프로세스 중심방식에 기반한 비동기식 유한상태기의 자동생성을 통한 분산 비동기식 제어부의 유도

(Deriving a Distributed Asynchronous Control Unit through Automatic Derivation of Asynchronous Finite State Machines based on the Process-Oriented Method)

김 의 석 † 이 정 근 †† 이 동 익 †††

(Euiseok Kim) (Jeong-Gun Lee) (Dong-Ik Lee)

요 약 본 논문에서는 비동기식 상위수준합성기 제작의 일환으로 효율적인 비동기식 제어부의 자동생성에 관한 방법을 제안한다. 제안된 방법은 목적시스템의 사양으로써 주어진 제어데이타흐름그래프로부터 일련의 체계적인 변환과정을 통하여, 제어부를 구성할 제어회로들에 대응하는 계층적으로 분할된 비동기식 유한상태기들의 집합을 유도한다. 유도된 비동기식 유한상태기들은 현존하는 비동기식 제어회로 합성기를 통하여 해저드 없는 비동기식 제어회로들로 합성되며, 이들은 상호간에 4단계 핸드셰이킹에 기반한 신호교환을 통하여 동작하면서 전체 시스템을 제어하는 계층적으로 분할된 비동기식 제어부를 구성한다. 획득한 제어부는 계층·분산적이며, 면적, 성능 및 합성시간의 측면에서 기존방식을 통하여 생성한 제어부에 비해 우월하다.

Abstract In this paper, we suggest a new method to build an asynchronous control unit automatically as a part of an asynchronous high-level synthesis procedure. In this method, a set of hierarchically decomposed asynchronous finite state machines(AFSMs) are derived from a control/data flow graph(CDFG) through a systematic procedure. Through an asynchronous logic synthesis tool, those derived AFSMs are synthesized into asynchronous controllers, which communicate with each other through 4-phase handshaking protocol and compose an asynchronous control unit. The suggested method produces superior asynchronous control units in the aspects of area, performance, implementability and synthesis time compared to previous methods.

1. 서 론

저 전력·고성능 시스템에 관한 관심이 점차 고조되면서 비동기식 시스템 설계 방식은 시스템 설계자들 가운데 널리 확산되고 있다. 특히 전역클록의 부재에 기인하여 비동기식 시스템이 제공하는 근원적인 장점인 무(無) 클록왜곡(clock skew), 저 전력 소모(low power consumption), 고성능(high-performance), 조립성(modularity) 및 저(低) EMI(low EMI)등의 장점들은 비동기식 시스템 설계 방식의 확산을 더욱 가속화시키고 있다[1]. 그러나 이와 같은 장점들에도 불구하고, 비동기식 시스템의 설계과정을 효과적으로 지원해 줄 설계자동화 도구의 부재는 비동기식 시스템 설계 방식의 확산에 가장 커다란 장애물로써 작용해 오고 있다. 이에 지난 10년 간 비동기식 시스템의 올바르고 효율적인 설계를 지원해 줄 설계자동화 도구에 관한 많은 연구가 진행되어져 왔으며, 소규모 비동기식 제어회로의 논리합성을 수행 할 수 있는 다양한 비동기식 제어회로 합성기들이 성공적으로 구현되었다

· 본 연구는 한국과학재단 한일국제공동연구(20006-302-01-2) 및 교육부 BK21 사업에 의한 지원으로 수행되었음.

† 학생회원 : 광주과학기술원 정보통신공학과
uskim@kjist.ac.kr
†† 비 회 원 : 광주과학기술원 정보통신공학과
eulia@kjist.ac.kr
††† 종신회원 : 광주과학기술원 정보통신공학과 교수
dilee@kjist.ac.kr

논문접수 : 2000년 8월 25일
심사완료 : 2001년 5월 10일

[2-6]. 다수의 성공적인 비동기식 논리합성기들이 존재함에도 불구하고 설계자가 전체 제어부를 구성하는 다수의 제어기들을 일일이 고안하고, 대응하는 동작을 비동기식 유한상태기(Asynchronous Finite State Machine, AFSM)와 신호전이그래프(Signal Transition Graph, STG)들을 이용하여 직접 기술하는 작업은 매우 어렵고 많은 시간을 소모하는 일이다. 이에 최근 들어 상위수준의 시스템 기술로부터 비동기식 제어부에 대응하는 동작기술을 자동적으로 유도하는 작업이 상위수준합성 과정의 일환으로 연구되기 시작하고 있다.

동기식 시스템에 대한 제어부의 자동생성에 관한 다수의 연구결과들이 있음에도 불구하고[7, 8], 전역클록의 부재라는 비동기식 시스템의 근원적 특성은 그러한 연구결과들의 적용을 불가능하게 한다. 비동기식 상위수준합성에 관한 초기논문인 [9]와 [10]에서는 자원제약하의 스케줄링과 제어부의 자동생성에 관한 연구결과를 보고하였다. Cortadella와 그의 연구팀은 [9, 10]에서 자기시간(self-timed) 하드웨어 모듈로 구성된 비동기식 시스템에 대한 제어회로를 시스템을 구성하는 하드웨어 모듈을 기준으로 분할·생성하는 방법을 제안하였다. 최근의 논문인 [11]에서는 Verilog로 기술된 목적시스템의 사양기술로부터 비동기식 제어부를 자동 생성하는 방법을 발표하였다. 상기한 방식들은[9-11] 제어부에 대한 신호전이그래프를 자동생성 한 후에 이를 현존하는 비동기식 제어회로 합성기를 이용하여 비동기식 제어회로로 합성하는 방식을 사용하고 있으나, 획득한 신호전이그래프의 크기가 매우 커서 합성이 어려우며, 합성의 결과로 나온 회로의 면적 및 성능도 좋지 않다. 비동기식 제어회로의 자동생성에 관한 다른 접근 방식으로 비동기식 매크로 모듈 기반의 방법이 있다. 이 방법은 [12]에서 제안된 이래로 구현 및 적용의 용이성으로 말미암아 널리 연구·사용되어 졌다. 그러나 이 방식은 사용되어지는 비동기식 매크로 모듈간의 기능 중복에 기인하여 회로면적의 손실을 야기할 수 있으며, 면적손실을 최소화하기 위하여 peephole최적화, 매크로 모듈의 재합성등의 부가적인 작업을 요구한다[13, 14].

본 논문에서는 비동기식 상위수준합성의 일환으로 목적시스템의 사양으로 주어진 제어데이타흐름그래프(Control/Data Flow Graph, CDFG)로부터 일련의 체계적인 변환과정을 통하여 제어부를 구성할 소규모 제어회로들의 사양에 대응하는 계층적으로 분할된 비동기식 유한상태기들의 집합을 유도하는 방법을 제안한다. 획득한 비동기식 유한상태기들은 현존하는 비동기식 제어회로 합성기들을 통하여 해저드 없이 올바르게 동작하는 비동기식 제어회로들로 합성된다. 이들은 상호간에 4단계 핸드셰이킹에 기반한 신호교환을 통하여 동작하면서 전체 시스템을 제어하는 계층적으로 분할된 비동기식 제어부를 구성한다. 제안된 방법을 통하여 획득한 제어부는 다음과 같은 주요한 특징을 가진다.

- '수행 제어부'와 '수행순서 제어부'로 완전히 분할된다.
- 계층적이며 균일하게 분할된 비동기식 제어부를 유도한다.
- 목적시스템의 알고리듬 수준의 사양기술로부터 체계적인 분할과정을 통하여 자동 생성된다.
- 기존의 방식에 기반하여 획득한 제어부에 비해 면적, 성능, 구현성 및 합성시간의 측면에서 우월하다.

특히, 첫 번째 특징인 '수행 제어부'와 '수행순서 제어부'의 완전한 분할은 제안된 방법의 가장 주요한 특징으로 나머지 다른 특징들을 가능하게 한다.

본 논문의 구성은 다음과 같다. 2장에서는 본 논문을 이해하는데 필요한 예비지식을 설명한다. 본 논문의 핵심인 3장에서는 제어데이타흐름그래프로부터 계층적으로 분할된 비동기식 유한상태기들을 자동 생성하는 과정을 상세히 설명한다. 4장에서는 3장에서 설명한 방법을 통하여 자동 생성한 비동기식 유한상태기들로부터 논리합성을 통하여 획득한 제어회로가 올바르게 전체 시스템을 제어하는데 필요한 시간제약들을 설명한다. 5장과 6장에서는 각각 실험결과와 결론을 제시한다.

2. 예비지식

2.1 제어데이타흐름그래프(CDFG)

본 논문에서는 기존의 상위수준언어인 VHDL 혹은 Verilog에 비하여 다소 원시적(Primitive)일 지라도 설계대상 시스템의 동작을 매우 자연스럽고 명료하게 표현할 수 있는 제어데이타흐름그래프를 설계대상 시스템의 초기사양으로써 선택하였다. 게다가 상위수준언어들이 설계자동화 도구의 입력으로 사용되기 위해서는 내부적으로 원시적인 형태의 표현으로 바뀌어야 하므로, 제어데이타흐름그래프는 상위수준언어들과 설계자동화도구를 연계시켜주는 중간형태로써의 표현으로 매우 적합하다. 먼저 제어데이타흐름그래프의 주요한 두 구성요소인 DFG-유닛과 CDFG-유닛을 정의한 후에 제어데이타흐름그래프를 정의한다.

정의 1[15] **DFG-유닛**은 $\Omega=(N, A, V, F)$의 4-튜플로써 정의한다. $N=\{n_1, n_2, ..., n_M\}$은 연산노드들의 집합이며, $A=\{a_1, a_2, ..., a_L\}\subseteq N\times N$는 연산노드들 사이의 수행순서를 나타내는 연결들의 집합이다. $V\subseteq V_1\times V_2\times...\times V_L$는 각각의 연결과 연관된 값들의 집합이며,

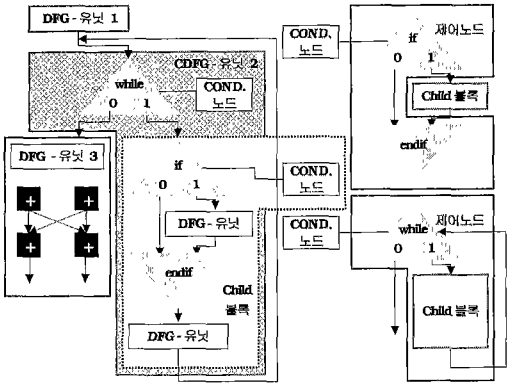


그림 1 (a) 제어데이타흐름그래프 (b) IF-노드와 WHILE -노드

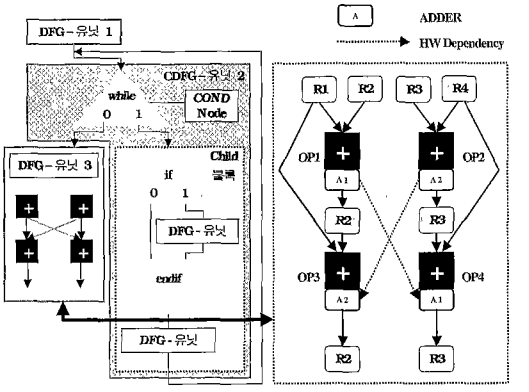


그림 2 SAB-제어데이타흐름그래프

$F=\{f_{n_i}: n_i \in N\}$는 각각의 연산노드 n_i에서 수행되어지는 연산들의 다중집합(multiset)이다.

정의 1에서 정의한 DFG-유닛을 구성하는 연산노드는 '덧셈', '곱셈' 등등의 연산(Operation) 혹은 프로세스(Process)를 나타낸다. 그러므로 본 논문에서는 DFG-유닛에 대하여 '연산노드'와 '프로세스'를 같은 의미로 사용한다.

정의 2 CDFG-유닛은 $\Gamma=(X, Y, Z, E)$의 4-튜플로써 정의한다. X={IF-노드, WHILE-노드}는 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 2의 제어노드들은 사전에 정하여진 기능에 따라 하위의 Child-블록을 수행하여 준다. 그림 1의 (b)는 정의 2에서 정의한 IF-노드와 WHILE-노드를 보여준다. 본 논문에서는 간략함을 위하여 단지 두 개의 제어노드만을 정의하였다. 정의 2의 조건노드는 일종의 DFG-유닛으로 제어노드의 수행조건을 표현한다. Child-블록은 제어노드의 제어 하에 수행 될 시스템의 동작을 표현하는 것으로 DFG-유닛과 CDFG-유닛의 집합으로 구성되는 일종의 제어데이타흐름그래프이다.

정의 3 제어데이타흐름그래프(CDFG)는 $\Sigma=(N, E)$의 2-튜플로써 정의한다. $N=\Omega \cup \Gamma$은 DFG-유닛과 CDFG-유닛에 대응하는 유닛들의 집합이며, E는 유닛들 사이의 연결들의 집합이다. 각각의 유닛은 많아야 하나의 선행유닛과 후행유닛을 가진다. 또한, 제어데이타흐름그래프가 오직 하나의 유닛으로 구성되는 경우를 제외하고는 적어도 하나의 선행유닛 혹은 후행유닛을 가진다.

정의 1, 2, 3을 통하여 정의한 제어데이타흐름그래프는 설계대상 시스템의 순차적인 동작을 계층적으로 표현할 수 있다. 그림 1의 (a)는 정의 3에서 정의한 DFG-유닛들과 CDFG-유닛의 순차적인 연결로 구성된 제어데이타흐름그래프를 보여준다. 또한 CDFG-유닛2가 Child-블록으로 제어데이타흐름그래프를 포함하는 것으로 정의 3에서 정의한 제어데이타흐름그래프가 목적시스템을 계층적으로 표현할 수 있음을 보여준다. 정의 3에서 정의한 제어데이타흐름그래프는 단지 동작만을 표현할 뿐 하드웨어 구현에 필요한 하드웨어 자원에 관한 정보는 전혀 포함하고 있지 않다. 따라서 스케줄링, 자원할당(Resource Allocation), 자원결합(Resource Binding)등의 상위수준합성(High-Level Synthesis)과정을 통하여 하드웨어 구현에 관련한 정보를 획득해 주어야 한다. 이와 같은 하드웨어 관련 정보들은 기존의 제어데이타흐름그래프에 새로운 자원의존관계(Resource Dependency Relation)를 생성하게 되므로 이들을 제어데이타흐름그래프에 명료하게 표현해 주어야 한다. 본 논문에서는 스케줄링, 자원할당, 자원결합의 세 가지 과정이 비동기식 제어부의 유도에 앞서서 이미 수행되어졌다고 가정한다. 본 논문에서는 정의 3에서 정의한 CDFG와 상기한 세 가지 가정을 통하여 획득한 하드웨어 구현에 관련한 정보를 통합하여 SAB(Scheduled, resource Allocated, resource Bound)-제어데이타흐름그래프로 정의하며, 비

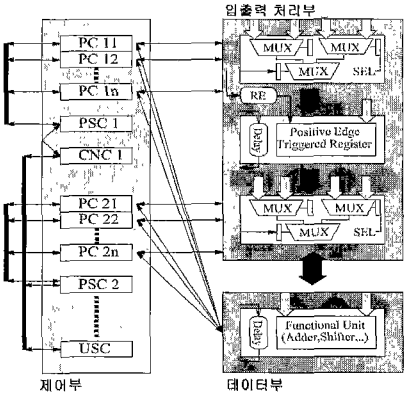


그림 3 목적시스템에 대한 아키텍처 모델

동기식 제어부의 자동생성을 위한 기본입력으로 SAB-제어데이타흐름그래프가 주어진다고 가정한다. 그림 2는 SAB-제어데이타흐름그래프를 보여준다. 이후부터 나오는 제어데이타흐름그래프는 표현의 간략함을 위하여 'SAB-'를 생략할 지라도 SAB-제어데이타흐름그래프를 의미한다.

2.2 비동기식 유한상태기

비동기식 유한상태기(Asynchronous Finite State Machine, AFSM)는 신호전이그래프와 더불어 비동기식 제어회로의 사양 기술에 널리 사용되어지는 모델이다[3, 6].. 비동기식 유한상태기는 일종의 Mealy머신으로 기본모드(fundamental mode)와 버스트모드(burst) 가정의 비동기식 제어회로를 기술하는데 널리 사용되어진다. 여기서 기본모드 가정은 회로에 입력이 가해진 후 회로가 안정화 된 연후에야 다음의 입력이 가해질 수 있음을 의미하는 것이다. 또한 버스트모드 가정은 현재 입력신호들의 집합인 입력버스트와 그에 대응하는 출력신호들의 집합인 출력버스트가 언제나 상호배제적으로(mutual exclusive) 발생함을 의미한다. 그림 5, 6, 7, 8은 비동기식 유한상태기를 보여준다.

정의 4 비동기식 유한상태기(Asynchronous Finite State Machine, AFSM)는 $\Lambda=(V, E, C, I, O, T)$의 6-튜플로써 정의한다. V는 상태들의 유한집합이며 $E\subseteq V\times V$는 연결들의 집합이다. C, I 와 O는 각각 조건입력, 입력 및 출력 신호들을 나타낸다. $T:E\rightarrow P(C)\times P(I)\times P(O)$는 전이함수로 각각의 연결들에 대하여 조건입력, 입력 및 출력신호들의 버스트를 돌려준다. $P(S)$는 집합 S의 멱집합(power set)을 의미한다.

2.3 아키텍처(Architecture) 모델

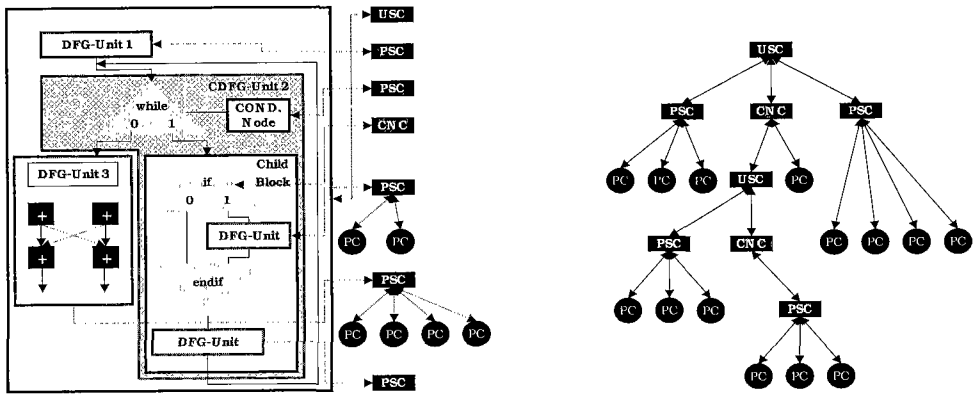
본 논문에서는 제안된 방법을 적용할 목적시스템(target system)에 대하여 그림 3과 같은 아키텍처 모델을 가정한다. 그림 3은 아키텍처 모델의 제어부(control unit)와 데이타부(datapath)를 나타낸다. 본 논문에서 제안한 방식에 기반하여 계층적으로 분할되어져 있는 제어부는 프로세스 제어기(Process Controller, PC), 프로세스 순서 제어기(Process Sequencing Controller, PSC), 제어노드 제어기(Control Node Controller, CNC), 유닛 순서 제어기(Unit Sequencing Controller, USC)로 구성되어져 있다. 제어부를 구성하고 있는 4가지 제어기에 관해서는 본 논문의 핵심인 3장과 4장에서 상세히 설명한다.

데이타부는 입출력 처리부(input/output processing part)와 데이타 처리부(data processing part)로 구성된다. 입출력 처리부는 MUX 기반의 연결부 및 양의 모서리 작동 레지스터(positive edge-triggered register)[1] 들로 이루어져 있다. 또한 MUX의 출력값을 선택하기 위한 선택 신호의 생성을 담당하는 MUX 선택 신호 생성기 (MUX selection signal generator, SEL)와 레지스터 인에이블 신호 생성기(RE)가 존재한다. 레지스터의 입출력 단자에 연결되어져 있는 MUX는 데이타 처리부와 레지스터 사이의 데이타 흐름을 담당한다.

데이타 처리부는 가/감산기(adder/subtractor), 승산기(multiplier)등의 연산모듈로써 구성되어진다. 일반적으로 비동기 시스템에 대한 연산모듈의 설계 방법에는 크게 묶음데이타(bundled data) 방식과 이중회선(dual-rail) 방식이 있다[1]. 후자의 방식은 출력값의 검사를 통하여 연산모듈의 수행완료를 결정할 수 있으므로 고성능 시스템에 유리하다. 그러나 기존의 동기식 연산모듈에 비해 많은 수의 트랜지스터 개수를 요구한다. 전자의 방식에서는 동기식 연산모듈과 연산모듈의 최악지연(worst case delay)에 대응하는 지연소자를 결합하여 연산모듈을 구성함으로써, 입력과 요구(request) 신호가 가해졌을 때 일정한 시간 후에는 결과값과 결과값이 생성되었음을 알려주는 응답(acknowledgement) 신호가 발생하게 된다. 묶음데이타 방식에 기반한 연산모듈은 이중회선 방식에 기반한 연산모듈에 비하여 성능은 다소 떨어지지만 회로구성의 비용은 지연소자를 제외하고는 동기식 모듈과 동일하다는 특징을 가진다. 본 논문에서는 면적의 측면에서 최적화를 위해 그림 3의 데이타부가 보여주는 것처럼 묶음데이타 방식을 사용하였다.

---

1) 본 논문에서 양의 모서리 작동 레지스터의 사용을 가정한 것은 연산모듈을 통하여 획득한 결과값의 저장을 프로세스 제어기가 생성하는 신호인 ReqWDR의 상승전이(ReqWDR+)에 연동시키고자 함에 기인한다.



(a) (b)

그림 4 (a) 제어데이타흐름그래프의 구성모듈들과 프로세스 중심의 제어기들 사이의 대응관계
(b) 프로세스 중심의 제어기들의 일반적인 계층도

그러나 본 논문에서 제안된 방법은 이중회선 방식에 대해서도 특별한 변형 없이 적용 가능하다.

3. 제어데이타흐름그래프에 대한 비동기식 제어부의 자동생성

3.1 기존의 방법들

비동기식 제어부의 자동생성과 관련하여 크게 다음의 두 가지 기존의 방법들, 중앙집중방식과 하드웨어 중심방식이 있다. 중앙집중방식[11]은 전체 시스템에 대한 제어부를 하나의 모듈로 구성하는 방식이다. 중앙집중방식 제어부는 수행과 관련한 모든 제어신호를 홀로 생성해야 하며 결과적으로 제어부에 대한 사양기술이 매우 복잡하게 된다. 복잡한 사양기술은 논리합성에 의한 비동기식 제어부의 제작에 있어서 성능감소, 회로면적의 증가, 합성시간의 급속한 증가, 구현성의 감소 등과 같은 치명적인 문제를 야기할 수 있다.

비동기식 제어부의 자동생성에 관한 다른 방식으로 하드웨어 중심방식이 있다[9][10]. 이 방식에서는 시스템을 구성하는 하드웨어를 기준으로 전체 제어부를 분할·유도하는 것으로 각각의 구성하드웨어에 대한 독립적인 제어기가 생성된다. 이와 같은 방식의 제어부 분할은 비동기식 시스템의 자율적 동작 특성을 자연스럽게 지원하며, 기존의 중앙집중방식의 설계가 야기하는 문제점을 경감할 수 있다는 장점을 가진다. 그러나 할당된 하드웨어 자원의 분량에 비해 시스템 사양의 크기가 클 경우 중앙집중방식이 야기하였던 문제를 동일하게 야기할 수 있다. 예를 들어, n개의 가산 연산을 가지고 있는 시스템 사양에 대하여 1개의 가산기가 할당되어져 있다면, 가산기에 대한 제어기는 n개의 가산 연산 모두를 수행하는데 필요한 모든 제어신호를 생성해야한다.

3.2 프로세스 중심방식에 의한 비동기식 제어부의 자동생성

본 논문에서는 3.1절에서 설명한 기존의 비동기식 제어부의 자동생성과 관련한 문제점들을 해결하기 위하여 목적시스템의 사양으로써 주어진 제어데이타흐름그래프로부터 일련의 체계적인 변환과정을 통하여 소규모 제어회로들로 구성된 분산 비동기식 제어부를 유도하는 방법을 제안한다. 소규모 제어회로들은 DFG-유닛의 프로세스를 기준으로 한 제어데이타흐름그래프의 계층적 분할에 대응한다. 그러므로 이후부터는 제안된 방법을 프로세스 중심방식이라고 명명한다. 프로세스 중심방식의 핵심은 목적시스템에 대한 사양기술을 프로세스에 기반한 계층성을 기준으로 '수행 제어부' 와 '수행순서 제어부'로 분할하는 것이다. 본 논문에서 목적시스템의 사양기술 방법으로 제안한 제어데이타흐름그래프는 정의 3에 의거하여 DFG-유닛과 CDFG-유닛으로 구성된다. DFG-유닛의 각 연산노드는 시스템이 수행할 연산을 나타내며, DFG-유닛, CDFG-유닛, 제어데이타흐름그래프의 연결들 및 제어노드는 시스템이 수행할 연산들의 수행순서를 나타낸다. 본 논문에서는 제어데이타흐름그래프로부터 다음의 4가지 형태의 제어회로, 프로세스 제어기(Process Controller, PC), 프로세스 순서 제어기(Process Sequencing Controller, PSC), 제어노드 제어기(Control Node Controller, CNC), 유닛 순서 제어기

(Unit Sequencing Controller, USC)를 유도한다. 프로세스 제어기와 프로세스 순서 제어기는 DFG-유닛으로부터 유도하는 제어기로써, 각각 연산노드의 수행 및 연산노드들의 수행순서를 제어한다. 또한 제어노드 제어기와 유닛 순서 제어기는 CDFG-유닛 및 제어데이타흐름그래프로부터 유도하는 것으로 유닛들의 수행순서를 제어한다. 이와 같은 4가지 종류의 제어기들은 목적시스템에대한 제어데이타흐름그래프의 계층적 분석을 통하여 계층적으로 분할된 비동기식 유한상태기들을 유도한 후, 이들을 기존의 비동기식 제어회로 합성기들을 이용하여 합성함으로써 획득할 수 있다. 그림 4(a)는 목적시스템의 사양기술인 제어데이타흐름그래프의 구성요소들과 프로세스를 중심으로 계층적으로 분할된 4가지 종류의 제어회로들의 대응관계를 보여주며, 그림 4(b)는 4가지 종류의 제어회로들의 일반적인 계층도를 보여준다.

3.3 계층적으로 분할된 비동기식 유한상태기들의 자동생성

3.3.1 프로세스 제어기에 대한 비동기식 유한상태기의 자동생성

프로세스 제어기(Process Controller, PC)는 DFG-유닛의 연산노드에 대응하는 프로세스를 수행하는데 필요한 제어신호들을 생성하는 제어기로써, 일반적으로 하나의 프로세스는 연산자 획득, 연산의 수행, 연산결과의 저장으로 이루어진다. 프로세스 제어기는 그림 5가 보여주듯이 프로세스 순서 제어기와 프로세스에 할당된 연산모듈 및 연산결과의 저장을 위한 레지스터 모듈들과의 신호교환을 통하여 프로세스를 수행한다. 프로세스 제어기의 동작을 입출력 신호들의 관점에서 기술하면 다음과 같다.

[동작 1] 프로세스 순서 제어기로부터 프로세스 제어기를 활성화시키는 ReqStart+의 입력에 의하여 프로세스 제어기가 동작을 시작한다.

[동작 2] 프로세스 제어기는 연산에 필요한 연산자 획득을 위해 연산자 요구 신호들 ReqOP1+ ... ReqOPn+를 발생시킨다.

[동작 3] 연산의 수행을 위해 프로세스 제어기는 프로세스에 할당된 기능모듈(functional module)을 활성화시키기 위하여 ReqFU+를 발생시킨다. ALU와 같이 OP code를 필요로 하는 기능모듈의 경우 OPcode+를 함께 발생시킨다. 본 논문에서는 묶음데이타 방식에 기반한 신호교환을 가정하고 있으므로 일정한 시간 후에 응답신호 AckFU+가 데이타 처리부로부터 프로세스 제어기에 입력으로 가해진다.

[동작 4] 연산결과의 저장을 위해 프로세스 제어기는

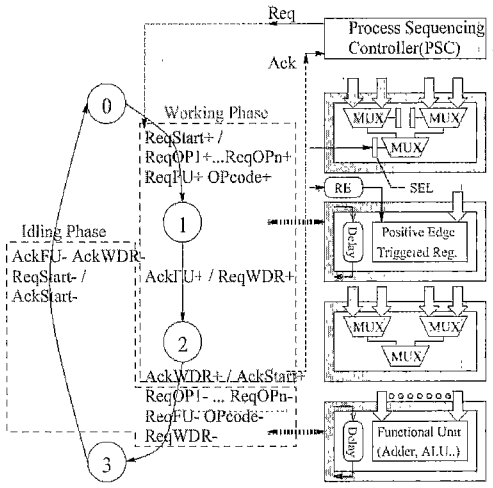


그림 5 프로세스 제어기에 대한 비동기식 유한상태기

목적 레지스터에 대하여 ReqWDR+를 발생시킨다. ReqFU+에 대한 응답신호인 AckFU+와 마찬가지로 응답신호 AckWDR+가 일정한 시간 후에 프로세스 제어기에 입력으로 가해진다.

[동작 5] 한 프로세스의 모든 작업이 완료되었으므로 프로세스 제어기는 프로세스 수행의 완료를 나타내는 AckStart+ 신호를 발생한 후 휴식단계(idling phase)로 들어간다. 휴식단계에서는 작업단계(working phase)에서 발생한 모든 신호들의 리셋이 발생한다.

상기한 프로세스 제어기의 신호수준에서의 동작기술은 정의 4에 의거하여 비동기식 유한상태기로 그림 5와 같이 기술할 수 있다. 그림 5의 비동기식 유한상태기는 모두 4개의 상태들, 0, 1, 2, 3으로 구성되어져 있다. 상태 '0'은 시스템의 초기상태를 표현하며, 상태 '1'과 상태 '2'는 각각 입력데이타에 대한 연산의 수행완료와 연산결과의 저장의 완료를 기다리는 상태들을 표현한다. 상태 '3'은 프로세스 제어기의 모든 출력신호들의 0값으로의 회귀에 대응하는 모든 입력신호들의 0의 값으로의 회귀를 기다리는 상태를 표현한다. 모든 입력신호들이 0의 값으로 회귀하면 비동기식 유한상태기는 초기상태 '0'으로 전이한다. 비동기식 유한상태기는 비동기식 제어회로로 올바르게 합성되기 위하여 다음의 두 가지 성질들, 최대집합성질(maximal set property)과 고유입구성질(unique entry point property)을 반드시 만족시켜야 한다[3][6].

정의 5 비동기식 유한상태기 A의 임의의 상태 u에 대하여, (u, v), (u, w)∈E라고 가정한다. 이때, (u, v)의

입력버스트와 (u, w)의 입력버스트 상호간에 포함관계가 없다면, Λ은 **최대집합성질**(maximal set property)을 만족시킨다. 즉, ((u, v)의 입력버스트 ⊆ (u, w)의 입력버스트 → v=w) ↔ (Λ은 최대집합성질을 만족시킨다.) 이다.

정리 1 프로세스 제어기에 대한 비동기식 유한상태기는 최대집합성질을 만족시킨다.

[**증명**] 프로세스 제어기에대한 비동기식 유한상태기의 임의의 상태 u는 언제나 하나의 후위상태만을 가진다. 즉, 임의의 상태 u의 다음 상태는 선택적이 아닌 결정적으로 정하여 진다. 그러므로 프로세스 제어기에 대한 비동기식 유한상태그래프는 언제나 최대집합성질을 만족시킨다.

정의 6 비동기식 유한상태기 Λ의 임의의 상태 u에 대하여, 상태 u에서의 입·출력 신호들의 현재값으로 구성된 벡터를 u의 이진 상태값이라 정의한다. 이때, 상태 u에 대하여 언제나 고유한 이진 상태값을 할당할 수 있다면 Λ은 **고유입구성질**(unique entry point property)을 만족시킨다.

정리 2 프로세스 제어기에 대한 비동기식 유한상태기는 고유입구성질을 만족시킨다.

[**증명**] 프로세스 제어기에 대한 비동기식 유한상태기는 오직 하나씩의 선행상태와 후행상태를 가지며, 결과적으로 비동기식 유한상태기는 하나의 사이클(cycle)을 형성한다. 그러므로 임의의 상태 u로부터 시작하여 다시금 u에 이르는 경로상에서 모든 신호들이 교차적으로 발생한다면 비동기식 유한상태기는 고유입구성질을 만족시킨다. 프로세스 제어기에 대한 비동기식 유한상태기의 모든 입·출력 신호들에 대하여 사이클 상에서 '+'와 '-'의 전이가 각각 한 번씩 발생한다. 그러므로 프로세스 제어기에 대한 비동기식 유한상태기는 고유입구성질을 만족시킨다.

정리 1, 2에 의거하여 프로세스 제어기에 대한 비동기식 유한상태기는 최대집합성질과 고유입구성질을 만족시키므로 현존하는 비동기식 제어회로 합성기를 이용하여 올바른 비동기식 제어회로로 합성가능하다. 본 부절에서 설명하고 있는 프로세스 제어기는 프로세스 혹은 연산의 종류에 따라 다소간 다른 AFSM에 대응된다. 예를 들어, 곱셈 프로세스의 수행을 제어하는 프로세스 제어기의 경우에 그림 5의 AFSM에서 OPcode의 신호가 빠진 형태의 AFSM에 대응된다. 프로세스 제어기에 대한 AFSM은 대응하는 프로세스의 종류에 따라 고유하게 결정되어지므로 언제나 자동생성이 가능하다.

3.3.2 프로세스 순서 제어기에 대한 비동기식 유한상태기의 자동생성

프로세스 순서 제어기(Process Sequencing Controller, PSC)는 DFG-유닛에 대한 일련의 프로세스 제어기들을 데이타 및 할당된 하드웨어 자원들의 의존관계를 기준으로 적당한 순서에서 활성화 시켜주는 회로이다. 프로세스 순서 제어기에 대한 비동기식 유한상태기는 임의의 DFG-유닛으로부터 다음의 알고리듬 1을 통하여 획득할 수 있다.

알고리듬 1 DFG-유닛으로부터 프로세스 순서 제어기에 대한 비동기식 유한상태기의 유도

[**과정1**] 초기상태 '0'을 생성한다. 현재 상태를 나타내는 변수 current-state를 0의 값으로 설정한다.

[**과정2**] 새로운 상태 'current-state+1'을 생성한 후 상태 'current-state'로부터 상태 'current-state+1'로의 연결 F를 생성한다.

이때, 입력버스트는

'current-state'='0'일 경우,

Req+로 구성한다.

'current-state'≠'0'일 경우,

'current-state-1'로부터 상태 'current-state'로의 연결의 출력버스트를 구성하는 신호들인 $ReqPC_i$+들에 대응하는 입력 신호들로, 즉 $AckPC_i$+들로 구성한다.

또한 출력버스트는

DFG-유닛에서 선행노드를 가지지 않는 노드가 존재할 경우,

그 노드를 전부 삭제한 후에 각각의 삭제된 노드에 대응하는 $ReqPC_i$+들로 출력버스트를 구성한다. current-state의 값을 current-state+1로 설정한 후 과정2를 재 수행한다.

선행노드를 갖지 않는 노드가 존재하지 않을 경우,

Ack+로 출력버스트를 구성한다. current-state의 값을 current-state+1로 설정한 후 과정 3으로 진행한다.

[**과정3**] 새로운 상태 'current-state+1'을 생성한 후 상태 'current-state'로부터 상태 'current-state+1'로의 연결 F를 생성한다. F의 입력버스트는 Req-이며, 출력버스트는 모든 $ReqPC_i$-들로 구성된다. current-state의 값을 current-state+1로 설정한다.

[**과정4**] 상태 'current-state'로부터 '0'으로의 연결 F를 생성한다. F의 입력버스트는 모든 $AckPC_i$-들로 구성하며, 출력버스트는 Ack-이다.

그림 6은 DFG-유닛과 알고리듬 1을 이용하여 DFG-유닛으로부터 유도한 프로세스 순서 제어기에 대한 비

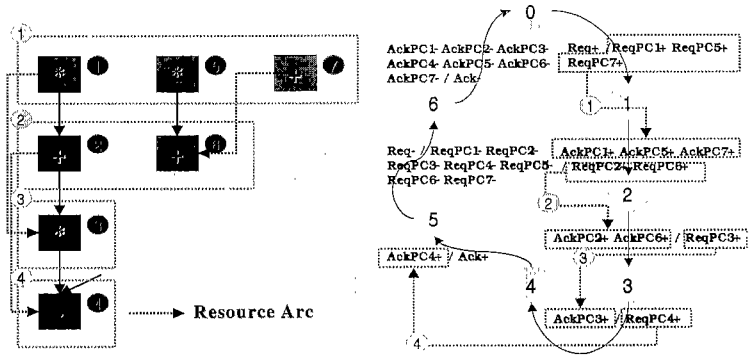


그림 6 DFG-유닛과 알고리듬 1을 이용하여 DFG-유닛으로부터 유도한 프로세스 순서 제어기에 대한 비동기식 유한상태기

동기식 유한상태기를 보여준다. 그림 6의 DFG-유닛에서 점선상자에 포함된 연산들의 집합은 병행적으로 수행가능한 연산들의 집합을 나타내며 이는 비동기식 유한상태기에 있어서 입출력 버스트로 표현된다. 예를 들어, 점선상자 ①의 연산 1, 5, 7은 비동기식 유한상태기의 상태전환 0→1→2에서 발생하는 출력·입력버스트 ReqPC1+, ReqPC5+, ReqPC7+와 AckPC1+, AckPC5+, AckPC7+에 대응한다. 프로세스 순서 제어기에 대한 비동기식 유한상태기들 역시 프로세스 제어기에 대한 비동기식 유한상태기와 마찬가지로 최대집합성질과 고유입구성질을 만족시켜야 한다.

정리 3 임의의 DFG-유닛으로부터 알고리듬 1을 통하여 획득한 비동기식 유한상태기는 최대집합성질과 고유입구성질을 만족시킨다.

[**증명**] 정의 1에서 정의한 DFG-유닛은 원천적으로 선택을 표현하지 않는다. 그러므로 프로세스 순서 제어기에 대한 비동기식 유한상태기의 임의의 상태 u는 언제나 하나의 후위상태만을 가진다. 즉, 임의의 상태 u의 다음 상태는 선택적이 아닌 결정적으로 정하여지며, 정리 1과 마찬가지로 프로세스 순서 제어기에 대한 비동기식 유한상태기는 언제나 최대집합성질을 만족시킨다. 고유입구성질의 만족은 정리 2와 동일한 방식으로 증명가능하다.

알고리듬 1을 통하여 획득한 비동기식 유한상태기는 정리 3에 의거하여 현존하는 비동기식 제어회로 합성기들을 이용하여 비동기식 제어회로로 올바르게 합성 가능하다.

프로세스 제어기와 프로세스 순서 제어기는 상호간에 4-단계 핸드셰이킹에 기반한 신호교환을 통하여 DFG-

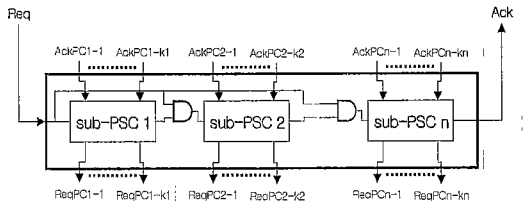
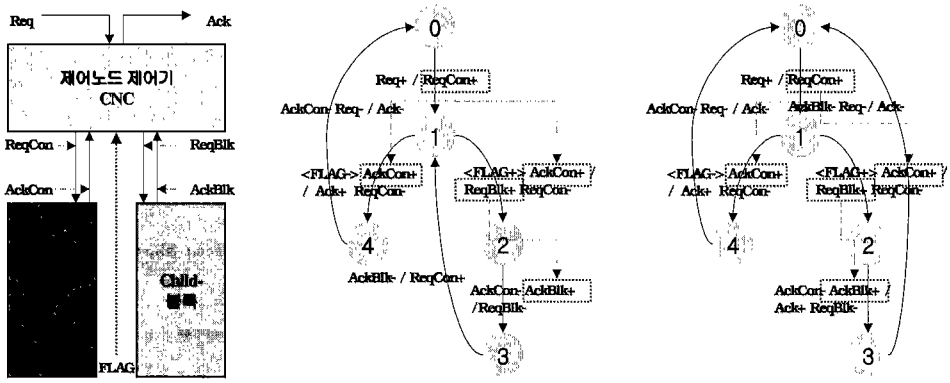


그림 7 분할된 프로세스 순서 제어기의 구조

유닛의 수행에 필요한 제어신호들을 발생시킨다. 이 과정에서 프로세스 제어기와 프로세스 순서 제어기는 최대한 빠르게 DFG-유닛을 수행하도록 설계되어져 있다. 즉, 프로세스 제어기의 경우, 작업단계(working phase)의 종료를 나타내는 Ack+의 신호를 출력한 후에 프로세스 순서 제어기로부터 Req-의 신호를 기다림이 없이 Ack 신호를 제외한 모든 출력신호들을 리셋시켜준다. 이와 같은 동작을 통하여 프로세스 제어기는 프로세스의 수행을 위해 사용한 모든 하드웨어 자원을 최대한 빠르게 다른 프로세스들이 사용할 수 있도록 하여준다. 또한 프로세스 순서 제어기도 DFG-유닛에 속한 모든 프로세스들에 대하여 작업단계만을 우선적으로 수행한 후에 최종적으로 휴식단계를 수행함으로써 최대한 빠르게 DFG-유닛을 수행할 수 있다.

본 논문에서는 하나의 DFG-유닛에 대하여 하나의 프로세스 순서 제어기를 생성하고 있으므로 DFG-유닛의 크기가 클 경우 대응하는 프로세스 순서 제어기의 크기도 클 수 있다. 결과적으로 프로세스 순서 제어기는 중앙집중방식과 하드웨어 중심방식에 의하여 유도된 비동기식 제어부가 야기할 수 있는 동일한 문제들을 야기할 수 있다. 그러므로 프로세스 순서 제어기의 크기가



(a) (b) (c)

그림 8 (a) 제어노드 제어기와 조건노드 및 Child-블록 사이의 신호교환 (b) WHILE-노드에 대한 비동기식 유한상태기 (c) IF-노드에 대한 비동기식 유한상태기

를 경우 본 논문에서는 그림 7과 같은 구조하에서 프로세스 순서 제어기의 분할을 수행한다. 소(小) 프로세스 순서 제어기(sub-PSC)들의 연결로 구성된 프로세스 순서 제어기는 분할 이전의 프로세스 순서 제어기와 동일한 동작양상을 가진다. 소(小) 프로세스 순서 제어기들의 유도는 DFG-유닛의 순차적 분할 후 각각의 소(小) DFG-유닛에 대하여 프로세스 순서 제어기를 유도함으로써 수행 할 수 있다. 예를 들어, 그림 6의 DFG-유닛에서 ①번 점선상자로 구성된 소(小) DFG-유닛과 ②, ③, ④번의 점선상자로 구성된 소(小) DFG-유닛의 분할을 통하여 두 개의 소(小) 프로세스 순서 제어기로 구성된 프로세스 순서 제어기를 유도할 수 있다.

3.3.3 제어노드 제어기에 대한 비동기식 유한상태기

제어노드 제어기(Control Node Controller, CNC)는 CDFG-유닛의 제어노드에 대응하는 제어기로 사전에 정하여진 고정된 동작을 수행한다. 본 논문에서 정의한 CDFG-유닛의 제어노드에는 IF-노드와 WHILE-노드의 두 가지가 있으며 각각 조건노드의 만족여부에 따라 정하여진 기능을 수행한다. 그러므로 제어노드 제어기는 먼저 조건노드의 수행을 담당하는 제어기를 활성화 시켜 수행조건의 참·거짓을 검사한 후 참일 경우 하위의 Child-블록의 수행을 담당하는 제어기를 활성화 시켜 Child-블록을 수행한다. 그림 8(a)는 제어노드 제어기와 조건노드 및 Child-블록 사이의 신호교환을 보여주며, 그림 8(b), (c)는 각각 WHILE-노드와 IF-노드에 대응하는 제어노드 제어기에 대한 비동기식 유한상태기를 보여준다. 제어노드 제어기에 대한 비동기식 유한상태기들은 원천적으로 최대집합성질과 고유입구성질을 만족시키도록 기술되어져 있다. 본 논문에서는 단지 두 가지 종류의 제어노드 제어기가 정의되어져 있으나 설계자의 필요에 따라 쉽게 확장 가능하다.

3.3.4 유닛 순서 제어기에 대한 비동기식 유한상태기의 자동생성

유닛 순서 제어기(Unit Sequencing Controller, USC)는 프로세스 순서 제어기가 DFG-유닛의 수행을 제어하듯이 제어데이타흐름그래프의 두 구성요소인 DFG-유닛과 CDFG-유닛에 대한 최상위 제어기들, 프로세스 순서 제어기와 제어노드 제어기들,을 적절한 순서에서 활성화 시킴으로 제어데이타흐름그래프를 수행하는 제어기이다. 유닛 순서 제어기에 대한 비동기식 유한상태기는 임의의 제어데이타흐름그래프로부터 다음의 알고리듬 2를 통하여 획득할 수 있다.

알고리듬 2 제어데이타흐름그래프로부터 유닛 순서 제어기에 대한 비동기식 유한상태기의 유도

[**과정1**] 초기상태 '0'을 생성한다. 현재 상태를 나타내는 변수 current-state를 0의 값으로 설정한다.

[**과정2**] 새로운 상태 'current-state+1'을 생성한 후 상태 'current-state'로부터 상태 'current-state+1'로의 연결 F를 생성한다.

이때, 입력버스트는

'current-state'='0'일 경우,

Req+로 구성한다.

'current-state'≠'0'일 경우,

'current-state-1'로부터 상태 'current-state'로의 연결의 출력버스트를 구성하는 신호 $ReqUnit_i$+에 대응하는 입력 신호로, 즉 $AckUnit_i$+로 구성한다.

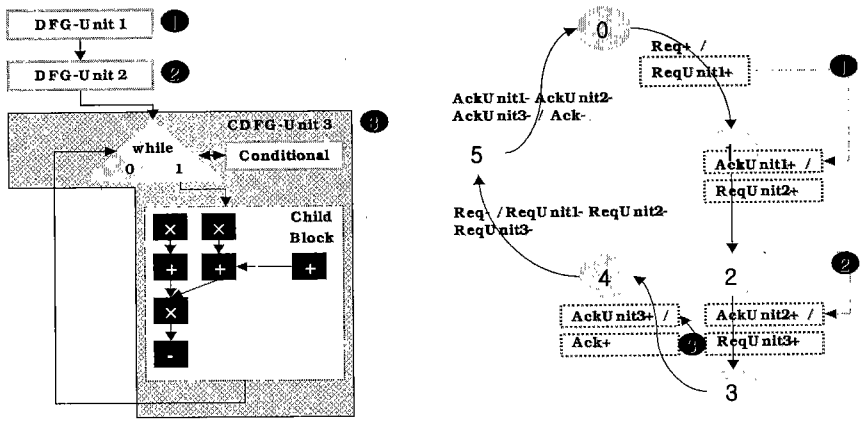


그림 9 제어데이타흐름그래프와 알고리듬 2를 이용하여 유도한 유닛 순서 제어기에 대한 비동기식 유한상태기

또한 출력버스트는

제어데이타흐름그래프에서 선행유닛을 가지지 않는 유닛이 존재할 경우,

그 유닛을 삭제한 후에 삭제된 유닛에 대응하는 $ReqUnit_i$+으로 출력버스트를 구성한다. current-state의 값을 current-state+1로 설정한 후 과정 2를 재 수행한다.

선행유닛을 갖지 않는 유닛이 존재하지 않을 경우,

Ack+로 출력버스트를 구성한다. current-state의 값을 current-state+1로 설정한 후 과정 3으로 진행한다.

[**과정3**] 새로운 상태 'current-state+1'을 생성한 후 상태 'current-state'로부터 상태 'current-state+1'로의 연결 F를 생성한다. F의 입력버스트는 Req-이며, 출력버스트는 모든 $ReqUnit_i$-들로 구성된다. current-state의 값을 current-state+1로 설정한다.

[**과정4**] 상태 'current-state'로부터 '0'으로의 연결 F를 생성한다. F의 입력버스트는 모든 $AckUnit_i$-들로 구성하며, 출력버스트는 Ack-이다.

임의의 제어데이타흐름그래프로부터 유도한 유닛 순서 제어기에 대한 비동기식 유한상태기는 최대집합성질과 고유입구성질을 만족시키며 이는 프로세스 순서 제어기에대한 비동기식 유한상태기들의 증명과 동일한 방식으로 증명할 수 있다. 그러므로 유닛 순서 제어기에 대한 비동기식 유한상태기는 현존하는 비동기식 제어회로 합성기들을 이용하여 올바르게 합성할 수 있다. 그림 9는 제어데이타흐름그래프와 알고리듬 2를 이용하여 제어데이타흐름그래프로부터 자동 생성한 유닛 순서 제어기에 대한 비동기식 유한상태기를 보여준다.

프로세스 순서 제어기와 마찬가지로 입력으로 주어진 제어데이타흐름그래프의 크기가 클 경우 대응하는 유닛 순서 제어기의 크기도 클 수 있으며, 그림 7과 같은 구조하에서 분할할 수 있다.

4. 비동기식 제어부의 올바른 동작을 위한 시간제약

본 논문에서 제안한 방식에 기반하여 유도한 계층적으로 분할된 비동기식 제어부가 전체 시스템을 올바르게 제어하기 위해서는 다음과 같은 타이밍 제약들을 만족시켜야한다.

[제약 1] 묶음데이타 방식에 기반한 연산모듈의 지연소자의 지연시간은 '연산자 획득 최대지연시간', '연산모듈의 수행 최대지연시간', '목적레지스터의 입력MUX들의 최대지연시간'의 합보다 커야한다.

[제약 2] 묶음데이타 방식에 기반한 레지스터의 지연소자의 지연시간은 레지스터 기록에 필요한 최대지연시간보다 커야한다.

[제약 3] 동일한 연산모듈 혹은 목적 레지스터를 사용하는 임의의 연속적인 두 프로세스 p_i와 p_j에 대하여, 두 프로세스의 작업단계와 휴식단계는 상호배제적이어야 한다.

제약 1과 제약 2의 필요성은 묶음데이타 방식의 가정에 따라 필요한 것이며 쉽게 만족시킬 수 있다. 만약, 묶음데이타 방식이 아닌 이중회선 방식을 가정한다면 제약 1, 2는 불필요하다. 제약 3은 묶음데이타 방식에 기반한 4단계 핸드셰이크 규약의 올바른 구현을 위해

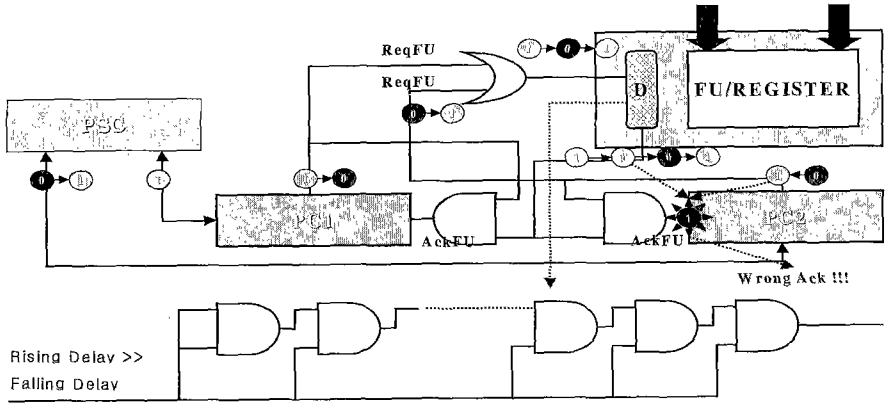


그림 10 동일한 연산모듈 혹은 목적레지스터를 순차적으로 사용하는 PC1과 PC2의 동작양상

필요하다. 본 논문에서는 독자의 이해를 돕고자 그림 10의 간단한 예제를 통하여 제약 3을 설명하고자 한다. 그림 10의 프로세스 제어기 PC1과 PC2가 각각의 프로세스를 수행하기 위하여 동일한 연산모듈을 순차적으로 사용한다고 가정하자. 먼저 PC1이 연산모듈을 활성화시키기 위하여 요구신호인 ReqFU+신호를 발생시키면, 묶음데이타 방식에 의하여 일정한 시간 후에 응답신호 AckFU+를 연산모듈의 지연소자로부터 받는다. 그림 10이 보여주는 것처럼 두 프로세스 제어기는 동일한 연산모듈을 사용하므로 동일한 응답신호 AckFU+를 받는다. 그러므로 자신에 대한 응답신호만을 받기 위하여 요구신호인 ReqFU와 연산모듈의 지연소자의 출력값의 AND값을 응답신호로 취한다. 이때, PC1이 작업단계를 끝마치고 휴식단계로 들어갔다고 가정하면, PC1은 ReqFU-신호를 발생시킨다. PC1과 PC2의 작업순서를 조정하는 프로세스 순서 제어기 PSC는 PC1이 휴식단계로 들어감에 따라 ReqPC2+를 발생시키는 것에 의하여 PC2를 활성화시키며 PC2는 즉시 ReqFU+를 발생시키면서 자신의 작업단계를 시작한다. PC1이 ReqFU-를 발생시켰음에도 불구하고 연산모듈의 지연소자의 지연으로 말미암아 지연소자의 출력값은 여전히 '1'의 값을 가지고 있을 수 있으며 PC2는 연산모듈의 수행 없이 ReqFU+의 발생과 더불어 즉시 AckFU+를 받게된다. 이러한 동작은 명백한 오동작이며 이는 동일한 연산모듈을 순차적으로 사용하는 두 프로세스 제어기 PC1과 PC2의 휴식단계와 작업단계가 겹치면서 발생한 것이다. 이러한 문제를 해결하기 위해서는 PSC와 PC2 사이에 적절한 크기의 지연소자를 삽입하여 PC2의 작업단계의 시작을 PC1의 휴식단계 이후로 연기해 주어야 한다(제약 3). PC2의 지연은 사실상 전체 시스템의 성능감소를 의미한다. 이와 같은 성능감소를 최소화시켜 주기 위해서는 PC1의 휴식단계의 길이를 최소화 시켜주어야 한다. 본 논문에서는 그림 10의 하단부에 보여지는 특별히 설계된 지연소자를 통하여 프로세스 제어기의 휴식단계의 길이를 최소화하였다. 그러므로 제약 3은 최소한의 성능감소만을 가지고 쉽게 만족시킬 수 있다.

5. 실험결과

본 논문에서는 프로세스를 중심으로 계층적으로 분할된 비동기식 유한상태기의 자동생성을 통하여 제어데이타흐름그래프로부터 효율적인 비동기식 제어부를 유도하는 방식을 제안하였다. 제안된 방법은 비동기식 제어부의 자동생성을 위한 설계자동화 도구로 구현되었다. 본 논문에서 제안한 방법을 통하여 획득한 비동기식 제어부는 기존방식을 통하여 획득한 비동기식 제어부에 비하여 회로의 면적, 성능, 구현성 및 합성시간의 측면에서 우월하다. 본 장에서는 세 가지 실험과 분석을 통하여 제안된 방법의 장점들과 올바름을 보이고자 한다. 먼저 첫 번째 실험에서는 다수의 일반적인 비동기식 유한상태기들의 합성결과들의 분석을 통하여 비동기식 유한상태기들의 크기 증가와 함께 비동기식 유한상태기들이 야기할 수 있는 문제점들을 살펴보았다. 또한 문제점들의 분석을 통하여, 주어진 시스템의 사양이 증가함에 따라 증가된 규모의 비동기식 유한상태기(*로 표시)를 생성하는 기존의 방식과 언제나 일정한 규모로 제한되는 비동기식 유한상태기들(**로 표시)을 생성하는 프로세스 중심방식의 비교를 수행하였다. 두 번째 실험에서는 상위수준합성과정에 널리 사용되어지는 벤치마크들

표 1 비동기식 유한상태기의 크기 증가에 따른 합성회로 및 합성시간의 변화

제어기 이름	상태 개수	전이 개수	주 입력	주 출력	상태변수의 개수	리터럴의 개수	fanin 6,7	fanin 8≥	합성시간
sbuf-read-ctl†	7	8	3	3	1	15	0	0	2.9 sec.
pscsi-ircv†	6	7	4	3	2	31	1	0	4.1 sec.
scsi-init-send†	9	11	5	3	3	83	2	0	6.9 sec.
nak-pa†	6	6	4	5	1	26	0	0	4.8 sec.
sbuf-ram-wri†	6	6	5	5	1	31	0	0	4.8 sec.
hp-ir-it-cont†	10	12	5	7	1	49	0	0	8.0 sec.
diff-alu2†	14	16	5	7	3	139	3	1	19.8 sec.
dram-control†	12	14	7	6	1	50	1	0	8.5 sec.
scsi†	70	93	9	5	-	-	-	-	-
pscsi†	45	62	10	5	7	380	5	14	218.5 sec.
hp-ir-sd-cont†	25	32	8	11	4	165	7	3	75.7 sec.
hp-ir-sc-cont†	33	42	13	14	3	389	12	11	908.0 sec.
3-FIR*	10	10	7	11	2	106	1	0	49.2 sec.
4-FIR*	13	13	7	13	3	180	4	2	163.2 sec.
5-FIR*	16	26	10	19	3	249	11	2	1738.7 sec.
2-IIR*	13	13	9	16	2	231	6	6	523.7 sec.
4-IIR*	28	28	13	27	-	-	-	-	-
PC**	4	4	3	5	0	17	0	0	3.8 sec.
PSC4**	4	4	5	5	0	16	0	0	4.2 sec.
PSC8**	5	5	9	9	0	44	4	1	97.5 sec.
D-PSC8**	8	8	9	9	0	32	0	0	8.4 sec.
IF-CNC**	5	6	4	3	2	29	0	0	4.4 sec.
WHILE-CNC**	5	6	4	3	2	26	0	0	4.1 sec.
USC4**	7	7	5	5	0	14	0	0	4.4 sec.
USC8**	11	11	9	9	0	30	0	1	112.1 sec.
D-USC8**	14	14	9	9	0	28	0	0	8.8 sec

- †은 일반적인 비동기식 유한상태기들의 예제들을 나타낸다.
- *와 **은 각각 중앙집중방식 제어기와 프로세스 중심방식의 제어기를 나타낸다.
- '-'은 합성실패를 나타낸다.
- D-PSC와 D-USC는 각각 분할된 PSC와 USC를 나타낸다.

인 FIR필터와 IIR필터에 대하여 중앙집중방식에 기반하여 유도한 비동기식 제어부와 프로세스 중심방식을 통하여 유도한 비동기식 제어부 사이의 비교·분석을 수행하였다. 마지막으로 세 번째 실험에서는 *Differential Equation Solver*에 대하여 비동기식 시스템 설계 전문가에 의하여 설계되어진 결과와 프로세스 중심방식을 통하여 획득한 결과의 비교·분석을 수행하였다. 하드웨어 중심방식의 경우, 하드웨어 자원의 제약 하에 목적시스템에 대한 사양의 크기가 증가하면 중앙집중방식과 동일한 문제를 야기하므로 본 실험에서는 중앙집중방식의 제어부와 프로세스 중심의 제어부의 비교를 통하여 제안된 방법의 효율성을 보이고자 한다.

3.1절에서 설명한 기존의 비동기식 제어부의 자동생성 방식의 문제점은 목적시스템의 사양증가에 따라 제어부의 사양인 신호전이그래프 혹은 비동기식 유한상태기들의 크기가 증가함에 기인한다. 첫 번째 실험결과인 표 1에서 일반적인 비동기식 유한상태기들(†로 표시)의 합성결과는 비동기식 유한상태기의 크기 증가에 따른 합성회로의 여러 가지 측면들, 상태변수의 개수, 리터럴의 개수, 고(高)-팬인 게이트(high-fanin gate)의 개수 및 합성시간의 변화결과를 보여주고 있다. 상태변수의 개수와 리터럴의 개수의 경우, 비동기식 유한상태기의 크기 증가에 따라 증가할 지라도 그 증가정도가 선형적이므로 문제가 되지 않는다. 그러나 합성시간은 비동기식 유한상태기의 크기의 증가에 대하여 급속히 증가하는 모습을 보여준다. 이는 비동기식 유한상태기의 논리합성 과정에서 필요한 상태공간의 급속한 증가에 기인한다. 예를 들어, 입력 혹은 출력신호의 개수가 1개 증가하면

논리합성 과정에 필요한 상태공간의 개수는 2배 증가한다. 결과적으로 합성시간은 비동기식 유한상태기의 크기 증가에 대해 지수적으로 증가하며 이는 비동기식 제어기의 논리합성 과정에 있어서 매우 심각한 문제이다. 또한 비동기식 유한상태기의 증가에 따른 고(高)-팬인 게이트 개수의 증가는 비동기식 제어기의 구현성을 감소시킨다. 동기식 제어회로와는 달리 비동기식 제어회로의 경우 해저드의 위험 때문에 고(高)-팬인 게이트의 분할을 수행하는 것이 매우 어렵다[16]. 그러므로 논리합성의 결과 고(高)-팬인 게이트가 존재할 경우에 이를 지원하는 셀 라이브러리가 존재하지 않는다면 설계자는 고(高)-팬인 게이트를 지원하도록 셀 라이브러리를 확장하거나 완전설계방식(full custom design)에 기반하여 전체 제어회로를 구현해야만 한다. 비동기식 유한상태기의 크기 증가는 고(高)-팬인 게이트의 개수 증가와 직접적인 연관을 가진다. 현재까지의 비동기식 유한상태기의 논리합성은 해저드의 문제를 해결하기 위하여 2 레벨 SOP(Sum-of-Product)의 제한된 구조를 가진다[6]. 그러므로 비동기식 유한상태기의 크기 증가는 필연적으로 고(高)-팬인 게이트의 개수 증가 및 그에 따른 구현성의 현저한 감소로 이어진다. 표 1에서 중앙집중방식에 기반한 비동기식 유한상태기(*)들과 프로세스 중심의 비동기식 유한상태기(**)들의 분석결과는 프로세스 중심의 비동기식 유한상태기가 합성시간 및 구현성의 측면에서 중앙집중방식의 비동기식 유한상태기에 대해 우월함을 보여준다. 표 1에 있는 중앙집중방식을 통하여 획득한 비동기식 유한상태기들과 프로세스 중심방식을 통하여 획득한 비동기식 유한상태기들이 특정한 대응관계를 통하여 정량적으로 비교되는 것이 아니라는 점을 주의해야 한다. 표 1은 일반적인 비동기식 유한상태기들의 규모증가에 따른 문제점의 제기와 중앙집중방식을 통한 비동기식 유한상태기의 자동생성이 그러한 문제점들을 피할 수 없는 반면에 프로세스 중심방식을 통한 비동기식 유한상태기의 자동생성은 그러한 문제점들을 언제나 효과적으로 피할 수 있음을 보여주는 것이다. 중앙집중방식의 경우, 단 한 개의 비동기식 유한상태기를 통하여 시스템의 제어부의 모든 동작을 기술하므로, 시스템의 사양이 증가할 경우에 대응하는 비동기식 유한상태기의 크기도 함께 증가한다. 표 1이 보여주는 것처럼 비동기식 유한상태기의 크기 증가가 선형적일지라도 대응하는 합성시간은 지수승으로 증가하므로 대응하는 합성시간은 지수적으로 증가한다. 그러므로 중앙집중방식에서 발생하는 자동생성된 비동기식 유한상태기의 크기의 증가는 합성시간의 지수적 증가라는 치명적인 문제를 야기한다. 게다가 기존의 비동기식 논리합성 방법의 기본가정인 2 레벨 SOP구조는 비동기식 유한상태기의 크기 증가와 더불어 고(高)-팬인 게이트의 개수를 증가시켜 구현성을 현저히 감소시킨다. 이에 반하여 프로세스 중심방식의 경우, 유도된 비동기식 유한상태기들의 각각의 크기는 프로세스 순서 제어기(PSC)와 유닛 순서 제어기(USC)를 제외하고는 언제나 일정한 규모로 제한된다. 게다가 프로세스 순서 제어기와 유닛 순서 제어기의 경우에도 3.3.2절의 그림 7과 같은 구조로 분할하여 줌으로써 결과적으로 모든 종류의 비동기식 유한상태기들은 언제나 일정한 규모로 제한된다. 그러므로 프로세스 중심방식에서는, 시스템의 사양의 증가와 더불어 비동기식 유한상태기들의 크기가 아닌 개수가 증가한다. 비동기식 유한상태기의 논리합성 시간이 유한상태기의 크기에 대해 지수적으로 증가한다는 사실을 고려할 때, 규모가 큰 비동기식 유한상태기 1개의 합성시간보다 규모가 작은 다수의 비동기식 유한상태기들의 합성시간의 합이 적음을 예측할 수 있다. 또한 규모가 작은 각각의 비동기식 유한상태기들은 고(高)-팬인 게이트의 사용을 필요로 하지 않으며, 결과적으로 전체적으로도 고(高)-팬인 게이트의 사용을 필요로 하지 않음으로 구현성을 크게 향상시킬 수 있다. 결론적으로 표 1은 비동기식 유한상태기의 규모에 따른 특성을 고려해 볼 때, 시스템 사양의 크기 증가에 따라 계속적으로 증가하는 중앙집중방식의 비동기식 유한상태기에 비하여 언제나 적절한 크기로 유지되는 프로세스 중심방식의 비동기식 유한상태기들이 합성시간과 구현성의 측면에서 우월함을 보여준다.

두 번째 실험에서는 제안된 방법의 장점들을 보이고자 상위수준합성 과정에서 많이 사용되어지는 예제인 FIR필터와 IIR필터의 제어데이타흐름그래프[2]에 대하여 중앙집중방식과 프로세스 중심방식을 적용하여 비동기식 제어부를 유도하였다. 표 2는 획득한 비동기식 제어부들의 리터럴의 개수, 면적, 평균 입출력 반응시간, 고(高)-팬인게이트의 개수 및 합성시간의 측면에서의 분석결과를 보여준다. 면적의 측면에 있어서 중앙집중방식의 비동기식 제어부는 프로세스 중심방식을 통하여 획득한 비동기식 제어부에 비해 약 2.1배 정도 크다. 본 연구팀은 실험을 수행하기 전에 전역 최적화(global optimization)를 수행 할 수 있는 중앙집중방식의 비동기식 제어부가 면적의 측면에서 보다 적을 것으로 예측하였으나 결과는 예상과는 달리 프로세스 중심방식에 기반한

2) 표 2의 예제들은 하나의 DFG-유닛으로 구성된 제어데이타흐름그래프들이다.

표 2 중앙집중방식의 비동기식 제어부*와 프로세스 중심방식의 비동기식 제어부**의 비교

제어부 이름	리터럴의 개수	면적	평균 입출력 반응시간	fanin		합성시간
				6,7	8≤	
3-FIR*	106	160.30	1.52 ns	1	0	49.2 sec.
3-FIR**	87	83.79	0.60 ns	0	0	18.5 sec.
5-FIR*	249	359.77	1.42 ns	11	2	1738.7 sec.
5-FIR**	164	157.27	0.61 ns	0	0	22.4 sec.
2-IIR*	231	309.13	1.70 ns	6	6	523.7 sec.
2-IIR**	143	143.99	0.62 ns	0	0	17.9 sec.

- *와 **은 각각 중앙집중방식 제어부와 프로세스 중심방식의 제어부를 나타낸다.

표 3 [17]에서 사용한 비동기식 제어부*와 프로세스 중심방식의 비동기식 제어부**의 비교

제어부 이름	리터럴의 개수	면적	평균 입출력 반응시간	fanin		합성시간
				6,7	8≤	
DIFF-EQ-SOL*	239	353.96	1.75 ns	5	1	32.3 sec.
DIFF-EQ-SOL**	204	199.85	0.63 ns	0	0	39.2 sec.

- *와 **은 각각 중앙집중방식 제어부와 프로세스 중심방식의 제어부를 나타낸다.

비동기식 제어부의 면적이 보다 적었다. 이러한 실험결과는 프로세스 중심방식에 기반한 비동기식 제어부들의 '수행부'와 '수행 순서 제어부' 사이의 효율적 분할에 기인한다. 유도된 비동기식 제어부의 성능을 평가하기 위하여 본 논문에서는 평균 입출력 반응시간을 성능의 측정기준으로 설정하였다.

동기식 시스템과는 달리 비동기식 시스템은 각각의 구성모듈들이 전역클록의 통제없이 각 모듈들의 최상의 속도에서 모듈간의 신호교환을 통하여 동작하므로 평균 입출력 반응시간이 성능평가의 방법으로 적절하다. 표 2의 결과는 프로세스 중심방식을 통하여 획득한 비동기식 제어부가 약 2.5배정도 빠르게 입출력 신호들을 주변 모듈들과 교환함을 보여준다.

고(高)-팬인 게이트의 개수 및 합성시간의 결과는 표 1에서 설명한 것과 같이 프로세스 중심방식을 적용하여 획득한 비동기식 제어부가 적다. 결과적으로 첫 번째 실험과 두 번째 실험의 결과는 프로세스 중심방식을 적용하여 획득한 비동기식 제어부가 기존의 방식을 적용하여 획득한 비동기식 제어부에 비하여 면적, 성능, 구현성 및 합성시간의 측면에서 효율적임을 보여준다.

세 번째 실험에서는 제안된 방법의 효율성 및 올바름을 보이기 위하여 제안된 방법을 통하여 획득한 비동기식 제어부를 가지고 있는 *Differential Equation Solver*를 VHDL코드수준에서 구현하였다. 또한 비교를 위하여 [17]에서 작성된 비동기식 제어부에 대한 비동기식 유한상태기를 동일한 합성기를 통하여 합성하여 비교·분석을 수행하였으며, 표 3은 분석결과를 보여준다. [17]의 비동기식 제어부는 경험있는 설계자에 의하여 분할·기술된 비동기식 유한상태기들을 합성하여 획득한 것임에도 불구하고, 프로세스 중심방식에 기반하여 획득한 비동기식 제어부에 비해 면적의 측면에서는 1.78배 크고, 평균 입출력 반응시간은 약 2.78배 크다. 게다가 다수의 고(高)-팬인 게이트를 포함하고 있다. 합성시간의 측면에서는 [17]의 결과가 다소 빠르다. 이는 [17]의 비동기식 제어부 또한 설계자에 의하여 분할되어져 있다는 점에 기인한다. 결과적으로 표 3은 경험있는 설계자가 직접 설계한 회로와 비견할 만한 비동기식 제어부를 프로세스 중심방식을 이용하여 유도할 수 있음을 보여준다.

그림 11은 프로세스 중심방식을 통하여 획득한 *Differential Equation Solver*의 비동기식 제어부의 수행결과를 보여주는 것으로 프로세스 중심방식의 비동기식 제어부는 구성요소간에 4 단계 핸드세이킹에 기반하여 통신하면서 전체시스템을 제어한다. 전체 제어부는 유닛 순서 제어기로부터 동작을 시작하여 제어노드 제어기, 프로세스 순서 제어기, 프로세스 제어기의 계층적 순서에서 신호를 상호 교환하며 동작한다. 그림 12는 비동기식 제어부에 의하여 제어되는 *Differential Equation Solver*의 입출력 값의 변화를 보여준다.

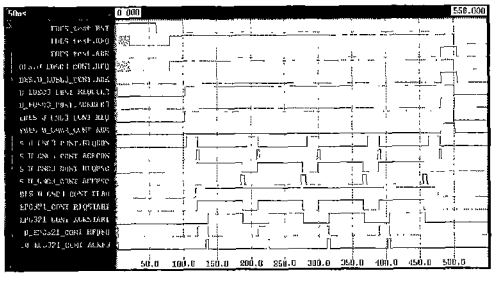


그림 11 *Differential Equation Solver*의 모의실험 결과 I

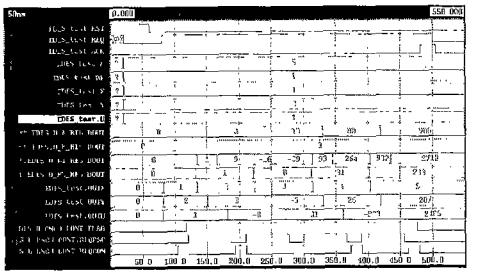


그림 12 *Differential Equation Solver*의 모의실험 결과 II

6. 결 론

본 논문에서는 비동기식 상위수준합성기 제작의 일환으로 효율적인 비동기식 제어부의 자동생성에 관한 방법을 제안하였다. 제안된 방법은 목적시스템의 사양으로써 주어진 제어데이타흐름그래프로부터 일련의 체계적인 분할과정을 통하여 제어부를 구성하는 소규모 제어회로들에 대응하는 계층적으로 분할된 비동기식 유한상태기의 집합을 유도한다. 유도된 비동기식 유한상태기의 집합은 비동기식 논리합성기에 의하여 프로세스 중심방식의 비동기식 제어부를 형성하는 소규모의 비동기식 제어기들로 합성된다. 획득한 비동기식 제어부의 분석은 다수의 실험을 통하여 수행되어졌으며, 분석결과는 본 논문에서 제안한 프로세스 중심의 비동기식 제어부가 기존의 방식을 통하여 획득한 비동기식 제어부에 비해 면적, 성능, 구현성 및 합성시간의 측면에서 우월함을 보여준다. 결론적으로 본 연구는 비동기식 상위수준합성기 제작의 일환으로 수행되어졌으며, 제안된 방법은 비동기식 시스템의 설계자가 상위수준의 시스템 기술인 제어데이타흐름그래프로부터 자동적으로 비동기식 제어부를 획득할 수 있도록 하여준다.

참 고 문 헌

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

[2] 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.

[3] S. M. Nowick, "Automatic Synthesis of Burst-Mode Asynchronous Controllers," Ph. D. thesis, Stanford University, 1995.

[4] 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.

[5] E. Pastor, J. Cortadella, A. Kondratyev and O. Roig, "Structural Methods for the Synthesis of Speed-Independent Circuits," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol.17, no.11, pp.1108-1129, 1998.

[6] K. Y. Yun and D. L. Dill, "Automatic Synthesis of Extended Burst-Mode Circuits: Part II (Automatic Synthesis)," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol.18, no.2, pp.118-132, 1999.

[7] D. D. Gajski, N. D. Dutt, A. C-H Wu and S. Y-L Lin, "High-Level Synthesis : Introduction to Chip and System Design," Kluwer Academic Publishers, 1991.

[8] G. DeMicheli, "Synthesis and Optimization of Digital Circuits," McGraw-Hill, 1994.

[9] 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.

[10] R. M. Badia, J. Cortadella, E. Pastor and A. Pardo, "A High-Level Synthesis System for Asynchronous Circuits," Sixth International Workshop on High-Level Synthesis, Nov., pp. 87-94, 1992.

[11] 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.

[12] K. V. Berkel, "Handshake Circuits. An asynchronous architecture for VLSI programming," International Series on Parallel Computation 5. Cambridge University Press, 1993.

[13] T. Kolks, S. Vercauteren and B. Lin, "Control Resynthesis for Control-Dominated Asynchronous Designs," In Proceedings of Second International

Symposium on Advanced Research in Asynchronous Circuits and Systems, Mar., pp.233-243, 1996.

[14] M. A. Peña, J. Cortadella, "Combining Process Algebras and Petri Nets for the Specification and Synthesis of Asynchronous Circuits," In Proceedings of Second International Symposium on Advanced Research in Asynchronous Circuits and Systems, Mar., pp.222-232, 1996

[15] D. D. Gajski, J. Zhu and R. Dömer, "Essential Issues in Codesign," In J. Staunstrup and W. Wolf, editor, Hardware/Software Co-Design : Principles and Practice, pp.1-45, Kluwer Academic Publishers, 1997.

[16] P. Siegel, G. De Micheli and D. Dill, "Automatic Technology Mapping for Generalized Fundamental Mode Asynchronous Designs," In Proceedings of the Design Automation Conference, June 1993.

[17] K. Y. Yun, P. A. Beerel, V. Vakilotojar, A. E. Dooply and J. Arceo, "The Design and Verification of A High-Performance Low-Control-Overhead Asynchronous Differential Equation Solver," IEEE Trans. VLSI Systems, vol.6, no.4, pp.643-655, 1998.

김 의 석
1995년 연세대학교 전산과학과 학사. 1997년 광주과학기술원 정보통신공학과 석사. 1997년 ~ 현재 광주과학기술원 정보통신공학과 박사과정. 관심분야는 병행 시스템(Concurrent Systems) 해석 및 설계, Petri Nets 이론, 비동기 회로 설계 및 CAD 등

이 정 근
1996년 한림대학교 전자계산학과 학사. 1998년 광주과학기술원 정보통신공학과 석사. 1998년 ~ 현재 광주과학기술원 정보통신공학과 박사과정. 관심분야는 비동기 회로, 병렬 및 분산 계산, formal methods 등

이 동 익
1985년 영남대학교 전기공학과 학사. 1989년 오오사카 대학 전자공학과 석사. 1993년 오오사카 대학 전자공학과 박사. 1993년 ~ 1994년 일리노이 대학 컴퓨터 공학과 방문연구원. 1990년 ~ 1995년 오오사카 대학 전자공학과 문부교관. 1995년 ~ 현재 광주과학기술원 정보통신공학과 부교수. 관심분야는 병행시스템(Concurrent Systems) 해석 및 설계, Petri Nets 이론, 이동 에이전트 시스템, 보안시스템, 비동기 회로 설계 및 CAD 등