실시간 상황 인식을 위한 하드웨어 룰-베이스 시스템의 구조

Real-Time Rule-Based System Architecture for Context-Aware Computing

[1]이승욱, [1]김종태, [2]손봉기, [2]이건명, [1]조준동, [1]이지형, [1]전재욱
[1]성균관대학교 정보통신공학부, [2]충북대학교 전기전자컴퓨터 공학부

[1]Seung Wook Lee, [1]Jong Tae Kim, [2]Bong Ki Sohn, [2]Keon Myung Lee,
[1]Jun Dong Cho, [1]Jee Hyung Lee, [1]Jae Wook Jeon
[1]School of Information and Communication Engineering
Sunkyunkwan University, Korea
[2]School of Electrical and Computer Engineering
Chungbuk National University, Korrea
E-mail : jtkim@yurim.skku.ac.kr

요 약

본 논문에서는 실시간으로 상수 및 변수의 병렬 매칭이 가능한 새로운 구조의 하드웨어 기반 룰-베이스 시스템 구조를 제안한다. 이 시스템은 context-aware computing 시스템에서 상황 인식을 위한 기법으로 적용될 수 있다. 제안된 구조는 기존의 하드웨어 기반의 구조가 가지는 룰의 표현 및 룰의 구성에서 발생하는 제약을 상당히 감소시킬 수 있다. 이를 위해 변형된 형태의 content adressable memory(CAM)와 crossbar switch network(CSN)가 사용되었다. 변형된 형태의 CAM으로 구성된 지식-베이스는 동적으로 데이터의 추가 및 삭제가 가능하다. 또한 CSN은 input buffer와 working memory(WM) 사이에 위치하여, 시스템 외부 및 내부에서 동적으로 생성되거나, 시스템 설정에 의해 지정된 데이터들의 조합 및 pre-processing module(PPM)을 이용한 연산을 통하여 WM을 구성하는 데이터를 생성시킨다. 이 하드웨어 룰-베이스 시스템은 SystemC 2.0을 이용하여 설계하였으며 시뮬레이션을 통하여 그 동작을 검증하였다.

1. 서론

Context-aware computing system은 주변 환경을 여러 센서를 통해 감지하여 context를 파악하고, 그 파악된 정보에 의해 적절한 서비스를 제공하는 시스템으로 정의 할 수 있다[1]. 이를 위해서는 적절한 센서 관리 및 상황 추론을 위한 효율적인 기법이 사용되어야 한다. 센서 관리는 센서의 종류 및 시스템이 적용되는 환경에 따라 적절한 기법이 사용 되어야 한다. 하지만 상황 추론은 센서의 종류 및 적용 환경에 따라 변하지 않은 기법이 적용 된다면, 여러 시스템에 동일한 상황 추론 구조를 적용 할 수 있을 것이다. 상황을 추론하기 위하여 시스템 내에 지식-베이스를 구축하여 입력되는 정보와의 비교를 통해 현재 상황을 추론하는 기법이 있다. 이는 과거 여러 연구에 의해 그 능력이 입증된 룰-베이스 시스템의 기능과 동일하다. 기존의 소프트웨어 알고리즘에 기초한 룰-베이스 시스템의 처리 속도는 실시간 처리가 필요한 응용 분야에는 적합하지 않다. 즉 context-aware computing system과 같은 실시간 처리가 필요한 분야는 소프트웨어 알고리즘에 기초한 룰-베이스 시스템과 같은 추론 기법은 적당하지 않다. 과거 몇몇의 연구에서 이와 같은 처리 속도의 문제점을 해결하기 위하여 룰-베이스 시스템을 위한 하드웨어적 접근을 시도했다. 여러 연구에서 하드웨어적 접근을 위하여 CAM의 적용은 한 가지 방법이 되어왔다. 하지만 기존의 구조에서 CAM의 적용은 RETE 네트워크를 하드웨어적으로 묘사하기 위한 방법의 일환으로 사용되어 완전히 소프트웨어에 독립적으로 동작하는 구조는 아니다[2-4]. 이로 인해 적용되는 룰-베이스 시스템을 소프트웨어로 우선 모델링 한 후 여기에서 필요한 시스템 구성을 위한 하드웨어의 크기를 결정 해야만 했다. 즉 적용되는 룰-베이스 환경이 변화하면 그에 대하여 하드웨어를 다시 설정해야 하는 단점을 가지고 있었다. 하지만 제안된 구조는 룰-베이스 시스템이 적용되는 환경에 구애

받지 않고 동일한 독립적인 하드웨어 구조로 적용이 가능 하다. 또한 상수 및 변수의 병렬 매칭이 가능하고 이전의 구조에 비하여 룰의 표현 및 룰을 구성하는 조건문 조합의 제약이 상당히 감소되었다. 이를 위해 된 변형된 CAM와 CSN을 적용하였다. 변형된 CAM는 시스템의 지식-베이스를 구성하는데 이때 지식-베이스의 내용의 동적인 추가 및 삭제가 가능하다. 또한 CSN는 input buffer와 WM사이에 위치하여 주어진 데이터를 가지고 룰을 구성하는 조건문의 생성에 제약을 감소시켰다.

2. 실시간 룰-베이스 시스템의 구조

2.1 제안된 구조에서의 룰의 표현 기법

이번 장에서는 시스템에 적용시키기 위한 룰의 구성 및 매칭 기법에 대해 다룬다. 예제로 사용되는 룰-베이스 시스템은 일종의 지능형 서비스 제공 로봇의 추론을 위한 시스템으로서 주변 상황을 로봇에 부착된 여러 센서를 통하여 수집한 후 현재 상황을 이해하여 적절한 서비스를 사용자에게 제공하는 시스템 이다. 이를 위해 로봇에 적용할 56개의 룰을 만들었다. 이 룰은 로봇 스스로 사용자의 건강상태, 실내 상태, 비상 상태 대처가 가능하게 하는 룰로서 우선 JESS를 통해 룰의 동작이 검증 되었다.

```
Rule 1
        IF   <OK-response>
        THEN reset abnormal-pulse-rate flag
             reset abnormal-temperature flag
             reset OK-response flag
Rule 2
        IF   <abnormal-pulse-rate> AND <abnormal-temperature>
        THEN set abnormal-state flag
             ask the master "How do you feel?"
             set event-time-1
Rule 3
        IF   <blood-pressure ≥ upper-normal-pressure>
        THEN set abnormal-blood-pressure flag
             prepare hypertensive drug
Rule 4
        IF   <wait-response> AND <clock-time - event-time-1 ≥ specified-time-1>
        THEN set abnormal-state flag
             reset wait-response flag
```

그림 1 적용된 룰의 예제

시스템에 적용될 수 있는 룰은 위의 그림 1과 같이 1) 단일 attribute로 표현 될 수 있는 룰, 2) 복수의 attribute가 AND 관계를 가지고 있는 룰, 3) 단일 조건문으로 이루어진 룰, 4) attribute와 조건문간의 AND 연산을 통해 이루어지는 룰로 구성 가능하다. 기존에 연구된 하드웨어 구조에서는 룰 및 WM를 구성하기 위하여 필요한 데이터들의 연산을 별도의 프로세서를 통해서 수행 하게 되어있다. 하지만 설계된 시스템 구조에서는 input buffer가 센서 및 시스템 설정에서 주어지는 모든 정보를 우선 저장한 후 별도의 전용 연산 PPM을 사용 하여 실제 룰 및 WM에서 필요한 데이터로 가공한다. 즉 input buffer의 정보는 룰-베이스 및 WM에서 사용되기 위한 attribute나 condition term들의 연산을 위한 미가공 정보이다. 만일 룰-베이스를 구성하는 메모리가 5개의 contents로 이루어져 있고, 각 contents의 위치는 고유의 속성을 가질 수 있다면, WM도 5개의 content로 구성되며 각 위치의 속성은 또한 룰-베이스의 형태와 같다. 이렇게 되면 다음에 설명 될 변형된 CAM의 연산에 의해서 룰의 병렬 매칭이 가능한 모습의 룰-베이스를 구성할 수 있다. 이와 같이 룰-베이스와 WM를 구성하기 위해서는 input buffer에 들어있는 정보들의 가공이 필요하다. 이는 CSN와 PPM을 통해 이루어진다. 이렇게 룰-베이스를 구성함으로써 이론적으로 룰-베이스의 모든 룰과 WM와의 비교를 하나의 연산 싸이클 내에서 수행이 가능하다.

2.2 시스템의 구조

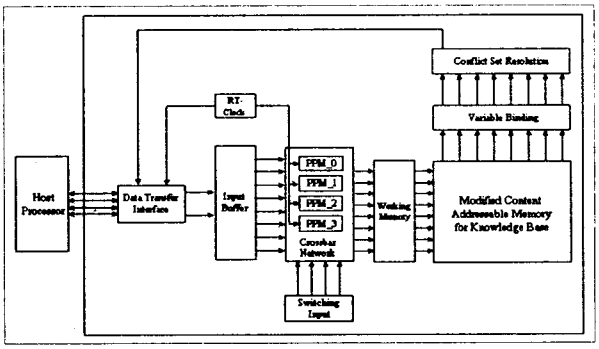

그림 2 제안된 룰-베이스 시스템의 구조

실시간 룰-베이스 시스템의 구조는 그림 2와 같다. 이 시스템은 input buffer, CSN, WM, variable binding block, conflict resolution block와 호스트 프로세서와는 별도의 RT-clock으로 구성 되어있다. 또한 이 시스템 구조는 단독으로 사용 될 수도 있고, 적용해야 할 룰의 개수가 증가 될 시에는 여러 개를 동시에 사용하여, 블럭 간의 병렬 구조로도 사용이 가능하다.

2.3 WM와 룰의 LHS 인코딩

효율적인 상수 및 변수 매칭을 위하여 WM와 룰의 LHS에 특별한 인코딩을 필요로 한다. 만일 WM의 각 contents가 m-bits고 그 것의 개수가 n개라면, 그 모

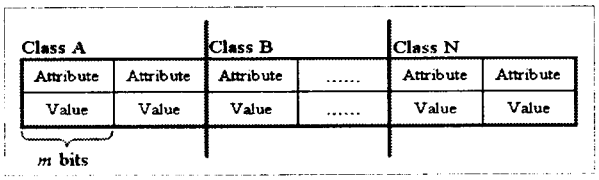

그림 3 WM의 인코딩

습은 그림 3와 같다. 그림 4은 룰의 LHS에 변수를 가지고 있는 형태로 그림과 같은 포맷으로 변수 여부, 변수의 종류, 변수의 개수를 표현 할 수 있다. 이전 절에서 논의 했듯이 WM contents로 표현 가능한 것은 objects class의 attribute 또는 룰에서 표현 가능한 condition term의 연산 결과를 표현 할 수 있다. 이렇게 함으로서 병렬 매칭을 위해 WM 부분과 이후에 설명될 변형된 CAM사이에 별도의 하드웨어 로직이 없어도 된다.

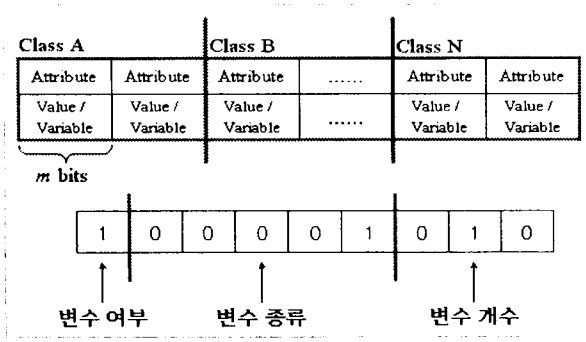

그림 4 변수를 포함한 룰의 LHS 인코딩

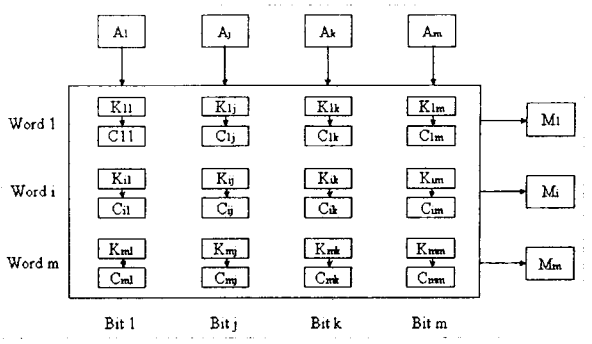

그림 5 제안된 CAM의 구조

2.4 Crossbar Switch Network

본 논문에서는 CSN를 사용하여, input buffer로부터 WM의 contents로 사용될 데이터의 전송 및 condition term의 조합을 통한 연산 결과를 구성할 수 있도록 하였다. 이는 이 룰-베이스 시스템의 적용 환경이 다를 경우 CSN의 스위칭 설정의 변경을 통하여 자유롭게 데이터의 전송 및 다양한 condition term의 구성이 가능하다. 또한 구조적 특성상 multi point switching이 가능 하므로 input buffer로부터의 데이터 패치 시 스케쥴링에 제약이 없게 할 수 있다. 또한 CSN 상에 존재하는 PPMs은 condition term의 연산 결과를 산출할 수 있는 일종의 ALU로서 별도의 호스트 프로세서의 작동 없이도 연산이 가능하므로 전체 룰-베이스 시스템의 연산이 병렬 프로세서 환경에서 작동하는 것과 같은 효과를 나타낼 수 있다. 이 PPMs의 연산의 종류 결정을 위하여 CSN의 스위칭 시 동시에 연산의 종류를 선택하는 input signal이 입력된다. 이 CSN와 PPMs의 연산 수행 시간은 PPMs의 개수와 계산해야 할 condition term의 개수에 의존한다. 별도의 CSN 및 PPMs의 설정 과정이 필요 없기 때문에 다음과 같이 연산 싸이클이 계산 가능 하다.

$$Cycles = \frac{\lceil \ Nmber\ of\ Condition\ terms\ \rceil}{Nmber\ of\ Pre-Processing\ Modules}$$

2.5 변형된 CAM의 구조

룰-베이스 시스템의 지식-베이스를 구성 할 때 사용되는 메모리 디바이스로서 CAM의 사용은 룰의 매칭 과정에서 속도의 향상을 제공 할 수 있다. 일반적인 CAM의 경우 key register를 두어 입력의 일정 부분을 masking을 할 수 있게 한다[6]. 이 masking 기능은 입력으로 주어지는 모든 bit를 저장된 메모리 content와 비교 하지 않고, masking된 bit만을 비교하는 기능이다. 즉 입력을 주어진 pattern중 일정 부분만 일치해도 output의 매치 bit를 set 한다. 하지만 본 논문에서 구성한 CAM는 masking의 기능이 개념상 일반적인 경우와 다르다. 그 이유는 WM content에는 표현된 attribute 및 condition term들의 연산의 결과들이 있을 것이다. 하지만 룰은 이들 attribute 및 condition term을 모두 가지고 구성되는 것이 아니라, 일부만을 사용하여 여러 룰을 만든다. 여러 룰들은 CAM으로 구성된 지식-베이스에 저장되고, 이 CAM의 입력으로 들어오는 WM의 content들 중에서 자신의 매칭 여부를 알 수 있는 일부 attribute 및 condition term들만을 비교하는 것이 필요하다. 이를 위해 그림 5와 같이 일반적인 CAM 에서 masking register의 기능이 확대된 구조로 masking register의 기능을 하는 로직을 각 memory cell 에 삽입한 구조를 제안한다. 그림 5의 CAM는 합성 가능한 SystemC 2.0 코드로 작성되어 그 기능 및 합성 후의 구조를 확인 가능하였다.

2.6 하드웨어 구조 기반의 변수 바인딩 기법

룰-베이스 시스템을 실시간 환경에 적용하기 어려운 이유는 실행 시간이 느리기 때문이었다. 이는 전체 실행 시간의 90% 정도를 차지하는 matching 연산이 그 주된 원인인데, (90%중 가장 많은 부분을 변수 바인딩을 위한 연산에 사용된다[7].) 이에 본 논문에서는 matching 연산 중 상당 부분을 차지하는 변수 바인딩 연산을 위한 하드웨어 구조 기반의 기법를 제안한다. 기존의 RETE와 같은 알고리즘이 효과적으로 변수 바인딩을 위한 기법을 제공 했으나, 변수 바인딩의 순서에 따라 연산량이 비효율적으로 증가 될 수 있는 단점이 있었다[2]. 하지만 이를 하드웨어 구조로 제안함으로서 변수 바인딩의 순서에 구애 받지 않고 연산량 또한 시스템이 설정될 때 고정적으로 결정 될 수 있다. 그림 6에서, 만일 WM와 지식-베이스에 저장된 룰의 모습이 그림과 같다면, 룰에서 X0, X1은 그림 4와 같은 형식에 의해 표현된 변수이다. 제안된 구조에서 변수 바인딩을 하는 기법은 우선 표현 할 수 있는 각 변수마다 룰 내에서 그 변수가 최대로 표현 될 수 있는 개수의 CAM를 가지고 있는 것이다. 그러면 2.3절의 룰의 형식에 의하여 룰 내의 변수는 자신의 CAM영역에 고유의 위치를 가리 킬 수 있다. 그러면 variable binding block은 룰에서 변수가 나타난 위치의 WM의 content를 룰 내의 변수가 지시하는 고유의 영역에 복사 한 후, 각 변수의 CAM에 동일한 content가 들어있는 지를 확인 하면 된다. 만일 그림 6과 같은 경우라면, 변수 X의 X0, X1의 content가 서로 다르므로 그림 6에 표현된 룰은 변수 바인딩을 실패하게 된다. 따라서 최종적으로 conflict set의 항목으로 추가 될 수 없게 된다. 그림 7은 변수를 가지고 있는 룰이 매치된 경우이다

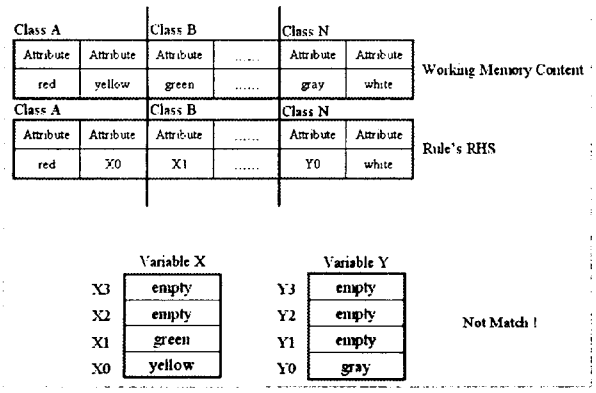

그림 6 변수 바인딩의 예 - 1

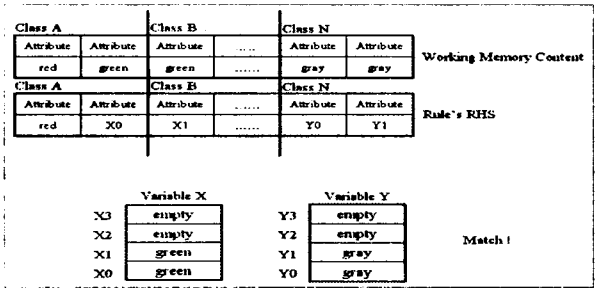

그림 7 변수 바인딩의 예 - 2

3. 실 험 결 과

본 논문에서 설계된 시스템은 2절에서 언급한 지능형 서비스 로봇을 위한 56 개의 환경 인식 가능한 룰-베이스 시스템으로의 적용을 목적으로 설계 되었다. 모든 설계는 SystemC 2.0의 behavioral 설계 기법을 사용 하였으며, SystemC 2.0 자체에서 제공하는 시뮬레이션 기법을 사용하여 그 동작을 확인 하였다. 작성된 코드는 Synopsys사의 System Compiler를 사용하여 합성 가능한 HDL 코드로 생성되어 적용되는 라이브러리 Technology에 상관없이 합성 가능 하다. 또한 작성된 코드는 SystemC의 문법적인 지원에 의해 헤더 파일의 파라미터 수치 변경만으로 다양한 조건의 하드웨어 구조를 생성 할 수 있다. 설계된 구조는 100 워드의 input buffer와 50워드의 WM, 4개의 PPMs 그리고 2.8k byte 의 룰-베이스 메모리로 설계 되었다. 시뮬레이션을 통해 그 동작 싸이클을 확인 하였는데, 하나의 룰을 매칭 해서 실행하는데 걸리는 연산 싸이클는 표 1과 같다.

Processing Path	Requited Cycles
From input buffer to WM	3
From WM to knowledge base	1
From knowledge base to variable binding	(avrg.) 5
From conflict resolution to input buffer	(avrg.) 5
Total	(avrg.) 14

표 1 시뮬레이션 결과

여기서 cycle은 시스템에 인가되는 clock 의 매 rising edge의 개수를 의미한다. 따라서 설계된 하드웨어 구조를 50 MHz 의 시스템 clock 환경에서 동작 시킨다면 평균적으로 매 초마다 3,500,000회의 추론을 수행할 수 있다. 비록 외부 호스트 프로세서와의 데이터 전송을 위한 overhead가 고려되지 않은 수치이지만, 발생하는 overhead가 로봇 외부의 센서 입력 데이터 전송에 의해서라고 가정 하면, 이 측정된 수치는 센서로부터의 입력 데이터에 실시간으로 대응 할 수 있는 처리 속도라 할 수 있다.

4. 결 론

본 논문에서는 지능형 서비스 로봇에 적용할 context-aware computing system의 구현을 위한 실시간 룰-베이스 시스템의 하드웨어적 구조를 제안 하였다. 시뮬레이션에 의하면 상황 인식을 위한 센서 데이터를 실시간으로 처리하는데 부족함이 없는 처리 속도가 가능 하다. 또한 변형된 형태의 CAM과 CSN의 적용으로 기존의 구조에서 문제시 되었던 룰의 조건문의 표현에서의 제약을 효과적으로 줄 일 수 있었다. 또한 호스트 프로세서와는 별도의 PPM을 두어 룰 내의 condition term의 연산 처리 속도를 향상 시켰다. 제안된 구조는 SystemC 2.0을 통하여 설계 되어 추후 구조적 수정이 용이 하다.

감사의 글: 본 논문은 21C 프론티어 "인간기능 생활 지원 지능로봇 기술개발" 과제에 의하여 지원되었으며 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] Pratibha and P.Dasiewicz, "A CAM Based Architecture for Production System Matching", reprinted in *VLSI for Artificial Intelligence and Neural Networks*, edited by J.G. Delgado-Frias and W.R. Moore, Plenum Press (1991) 57-66.

[4] Dou, C. "A highly-parallel match architecture for

AI production systems using application-specific associative matching processors", Application-Specific Array Processors, Proceeding, International Conference (1993) 180 183
[5] Kai Hwang "Advanced Computer Architecture : Parallelism, Scalability, Programmability", McGRAW-HILL (1993) 329-402
[6] M. Morris Mano "Computer System Architecture 3rd edn", Prentice Hall (1997) 456-768
[7] Joseph Giarratano, Gary Riley "EXPERT SYSTEM Principles and Programming", PWS-KENT Publishing Company (1989) 501-532