험에서는, 기존의 신호전이그래프 자동생성 방법인 하드웨어 중심방식[5][1]과 본 논문에서 제안된 방식을 통하여 각각 생성된 신호전이그래프의 비교 분석을 수행하였으며, <표 1>과 <표 2>는 분석결과를 보여준다. <표 1>의 신호전이그래프는 하나의 연산기(혹은 레지스터)가 N번의 연산(혹은 연산 결과의 저장)을 수행한다고 가정하였을 때, 연산기(혹은 레지스터)에 대한 비동기식 제어회로에 대응하는 신호전이그래프(CP_{fuN} 혹은CP_{regN})이다. 생성된 신호전이그래프는 완전상태코딩 성질에 관한 고려가 없으며, 결과적으로 다수의 완전상태코딩 성질의 위배가 발생한다. [5]에서는 완전상태코딩 성질의 위배 문제를 기존의 비동기식 설계자동화 도구에 전적으로 의존하고 있으나, 이는 부가적인 합성시간과 회로면적을 요구한다. <표 1>은 완전상태코딩 성질의 해결을위해 전체 회로의 평균 41.53%를 차지하는 부가회로가 필요함을 보여준다. <표 1>의 신호전이그래프들은 자원의 제약하에서 시스템의 사양인 CDFG가 증가한다면, 유도되는 신호전이그래프들의 크기도 그에 따라 선형적으로 증가함을 보여준다(CP_{fu1}~CP_{fu5}, CP_{reg1}~CP_{reg5}). 비동기식 제어회로의 입·출력 신호들의 병행적인 동작을 기술하는 신호전이그래프 크기의 선형적 증가는 합성에 필요한 상태공간 크기의 지수적 증가를 야기하며, 결과적으로 상태공간폭발 문제를 일으킬 수 있다. 상태공간폭발의 문제와 완전상태코딩 성질의 불만족은 비동기식 제어회로의 논리합성 시간의 급속한 증가를 야기하며, <표 1>은 이를 명확히 보여준다. 상기한 두 가지 문제 이외에도 <표 1>의 신호전이그래프들은 구현성의 측면에서 문제점을 가진다. 일반적으로, 비동기식 제어회로들은 해저드없는 제어회로의 구현을 위해 2-레벨 SOP 구조에 기반한 회로구현을 가정한다. 그러므로 합성된 회로의 규모가 클 경우에는, 합성된 회로의 구현과정에서 회로 라이브러리가 지원하지 아니하는 다수의 고-입력 팬인게이트들이 요구될 수 있다. 그러나 비동기식 제어회로를 구성하는 게이트들은 해저드의 위험으로 말미암아 함부로 게이트 분할을 수행할 수 없다. 그러므로 합성된 회로의 구현과정에서 다수의 고-입력 팬인 게이트(fanin ≥ 6)들이 요구될 경우, 해저드 없는 올바른 비동기식 제어회로의 구현은 매우 어려워진다. <표 1>은 해당 신호전이그래프들의 합성 결과들이 다수의 고-입력 팬인 게이트를 요구하고 있음을 보여준다. 하드웨어 중심 방식을 이용하여 생성된 신호전이그래프들과는 달리 본 논문에서 제안된 방식을 이용하여 생성된 신호전이그래프들은 상기한 세 가지 문제점을 가지지 않는다. <표 2>가 보여주는 것처럼 해당 신호전이그래프들은 언제나 완전상태코딩 성질을 만족시키므로[2] 부가적인 회로나 합성시간을 요구하지 않는다. 또한, <표 2>의 신호전이그래프들은 프로세스 순서 제어기(PSC)와 유닛 순서 제어기(USC)에 대응하는 것들을 제외하고는 언제나 일정한 크기로 한정되며, 프로세스 순서 제어기와 유닛 순서 제어기의 경우에도 3장에서 설명한 방법을 이용하여 분할함으로써 언제나 제한된 규모의 신호전이그래프들을 이용하여 기술될 수 있다. 그러므로 <표 2>의 신호전이그래프들은 상태공간폭발 문제를 야기하지 않는다. 완전상태코딩 성질의 근본적 만족과 상태공간폭발 문제의 효과적인 해결은 고-입력 팬인 게이트를 요구하지 않는 단순한 형태의 비동기식 제어회로의 합성을 가능하게 함으로써 자동생성된 신호전이그래프들로부터 합성된 비동기식 제어회로의 구현성을 크게 향상시킬 수 있다. 상호간에 입도(粒度)(Granularity)가 다르므로 <표 1>과 <표 2>의 신호전이그래프들을 일대일로 비교하는 것은 무의미하다. 그러나 각각의 신호전이그래프들의 합성결과들은 하드웨어 중심 방식과 프로세스 중심 방식을 통하여 유도하고자 하는 비동기식 제어부들의 구성회로가 되므로 <표 1>과 <표 2>의 합성결과들로부터 최종적인 비동기식 제어부들의 품질(Quality)을 추정하여 볼 수 있다. <표 1>의 결과들은 비동기식 제어회로의 논리합성 과정에서 발생할 수 있는 고유한 문제들인 완전상태코딩 성질의 만족과 상태공간폭발 문제를 신호전이그래프의 생성단계에서 고려하지 않은 하드웨어 중심 방식[5]이 전체 회로면적의 상당 부분을 불필요하게 차지하는 부가회로와 합성시간의 급속한 증가 및 구현성의 감소와 같은 문제를 가지고 있음을 보여준다. 이에 반하여 <표 2>의 결과들은 본 논문에서 제안된 방법이 상기한 두 가지 문제점들을 신호전이그래프의 생성단계에서 고려함으로써 면적, 성능, 구현성 및 합성시간의 측면에서 효율적인 비동기식 제어부를 합성할 수 있음을 보여준다.

두 번째 실험에서는, 제안된 방법의 장점을 보다 명시적으로 보여주기 위하여 상위수준합성의 과정에서 자주 사용되는 FIR, IIR 필터들에 대하여 하드웨어 중심 방식[5]과 본 논문에서 제안된 방식을 각각 적용하여 그 결과들을 비교하였으며, <표 3>은 비교결과들을 보여준다. 첫 번째 실

1) [5]에서는 목적 시스템을 구성하는 각각의 하드웨어 모듈들에 대하여 신호전이그래프가 생성되므로 본 논문에서는 [5]에서 제안된 방법을 하드웨어 중심방식이라 호칭한다.

2) 완전상태코딩 성질의 만족은 3장에서 증명하였다.

험을 통하여 예측하였던 것처럼, 제안된 방법을 통하여 유도된 비동기식 제어부는 면적, 평균 입·출력 반응시간, 합성시간의 측면에서 매우 우수하였다. 특히, 면적과 평균 입·출력 반응시간은 하드웨어 중심 방식을 통하여 유도된 비동기식 제어부에 비하여 대략 20%와 27%에 불과하였다. 이러한 놀라운 결과는 제안된 방법을 통하여 유도된 신호전이그래프가 완전상태코딩 성질을 원천적으로 만족하여 부가적인 회로를 요구하지 않음에 가장 큰 원인이 있다. 또한 신호전이그래프의 생성방법으로써 사용한 프로세스 중심 방식이 계층적이며 조직적으로 비동기식 제어부를 적절한 규모를 가진 비동기식 제어회로들로 분할함에 기인한다. 게다가 첫 번째 실험의 결과들이 보여준 것처럼 상태공간 폭발 문제의 유무는 합성시간의 큰 차이로 나타남을 볼 수 있다.

본 논문에서는 제어 데이터 흐름 그래프(CDFG)로부터 최근에 제안된 프로세스 중심 방식[8]을 적용하여 비동기식 제어부에 대한 주요한 사양기술중의 하나인 신호전이그래프를 자동으로 생성하는 방법을 제안하였다. 제안된 방법의 주요한 특징은 신호전이그래프의 자동생성 과정에서 신호전이그래프의 논리합성의 고유한 특성들을 주의깊게 고려하여 줌으로써, 자동생성된 신호전이그래프로부터 면적, 합성시간, 성능, 구현성의 측면에서 매우 우수한 비동기식 제어부를 유도하는 것을 가능하게 한다. 본 절에서 수행한 두 가지 실험들은 제안된 방법의 이러한 특징을 명확하게 보여준다. 결론적으로 제안된 방법은 프로세스 중심 방식의 적용범위를 비동기식 유한상태기의 자동생성으로부터 신호전이그래프의 자동생성으로 확장하였으며, 자동생성된 신호전이그래프는 신호전이그래프 기반의 비동기식 논리합성에 매우 적합하다.

참 고 문 헌

[1] T. A. Chu, "Synthesis of self-timed VLSI circuits from graph-theoretic specifications," Ph.D. Dissertation, MIT, Jun., 1987.

[2] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, and A. Yakovlev, "Petrify : A tool manipulating concurrent specifications and synthesis of asynchronous controllers," IEICE Trans. Inf. & Syst., Vol.E.80-D, No.3, pp.315-325, Mar., 1997.

[3] E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton and A. Sangiovanni-Vincentelli, "SIS : A System for Sequential Circuit Synthesis," May, 1992.

[4] S. Jung and C. J. Myers, "Direct Synthesis of Timed Asynchronous Circuits," In IEEE International Conference on Computer Aided Design (ICCAD), pp.332-337, Nov., 1999.

[5] J. Cortadella and R. M. Badia, "An Asynchronous Architecture Model for Behavioral Synthesis," In Proceedings of European Conference on Design Automation, Mar., pp.307-311, 1992.

[6] I. Blunno and L. Lavagno, "Automated Synthesis of Micro-Pipelines from Behavioral Verilog HDL," In Proceedings of Sixth International Symposium on Advanced Research in Asynchronous Circuits and Systems, Apr., pp.84-92, 2000.

[7] E. Kim, J.-G. Lee and D.-I. Lee, "Building a Distributed Asynchronous Control Unit through Automatic Derivation of Hierarchically Decomposed AFSMs from a CDFG," In Proceedings of Advanced Research in VLSI 2001, pp.2-15, 2001.

[8] 김의석, 이정근, 이동익, "프로세스 중심방식에 기반한 비동기식 유한상태기의 자동생성을 통한 분산 비동기식 제어부의 유도", 정보과학회논문지 : 시스템 및 이론, 제28권 제7·8호, 2001.

[9] A. Kondratyev, M. Kishinevsky, B. Lin, P. Vanbekbergen and A. Yakovlev, "Basic Gate Implementation of Speed-Independent Circuits," In Proceedings of ACM/IEEE Design Automation Conference, pp.56-62, Jun., 1994.

[10] 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, pp.315-325, 1997.

[11] S. Hauck, "Asynchronous Design Methodologies : an Overview," Proceedings of the IEEE, Vol.83, No.1, pp.69-93, 1995.

[12] T. Murata, "Petri Nets : Properties, Analysis and Applications," Proceedings of the IEEE, Vol.77, No.4, pp.541-580, 1989.

김 의 석

e-mail : uskim@kjist.ac.kr

1995년 연세대학교 전산과학과(학사)

1997년 광주과학기술원 정보통신공학과 (공학석사)

2001년 광주과학기술원 정보통신공학과 (공학박사)

2002년~현재 광주과학기술원 초고속 광 네트워크 연구센터 연구교수

관심분야 : 비동기식 시스템 설계, 설계 자동화 도구(CAD), 병행시스템(Concurrent Systems) 해석 및 설계, Petri Nets 이론 등

이 정 근

e-mail : eulia@kjist.ac.kr

1996년 한림대학교 전자계산학과(학사)

1998년 광주과학기술원 정보통신공학과 (공학석사)

1998년~현재 광주과학기술원 정보통신 공학과 박사과정

관심분야 : 비동기식 시스템 설계, 병렬 및 분산 계산, Formal Methods 등

이 동 익

e-mail : dilee@kjist.ac.kr

1985년 영남대학교 전기공학과(학사)

1989년 오사카 대학 전자공학과(공학석사)

1993년 오사카 대학 전자공학과(공학박사)

1993년~1994년 일리노이 대학교 컴퓨터 공학과 방문연구원

1990년~1995년 오사카 대학 전자공학과 문부교관

1995년~현재 광주과학기술원 정보통신공학과 부교수

관심분야 : 병행시스템(Concurrent Systems) 해석 및 설계, Petri Nets 이론, 이동 에이전트 시스템, 보안시스템, 비동기 회로 설계 및 CAD 등