가변지연시간 연산기를 가진 데이터 경로에 대한 동기식 제어기의 설계

*김 의 석, **이 정 근, **이 동 익
*초고속 광 네트워크 연구센터, **광주과학기술원 정보통신공학과
전화 +82-62-970-2267

Design of a Synchronous Control Unit for a Datapath with Variable Delay Arithmetic Units

*Euiseok Kim, **Jeong-Gun Lee and **Dong-Ik Lee
*Ultrafast Fiber-Optic Networks Research Center, **Dept. of Info. & Comm., K-JIST
E-mail: uskim@kjist.ac.kr, eulia@kjist.ac.kr and dilee@kjist.ac.kr

Abstract

Nowadays variable delay arithmetic units have been used for implementing a datapath of a target system in pursuit of performance improvement. However, adoption of variable delay arithmetic units requires modification of a typical synchronous control unit design methodology. There is a representative approach, which is called a monolithic approach. Although its results are good, its proposed methodology may cause critical problems in the aspects of area and performance with the size increase of initial system specifications. In order to solve this problems, a distributed approach is suggested. Experimental results show that the proposed method can guarantee original performance of an initial system specification with minimized additional area increase.

I. 서론

전형적인 동기식 시스템은 주기적으로 발생하는 전역클록을 기준으로 기술된 유한상태기(Finite State Machine, FSM)로부터 합성된 동기식 제어부와, 동기식 연산기들로 구성된 데이터 경로로 구성되어져 있다. 동기식 연산기들의 연산시간이 실제로는 피연산자들의 종류에 따라서 다양하게 변화할 수 있음에도 불구하고, 전역클록에 따른 제약으로 시스템 설계자는 언제나 고정된 연산시간을 가정하여 데이터 경로에 대한 제어부를 설계하여 왔다. 이러한 설계 방식은 연산기들의 아이돌타임(idle time)을 증가시키며, 궁극적으로 연산기들의 사용 효율성과 전체적인 시스템의 성능을 감소시킬 수 있다.

이러한 문제를 해결하기 위해서는 두 가지 연구가 수행되어져야 한다. 첫째는, 가변지연시간 연산기의 구현이다. 둘째는, 가변지연시간 연산기를 포함한 데이터 경로를 효과적이며 올바르게 제어할 수 있는 제어부의 설계 및 개발이다. 가변지연시간 연산기의 구현은 전역클록을 사용하지 아니하고 시스템을 설계하는 비동기식 시스템 설계 분야를 비롯하여 다양한 분야에서 활발히 연구되어져 오고 있으므로, 본 논문에서는 동기식 시스템에서 올바르게 사용되어질 수 있는 가변지연시간 연산기의 라이브러리가 이미 존재한다고 가정하며, 두 번째 문제를 집중적으로 다루도록 한다. 즉, 본 논문에서는 가변지연시간 연산기를 시스템의 제작에 사용하였을 때, 이들의 가변시간적 동작을 통하여 획득할 수 있는 국부적 성능향상을 시스템 전체의 성능향상으로 전환하여 주는 동기식 제어부의 설계 방법을 제안하고자 한다. 일반적으로 가변지연시간 연산기를 포함하고 있는 데이터 경로는 보다 복잡한 시간행위(timing behavior)를 보여주며, 이에 대응하는 제어기 또한 급속도로 복잡하여 질 수 있다. 제어기의 복잡도 증가는 제어기의 면적의 급속한 증가 및 성능의 감소를 야기할 수 있다. 본 논문에서는 정량적인 분석

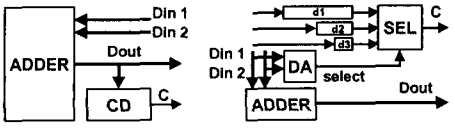

그림 3 대표적인 가변지연시간 연산기(C와 CD는 각각 연산완료 신호와 연산완료 감지모듈을 나타낸다.)

을 통한 이러한 문제의 지적과 함께, 분산된 형태의 제어기를 생성하여 주는 방법을 제안 및 개발함으로써 이러한 문제를 효과적으로 해결하고자 한다.

II. 가변지연시간 연산기

전형적인 동기식 시스템 설계자는 각각의 구성 연산기들에 대하여 피연산자의 종류에 상관없이 최악지연시간의 연산시간을 가정하여 목적시스템을 설계한다. 그러나 사실상 각각의 연산기는 피연산자의 종류에 따라 다양한 연산시간을 가진다. 예를 들어, N bit RCA(Ripple Carry Adder)의 경우, 캐리 전파 경로의 길이가 1~N 비트의 값 중 임의의 한 값을 가질 수 있으며, 연산시간 또한 전파 경로의 길이에 따라 크게 변할 수 있다. 참고로, 연산시간은 피연산자가 주어진 시점부터 올바른 결과값이 생성되기까지의 시간을 의미한다.

연산기의 연산시간 가변성을 이용하기 위해서는 연산기가 대응하는 제어기에게 연산의 완료를 인식시킬 수 있어야하며, 이를 위한 연산완료 신호의 생성은 가변지연 연산기 구현의 핵심기술이다. 이러한 기능을 가진 대표적인 연산기로 이중회선 연산기[1], 스페큘러티브(speculative) 연산기[2], 텔레스코픽(telescopic) 연산기[3]가 있다. 이중회선 연산기는 입력 피연산자의 각 비트가 2 비트로 즉, 0은 01로, 1은 10으로, 코딩된 입력 피연산자를 입력으로 받아 연산을 수행하며, 연산결과의 모든 비트가 01 혹은 10이 되었을 때 별도의 연산완료 감지모듈(Completion Detection Logic)이 연산이 완료되었음을 감지하여 연산완료 신호를 생성함으로써 제어기에게 연산의 완료를 알려준다. 스페큘러티브 연산기는 먼저 피연산자의 분석을 수행하며, 분석결과에 따라서 다단계의 지연시간들 중에서 적절한 것을 선택하여 연산완료 신호를 생성한다. 텔레스코픽 연산기는 사실상 2-단계의 지연시간을 생성하는 스페큘러티브 연산기이다. 상기한 가변지연시간 연산기중에서, 이중회선 연산기는 연속적인 시간영역에서 연산완료 신호를 생성하며 이는 동기식 제어부와의 연동시 셋업-홀드 타임의 위배에 의한 메타스터빌러티

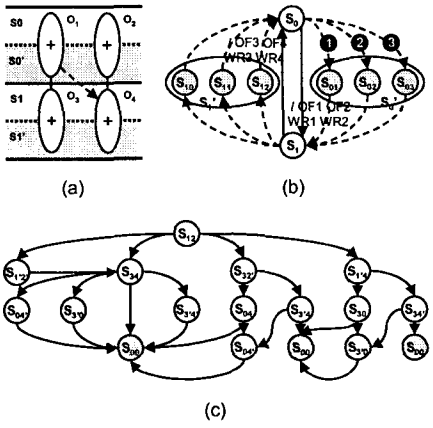

그림 4 (a) 데이터 흐름그래프(Dataflow Graph, DFG) (b) 전형적인 유한상태기 (c) 요구-응답 신호를 가진 유한상태기

(metastability) 문제를 야기할 수 있으므로 동기식 시스템 설계 환경하에서는 사용될 수 없다. 이에 반해, 스페큘러티브 연산기와 텔레스코픽 연산기는 이산적인 시간영역에서 연산완료 신호를 생성함으로써 동기식 시스템 설계 환경하에서도 사용이 가능하다. 그러므로 본 논문에서는 가변지연시간을 가진 연산기로 스페큘러티브 연산기와 텔레스코픽 연산기를 사용한다고 가정하며, 이후 편의상 이들을 다단계 지연시간 연산기로 통칭한다. 그림 1은 상기한 가변지연시간 연산기들의 구조를 보여준다.

III. 가변지연시간 연산기를 가진 데이터 경로에 대한 제어회로의 유도

3.1 전형적인 유한상태기의 구조

유한상태기는 전역클록의 발생에 따라 상태를 천이하며 적절한 입출력 신호를 받아들이는 동기식 순차회로의 동작을 기술하는데 사용되어지는 가장 대표적인 기술모델이다. 본 논문에서는 목적시스템의 초기기술로써 상위수준합성과정에서 가장 널리 사용되어지는 모델중의 하나인 DFG(그림 2(a))가 주어진다고 가정한다. 이때, 유한상태기의 출력신호는 크게 두 가지, 적절한 피연산자의 선택을 위한 OF(Operand Fetch) 신호와 연산의 수행결과의 저장을 위한 WR(Writing Result) 신호로 구성된다. 그림 2(b)(S_0, S_1 및 실선 연결부분만 해당)는 그림 2(a)에 대응하는 유한상태기이다.

3.2 연산기에 대한 요구-응답신호를 가진 유한상태기의 구조

3.1에서 설명한 전형적인 유한상태기는 데이터 경로를 구성하는 모든 연산기들의 연산시간이 고정되어져 있음을 가정하고 있다. 그러나 가변지연시간 연산기를 데이터 경로에 사용한다면 연산기의 연산시간은 피연산자의 종류에 따라 변화할 수 있다. 가령, 최악 연산시간 10ns에 2-단계의 지연시간을 가진 2-단계 지연시간 연산기는 피연산자의 종류에 따라 각각 5ns와 10ns의 지연시간을 가진다. 이와 같은, 가변지연시간 연산기의 가변적인 연산완료에 따른 국부적인 성능향상을 시스템 전체로의 성능향상으로 전환시켜주기 위해서는 유한상태기가 가변지연시간 연산기의 연산완료 신호를 감지하여 이에 따라 적절한 제어신호를 생성할 수 있어야 한다. 그러므로 가변지연시간 연산기로 구성된 데이터 경로를 제어하는 제어기에 대응하는 유한상태기는 기존의 입·출력 신호들에 더하여 연산기를 활성화시키는 요구(Req) 신호와 연산의 완료를 알리는 응답(Ack) 신호를 추가적으로 가진다. 그림 2(b)의 유한상태기에서 점선 연결 및 상태 S_{01}, S_{02}, S_{03}, S_{11}, S_{12}, S_{13}은 그림 2(a)의 DFG를 2-단계 지연시간 가산기를 이용하여 구현하였을 때 새로이 추가되어지는 연결 및 상태들을 나타낸다. 이때, 연결 1, 2, 3은 O_1만 끝났을 때, O_2만 끝났을 때, O_1과 O_2가 동시에 끝났을 때를 각각 나타낸다. 이처럼 가변지연시간 연산기의 사용은 기존에 비해 보다 복잡한 데이터 경로의 동작을 야기하면서 대응하는 유한상태기의 복잡도를 증가시킨다. 그림 2(b)의 유한상태기에서 S_{01}, S_{02}, S_{03}와 S_{11}, S_{12}, S_{13}은 각각 새로운 S_0'과 S_1'으로 통합될 수 있음을 주목하라. 그러나 상태 통합 이전의 유한상태기와 통합된 상태를 가진 유한상태기 모두 동기화에 따른 성능의 감소를 야기할 수 있다. 원래의 DFG가 가지고 있는 병행성을 보장하는 유한상태기는 그림 2(c)와 같으며 기존의 유한상태기에 비하여 훨씬 복잡한 모습을 보여준다. 그림 2(c)의 유한상태기에서 S_{ab}은 연산 a의 시작과 연산 b의 긴 연산이 수행되어질 것임을 의미한다. 또한 a, b가 0의 값을 가지는 것은 해당 연산기에 할당된 모든 연산이 완료되었음을 의미한다. 이처럼 가변지연시간 연산기를 사용하여 구성된 데이터 경로가 DFG의 병행성을 온전히 보장할 수 있도록 유한상태기를 기술하는 것은 상태개수의 급속한 증가와 같은 유한상태기의 복잡도의 급속한 증가를 야기할 수 있다. 3.3에서는 이러한 문제를 해결하기 위해 본 논문에서 새로이 제안하는 방법을 설명하도록 한다.

3.3 분산된 유한상태기들의 집합의 생성

3.2에서 제안한 요구-응답 신호를 가진 유한상태기는 중앙집중적인 형태를 가지며, 이는 DFG의 크기가 증가하거나 혹은 내재한 병행성이 높아질 경우 상태수의 급속한 증가를 유발할 수 있다. 일반적으로 상태수의 증가는 논리합성 시간의 증가, 합성된 회로 면적의 증가 및 성능의 감소를 야기할 수 있다. 본 저자들은

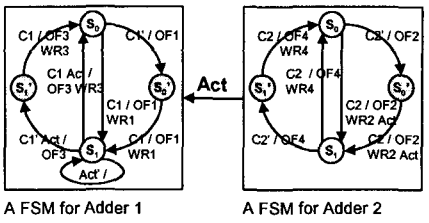

그림 5 할당된 연산기를 기준으로 분할 유도된 유한상태기들의 집합 및 관계

이러한 문제를 해결하기 위하여 분산된 형태의 유한상태기 집합을 유도하는 방법을 제안하며, 다음은 이를 위한 알고리즘이다.

[알고리즘 1] 분산된 유한상태기 집합의 유도

단계-1 주어진 DFG에 대하여 자원결합을 수행한다.

단계-2 동일한 자원과 결합한 연산들에 대하여 요구-응답 신호를 가진 독립적인 유한상태기를 생성한다.

단계-3 단계-2에서, 연산 O_i가 다른 자원을 할당받은 연산 O_j를 선행 연산으로 가질 경우에 연산 O_i를 수행하는 상태 S는 O_j의 연산완료에 대응하는 신호를 O_i의 연산 시작을 위한 조건신호로 받아들인다.

그림 3은 그림 2(a)의 DFG(점선 연결을 포함)에 대한 분산된 형태의 유한상태기를 보여준다. 그림 3의 유한상태기는 연산 O_1, O_3와 O_2, O_4에 각각 1개씩의 2단계 지연시간 가산기를 할당한 것을 가정하였다. 가변지연시간 연산기를 포함한 데이터 경로에 대한 제어부에 대응하는 유한상태기의 상태수의 증가원인은 다음의 두 가지이다. 첫째, N-단계 연산시간 연산기를 사용할 경우에 연산은 N가지 경우의 지연시간 후에 완료되어질 수 있으며 이를 해당 제어부가 인지하기 위해서는 대응하는 유한상태기의 상태개수가 증가하게 된다. 둘째, DFG는 입력 연산자의 순차적인 연산과 병행적인 연산을 모두 표시할 수 있으며 연산간의 병행적인 동작은 가변지연시간 연산기의 사용과 더불어 급속도의 상태수 증가를 야기할 수 있다. 예를 들어, 그림 2(a)의 DFG의 O_1과 O_2의 연산완료는 2-단계 지연시간 연산기를 사용할 경우에 (S_0, S_0), (S_0, S_0'), (S_0', S_0), (S_0', S_0'')의 4가지 경우를 가질 수 있다. 사실상, 첫번째 경우는 가변지연시간 연산기의 사용에 따른 불가피한 경우이며 상태수의 증가 또한 그리 크지 않다. 그러나 두 번째 경우는 연산간의 병행성에 기인한 것이므로 유한상태기의 분할을 통하여 효과적으로 대처할 수 있다. 동일한 연산기를 사용하는 연산들은 자원의존관계에 따라 언제나 순차적으로 수행되어져야 한다. 그러므로 동일한 연산기를 할당받은 연산들에 대하여 생성된 유한상태기는 병행성을 지원할 필요가 없으며 결과적으로 크기의 증가가 크지 않다. 게다가 데이터 및 자원의 측면에서 병행적인 관계에 있는 연산

표 2 그림 2(a)의 DFG에 대해 생성된 FSM의 비교

	실행시간(ns) BD/WD	면적 S / C	상태 개수	FF의 개수
FSM	20 / 20	22 / 3	2	1
FSM_OLD1	10 / 20	44 / 37	4	2
FSM_OLD2	10 / 20	88 / 317	12	4
FSM_NEW	10 / 20	102 / 104	8	5

2-단계 지연시간 가산기: 단(短)지연(5ns)/장(長)지연(10ns)
BD: Best Case Delay WD: Worst Case Delay

들 사이의 병행성은 분할된 유한상태기 사이의 독립적인 동작으로 표현될 수 있다. 그러므로 본 논문에서 제안하는 할당된 연산기를 기준한 유한상태기의 생성은 가변지연시간 연산기의 사용에 따른 유한상태기의 상태공간의 급속한 증가에 대한 효과적인 해결책이라고 할 수 있다.

IV. 실험결과 및 논의

본 논문에서는 가변지연시간 연산기를 포함한 데이터 경로에 대한 효율적인 동기식 제어기의 설계 방법을 제안하였다. 제안된 방법은 할당된 연산기들을 기준으로 분할된 유한상태기들의 집합을 유도함으로써 기존의 방법에 비하여 보다 적은 면적증가만으로 DFG의 연산들 사이의 병행성을 온전히 보장하는 동기식 제어부를 설계할 수 있다. 표 1은 그림 2(a)의 DFG에 대하여 기존의 방법들과 제안된 방법을 통하여 유도된 유한상태기들의 분석결과를 보여준다. 표 1에서 FSM, FSM_OLD1, FSM_OLD2, FSM_NEW는 각각 전형적인 동기식 유한상태기(그림 2(b): 실선부분), 가변지연시간 데이터 경로를 제어할 수 있는 기능을 가진 기존의 유한상태기(그림 2(b): 점선부분 포함), 연산간의 병행성을 온전히 보장하기 위한 유한상태기(그림 2(c)), 제안된 방법에 기반하여 유도된 분산된 형태의 유한상태기를 나타낸다. FSM_OLD2의 결과가 보여주는 것처럼 DFG에 속한 연산들 사이의 병행성을 단일한 유한상태기의 형태로 기술하는 것은 유한상태기의 복잡도를 크게 증가시킬 수 있으며 이는 면적의 증가를 야기한다. 이에 반해, 두개의 유한상태기로 분할 유도된 FSM_NEW의 경우는 FSM_OLD2에 비하여 면적의 증가가 적음을 알 수 있다. 그러나 FSM_NEW의 면적은 FSM_OLD1에 비하여 상당히 크며, 이는 연산간의 병행성을 온전히 보장하여 주기 위해 요구되어지는 부가적인 회로의 면적이다. 그림 4는 그림 2(a)의 DFG에 대하여 가산기의 단(短)지연과 장(長)지연의 비율을 변화시켜 가면서 수행시간의 변화를 측정한 것이다. 그림 4에서 볼 수 있는 것처럼 FSM_NEW의 수행시간

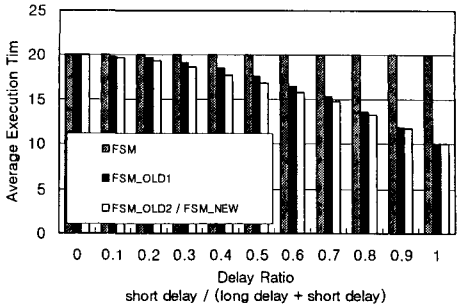

그림 6 가변지연시간 가산기의 단(短)지연과 장(長)지연의 비율의 변화에 따른 그림 2(a)의 DFG의 수행시간의 변화

은 FSM_OLD1에 비하여 작으며, 이는 FSM_NEW가 DFG에 속한 연산들 사이의 병행성을 온전히 보장하는데 기인한다. 이러한 성능의 향상은 사용되어지는 가변지연시간 연산기의 지연시간 단계의 개수 증가, DFG의 규모의 증가 및 다양한 종류의 가변지연시간 연산기의 사용과 함께 보다 크게 증가할 것으로 기대된다.

Acknowledgment

본 연구는 한국과학재단의 한일국제공동연구과제 (20006-302-01-2) 및 광주과학기술원 초고속 광 네트워크 연구센터를 통한 한국과학재단 우수연구센터 지원금에 의한 것입니다.

참고문헌

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

[2] S. M. Nowick, K. Y. Yun, A. E. Dooply, P. A. Beerel, "Speculative Completion for the Design of High-Performance Asynchronous Dynamic Adders," In Proceedings of Async'97, pp.210-223, 1997.

[3] L. Benini, E. Macii, M. Poncino and G. DeMicheli, "Telescopic Units: A New Paradigm for Performance Optimization of VLSI Designs," IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, Vol.17, No.3, Mar. 1998.

[4] L. Benini, E. Macii, and M. Poncino, "Efficient controller design for telescopic units," In Proceedings of IEEE International Conference Innovative Systems in Silicon, pp.290-299, Oct. 1997.