병렬 컴퓨터를 위한 저지연 프로그램형 조견표 경로지정 엔진

(Low-Latency Programmable Look-Up Table Routing Engine for Parallel Computers)

장 래 혁 †

(Naehyuck Chang)

요 약 병렬 컴퓨터의 메시지 전달에서 응용에 관계없이 일반적으로 우수한 경로 지정 및 스위칭 정책은 존재하지 않으므로, 사용자가 응용에 따라서 정책을 변경할 수 있게 하는 것이 바람직하다. 본 논문에서는 마이크로프로세서 구조에 기초한 경로 지정 엔진과는 달리, 성능의 감소 없이 융통성 있는 경로 지정과 스위칭 기능을 수행할 수 있는 조견표(look-up table) 경로 지정 엔진의 구현에 대하여 기술한다. 제안된 경로 지정 엔진은 조견표의 내용을 바꿈으로써 웜홀(wormhole), 가상 컷스루우(virtual cut-through) 및 패킷 스위칭(packet switching) 등은 물론, 다양한 경로 지정 알고리즘의 혼성(hybride) 스위칭을 구현할 수 있다. 경로 지정 엔진의 조견표는 파이프라인 구조로 되어 있어, 하나의 플릿(flit) 정도의 저 지연을 가지므로, 단일 경로 지정 및 스위칭 정책을 하드와이어(hardwired)로 구현한 경우 보다 큰 성능의 감소 없이 다중의 경로 지정 동작을 중첩할 수 있다. 제안된 4개의 파이프 라인단은 해저드(hazard)를 일으키지 않으므로, 고 비용의 포워딩(forwarding) 회로가 필요 없다. 경로 지정 엔진은 시간 공유의 컷스루우 버스나 크로스바(crossbar) 스위치를 갖는 단일 경로로 되어 있는 4개의 물리적 경로를 수용할 수 있다. 제안된 경로 지정 엔진은 Xilinx 4000XL 시리즈 FPGA를 사용하여 구현되었다.

Abstract Since no single routing-switching combination performs the best under various different types of applications, a flexible network is required to support a range of polices. This paper introduces an implementation of a look-up table routing engine offering flexible routing and switching polices without performance degradation unlike those based on microprocessors. By deciding contents of look-up tables, the engine can implement wormhole routing, virtual cut-through routing, and packet switching, as well as hybrid switching, under a variety of routing algorithms. Since the routing engine has a piplelined look-up table architecture, the routing delay is as small as one flit, and thus it can overlap multiple routing actions without performance degradation in comparison with hardwired routers dedicated to a specific policy. Because four pipeline stages do not induce a hazard, expensive forwarding logic is not required. The routing engine can accommodate four physical links with a time shared cut-through bus or single link with a cross-bar switch. It is implemented using Xilinx 4000 series FPGA.

1. 서 론

고성능 연결 망과 메시지 전달에 관한 연구는 병렬 분산 시스템에서 효율적인 연산을 가능하게 하였으며, 다양한 구조의 상호 연결 망과 경로 지정(routing) 및 스위칭 정책에 관한 연구 결과를 도출하였다. 병렬 분산 시스템의 성능을 극대화시키기 위해서는 이와 같은 연구에 기초한 적절한 연결 망을 구축하여야 한다. 그렇지만 동일한 병렬 분산 시스템에서 수행해야 하는 응용 프로그램들은 매우 다양한 패킷 길이와 도착 간격 시간, 목적지를 갖는 메시지를 전달하므로, 수행해야 하는 응

· 본 연구는 한국과학재단 해외 박사후 연수과정 프로그램을 지원받았습니다.

† 종신회원 : 서울대학교 컴퓨터공학과 교수
nachyuck@snu.ac.kr

논문접수 : 1999년 8월 10일
심사완료 : 1999년 11월 29일

용 프로그램들의 특성들을 고루 만족시키는 연결 망의 설계는 매우 중요하고도 어려운 일이다.

예를 들어, 적응 경로 지정(adaptive routing) 방법은 종단간 지연 시간을 줄일 수 있으나, 순서가 맞지 않는 패킷의 도착은 프로토콜의 처리를 복잡하게 한다. 웜홀 스위칭은 패킷 버퍼의 필요 없이 전달 지연 시간(latency)을 적게 할 수 있으나, 보다 높은 부하에서는 가상 컷스루우 방식과 패킷 스위칭 방식이 보다 나은 처리율(throughput)을 낼 수 있다. 결국 이상적인 연결 망은 모든 응용 프로그램의 특성을 만족시켜야 하나, 이와 같이 모든 조건하에서 최선의 동작을 하는 단일의 경로 지정과 스위칭 정책의 조합은 존재할 수 없다[1, 2, 3, 4].

따라서, 응용 프로그램의 특성에 따라 융통성 있는(flexible) 경로 지정 및 스위칭 정책을 사용할 수 있는 연결 망은 이러한 요구 사항을 잘 만족시키는 방법이 된다. 이를 실현하려면 연결 망 정책을 수행하는 경로 지정 장치(router)는 광범위한 경로 지정 및 스위칭 정책들을 응용 프로그램에 따라서 지원해야 한다[5, 6, 7]. 그러나, 기존의 대부분의 경로 지정 장치에서는 단일의 경로 지정 및 스위칭 조합이 하드와이어 형태로 구현되므로 기존 기법을 사용하여서는 융통성 있는 경로지정 장치를 구성하기 힘들다[5].

하드와이어 기법으로 경로 지정 장치를 구성하는 것과 비교하여 융통성 있는 경로 지정 장치를 구성하는 방법으로, 마이크로프로세서 구조를 채택하고 응용 프로그램에 따라서 마이크로프로그램을 변경하는 것을 고려할 수 있다. 이러한 경로 지정 장치는 연결 망 정책의 융통성 면에서는 매우 우수한 성능을 제공하는 반면[3, 4], 마이크로프로그램 방법에서 발생하는 지연 시간이 문제가 된다. 일반적인 마이크로프로세서 기반의 경로 지정 엔진에서는 도착한 패킷에 경로를 지정하고, 이를 전송하기 위하여 목적 노드로 향한 송신 자원을 예약하며, 패킷 헤더를 전송하는데 필요한 사이클의 하한이 대략 16 사이클 정도가 된다. 보다 복잡한 연결 망 정책을 위하여 마이크로프로그램의 크기가 더 커지는 경우에는 경로 지정 지연 시간이 수행하여야 마이크로프로그램의 크기에 따라 증가하게 된다. 예를 들어 적응 경로 지정의 경우, 경로를 지정하는 데에 22 사이클을 사용하며, 송신 채널을 예약하기 위해 스위치에 접근하여 패킷 헤더를 전송하는 것은 적어도 8 개의 추가 사이클을 사용하게 된다[5, 6].

이러한 경로 지정 엔진의 성능 저하는 기존의 하드와이어로 구성된 경로 지정 엔진 대신 유연한 경로 지정 엔진을 선정하는 이점을 모두 상쇄시킬 수 있을 만큼 심각하다. 하드웨어로 구성되어 단일 경로 지정 및 스위칭 정책을 갖는 경로 지정 엔진은 대개 몇 개의 플릿(flit) 사이클 내에 패킷에 경로를 지정한다[8]. 이 경로 지정 지연 시간은 컷스루우 스위칭 연결 망에서는 전체의 통신망 지연 시간의 중요한 요소이다. 따라서, 연결 망 정책의 융통성 있는 선택만으로는 이와 같은 큰 성능 격차를 항상 극복해 낼 수는 없다. 본 논문에서는 성능 저하를 야기하지 않고도 경로 지정 및 교환 정책을 융통성 있게 선택할 수 있는 경로 지정 엔진의 구조를 제안하고 아울러 이를 FPGA로 구현하여 실용 가능성을 함께 보여준다.

본 논문에서 구현하는 경로 지정 엔진의 구조는 조견표를 사용하는 것이나, 전통적인 조견표 경로 지정 엔진에서와 같이 헤더에 있는 목적지 ID를 통하여 단순히 경로를 참조하는 것[3]과 크게 다르다. 목적지를 저장한 조견표 경로 지정 엔진은 연결 망내의 모든 노드들에 대한 정보를 가지고 있는 경로 지정 조견표를 필요로 하므로 통신망내의 노드 수가 조견표의 크기에 따라 제한된다. 목적지 주소대신에 비트 패턴의 조합을 이용하여 조견표의 크기를 감소시키는 방식[6]은 조견표의 크기가 링크의 차원에 비례하나 유연성이 많이 떨어진다. 구현된 경로 지정 엔진은 연결 망 정책이 조견표에 저장되어 하드웨어로 구성된 경우처럼 동작하므로, 단지 기존의 조견표 경로 지정 엔진보다 메모리 요구량 만을 줄이는 것이 아니라 보다 나은 융통성을 아울러 제공한다. 이와 더불어 제안된 경로 지정 엔진의 조견표 참조에 따른 지연시간을 줄이기 위하여 파이프라인 구조를 채택하여 성능을 높여 하드웨어로 구성된 경로 지정 엔진에 근접하도록 한다.

본 논문은 다음과 같이 구성된다. 2 장에서는 제안하는 경로 지정 엔진의 원리를 언급하고, 3 장에서는 경로 지정 엔진의 구조에 대하여 설명한다. 4 장에서는 원형 시스템의 구현에 대하여 서술한다. 결론은 5 장에서 맺는다.

2. 조견표를 사용한 경로 지정 엔진

구현되는 조견표를 사용한 경로 지정 엔진은 그림 1에서 보는 것과 같이 입력 버퍼만 가지고 있는 일반적 경로 지정 모델[9]을 따른다. 경로 지정 엔진은 조견표와 제어 논리 회로로 구성된다. 스위치는 요구되는 대역폭에 따라 크로스바 스위치 또는 시분할 버스 모두 가능하다. 본 논문에서 다루는 경로 지정 엔진은 4 개의 입력과 출력 경로와 호스트 인터페이스 경로를 마련하

여 그물(mesh) 형태의 연결 망 구조를 지원한다. 본 논문에서는 그림 1의 경로 지정 엔진 중에서 조건표를 사용한 경로 지정 엔진의 구조에 집중하여 서술한다. 단순성을 위해 호스트 인터페이스는 다른 경로들과 동일한 것으로 간주하며, 따라서 자세한 호스트 인터페이스에 관한 서술은 제외한다.

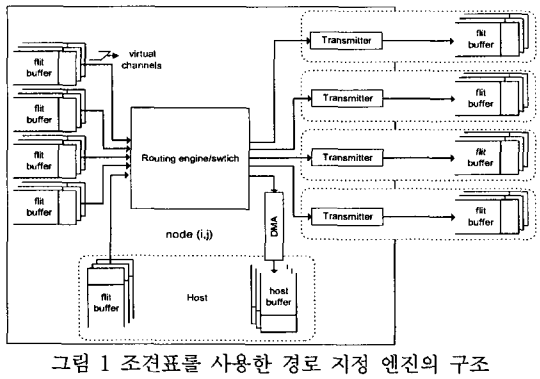

그림 1 조건표를 사용한 경로 지정 엔진의 구조

본 논문에서 제안하는 경로 지정 엔진에서는 단일한 경로 지정과 스위칭 정책의 조합을 구현하지 않는 대신에, 적당한 경로 지정과 스위칭 정책의 조합이 호스트에 의해 제한된 범위 내에서 프로그램 될 수 있다. 경로 지정 엔진은 패킷 경로 지정의 주요한 4 단계 중에서 3 단계를 수행한다. 첫째로, 헤더 분석(header parsing)과, 둘째로 목적지 노드로의 경로 연산(route computation), 그리고 마지막으로 연산을 통하여 얻은 가능한 경로 중에서 사용 가능한 적절한 경로 선택(route selection) 기능을 수행한다[5].

본 논문에서 제안하는 경로 지정 엔진은 경로 지정의 유연성 못지 않게 경로 지정 시간 지연 및 구현 복잡도의 감소에 큰 비중을 두어, 실제로 효율적인 구현 가능성을 목표로 한다. 대신에, 마이크로프로그램 방식과는 달리 가능한 경로 지정 명령어를 선택 범위를 제한한다. 이를 위하여 헤더의 형식을 표 1과 같이 간략화하여 조건표와 이를 제어하는 하드웨어로 구성된 스테이트 머신에 의하여 효율적으로 처리가 가능하도록 한다. 패킷의 목적지를 표현하는데는 절대 주소와 상대 주소(offset address)를 사용하는 방법이 있는데, 본 논문에서는 경로 지정 알고리즘이 사용되는 노드의 주소에 독립적으로 유지될 수 있도록 상대 주소 방법을 사용하여 표현한다. 패킷 상태(packet status)는 보다 복잡한 경로 지정 알고리즘을 위한 것으로, 다양한 스위칭 정책들을 위한 플래그(flag)나 카운터로 사용된다. 실제로도 많은 경로 지정 방법들이 헤더 내에 패킷 상태를 유지한다. 예를 들면, 한 상태는 한 패킷이 특정한 집합으로부터 어떠한 채널을 통과 했는지의 여부나 혹은 잘못된 경로가 지정된 횟수를 지시할 수도 있다[5].

표 1 헤더 서식

패킷 목적지 (packet destination)		패킷 상태 (packet status)	스위칭 제어 (switching control)
Xoffset	Yoffset	Z	H

대부분의 경로 지정 알고리즘은 목적지 주소, 착신 경로(incoming link), 가상 채널 번호, 그리고 사건 카운터, 부울(Boolean) 플래그 등과 같은 패킷 상태의 함수로 표현이 가능하므로 대부분 계층적 CASE 문의 집합으로도 표현이 가능하다. 알고리즘의 표현을 간단히 하기 위하여 표 2와 3에서와 같은 기호를 사용하여 표현한다. 표 4에서 기본적인 이차원 경로 지정 알고리즘인 XY-경로 지정 알고리즘을 계층적 CASE문으로 변형한 것을 볼 수 있다. 계층적 CASE 문으로 표현된 알고리즘은 표현상으로는 복잡하게 보이나, 다단의 참조표를 사용하여 표현하는데 알맞은 구조가 된다.

표 2 입력 방향 부호화

InDir	X-	Y-	X+	Y+
	E	S	W	N
Encode	000B	010B	100B	110B

표 3 출력 방향 부호화

OutDir	X-	X-,Y-	Y-	X+,Y-	X+	X+,Y+	Y+	X-,Y+
	E	SE	S	SW	W	NW	N	NE
Xoffset	+	+	0	-	-	-	0	+
Yoffset	0	-	-	-	0	+	+	+
Encode	000B	001B	010B	011B	100B	101B	110B	111B

첫 번째의 CASE 문은 패킷이 도착한 가상 채널에 따라 서로 다른 경로 지정 알고리즘의 적용을 가능하게 한다. 가상 채널 번호는 조건표의 어드레스로서 다음 단으로 전송된다. 두 번째 CASE 문은 패킷이 도착한 경로의 방향에 따라 적절한 경로 지정 알고리즘을 적용할 수 있게 한다. 아울러 도착한 패킷의 주소 오프셋을 추

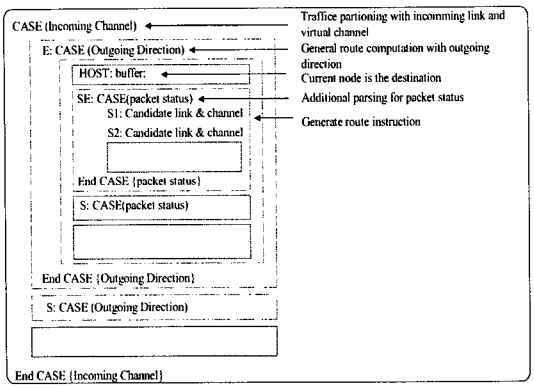

그림 2 계층적 CASE 문으로 표현된 경로 지정 알고리즘

출함으로써 목적지의 방향을 추출한다. 세 번째 CASE 문은 보다 복잡한 경로 지정 알고리즘을 위해 패킷 상태를 사용하는 부가적인 헤더 해석 기능을 제공한다. 호스트에서는 사용하고자 하는 경로 지정 알고리즘에 따라서 적절히 헤더의 상태를 검사하는 알고리즘을 조견표에 다운로드 함으로써 필요한 헤더 상태 검사를 위한 연산을 수행한다. 결국, 3 계층의 CASE 문을 통하여 목적 노드로 패킷을 전송하는데 사용할 수 있는 경로의 집합(경로 지정 명령어, route instruction)을 생성해 낸다. 스위칭 제어부는 경로 지정 명령어 중에서 가능한 하나를 예약하며, 예약이 가능하지 않을 경우 지정된 스위칭 정책에 따라서 패킷을 호스트에 임시로 저장하거나, 패킷 흐름을 정지시키거나 한다.

결론적으로 본 논문에서 제안하는 조견표를 사용하는 경로 지정 엔진은 그림 2의 형식을 따르는 계층적 CASE 문으로 표현될 수 있는 경로 지정 알고리즘만 수용이 가능하다. 알고리즘의 수행 속도는 수용 가능하기만 하면 알고리즘의 종류와 무관하며, 그 지연 속도 또한 매우 작게 된다. 다시 말하면, 그림 3의 구조와 같은 계층적 CASE 문으로 표현될 수 없는 경로 지정 알고리즘은 제안된 경로 지정 엔진에서 수행 될 수 없다. 즉, 본 논문에서 제안하는 경로 지정 엔진의 구조는 수용할 수 있는 알고리즘의 자유도 면에서 매우 뛰어난 마이크로프로그램 방식의 경로 지정 엔진과, 매우 적은 지연 시간을 보이는 단일 알고리즘의 하드웨어 경로 지정 엔진의 장점을 수용한 구조로, 전달 지연 시간이 단일 알고리즘의 하드웨어 경로 지정 엔진과 유사하도록 수용 가능한 유연성을 적절히 희생한 구조가 된다.

제안되는 경로 지정 엔진은 계층화된 CASE문을 하드웨어 수준으로 구현하여 적은 연산 지연 시간을 보이도록 하며, 조견표는 사용자 프로그램에서 변경이 가능한 구조로 되어 있어 사용자 수준에서 경로 지정 알고리즘을 변경할 수 있도록 한다. 복잡한 마이크로프로세서 대신에 고속 메모리, 멀티플렉서, 플립플롭 및 기본 논리 게이트들로만 구성된 하드웨어로 기존의 단일 알고리즘 하드웨어보다 크게 뒤지지 않는 특성을 보인다.

표 4 XY-routing 알고리즘

XY-Routing 알고리즘	계층적 CASE 문으로 표현된 XY-routing 알고리즘

Algorithm: XY-Routing for 2-D meshes
Inputs: Xoffset, Yoffset
Output: Channel
Procesure:
If (InDir == E)
Xoffset = Xoffset + 1;
if (InDir == W)
Xoffset = Xoffset - 1;
if (InDir == S)
Yoffset = Yoffset + 1;
if (InDir == N)
Yoffset = Yoffset - 1;
if (Xoffset < 0)
Channel = W;
if (Xoffset > 0)
Channel = E;
if (Xoffset == 0 and Yoffset < 0)
Channel = N;
if (Xoffset == 0 and Yoffset > 0)
Channel = S;
if (Xoffset == 0 and Yoffset == 0)
Channel = H;
	Algorithm: XY-Routing for 2-D meshes
	Inputs: Xoffset, Yoffset
	Output: Channel
	Procesure:
	CASE (InDir)
	E: Xoffset = Xoffset + 1;
	Channel = Dest_Calc(DestDir);
	W: Xoffset = Xoffset - 1;
	Channel = Dest_Calc(DestDir);
	S: Yoffset = Yoffset + 1;
	Channel = Dest_Calc(DestDir);
	N: Yoffset = Yoffset - 1;
	Channel = Dest_Calc(DestDir);
	
	Function Dest_Calc(DestDir)
	CASE (DestDir)
	E: Channel = return(E);
	SE: Channel = return(E);
	S: Channel = return(S);
	SW: Channel = return(W);
	W: Channel = return(W);
	NW: Channel = return(W);
	NE: Channel = return(E);

3. 경로 지정 엔진의 구조

3.1 경로 지정 엔진의 구성 요소

경로 지정 알고리즘의 주요 단계와 소요되는 주요 자원은 다음과 같다.

3.1.1 오프셋 갱신

패킷 헤더의 주소는 목적지 주소의 상대값을 의미하므로 패킷이 착신된 방향에 따라서 상대값을 갱신하여 목적지와 현재 노드의 상대 위치를 파악한다. 즉,

```
CASE (InDir)
        E: Xoffset = Xoffset + 1;
        W: Xoffset = Xoffset - 1;
        S: Yoffset = Yoffset + 1;
        N: Yoffset = Yoffset - 1;
END
```

상대 주소 갱신을 위하여서는 1 비트의 상수와 8 비

트의 변수를 덧셈 또는 뺄셈하는 상수 가감기가 2 개 소요된다.

3.1.2 목적지 노드 방향 계산

목적지 노드의 방향은 현재 노드와 목적지 노드의 상대 위치를 나타내는 것으로 상대 주소값으로 결정되며, 표 3과 같이 표현된다. 목적지 노드의 방향은 단순히 목적지 노드의 상대 위치를 나타내며, 이것이 바로 패킷이 출력되는 자원을 나타내지는 않는다. 만일 X와 Y의 상대값이 모두 0이면 현재 노드가 목적 노드와 일치하는 것을 의미하며 패킷은 호스트의 버퍼 공간으로 복사되므로 별도의 부호를 할당하지 않는다. 방향 계산을 위하여서는 0과 크기를 비교하는 8비트 크기 비교기가 2 개 소요된다. 방향 계산 논리만으로도 비기억형 차원순 알고리즘(oblivious dimen- sion-ordered algorithm)이나 적응 최단 경로 알고리즘(adaptive minimal path algorithm)과 같은 널리 사용되는 경로 지정 알고리즘을 구현할 수 있다.

3.1.3 패킷 상태 검사 및 검사 명령어 표

세 번째 계층의 CASE 문으로, 착신된 패킷의 착신 방향과 가상 채널 및 목적지의 주소 이외에 경로 지정을 위하여 추가로 사용할 수 있는 기능을 제공한다. 이를 위하여 패킷 상태 검사 회로와 패킷 상태 검사 명령어 표를 사용한다. 패킷 상태 검사 명령어 표는 사용자가 변경할 수 있다.

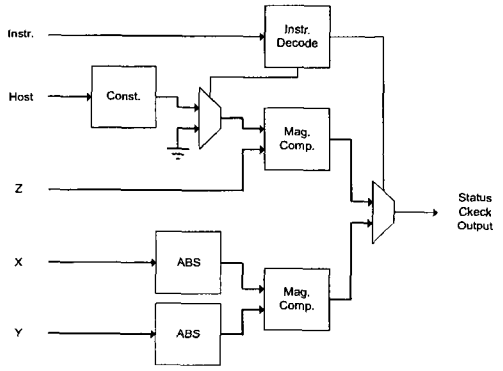

그림 3 패킷 상태 검사 회로

패킷 상태 검사 회로는 그림 3과 같은 구조로 되어 있으며, 표 5에서 나타낸 것과 같은 제한적인 종류의 연산을 수행할 수 있다. 보다 다양한 알고리즘을 모두 수용하려면, 표 5에 나타낸 것 이외에 가능한 모든 조합의 연산을 수행할 수 있어야 하며, 이를 위해서는 마이크로프로세서 구조의 연산 장치를 구비하여야 한다. 본 논문에서는 유연성을 약간 희생하면서 저 비용으로 고성능을 얻기 위한 목적으로 유용한 연산 기능을 한정적으로 지원한다.

표 5 패킷 상태 검사 연산 명령어 구성

Instruction	Encoding	Output = 0	Output = 1
ZLTC	000	$z > c$	$z \leq c$
ZEQC	001	$z \neq c$	$z = c$
FALSE	010	0	×
TRUE	011	×	1
XLTY	100	$\|x\| > \|y\|$	$\|x\| \leq \|y\|$
XEQY	101	$\|x\| \neq \|y\|$	$\|x\| = \|y\|$

패킷 상태 검사 회로를 위해서는 상수 저장을 위한 8 비트 레지스터 1 개, 상수와 0을 선택하는 2 비트 AND 소자 8 개, 두 개의 8 비트 변수의 크기를 비교하는 비교기 1 개와, 2 개의 7 비트 변수의 크기를 비교하는 비교기 1 개, 8 비트의 2의 보수 변수의 절대값을 추출하는 절대값 연산기 2 개가 소요된다. 절대값 연산기는 7개의 2 비트 XOR와 7 비트의 상수 덧셈기가 소요된다.

상태 검사 명령어 표는 사용자가 패킷의 착신 방향과 가상 채널에 따라서 각기 다른 명령어를 사용할 수 있도록, 착신 방향 2 비트, 가상채널 2 비트, 목적 노드 상대 위치 3 비트, 총 7 비트의 주소 공간 및 3 비트의 명령어 길이를 가지는 조건표로 구성된다.

3.1.4 경로 지정 후보 자원 추출

경로 지정 명령어(RTI, routing instruction)는 경로 지정 자원과 스위칭 제어 명령어로 정렬된 순열이며, $P_1P_2P_3..P_nH$ 와 같은 형식을 갖는다[5]. 여기서 P_i는 패킷을 목적 노드로 보낼 수 있는 경로 지정 자원이다. 각 순열의 앞에 있는 자원이 패킷을 목적 노드로 효율적으로 보내는데 유리한 자원이고, 뒤에 나오는 자원은 앞의 자원이 사용 불가능할 경우에 차선책으로 사용할 수 있는 자원을 뜻한다. H는 스위칭 제어 명령어로, 나열된 모든 자원이 사용 불가능할 경우에 취해야 하는 스위칭 동작을 명시한다. 예를 들어 스위칭 제어 명령어가 가상 컷스루우 스위칭을 명령하면 모든 자원이 사용 불가능할 경우에 패킷을 일단 호스트의 메모리에 저장한다. P_i는 경로 할당을 위한 자원의 ID뿐 아니라 자원을 성공적으로 할당받았을 경우에 패킷의 상태를 적절

히 변경하여 다음 노드에서 경로 지정 시에 적절한 동작을 할 수 있도록 한다.

패킷 상태를 검사한 결과는 한 개의 부울 변수로 출력된다. 경로 지정 명령어는 가장 내부에 위치한 CASE 문의 각 항목에 각기 다른 내용을 저장할 수 있어야 다양한 경로 지정 알고리즘을 구현할 수 있으므로, 최종적으로 착신 방향 2 비트, 가상채널 2 비트, 목적 노드 상대 위치 3 비트에 패킷 상태 부울 변수 1 비트가 추가되어 주소공간 8 비트의 조견표가 소요된다. 조견표의 데이타는 후보 자원의 ID와 패킷 상태 갱신 명령어 및 스위칭 정책 명령어를 포함한다. 각 방향에 3 개의 가상채널이 존재할 때, 호스트를 포함하여 15 개의 자원이 존재하므로 4 비트로 표현이 가능하고, 스위칭 정책과 패킷 상태 갱신을 위한 명령어[5]가 각기 2 비트 소요된다. 표 6에서는 경로 지정 명령어표의 구현 예를 보여준다. 경로 지정 자원은 최대 3 개까지 지정이 가능하며, 스위칭 제어는 웜홀, 패킷, 혼성(Hybrid) 스위칭[9]을 지원하며, 혼성 스위칭 제어용 카운터는 2 비트를 할당하여 총 20비트의 데이타 폭을 갖는다. 경로 지정 명령어 표는 경로 지정

자원을 하나 더 추가할 때마다 6 비트의 데이타 폭이 증가된다.

표 6 경로 지정 명령어표

경로 지정 자원 1		경로 지정 자원 2		경로 지정 자원 3		스위칭	
자원 ID	상태 갱신	자원 ID	상태 갱신	자원 ID	상태 갱신	스위칭 정책	혼성 카운터
4 비트	2 비트	4 비트	2 비트	4 비트	2 비트	2 비트	2 비트

3.2 경로 지정 엔진의 파이프라인 구조

그림 4는 제안된 경로 지정 엔진의 구조를 나타낸다. 각 조견표 및 이를 제어하는 회로의 이용을 극대화하기 위해 각 조견표들은 파이프라인 레지스터(pipeline register)들에 의해 분리되어 있다. 그림 5와 같이 제안된 경로 지정 장치의 경우에는 조견표나 제어 회로의 전달 지연 시간의 균형을 고려해 보면 2 단 또는 4 단의 파이프라인 구조가 바람직하다. 2 단의 파이프라인은 동시에 2 개까지의 패킷 헤더의 경로 지정을 중첩할 수 있고, 4 단의 파이프라인에서는 동시에 4 개까지의 패킷 헤더의 경로 지정을 중첩할 수 있다. 파이프라인의 단계와 최악 지연 시간은 무관하므로 2 단, 4 단의 파이프라인 모두 같은 최악의 지연 시간을 갖는다. 패킷이 플릿 단위로 들어오며, X, Y, X, H가 각각 한 플릿으로 구성되므로 그림 6과 7에서와 같이 4 단까지의 파이프라인은 해저드(hazard)를 일으키지 않고, 따라서 고 비용의 포워딩 회로(forwarding logic)를 필요로 하지 않는다. 그림 4에서 어둡게 표현된 블록이 파이프라인 레지스터를 나타낸다.

그림 7에서와 같이 4 단의 파이프라인으로 경로 지정 엔진은 동시에 들어오는 4 개까지의 패킷에 대해 추가의 지연 없이 경로 지정을 수행하며, 이 경우의 경로 지정 지연 시간은 3/4에서 4/5 플릿 주기 사이로 한정된다. 그러나, 2 차원 그물 구조에서 호스트의 접근을 포함하여 최악의 경우에는 동시에 존재하는 패킷 헤더가 5 개까지 있을 수 있으므로, 연속적인 최악의 경우는 경로 지정 지연 시간을 누적시킨다. 만일 경로 지정 지연 시간이 2 플릿 지연 시간에 도달하면, 2 플릿 깊이의 버퍼가 이를 수용할 수 없게 되므로 흐름 제어를 통하여

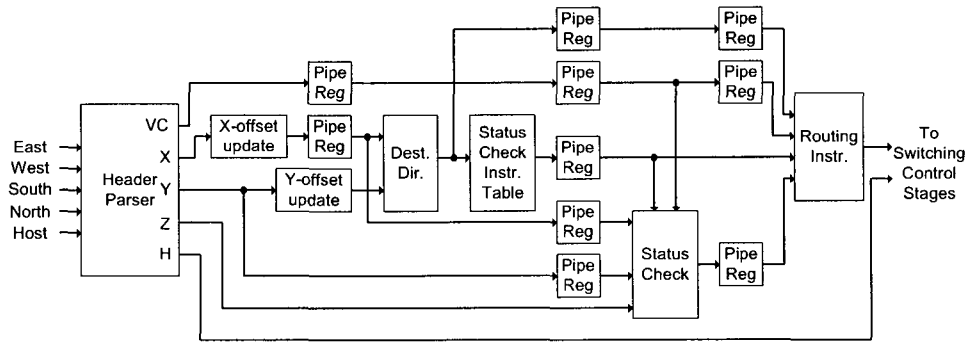

그림 4 조견표를 사용한 경로 지정 엔진의 구조

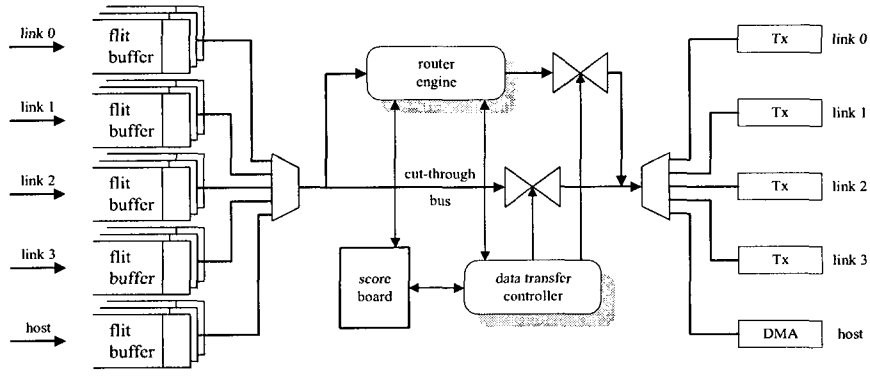

그림 5 시간 공유의 컷 스루우 버스를 갖는 경로 지정 장치의 구조

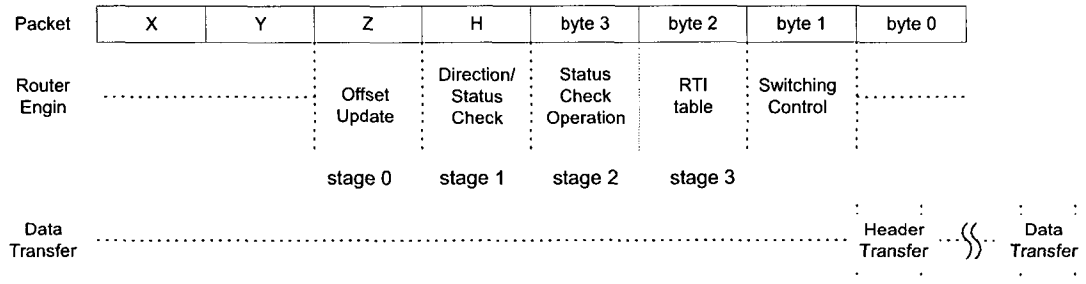

그림 6 파이프라인의 구성 (피트(Phit) 크기의 정수배 = 플릿 크기)

안정된 동작을 얻게되나, 동시에 성능이 감소된다. 이렇게 누적된 경로 지정 지연 시간은 동시에 들어오는 패킷의 헤더의 개수가 줄거나, 패킷 몸체(body)가 수신된 경우에 감소한다. 실제로는 패킷의 헤더만이 연속해서 모든 경로를 통하여 수신되는 확률은 매우 적으므로, 4 단의 파이프라인 구조의 경로 지정 엔진은 2 차원 그물 구조의 연결 망에서 무리 없이 동작하게 된다.

3.3 스위칭 제어 및 주변 장치

스위칭 제어 및 스코어보드와 데이타 전송 스테이트 머신의 설계 및 구현에 대한 내용은 본 논문의 범위를 벗어나나, 경로 지정 엔진의 구조 설명을 위하여 간단히 언급한다. 마이크로프로세서 기반의 경로 지정 엔진에서는 경로 지정 명령어를 토대로 가장 바람직한 자원을 할당받기 위해서 최우선 순위로부터 차례로 각 자원의 사용 가능성을 검사한다[5]. 교환되는 패킷의 양이 그리 많지 않을 때에는 이러한 방법이 크게 성능 저하를 가져오지 않지만, 단일 알고리즘 기반의 하드웨어 경로 지정 엔진과 비교할 만한 지연 속도를 가지려면 이와 같은 순차적인 검사 방법은 바람직하지 않다. 제안된 조견표 경로 지정 엔진은 후보 자원의 사용 가능성을 동시에 검사하고, 사용 가능한 자원의 허가를 우선 순위 인코더로 조합하여 사용 가능한 가장 우선 순위가 높은 자원을 할당받는다. 자원의 할당이 끝난 이후에는 예약된 자원에 따라서 패킷의 상태 값을 가감하거나 세트 또는 리세트를 통하여 적절히 패킷의 상태를 갱신한다. 만일 자원을 할당받지 못한 경우에는 적절한 스위칭 정책에 따른 동작과 스위칭 카운터를 갱신하게 된다[5].

자원의 예약이 끝나면 예약 상태를 스코어보드(score board)에 등록하고, 경로 지정 엔진과 독립적으로 운용되는 데이타 전송 스테이트 머신이 데이타를 전송하게 된다. 전송 사이클은 4 단의 파이프라인을 구성하였을 경우에 2 배의 클록, 즉 $\frac{1}{2}$ 파이프라인 주기에 동기시키면 시분할 버스가 병목이 되지 않게 된다.

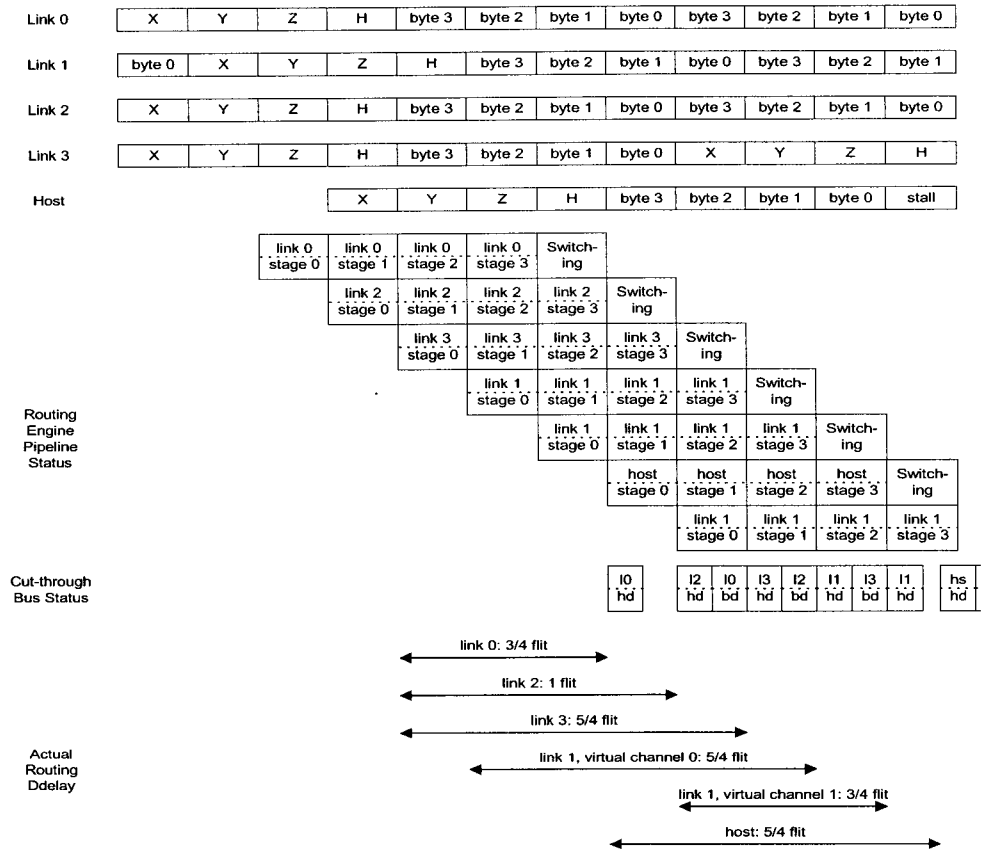

그림 7 다중 경로를 갖는 파이프라인의 상태 (피트(Phit) 크기의 정수배 = 플릿 크기)

4. 구 현

제안된 경로 지정 엔진의 원형은 Xilinx사의 XC4000XL 시리즈 FPGA(field programmable gate array)를 사용하여 구현하였다. 조견표를 위한 메모리 모듈은 XC4000XL에서 CLB(configurable logic block)의 RAM 설정 기능에 의해 구현되었다. 조견표는 이중 포트 RAM으로 구성되어 호스트에서 자유롭게 다운로드 할 수 있다.

제안된 경로 지정 엔진의 제어 회로는 ABEL HDL로 합성하였고, 덧셈기 등 데이타 경로는 FPGA 공급자가 제공하는 데이타 경로 합성기인 LogiBlox를 이용하여 합성하였다. 원형 시스템에서는 구현의 용이성을 위하여 조견표를 구성하는 메모리 셀은 Xilinx 4000XL 시리즈 LCA에서 내부 LUT를 RAM 또는 ROM으로 설정할 수 있는 기능을 사용하여 FPGA에 내장하였다. 결국, FPGA 자원 중에서 가장 많은 비중을 차지하는 것은 조견표를 구성하는 메모리 셀이된다. 그 중에서도 가장 큰 비중을 차지하는 것은 크기가 가장 큰 경로 지정 명령어 표로, 표 7에서 보이듯이 전체 자원의 64% 정도를 차지한다. 표 7에서 조견표 중에서 경로 지정 명령어 표만을 제외한 것이 왼편 열에 나와 있다. XC4010XLPQ160-1의 총 400개의 CLB 중에서 71%를 사용하였다. 사용자 입출력은 88개로 68%를 사용하였다. 이와 비교하여 경로 지정 명령어 표를 내장한 것은 XC4020XLPQ160-1의 총 784개의 CLB 중에서 784개

표 7 구현된 경로 지정 엔진 디바이스 데이타

Index		Without RTI Table		With RTI Table	
		XC4010XLPQ160-1		XC4020XLPQ160-1	
Number of External IOBs	Total	88 out of 129	68%	104 out of 129	80%
	Flip Flops	71		73	
	Latches	0		0	
Number of Global Buffer IOBs	Total	1 out of 8	12%	1 out of 8	12%
	Flip Flops	0		0	
	Latches	0		0	
Number of CLBs	Total	284 out of 400	71%	784 out of 784	100%
	Latches	0 out of 800	0%	0 out of 1568	0%
	CLB Flip Flops	65 out of 800	8%	65 out of 1568	4%
	4 input LUTs	347 out of 800	43%	1339 out of 1568	85%
	3 input LUTs	30 out of 400	7%	204 out of 784	26%
Average Connection Delay		3.872 ns		7.807 ns	
Maximum Pin Delay		23.562 ns		22.519 ns	
Average Connection Delay on the 10 Worst Nets		8.915 ns		17.666 ns	
Average Clock Skew		0.181 ns		0.367 ns	

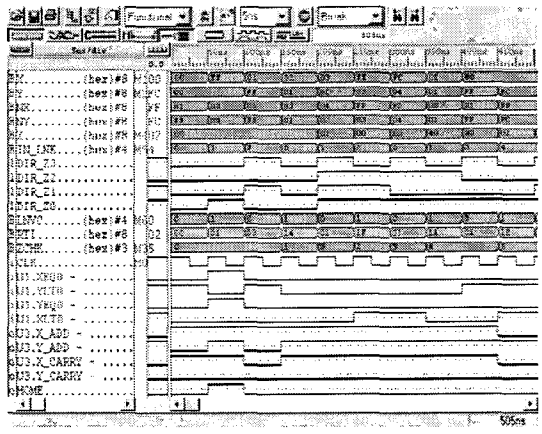

그림 8 경로 지정 엔진의 주요부분 동작

모두 사용하였다[1]. 실제 이 정도의 RAM 사용은 Xilinx 4000XL 시리즈 LCA에서 비용을 고려할 때 바람직한 구성이 아니므로, 원형 시스템이 아닌 실제 구현에서는 FPGA 대신에 비교적 비용이 적게 드는 표준 셀(standard cell) 라이브러리에서 제공하는 RAM을 사용하거나 외부에 별도의 RAM을 두어 비용을 절감하는 방법을 선택할 수 있다. 그림 8에서 구현된 경로 지정 엔진의 주요 부분 동작을 볼 수 있다. X, Y, Z는 입력되는 패킷이며, 결과는 DIR_Z에 나온다. HOME은 현재 노드가 목적지 노드임을 나타낸다. 그림 8에서와 같이 파이프라인 동작을 통하여 매 클럭 마다 새로운 패킷 헤더를 받을 수 있다.

제안된 경로 지정 엔진은 경로 지정이나 스위칭 정책을 수용할 수 있으면 이를 수행하는데 걸리는 지연시간은 1 플릿의 지연으로 일반 하드와이어 경로 지정 엔진과 같은 수준의 성능을 보인다. 사용되는 조견표의 크기는 목적지 노드의 주소를 기억하는 방식이 $O(n)$, (n은 노드 수)인 것인 반면에, $O(m)\times O(\log_2 z)$, (m은 링크 수, z는 패킷상태 검사 종류)이다. 비트 조합을 이용한 것은 비슷한 복잡도를 가지나[6] 유연성 면에서는 제안된 경로 지정 엔진이 계층적 CASE 문과 패킷 상태 검사를 수행할 수 있으므로 더욱 우수하다.

제안된 경로지정 엔진은 그림 7에서와 같이 대개 3/4 내지 4/5 플릿의 지연시간을 가진다. 구현된 결과물에서 RTI 조견표가 없는 경우(표 7), 25MHz에서 안정된 동작을 보였다. ASIC을 이용하여 동작속도를 높일 경우에는 스위칭 정책 및 공유버스 전송에 따른 지연을 포함하여 50MHz 동작시 150ns 정도의 지연을 보이게 되므로 단일 경로지정 엔진[8]과 필적할 만한 지연속도를 가지게된다. 반면 마이크로프로세서를 기반으로 하는 경로지정 엔진은 알고리즘에 따라서 차이가 있으나 0.5 μs

1) 기술매핑 시에 디바이스에 맞추는 옵션을 채택하였다.

내지 1 μs 정도의 지연을 보인다[4, 5].

5. 결 론

본 논문에서는 고성능의 조견표 경로 지정 엔진의 구조와 설계 및 구현에 관한 내용을 소개하였다. 조견표 경로 지정 엔진은 계층화된 CASE 문장으로 기술되는 다양한 스위칭 알고리즘을 가지고 있는 다양한 종류의 경로 지정 및 알고리즘을 구현한다. 다른 수신지 조견표 경로 지정 엔진과는 달리, 제안된 경로 지정 엔진은 수신지 주소만을 가지는 경로 지정 알고리즘을 구현하지는 않는다.

제안된 경로 지정 엔진은 분산된 경로 지정 알고리즘을 구현할 수 있는데, 이 알고리즘은 규칙적(regular) 혹은 유사 규칙적(near-regular) 망의 형태의 오프셋 기반의 주소 기법을 사용하고, 패킷 수신지, 현재의 채널, 현재 노드의 예약 상태 및 패킷의 헤더 부분 상태에 따라 기초적인 경로 지정 결정을 하며, 패킷의 헤더 상태에 따라 부가적인 해석을 추가로 수행하여 다양한 비교 함수의 조합을 실행한다.

제안된 경로 지정 엔진은 마이크로프로세서 기반의 융통성 있는 경로 지정 엔진이 갖는 성능의 한계를 극복할 수 있다. 경로 지정 대기 시간은 대부분의 단일 알고리즘의 하드웨어 경로 지정 엔진과 거의 동등한 반면, 하드웨어적으로 효율적인 많은 체계를 지원한다.

제안된 경로 지정 엔진은 4 단 파이프라인 구조를 채택하여, 동시에 여러 개의 헤더를 효율적으로 처리할 수 있다. 이러한 구조는 하드웨어 자원을 절감할 수 있고, 최악의 경로 지정 대기 시간을 줄일 수 있는 조정기(arbiter)를 사용함으로써 스위칭 제어기에 보다 많은 자원이 사용될 수 있게 해 준다. 제안된 경로 지정 엔진은 Xilnx 4000XL 시리즈의 FPGA를 사용하여 구현하였다.

참 고 문 헌

[1] M. L. Fulgham and L. Snyder, "Integrated multi-class routing," in *Proceedings of Parallel Computer Routing and Communication Workshop*, pp. 17 - 28, June 26-27, 1997.

[2] J. Rexford, J. Hall, and K. G. Shin, "A router architecture for real-time communication in multi-computer networks," *IEEE Transactions on Computers*, pp. 1088-1101, October 1998,

[3] Wu-chang, Feng and Kang G. Shin, "Impact of selection functions on routing algorithm performance in multicomputer networks," In *Proceedings of 11th Annual Conference on Supercomputing*, pp. 132 - 139, July, 1997.

[4] J. Dolter, S. Daniel, A. Mehra, J. Rexford, W. Feng, and K. Shin, "SPIDER: Flexible and Efficient Communication Support for Point-to- point Distributed Systems," In *Proceedings of Int. Conference on Distributed Computing Systems*, pp. 574-580, June 1994.

[5] S. W. Daniel, "Flexible router architecture for point-to-point networks," Ph.D Thesis. EECS div., University of Michigan, 1996.

[6] Douglas H. Summerville, Jose G. Delgado-Frias, and Stamatis Vassiliadis, "A flexible bit-pattern associative router for interconnection networks," *IEEE Transactions on Parallel and Distributed Systems*, vol 7, no 8, pp. 477 - 485, May, 1996.

[7] K. G. Shin, and S. W. Daniel, "Analysis and Implementation of Hybrid Switching," *IEEE Transactions on Computers*, vol. 45, no. 6, pp.684-692, June, 1996.

[8] Kevin Bolding, Sen-Ching Cheung, Sung-Eun Choi, Carl Ebeling, Soha Hassoun, Ton Anh and Robert Wille, "The Chaos Router Chip: Design and Implementation of Adaptive Router," In *Proceedings of VLSI '93, IFIP*, pp. 311 - 320, 1993.

[9] J. Duato, S. Yalamanchili, and L. Ni, *Interconnection networks: An engineering approach*, IEEE Computer Society, September, 1997.

장 래 혁

1989년 2월 서울대제어계측공학과 졸업. 1992년 2월 동대학원 석사. 1996년 2월 동대학원 박사. 1997년 1월 미시간대학교 Research Fellow. 1997년 10월 ~ 현재 서울대학교 컴퓨터공학과 전임강사. 관심분야는 내장형 시스템, 저전력 시스템, 실시간 시스템, 디지털 시스템 설계 및 구현