실시간 상황 인식 시스템을 위한 RETE 네트워크 하드웨어 가속기의 구조

Architecture of RETE Network Hardware Accelerator for Real-Time Context-Aware System

[1]이승욱, [1]김종태, [2]이건명, [1]이지형, [1]전재욱
[1]성균관대학교 정보통신공학부, [2]충북대학교 전기전자컴퓨터공학부

[1]Seung Wook Lee, [1]Jong Tae Kim, [2]Keon Myung Lee, [1]Jee Hyung Lee, [1]Jae Wook Jeon
[1]School of Information and Communication Engineering
Sungkyunkwan University, Korea
[2]School of Electrical and Computer Engineering
Chungbuk National University, Korea
E-mail : swlee@ece.skku.ac.kr

요 약

지능 홈-케어 시스템 또는 외부 통신 채널의 환경 인식이 가능한 모바일 통신기기와 같은 상황 인식 시스템이 외부 상태를 감지하여 현재 상황을 인식하고 대처하기 위해서는 수 백개 이상의 규칙들을 이용한 추론을 필요로 한다. 이들 규칙들의 효과적인 추론을 위해서는 룰-베이스 시스템에 기반을 둔 추론 기법을 적용 시킬 수 있다. 이 룰-베이스 시스템의 추론 규칙의 매칭을 위해서 RETE 알고리즘이 사용되어 왔다. 하지만 RETE 알고리즘은 그 특성상 Von Neumann 구조의 컴퓨터 시스템에서는 규칙의 증가에 따른 그 성능의 저하가 필연적이다. 본 논문에서는 RETE 네트워크를 이용한 추론을 효과적으로 수행할 수 있는 RETE 네트워크 하드웨어 가속기의 구조에 대해서 논한다. 이 RETE 네트워크 하드웨어 가속기은 Von Neumann의 구조적 제약점을 병렬처리 구조를 사용하여 제거하였다.

1. 서 론

상황 인식 시스템은 외부 센서의 입력 데이터와 같은 시스템의 상태를 나타내는 정보 즉 상황(context) 정보를 이용하여 적절한 서비스를 제공하는 시스템으로 정의할 수 있다[1]. 이 상황 인식 시스템이 주어진 정보를 이용하여 제공할 서비스를 선택 및 시스템 내부의 정보의 증강을 요하는 작업을 수행하기 위해서는 미리 저장된 규칙들의 조합 및 추론을 이용하게 된다. 여기서 추론을 요하는 동작은 룰-베이스 시스템의 그것과 동일한 연산을 요구한다. 즉 주어진 규칙들과 입력된 정보를 이용하여 적절한 추론을 통하여 그 결과의 생성 및 새로운 지식의 추가 등의 동작이 룰-베이스 시스템의 그것과 동일하다. 룰-베이스 시스템의 효율적인 추론을 위하여 Forgy에 의해 제안된 RETE 네트워크 알고리즘이 대표적이다[2]. 이는 시스템에서 사용될 규칙들을 네트워크 형태로 연관지어 중복되는 연산을 제거함으로 그 수행 속도의 향상을 목적으로 한다. 하지만 이 RETE 알고리즘의 순차적인 수행 방식은 Von Neumann 구조의 컴퓨터 시스템에서 소프트웨어로 구현될 시에 본질적인 성능의 저하를 초래한다. 이를 극복하기 위한 방법으로 RETE 알고리즘을 병렬 프로세서 환경을 기반으로 그 구현이 시도되기도 하였다[3]. 하지만 병렬 프로세서의 사용은 시스템의 비용을 증가 시키는 문제가 발생한다. 즉, 배터리를 사용하는 소형의

모바일 기기들에서는 그 구현이 본질적으로 비효율적이라 할 수 있다. 이에 본 논문에서는 RETE 알고리즘의 실시간 수행을 위한 RETE 네트워크 하드웨어 가속기 (RETE network hardware accelerator : RETENHA)의 구조를 제안한다. RETENHA는 범용 프로세서와의 연동을 통해 RETE 알고리즘을 병렬적으로 수행 가능하다. 이를 위해서는 주어진 규칙의 RETE 네트워크를 하드웨어에서 인식 가능한 형태의 형식으로 변환시켜야 한다. 이 변환 과정은 PC 환경에서 자동 RETE 네트워크 변환 도구를 이용해 가능하다. RETENHA는 연동되는 프로세서에서의 RETE a알고리즘의 수행 부담을 제거하여 전체 시스템의 성능을 향상 시킬 수 있다.

2. RETE 네트워크 하드웨어 가속기의 구조

RETENHA는 범용 규칙 처리를 위한 하드웨어 가속기로서 처리되는 규칙의 규모 및 표현 구조의 제약을 최소화 하도록 설계되었다. 즉, 기존의 하드웨어 규칙 처리기가 가지는 규칙 표현의 제약을 제거함으로서 소프트웨어 기반에서 개발되는 시스템과 동일한 시스템 융통성을 제공할 수 있다. 그림 1은 RETENHA와 응용 시스템의 연동 구조의 개념을 나타낸다.

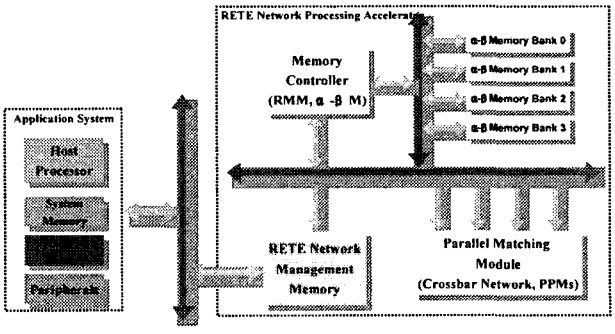

그림 1 RETE 네트워크 하드웨어 가속기의 구성

그림 1에서 RETENHA는 응용 시스템으로부터 추론에 이용될 데이터를 제공 받고, RETENHA에서의 RETE 알고리즘을 통한 추론을 통해 생성된 결과를 응용 시스템으로 전송한다. 만일 응용 시스템이 실시간으로 외부 상황에 따라 반응 동작을 수행해야하는 지능 로봇 시스템이라면, 로봇의 주제어 시스템은 비젼 및 음성, 기타 외부 센서를 이용하여 획득된 데이터를 추론에 적용할 수 있는 형태의 데이터 형식으로 변환 시킨 후 이 데이터를 시스템 버스를 통해 RETENHA로 전송한다. RETENHA에서는 실시간 병렬 추론을 통해 적절한 결과를 생성 시킨 후 이를 로봇의 주제어 시스템으로 전송하면, 주제어 시스템은 전송받은 데이터를 반응 동작에 이용할 수 있는 데이터로 처리하여 그 동작을 수행 하게 된다. 보다 자세한 과정은 다음의 절들에서 논한다.

2.1 RETE 네트워크

그림 2는 전형적인 RETE 네트워크의 형태를 나타낸다.

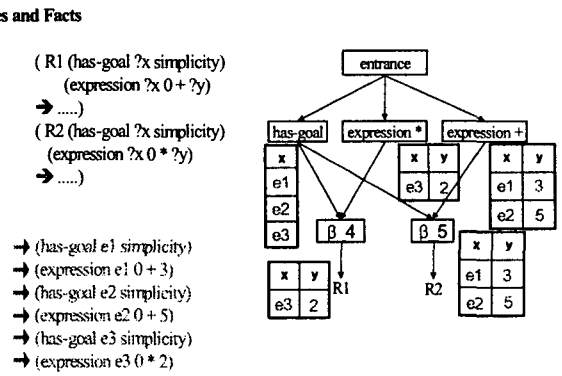

그림 2 RETE 네트워크의 예

그림 2에서 추론되는 규칙은 R1과 R2로 각 규칙에서 공통되는 조건을 공유하며, 각 α-메모리와 β-메모리를 가지고 있다. 여기서 α-메모리는 입력되는 데이터들 중에서 이미 저장된 정보를 제외한 정보만을 가지고 있으면서 다른 노드의 α-메모리와 매칭 연산을 통해서 β-메모리의 데이터를 구성하게 된다. 이 매칭 연산이 전체 RETE 알고리즘 연산량에서 80%~90%을 차지하게 된다. 즉 이와 같은 연산을 Von Neumann 구조의 컴퓨터 시스템에서 수행 할 경우에 구조적으로 그 연산이 비효율적일 수밖에 없다. Von Neumann 구조에서는 각 α-메모리에 새로운 데이터가 입력되는 경우 이미 저장된 데이터들을 순차적으로 레지스터에 저장시킨 후 비교 연산을 통해 새로운 데이터가 기존의 데이터와 중복되지 않음을 확인한 후 매칭 연산을 수행할 다른 α-메모리의 데이터들과 순차적으로 바인딩 연산을 수행한다. 즉 이모든 과정에서 메모리와 레지스터간의 중복적인 데이터의 load/store 과정이 필요하며, 이 과정에서 발생하는 전송 오버로드는 전체 시스템의 성능을 좌우하게 된다. 즉 추론을 위한 규칙의 증가는 급속한 시스템의 성능의 저하를 발생시킨다.

2.2 RETE 네트워크 관리 메모리

RETENHA에서는 Von Neumann 구조의 시스템에서의 데이터의 중복적인 전송을 제거하고 병렬 RETE 알고리즘 연산을 수행하기위하여 RETE 네트워크를 RETENHA에서 인식 가능한 형태로 변형해야한다. 그림 3은 2.1절의 그림 2와 같은 RETE 네트워크를 RETENHA 내에서 RETE 네트워크 관리 메모리의 구성 내용을 나타낸다.

속성		노드말단 원소의 주소	매칭대상 노드	노드간 연산	결과저장 노드	바인딩할 변수의 개수
1	has-goal	-	2	=	4	1
2	has-goal	-	3	=	5	1
3	expression *	-	1	=	4	1
4	expression +	-	1	=	5	1
5	β_4	-	no	=	terminal	0
6	β_5	-	no	=	terminal	0

그림 3 RETE 네트워크 관리 메모리

그림 3과 같이 각 α-노드와 β-노드는 고유의 노드 번호가 부여되며, 각 노드간의 연결 관계와 매칭이 필요한 변수의 개수 및 매칭시 수행되는 연산의 종류를 나타내었다. 또한 그림 1과 같이 α-β 메모리에서 각 노드 내의 원소의 위치를 지정하기위한 각 노드의 마지막 원소의 주소를 저장한다. 이는 α-β 메모리에 각 노드의 데이터 및 각 노드의 이전 데이터의 주소를 함께 저장함으로서 말단 원소의 주소를 가지고 있으면 이전에 저장된 모든 원소의 위치를 얻을 수 있다.

2.3 병렬 매칭 연산 모듈

RETENHA는 RETE 네트워크의 노드간 매칭 연산을 하드웨어를 통해 병렬적으로 연산시킬 수 있다. 그림 4는 RETENHA 내의 병렬 매칭 연산 모듈의 구조이다. 그림의 PPM은 병렬처리 모듈로 노드간에 수행되는 산술 및 논리 연산을 수행하는 모듈로서 복수개의 PPM이 병렬적으로 그 연산을 수행한다. 변수 버퍼로 새롭게 입력되는 노드의 데이터와 매칭이 수행될 노드의 데이터가 입력되면 스위칭 제어기는 RETE 네트워크 관리 메모리에 저장된 데이터를 기반으로 매칭이 수행되는 변수들의 지정과 매칭 시 필요한 연산의 종류를 PPM을 통해 설정하게 된다. 이 일련의 설정 과정은 crossbar switching network의 스위칭 동작을 통해 각 데이터들을 위한 능동적인 데이터 전송 경로의 생성을 통해 이루어진다. PPM의 연산 결과에 따라 매칭 판별기는 데이터의 매칭 여부를 판별하여 새로이 생성된 데이터를 RETE 네트워크 관리 메모리로 전송하게 된다.

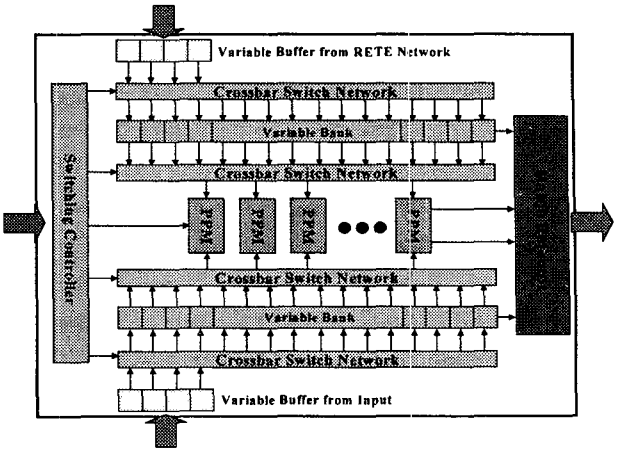

그림 4 병렬 매칭 모듈

3. 시스템 개발 과정

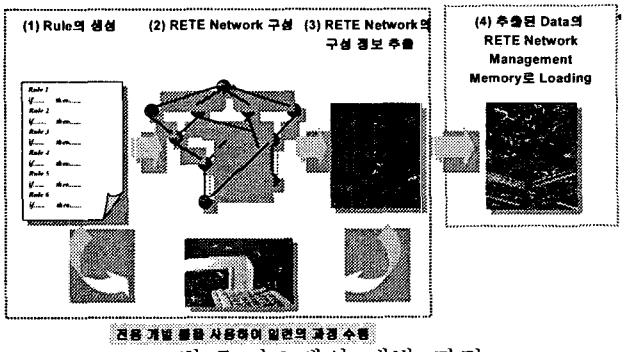

그림 5 시스템의 개발 과정

RETENHA를 사용하여 응용 시스템에서 이용하기 위해서는 RETENHA의 설정 과정이 필요하다. 우선 응용 시스템에서 사용되는 규칙을 생성해야한다. 생성되는 규칙은 일반적인 *if~then~* 형태의 규칙이다. 이를 전용의 컴파일러 도구를 사용하여 RETE 네트워크를 구성하고 RETE 네트워크 관리 메모리를 구성하는 데이터를 추출해야 한다. 이 컴파일러 도구는 또한 추출된 데이터를 RETENHA로 전송 가능한 데이터 형태로 변환하여 RETENHA를 설정한다. 그림 6은 컴파일러 도구의 인터페이스를 보여준다. 즉 규칙과 규칙의 실행을 위한 전처리부와 실행부를 기술하여 RETENHA에서 인식 가능한 형태의 데이터로 변환한 후 PC의 시리얼 링크를 통해 RETENHA를 설정하게 된다. 이 일련의 과정은 별도의 PC 환경에서 수행된다. 이후 설정이 끝난 RETENHA는 실제 응용 시스템에 장착되어 그 기능을 수행하게 된다.

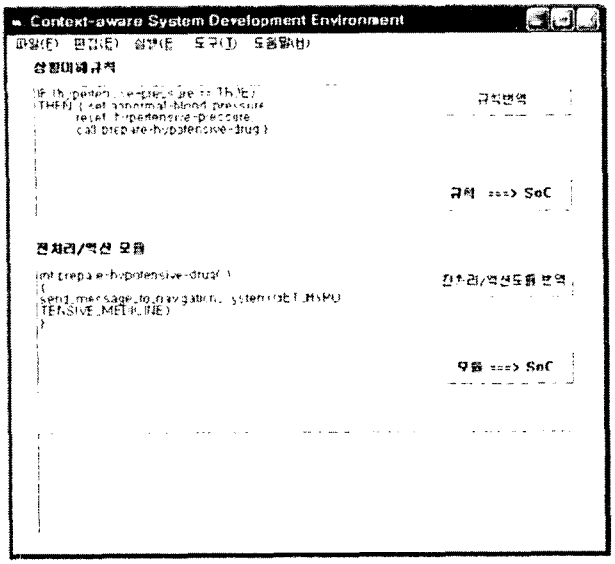

그림 6 시스템 개발 환경

4. 성 능 평 가

RETENHA는 시스템 설계 및 검증 언어인 SystemC ver 2.0을 사용하여 구현되고 그 성능이 측정되었다. 성능의 비교를 위하여 [4]과 같은 실시간 주변 상황 인식용 56개의 규칙을 생성하여 JESS(JAVA expert system shell), 최적화된 C 코드와 그 수행 속도를 비교하였다. 그 결과는 표 1과 같다.

추론환경		CPU 점유율	수행시간
Pentium-4 2.4GHz	JESS[1] (V-Tune[2] 사용)	7% (OS + JESS tool)	평균 1초내외
		100% (추정)	평균 70 *msec*
	Optimized C-code (Visual C++ profiling tool[3] 사용)	100%	평균 3.2 *msec*
RETE Network Processing Accelerator	SystemC[4] ver. 2.0 simulator 사용	-	System frequency 10MHz 평균 98-clock 소요 평균 9.8 *usec*

표 1 성능평가

성능평가를 위한 실험 조건은 Pentium-4 2.4GHz의 시스템 하에서 JESS와 C 코드로 구현된 56개의 규칙을 추론 하였으며, 수행시간의 측정은 하나의 입력에 의해 하나의 결과를 추론하기까지의 수행시간이 측정되었다. 참고로 하나의 결과를 추론하는데 평균적으로 7개의 규칙의 매칭 연산이 수행 되었다. JESS를 통한 수행시간의 측정은 Intel CPU의 프로그램 수행 성능 측정 도구인 V-Tune을 사용하였으며, 최적화된 C 코드는 Visual C++ 플랫폼에 내장된 수행 코드 분석기능 통해 측정 되었다. 또한 RETENHA의 수행 시간은 SystemC의 검증 기능을 사용하여 10MHz의 시스템 동작 주파수를 인가하였을 경우의 수행 시간이다. 표1에서와 같이 RETENHA는 C 코드 보다 약 330배 정도의 연산 속도의 증가가 됨이 측정되었다. 이는 C 코드가 2.4GHz의 고성능 CPU에서 수행되었음을 감안할 때 10MHz에서 동작하는 RETENHA의 그 성능은 소프트웨어로 구현된 시스템에서는 얻을 수 없는 성능이라 할 수 있다.

5. 결 론

본 논문에서는 실시간 상황 인식 시스템에 이용 가능한 RETE 알고리즘을 하드웨어적으로 수행 가능한 RETE 네트워크 하드웨어 가속기의 구조에 대해서 논하였다. 구현된 RETENHA는 PC에서 자동화 도구을 사용하여 생성된 규칙을 RETE 네트워크로 변환시켜 RETENHA를 설정하여 여러 응용 시스템에 범용적으로 사용될 수 있다. 또한 그 성능은 기존의 Von Neumann 구조의 시스템에 비해 매우 탁월하다.

감사의 글: 본 논문은 21C 프론티어 "인간기능 생활지원 지능로봇 기술개발" 과제와 반도체 설계 교육 센터(IDEC)의 지원으로 연구되었으며 이에 감사드립니다.

5. 참 고 문 헌

[1]G.D. Abowd, A.K. Dey "Towards a Better Understanding of Context and Context-Awareness" Proc. 1st Int'l Symp. Handheld and Ubiquitous Computing (HUC 99), Lecture Notes in Computer Science, no 1707 Springer-Verlag, Germany, (1999) 304-307

[2]C.L. Forgy, "RETE : A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem" , Artificial Intelligence (1982) vol. 19 17-37.

[3]Dou, C. "A highly-parallel match archi-tecture for AI production systems using application-specific associative matching processors",, Application-Specific Array Proce-ssors, Proceeding, International Conference (1993) 180 183

[4]이승욱, 김종태, 이건명 "실시간 상황 추론을 위한 하드웨어 룰-베이스 시스템의 구조", KFIS 2004, 17-21