1998년도 대한전기학회 정기총회 및 추계학술대회 논문집 1998. 11. 28

Self-timed 기반의 Node Label Data Flow Machine 설계

°김 희숙, 정 성태, 박 희순
원광대학교 컴퓨터공학과

Design of a Node Label Data Flow Machine based on Self-timed

Hee-sook Kim, Sung-tae Jung, Hee-soon Park,
Dept. of Computer Eng. Wonkwang Univ.

Abstract - In this paper we illustrate the design of a node label data flow machine based on self-timed paradigm. Data flow machines differ from most other parallel architectures, they are based on the concept of the data-driven computation model instead of the program store computation model. Since the data-driven computation model provides the excution of instructions asynchronously, it is natural to implement a data flow machine using self timed circuits.

1. 서 론

Data Flow Machine(DFM)들은 전역클럭으로 동기화되어 실행되는 프로그램 내장 방식 컴퓨터와는 다른 구조를 가지고, 임의의 노드에 데이터 도착 여부로 실행을 하는 비동기 방식의 병렬처리 장치이다.

노드들은 항상 자신의 모든 입력 단자에 필요한 모든 데이터 비트들이 도착할때만 firing을 하는데, 어느 시점에서 이 모든 데이터들이 도착되었음을 노드에게 신호할 것인가의 문제는 동작 속도와 동기화 문제에 있어서 중요하다. 따라서 기 설계된 Node Label Data Flow Machine(NLDFM)에서 Enable Token Flag(ETF)가 111일때만 firing할 수 있었던 조건에 C-element를 적용함으로써 Self-timed 회로로 구현하고 firing의 동기화 문제를 해결한다. 본 논문은 DFM의 여러 가지 모델들 중 Davis 요구 사항에 만족하고, Dennis와 Manchester 모델에 근거하여 설계하였다. 각 모듈 사이의 통신 기법은 2-cycle을 사용한다.

2. 본 론

2.1 NLDFM의 전체적인 시스템 구조

2.1.1 NL의 생성

연산 과정에서 발생하는 데이터의 직렬성, 제어성, 동시성을 판별하기 위해 입력언어로는 Data Flow Graph(DFG)를 사용하였고, Node Label(NL)을 이용하여 Node token(NT)이 실행되도록 구성하였다.
NL은 임의 노드의 직렬순서(i)와 병렬순서(j)를 2차원 첨자로 구성된다.

$$NL = i,j \ (1 \le i \le S_{max},\ 1 \le j \le P_{max}) \qquad (1)$$

i는 데이터의 의존도를 나타내고, j는 병렬처리의 동시성을 나타낸다. S_{max}는 순차 처리시 최하위 순서가 되고, P_{max}는 시스템의 총 프로세서 수가 된다.
NT을 생성하기 위해 목적지 입력 아크는 목적지 입력 시간에 따른 프로세서수와 목적지 노드 입력 단자를 갖는다. 따라서 출력 아크 DA는 다음과 같이 구성되고, Dk는 L,R,C의 값을 갖는다.

$$DA = Di,\ Dj,\ Dk \qquad (2)$$

위의 정의에서 NT를 명령 패킷 형태로 표현하면 다음과 같다.

$$NT = f,\ NL,\ DA, DA' \qquad (3)$$

DA, DA'는 출력 아크의 수를 나타내고, 모든 명령 NT는 해당 결과의 목적지 아크를 가지고 있다. 따라서 데이터의 소스 아크는 코딩할 필요가 없다. 모든 노드들의 실행을 위한 명령어의 구조는 그림1과 같다.

Node Token	Opernad Data	Control Data	Flag

그림1. 명령어 구조

Operand data와 Control data는 경우에 따라1개 또는 2개가 되고, Flag에 따라 데이터의 유무를 판단한다.

2.1.2 전체적인 시스템 구성

NLDFM은 Davis 요구 사항에 만족하고, Dennis와 Manchester 모델에 근거하여 설계하였다. 전체적인 시스템은 입출력 제어부분과 Processing site인 PM 그리고, 각 PM에서 생성된 결과 토큰을 목적지로 보내는 Distributor로 구성되어 그림2와 같다.

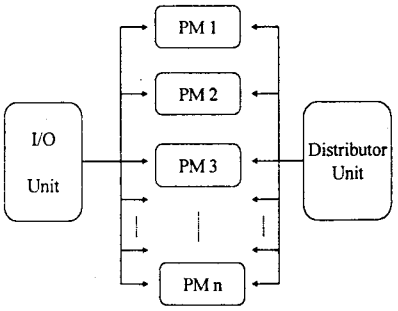

그림 2.. 전체적인 시스템 구조

2.1.3 PM의 구조

각 PM의 구조는 그림 3과 같다. Node Token, Operand, ETF값을 보관하는 Local memory와 Processing unit 및 Distributor를 위한 레지스터들로 구성되어 있고, 링타입의 실행 사이클을 갖는다.
각 장치들의 기능을 살펴보면, NTM은 Op code와 1개 또는 2개의 목적지 NL값을 보관하고, ETF로부터 enabled 토큰의 주소를 공급 받는다. DTML/R은

NTM과 동일 주소에 위치한 노드가 사용할 Operand 데이터를 보관하는 메모리로서 ETF로부터 addressing 된다. CF는 Control node가 필요로 하는 제어용 데이터를 보관하는 1비트 메모리이다. ETF는 노드의 실행 가능 여부를 표시하는 flag로서 L/R/C값은 각각 DTML/R 및 CF내의 데이터가 도착하였는지의 여부를 나타낸다. 결과의 read/write 과정마다 ETF=111인지를 체크한다. PU는 노드 토큰의 Op code를 디코드하고, operand 데이터에 대하여 산술, 논리, 제어연산을 수행한다. 실행결과는 Distributor로 보내져서 목적지 주소로 보내진다.

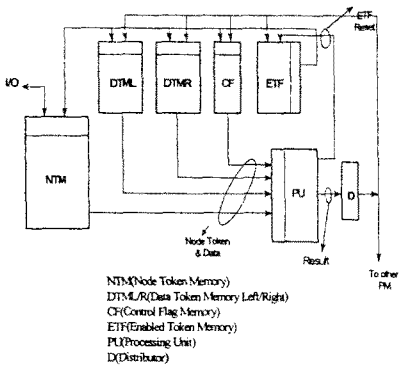

그림 3. PM의 기본 구조

2.2 Self-timed 기반의 NDFM 설계

2.2.1 Self-timed 회로

Self-timed 회로는 비동기적인 하드웨어 설계를 위한 하나의 방법이다. 전역 클럭을 사용하지 않고 시스템 구성요소들 사이의 특정 프로토콜에 의하여 동작이 제어되므로 VLSI소자를 더 효율적으로 만든다. Self-timed system은 Data 채널, Request 채널, Acknowledge 채널로 구성되고, 모듈 사이의 비동기 통신 프로토콜을 사용한다.

모듈사이의 통신 기법은 2-cycle과 4-cycle이 있다. 2-cycle은 유효한 데이터만 전달하므로 4-cycle보다 속도는 빠르고 전력도 덜 소모하지만 회로가 커질수 있는 단점이 있다. 본 논문에서는 2-cycle 기법을 사용했다. 그림 4는 Self-timed system 모듈 구성도이다.

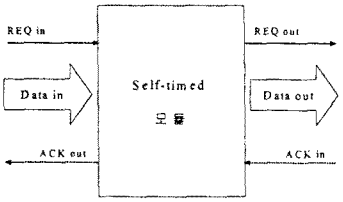

그림 4. Self-timed 모듈

2.2.2 구현

그림 5는 NLDFM의 Self-timed 모듈화된 PM구조의 블록도이다.

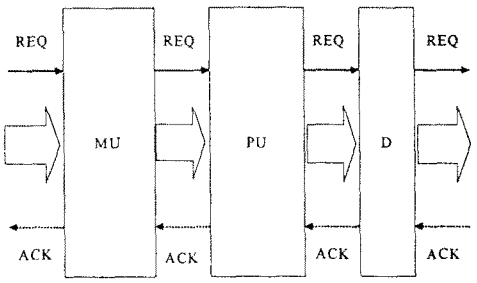

그림 5. Self timed 모듈로 구현된 PM 구조

Local memory인 NTM과 DTML/R, CF을 하나의 블럭 MU로 만들고, PU와 D로 각각 모듈화시켜 구성하였고, 파이프라인 구조를 이루도록 하였다. 각 Self-timed 모듈들은 Data 채널, Request 채널, Acknowledge 채널로 구성되고, 모듈 사이의 2-cycle 비동기 통신 프로토콜을 사용한다.

이 파이프라인에서 모듈사이의 동기화는 Muller의 C-element, TOGGLE 모듈, CALL모듈등을 이용하여 구현하였다.

Muller의 C-element는 표 1과 같은 동작을 한다. AND 게이트와 같이 동작하지만 두 개의 입력값이 같은 논리는 결과도 같은 값을 그대로 유지하고, 두 개의 입력값이 다를때는 그 이전 상태의 값을 유지하면서 출력이 변하지 않고 값을 그대로 유지하기 위한 기억장치와 같은 기능을 하고 있다. 따라서 두 개의 입력중 하나의 event만을 취한후 결과도 하나의 event만 생성한다.

Input 1	Input 2	Result
0	0	0
0	1	Last Value
1	0	Last Value
1	1	1

표 1. Muller의 C-element 동작

Muller의 C-element는 dynamic과 static을 구분하기 위하여 bubble기호를 사용하기도 한다.

TOGGLE 모듈은 입력 아크에 토큰이 도착하면 두 개의 출력중 한쪽으로 번갈아가면서 firing하는 기능을 갖는다. Select 모듈은 입력 아크에 들어온 토큰을 선택조건에 따라 한쪽 출력으로 firing하거나, 다른 한쪽 출력으로 firing한다. CALL은 입력 토큰중에 가장 나중에 들어온 토큰을 기억한다. 즉 결과 토큰을 기억한다. ARBITER는 도착 순서가 정해지지 않은 두 개의 토큰의 순서를 결정한 뒤 결과를 Distributor로 보내는 결정적인 역할을 한다. 만약 같은 시간에 토큰 결과가 도착했어도 하나의 토큰만 firing하도록 결정해야하는 기능을 갖는다. ARBITER와 CALL은 직접적으로 연결이 되어서 완전히 독립적인 프로세스 수행이 가능하다.

그림 6은 Muller C-element, ARBITER, CALL, SELECT간의 Request, Acknowledge신호를 통하여 PU에서 나온 result 토큰을 Distributor로 보내는 Self-timed 회로이다.

R1에 첫 번째 토큰이 도착하면 S=1로 set되고 G1을 생성한다. ARBITER에서는 D1이 생성됐다는 신호를 받을때까지 R2에서 토큰을 받는다. 반대로 R에서 토큰이 발생하는 bubbled Muller C-element와 CALL에서는 토큰이 firing하게 된다.

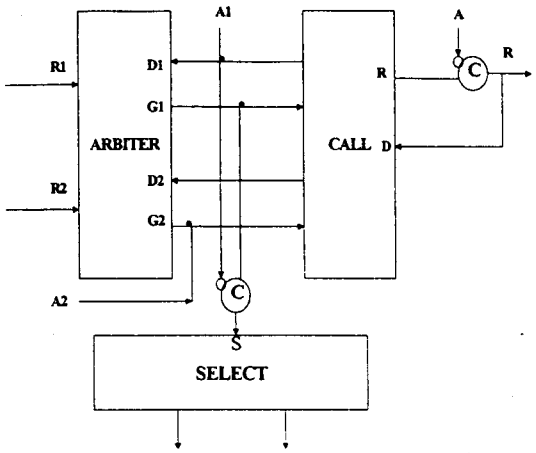

그림 6. Result token의 firing 회로도

3. 결 론

본 논문에서는 Muller의 C-element를 이용한 Self-timed Node Label Data Flow Machine을 설계하였다. DFM은 임의노드에 데이터 도착 여부의 판단을 어떠한 방법으로 활용할 것인가가 중요한 의미를 지닌다. 노드들은 항상 자신의 모든 입력 단자에 필요한 모든 데이터 비트들이 도착할때만 firing을 하는데, 어느 시점에서 이 모든 데이터들이 도착되었음을 노드에게 신호할 것인가의 문제는 동작 속도와 동기화 문제에 있어서 중요하다.

본 논문에서는 Self-timed 회로로 설명되는 Muller의 C-element를 이용하여 기존의 NLDFM에 적용시킴으로써 동기방식의 ETF를 이용할때보다 정확한 firing 과정을 확인할 수 있었다. 시뮬레이션은 VHDL을 이용하였다. 설계가 대규모이기 때문에 전체적인 모듈과 리스트를 작성하는데 어려움이 있었고, 여러개의 PM들에서 병렬처리를 해야하는 관계로 어려움이 있었다. 아직도 많은 연구가 되고, 좀더 더 보완된 실험을 해야할 것이다.

[참 고 문 헌]

[1] Shih-Len Lu, Chih-Ming Cahng, "Modeling of a Selftimed Dataflow Processor in VHDL", Proc 6th IEEE ASIC Conference, 1993

[2] Ilana David, Ran Ginosar, Michael Yoeli, "Implementing sequential machines as self-timed circuits", IEEE transactions on computers, VOL 41.NO.1, 1992,

[3] G.Theodoropoulos, J.V.Woods, "Distributed simulation of asynchronous computer architectures:The program driven approach"

[4] Arvind, Agerwala, "Data Flow system", 1982, computer

[5] Guang R.Gao, Gabriel M.silberman, "Parallel Architectures and Compilation Techniques", MICHEL, 1994

[6] Ivan E.Sutherland, "Micropipelines", communications of the ACM, June 1989, volume32, Number 6.