퍼지 및 지능시스템학회 논문지 2004, Vol. 14, No. 5, pp. 587-592

실시간 상황 인식을 위한 하드웨어 룰-베이스 시스템의 구조

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

이승욱* · 김종태* · 손봉기** · 이건명** · 조준동* · 이지형* · 전재욱*
Seung Wook Lee* · Jong Tae Kim* · Bong Ki Sohn** · Keon Myung Lee**
Jun Dong Cho* · Jee Hyung Lee* · Jae Wook Jeon*

* 성균관대학교 정보통신공학부
** 충북대학교 전기전자컴퓨터공학부

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. 제안한 구조는 기존의 하드웨어 기반의 구조가 가지는 룰의 표현 및 룰의 구성에서 발생하는 제약을 상당히 감소시킬 수 있다. 이를 위해 변형된 형태의 content addressable 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, 시스템의 설정에 의해 지정된 데이터들의 조합 및 pre-processing module(PPM)을 이용한 연산을 통하여 WM을 구성하는 데이터를 생성시킨다. 이 하드웨어 룰-베이스 시스템은 SystemC ver. 2.0을 이용하여 설계되었으며 시뮬레이션을 통하여 그 동작을 확인 및 검증하였다.

Abstract

Context-aware computing systems require real-time context reasoning process for context awareness. Context reasoning can be done by comparing input information from sensors with knowledge-base within system. This method is identical with it of rule-based systems. In this paper, we propose hardware rule-based system architecture which can process context reasoning in real-time. Compared to previous architecture, hardware rule-based system architecture can reduce the number of constraints on rule representations and combinations of condition terms in rules. The modified content addressable memory, crossbar switch network and pre-processing module are used for reducing constraints. Using SystemC for description can provide easy modification of system configuration later.

Key words : 룰-베이스 시스템, context-aware computing system, context reasoning

1. 서 론

Context-aware computing 시스템은 주변 환경을 여러 센서를 통해 감지하여 context를 파악 및 인식하고, 그 인식된 정보에 의해 적절한 서비스를 제공하는 시스템으로 정의할 수 있다[1-3]. 이를 위해서는 적절한 센서 관리 및 상황 추론을 위한 효율적인 기법이 사용되어야 한다. 센서 관리는 센서의 종류 및 시스템이 적용되어야 하는 응용 환경에 따라 적절한 기법이 사용되어야 한다. 하지만 상황 추론을 위해 센서의 종류 및 적용 환경에 따라 변하지 않는 기법이 사용된다면, 응용 환경이 다른 시스템들에 동일한 상황 추론 구조를 사용할 수 있을 것이다. 상황을 추론을 위한 방법으로서 시스템 내에 지식-베이스를 구축하여 입력되는 정보와의 비교를 통하여 현재의 상황을 추론하는 기법이 적용 가능하다. 이는 과거 여러 연구에 의해 그 성능이 입증된 룰-베이스 시스템의 동작과 동일하다. 룰-베이스 시스템은 인공지능 분야에서 성공적으로 실제 응용 분야에 적용된 기법으로 그 우수성이 잘 알려져 있다[4]. 하지만 기존의 소프트웨어 알고리즘에 기초한 룰-베이스 시스템의 처리 속도는 context-aware computing 시스템과 같은 실시간 처리가 필요한 응용 분야에는 적합하지 않다. 과거 몇몇의 연구에서 이와 같은 처리 속도의 문제점을 해결하기 위하여 룰-베이스 시스템을 위한 하드웨어적 접근을 시도했다[5]. 여러 연구에서 하드웨어적 접근을 위하여 CAM의 사용은 한 가지 해법이 되어왔다[6-7]. 하지만 기존의 구조에서 CAM의 적용은 RETE[8-9] 네트워크를 하드웨어적으로 묘사하기 위한 방법의 일환으로 사용되어 완전히 소프트웨어에 독립적으로 동작하는 구조는 아니다. 이로인해 적용되는 룰-베이스 시스템을 소프트웨어로 우선 모델링한 후 여기에서 필요한 시스템 구성을 위한 하드웨어의 규모 및 복잡도를 결정해야만 했다. 즉 적용되는 룰-베이스 응용 환경이 변화하면 그에따라 적당한 하드웨어를 다시 설정해야 하는 단점을 가지고 있었다. 하지만 제안된 구조는 룰-베이스 시스템이 적용되는 응용 환경에 구애를 받지 않고 동일한 독립적인 하드웨어 구조

접수일자 : 2004년 3월 31일
완료일자 : 2004년 5월 31일
감사의 글 : 본 논문은 21C 프론티어 "인간기능 생활지원 지능로봇 기술개발" 과제에 의하여 지원되었으며 이에 감사드립니다.

로 적용이 가능하다. 또한 상수 및 변수의 병렬 매칭이 가능하고 이전의 구조에 비하여 룰의 표현 및 룰을 구성하는 조건문 조합의 제약이 상당히 감소되었다. 이를 위해서 변형된 형태의 CAM과 CSN을 사용하였다. 변형된 CAM은 시스템의 지식-베이스를 구성하는데 이때 지식-베이스의 내용의 동적인 추가 및 삭제가 가능하다. 또한 CSN은 input buffer와 WM 사이에 위치하여 주어진 데이터를 가지고 룰을 구성하는 조건문의 생성에서의 제약을 감소시켰다.

2. 실시간 룰-베이스 시스템의 구조

2.1 시스템의 구조

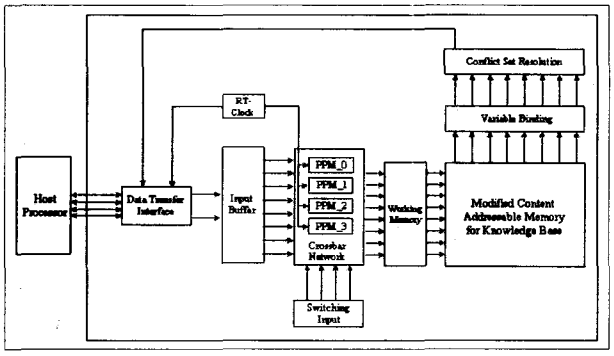


그림 1. 제안된 룰-베이스 시스템의 구조.
Fig. 1. Overview of real-time rule based system architecture.

실시간 룰-베이스 시스템의 구조는 그림 1와 같다. 이 시스템은 호스트 프로세서와의 인터페이스를 위한 data transfer interface block, input buffer, 변형된 CSN, WM, 지식-베이스의 구성을 위하여 변형된 CAM, variable binding block, conflict resolution block와 호스트 프로세서와는 별도의 RT-clock으로 구성 되어있다.

2.2 룰의 표현 기법

이번 절에서 시스템에 적용시키기 위한 룰의 구성 및 매칭 기법에 대해 논한다. 예제로 사용되는 룰-베이스 시스템은 일종의 지능형 서비스 제공 로봇의 상황 추론을 위한 시스템으로서 주변 상황을 로봇에 부착된 여러 센서를 통하여 수집한 후 현재 상황을 이해하여 적절한 서비스를 사용자에게 제공하는 시스템이다. 이를 위해 로봇에 적용할 56개의 룰을 결정하였다. 이 룰은 로봇 스스로 사용자의 건강상태, 실내상태, 비상상태의 대처가 가능하게하는 룰로서 우선 JESS(JAVA expert system shell)를 통해 소프트웨어적으로 그 룰의 동작이 검증 되었다.

시스템에 적용시킬 수 있는 룰은 그림 2과 같이 (1) 단일 attribute로 표현될 수 있는 룰, (2) 복수의 attribute가 AND 관계를 가지고 있는 룰, (3) 저장된 데이터들 간의 연산을 통한 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 위한 별도의 프로세서를 통해서 수행하게 되어있다[8-9]. 하지만 설계된 시스템 구조에서는

```
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 2. 적용된 룰의 예제.
Fig. 2. Example of rules.

input buffer가 센서 및 시스템 설정을 위해서 주어지는 모든 정보를 우선 저장한 후 별도의 조건문 전용 연산 PPMs을 사용하여 실제 룰 및 WM에 필요한 데이터로 가공한다. 그림 2의 룰들을 구성하기 위해서는 사용자의 맥박수 정보 및 사용자의 응답이 로봇의 센서로부터 주어지며, 시간정보 및 시스템 설정정보가 시스템 초기화 시에 저장되다. 즉 input buffer의 데이터는 지식-베이스 및 WM에서 사용되기 위한 attribute나 조건문들의 연산을 위한 미가공 데이터이다. 만일 룰이 다음과 같이 표현 된다면,

If A **AND** B ***then*** Action

여기서 A, B는 단일 attribute일 수도 있고, 위의 그림 2에서와 같이 어떤 연산의 결과일 수도 있다. 예를 들면 그림 2의 Rule 4에서 A는 <wait-response>로 외부센서 로부터 입력된 단일 attribute를 나타내고 B는 <clock-time - event-time-1 ≥ specified-time-1>로서 input buffer에 저장된 데이터를 가지고 연산을 수행한 결과이다. 만일 지식-베이스를 구성하는 메모리가 5개의 contents로 이루어져 있고, 각 content의 위치는 고유의 속성을 나타낸다면 위의 룰은 다음과 같이 지식-베이스 메모리에 저장된 모습으로 나타낼 수 있다.

Address	0	1	2	3	4
Content	A	B	d'c	d'c	d'c

여기서 0, 1, 2, 3, 4는 각 content에 부여된 주소 값이고, d'c는 해당 룰에서는 그 위치의 속성을 룰의 LHS에 가지고 있지 않음을 나타낸다. 또 다음과 같은 모습의 룰

If A **AND** E **AND** C ***then*** Action

은 아래와 같이 지식-베이스 메모리에 저장된다.

Address	0	1	2	3	4
Content	A	d'c	C	d'c	E

그러면 WM도 위와 같이 5개의 contents로 구성되며 각 위치의 속성은 룰-베이스의 그것과 같다. 이렇게 시스템의 메모리를 구성하면 다음에 설명될 변형된 CAM의 연산에 의해서 룰의 병렬 매칭이 가능한 모습의 지식-베이스를 구성 할

수 있다. 이와 같이 지식-베이스와 WM를 구성하기 위해서는 input buffer에 저장되어있는 데이터들의 가공이 필요하다. 이는 CSN과 PPM을 통해 이루어 진다. 이와 같이 지식-베이스를 구성함으로서 이론적으로 지식-베이스의 모든 룰과 WM와의 비교를 하나의 연산 싸이클 내에서 수행이 가능하다.

2.3 WM와 룰의 LHS 인코딩

효율적인 상수 및 변수 매칭을 위하여 WM와 룰의 LHS에 특별한 인코딩을 필요로 한다. 만일 WM의 각 content가 m-bits이면 A~N가지의 서로다른 class를 가지는 WM의 모습은 그림 3과 같이 표현 될 수 있다.

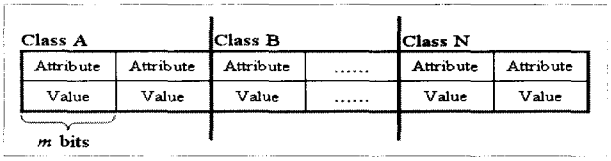


그림 3. WM의 인코딩.
Fig. 3. The form of working memory.

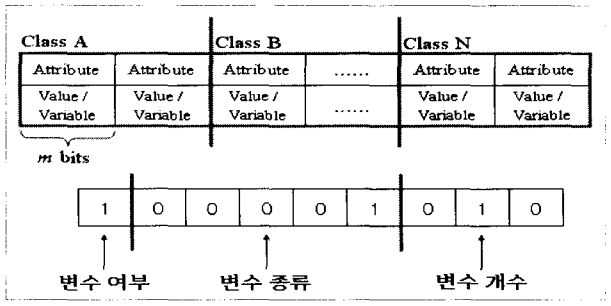


그림 4. 변수를 포함한 룰의 LHS 인코딩.
Fig. 4. The form of rule's LHS.

그림 4은 룰의 LHS에 변수를 가지고 있는 형태로 그림과 같은 포맷으로 저장된 데이터의 변수 여부, 변수의 종류, 변수의 개수를 표현할 수 있다. 2.2절에서 논의하였듯이 룰의 LHS를 구성하는 contents는 그 형태가 WM과 같다. WM contents로 표현 가능한 것은 attribute 또는 룰에서 표현 가능한 조건문의 연산 결과를 표현 할 수 있다. 이와 같은 방법으로 병렬 매칭을 위해 WM 부분과 이후에 설명될 지식-베이스 구성을 위한 변형된 CAM 사이에 별도의 하드웨어 로직이 없어도 된다. 즉 지식-베이스를 구성하는 CAM에 WM의 각 content로 어드레싱을 수행하면 변수 바인딩이 수행되지 않은 부분 매칭된 룰들을 병렬적으로 매칭 가능하다. 여기서 부분 매칭 된 룰들은 이후 variable binding block으로 전달 되어 최종 매칭 여부를 판별하게 된다. 병렬 변수 바인딩 기법은 이후의 절에서 논한다.

2.4 변형된 Crossbar Switch Network

CSN은 동적인 스위칭을 통하여 높은 대역폭을 가지는 노드간 인터커넥션을 제공할 수 있는 네트워크 구성의 하나이다[10]. 하드웨어 룰-베이스 시스템 구조에서는 CSN을 사용하여 input buffer로부터 WM의 contents로 사용될 데이터의 전송 및 조건문의 조합을 통한 연산 결과를 구성할 수 있다. 이는 이 룰-베이스 시스템의 적용 환경 및 룰의 구성이 변경될 경우 CSN의 스위칭 설정의 변경만으로 자유로운 데이터의 전송 및 다양한 조건문의 구성이 가능하게 한다. 또한 구조적 특성상 multi-point switching이 가능하므로 input buffer로부터 데이터 패치 시 스케쥴링의 제약을 피할 수 있다. CSN 상에 위치하는 PPMs은 조건문의 연산 결과를 산출 할 수 있는 일종의 ALU로서 별도의 호스트 프로세서의 추가 없이도 연산이 가능하므로 전체 룰-베이스 시스템의 연산이 병렬 프로세서 환경에서 작동되는 것과 같은 성능을 얻을 수 있다.

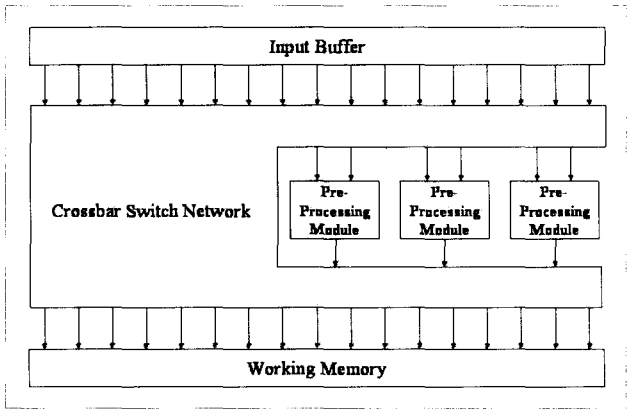


그림 5. Crossbar switch network의 구성.
Fig. 5. The crossbar switch network organization.

이 PPMs의 연산의 종류 결정을 위하여 CSN의 스위칭 시동시에 연산의 종류를 선택하는 input signal이 입력된다. 이 CSN와 PPMs의 연산 수행 시간은 PPM의 개수와 계산되어야 하는 조건문의 개수에 의존한다. 추가적인 CSN 및 PPMs의 설정 과정이 필요 없기 때문에 다음과 같은 연산 싸이클 수의 계산이 가능하다.

$$No.\ of\ cycles = \left\lceil \frac{Nmber\ of\ Condition\ terms}{Nmber\ of\ Pre-Processing\ Modules} \right\rceil$$

예를 들면 100개의 조건문과 30개의 PPMs가 있다면, 총 4회의 CSN 스위칭만으로 모든 조건문의 연산이 가능하다. 그림 5는 하드웨어 룰-베이스 구조에 CSN이 적용된 모습이다.

2.5 변형된 CAM의 구조

룰-베이스 시스템의 지식-베이스를 구성할 때 사용되는 메모리 디바이스로서 CAM의 사용은 룰의 매칭 과정에서 속도의 향상을 제공할 수 있다. 일반적인 CAM의 경우 key register를 두어 입력의 일정 부분을 masking을 할 수 있게 한다[11]. 이 masking 기능은 입력으로 주어지는 모든 bit을 저장된 메모리의 모든 content와 비교하지 않고, masking된 bit만을 비교하는 기능이다. 즉 입력으로 주어진 패턴중 일정 부분만 일치해도 출력의 매칭 bit를 세트하게 한다. 하지만 본 논문에서 구성한 CAM은 이 masking 기능이 개념상 일반적인 경우와 다르다.

그 이유는 WM contents에는 표현될 수 있는 attribute 및 조건문들의 연산의 결과들이 저장되어 있다. 하지만 룰은 이들 attribute 및 조건문을 모두 가지고 구성되는 것이 아니라 일부만을 조합하여 다양한 룰을 만든다. 여러 룰들은 CAM으로 구성된 지식-베이스에 저장되고, 이 CAM의 입력으로 들어오는 WM의 contents들 중에서 자신의 매칭 여부를 알 수 있는 일부 attribute 및 조건문들만을 비교하는 것이

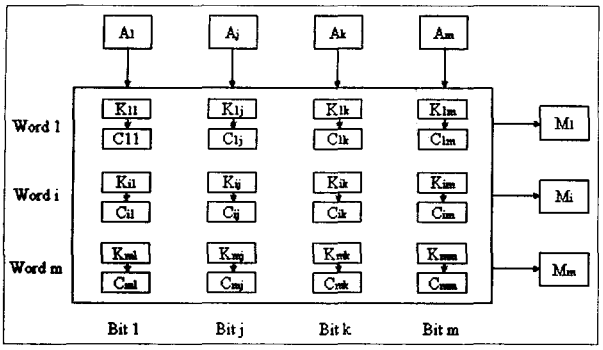


그림 6. 제안된 CAM의 구조
Fig. 6. The modified form of CAM

필요하다. 이를 위해 그림 6와 같이 일반적인 CAM에서 masking register의 기능이 확대된 구조로 masking register의 기능을 하는 로직을 각 memory cell에 삽입한 구조를 제안한다. 그림 6의 CAM은 로직 합성 가능한 SystemC ver. 2.0 코드로 작성되어 그 기능 및 로직 합성 후의 구조를 확인 가능하였다.

2.6 하드웨어 구조 기반의 변수 바인딩 기법

룰-베이스 시스템을 실시간 환경에 적용하기 어려운 이유는 그 실행 시간이 느리기 때문이었다. 이는 전체 실행 시간의 90% 정도를 차지하는 매칭 연산이 그 주된 원인인데, (90%중 가장 많은 부분을 변수 바인딩을 위한 연산에 사용된다[4].) 이에 본 논문에서는 매칭 연산 중 상당 부분을 차지하는 변수 바인딩 연산을 위한 하드웨어 구조 기반의 기법을 제안한다. 기존의 RETE와 같은 알고리즘이 효과적으로 변수 바인딩을 위한 기법을 제공하였으나, 변수 바인딩의 순서에 따라 연산량이 비효율적으로 증가될 수 있는 단점이 있었다[8].

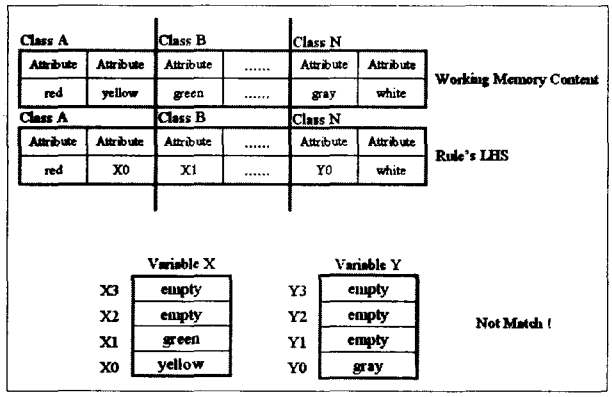


그림 7. 변수 바인딩의 예 - 1
Fig. 7. Example of variable binding process -1

하지만 이를 독립된 형태의 하드웨어 구조로 제안함으로서 변수 바인딩의 순서에 구애 받지 않고 연산량 또한 시스템이 설정될 때 고정적으로 결정될 수 있다. 그림 7에서, 만일 WM와 지식-베이스에 저장된 룰의 모습이 그림과 같다면, 룰에서 X0, X1은 그림 4와 같은 형식에 의해 표현된 변수이다. 하드웨어 구조에서 변수 바인딩을 하는 기법은 우선 표현할 수 있는 각 변수마다 룰 내에서 그 변수가 최대로 표현 가능한 개수의 CAM을 가지고 있는 것다. 그러면 2.3절의 룰의 형식에 의하여 룰 내의 변수는 자신의 CAM 영역에 고유의 위치를 가리킬 수 있다. 이때 variable binding block은 룰에서 변수가 나타난 위치의 WM의 content를 룰 내의 변수가 지시하는 고유의 영역에 복사한 후, 각 변수의 CAM에 동일한 content가 들어 있는지를 확인하면 된다. 만일 그림 7과 같은 경우라면, 변수 X의 X0, X1의 content가 서로 다르므로 그림 7에 표현된 룰은 변수 바인딩을 실패하게 된다. 따라서 최종적으로 충돌집합의 항목으로 추가될 수 없게 된다.

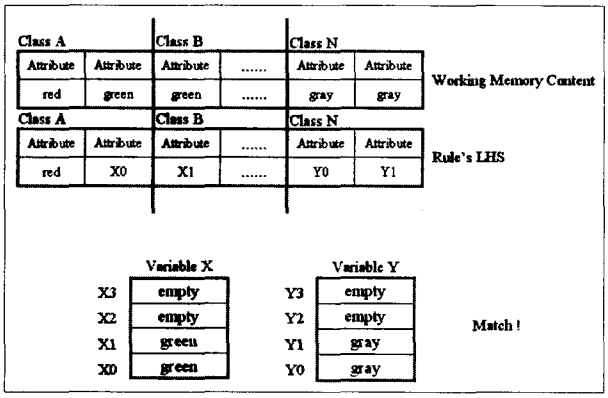


그림 8. 변수 바인딩의 예 - 2
Fig. 8. Example of variable binding process - 2

그림 8은 변수 X와 Y의 각 content에 동일한 항목이 저장되어 있으므로 변수 바인딩이 성공하여 룰이 매칭된 경우이다.

2.7 충돌해결

Conflict resolution block은 이전 단에서 생성된 충돌집합에서 실행시킬 룰을 선택하는 부분이다. 이 conflict resolution을 수행하는 정책으로 다양한 방법이 있다. 일반적으로 다음과 같은 방식들이 적용되는데, 보통 이들중에서 2~3개 정도를 동시에 적용시킨다[4].

(1) Agenda에서 첫 번째 규칙을 선택한다.
(2) Condition의 수가 가장 많은 규칙을 선택한다.
(3) 가장 최근에 변경된 object와 관련된 규칙을 선택한다.
(4) 가장 높은 우선순위의 규칙을 선택한다.
(5) 이전 싸이클에서 실행된 규칙은 agenda에 추가하지 않는다.
(6) 동일한 순위의 규칙이 하나 이상일 때는 임의로 선택한다.

위에 열거한 6가지의 정책이 하드웨어적으로 모두 구현 가능하나, 본 논문에서는 정책 (4), (5)가 적용된 하드웨어를 설계 적용하였다. 이의 적용을 위해 각 룰마다 우선순위를 부여하기 위한 버퍼와 이전의 실행 유무를 확인할 수 있는 별도의 카운터 로직을 삽입하였다. 또한 conflict resolution block은 실행이 결정된 룰이 WM의 내용을 변경해야 할 경우 데이터를 호스트 프로세서로 보내지 않고, 직접 input buffer에 접근하여 데이터의 업데이트가 가능하다. 이렇게 함으로서 호스트 프로세서와 데이터 전송으로 인한 오버헤드를 감소시킬 수 있다.

3. 시뮬레이션 및 결과

본 논문에서 설계된 시스템은 2장에서 언급한 지능형 서비스 로봇을 위한 56개의 환경 인식 가능한 룰-베이스 시스템으로의 적용을 목적으로 설계되었다. 모든 설계는 SystemC ver. 2.0의 behavioral 설계 기법을 사용하였으며, SystemC 자체에서 제공하는 시뮬레이션 기법을 사용하여 그 동작을 확인하였다. 작성된 코드는 Sysnopsys사의 System Compilier를 사용하여 로직 합성 가능한 HDL 코드로 생성되어 적용되는 라이브러리 technology에 상관없이 로직 합성이 가능하다. 또한 SystemC의 문법적 지원에 의해 헤더 파일의 파라미터 수치의 변경만으로 다양한 조건의 하드웨어 구조를 생성할 수 있다. 시뮬레이션에 사용된 구조는 100 워드의 input buffer와 50 워드의 WM, 4개의 PPMs 그리고 2.8kB의 지식-베이스 메모리로 설계되었다. 실험결과로서 시뮬레이션을 통해 그 동작 싸이클을 확인하였다. 하나의 룰을 매칭해서 실행하는데 걸리는 연산 싸이클의 수는 표 1과 같다.

표 1. 측정된 각 경로당 연산 싸이클
Table 1. Measured processing cycles for each path

Processing path	Required # of 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 1에 의하면 설계된 하드웨어 구조를 50MHz의 시스템 클럭 환경에서 동작 시킨다면 평균적으로 매 초당 3,500,000회의 추론을 수행할 수 있다. 비록 외부 호스트 프로세서와의 데이터 전송을 위한 오버헤드가 고려되지않은 수치지만, 발생하는 오버헤드가 로봇의 외부 센서 입력 데이터 전송에의해서라고 가정하면, 이 측정된 수치는 센서로부터의 입력 데이터에 실시간으로 대응할 수 있는 처리 속도라 할 수 있다.

4. 결 론

본 논문에서는 지능형 서비스 로봇에 적용할 context-aware computing 시스템의 구현을 목적으로 실시간 룰-베이스 시스템의 하드웨어적 구조를 제안하였다. 측정된 시뮬레이션 결과 데이터에 의하면 상황 인식을 위한 로봇의 외부 센서 데이터를 실시간으로 처리하는데 부족함이 없는 처리 속도가 가능함이 확인되었다. 또한 변형된 형태의 CAM과 CSN의 적용으로 기존의 구조에서 문제시 되었던 룰의 조건문의 표현에서의 제약을 효과적으로 줄일 수 있었다. 호스트 프로세서와는 별도의 ALU형 PPMs을 두어 룰의 조건문의 연산 처리 속도를 향상시켰다. 설계된 구조는 SystemC ver. 2.0을 통하여 설계되어 추후 시스템의 변경 및 확장이 용이하다.

참 고 문 헌

[1] G.D. Abowd, A.K. Dey "Towards a Better Under-standing 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, pp. 304-307, 1999.

[2] Abowd, G.D.; Ebling, M.; Hung, G.; Hui Lei; Gellersen, H.-W. "Context-aware computing" Pervasive Computing, IEEE , Volume: 1 , Issue: 3 , July-Sept. pp. 22-23, 2002.

[3] Munoz, M.A.; Rodriguez, M.; Favela, J.; Mar-tinez-Garcia, A.I.; Gonzalez, V.M. "Context-aware mobile communication in hospitals" Computer , Volume: 36 , Issue: 9 , Sept. pp. 38-46, 2003.

[4] Joseph Giarratano, Gary Riley "EXPERT SYSTEM Principles and Programming", PWS-KENT Publishing Company pp. 501-532, 1989.

[5] P. Kogge, J. Oldfield, M. Brule, and C. Stormon, "VLSI and rule-based systems," in VLSI for Artificial Intelligence, J. G. Delgado-Frias and W. R. Moore, Eds. Norwell, MA: Kluwer pp. 95-108, 1989.

[6] Pratibha and P.Dasiewicz, "A CAM Based Archi-tecture 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 pp. 57-66, 1991.

[7] Dou, C. "A highly-parallel match architecture for AI production systems using application-specific asso-ciative matching processors", Application-Specific Array Processors, Proceeding, International Conference pp. 180-183, 1993.

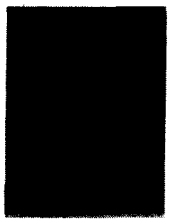
[8] C.L. Forgy, "RETE : A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem", Artificial Intelligence vol. 19 pp. 17-37, 1982.

[9] Mostafa M. Aref, Mohammed A. Tayyib, "Lana-Match Algorithm: A Parallel Version of the Rete-Match Algorithm.", Parallel Computing vol. 24. pp. 763-775, 1998.

[10] Kai Hwang "Advanced Computer Architecture : Parallelism, Scalability, Programmability", McGRAW -HILL pp. 329-402, 1993.

[11] M. Morris Mano "Computer System Archite -cture 3rd edn", Prentice Hall pp. 456-768, 1997.

저 자 소 개



이승욱(Seung Wook Lee)
2000년 : 성균관대학교 전기전자 및 컴퓨터 공학부 졸업
2002년 : 성균관대학교 대학원 전기전자 및 컴퓨터공학과 석사
2002년~현재 : 성균관대학교 대학원 전자전기공 박사과정

관심분야 : 지능시스템, 임베디드시스템, 시스템온칩
Phone : 031-290-7173
Fax : 031-290-7173
E-mail : swlee@ece.skku.ac.kr

김종태(Jong Tae Kim)
1982년 : 성균관대학교 전자공학과 졸업
1987년 : University of California, Irvine, Electrical Engineering 석사
1992년 : University of California, Irvine, Electrical and Computer Engineering 박사
1993년 : The Aerospace Corp. U.S.A., Member of technical staff.
1995년 : 전북대학교 컴퓨터공학과 전임강사
1995년~현재 : 성균관대학교 정보통신 공학부 교수

관심분야 : 임베디드시스템, 시스템온칩
Phone : 031-290-7130
Fax : 031-290-7179
E-mail : jtkim@yurim.skku.ac.kr

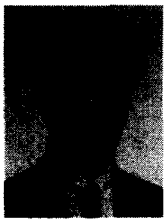
손봉기(Bong Ki Sohn)
1998년 : 서원대학교 전산학과 졸업.
2000년 : 충북대학교 대학원 전자계산학과 석사
2000년~현재 : 충북대학교 대학원 전자계산학과 박사과정

관심분야 : 에이전트 시스템, 기계학습, 워크플로우 시스템

Phone : 043-264-2263
Fax : 043-273-2265
E-mail : dobest@aicore.chungbuk.ac.kr

이건명(Keon Myung Lee)
1990 : KAIST 전산학과 학사
1992 : KAIST 전산학과 석사
1995 : KAIST 전산학과 박사
1995~1996 : 프랑스 INSA de Lyon, Post-Doc
1996~현재 : 충북대학교 전기전자컴퓨터 공학부 부교수

관심분야 : 기계학습, 데이터 마이닝, 소프트 컴퓨팅, 정보보안, 생물정보학
Phone : 043-261-2263
Fax : 043-273-2265
E-mail : kmlee@cbucc.chungbuk.ac.kr



조준동(Jun Dong Cho)
1989년 : Polytechnic Univ.에서 전산학 석사
1993년 : Northwestern Univ. 에서 전기 전산학 박사
1996년 6월 : IEEE Senior Member
1995년~현재 : 성균관대학교 정보통신 공학부 교수

관심분야 : Low Power SoC/CAD, SDR
Phone : 031-290-7127
Fax : 031-290-7191
E-mail : jdchoi@skku.ac.kr

이지형(Jee Hyung Lee)
1993년 : 한국과학기술원 전산학과 학사
1995년 : 한국과학기술원 전산학과 석사
1999년 : 한국과학기술원 전산학과 박사
2002년 : SRI International, International Fellow
2002년~현재 : 성균관대학교 정보통신공학부 조교수

관심분야 : 퍼지이론 및 응용, 지능시스템, 지능정보처리

Phone : 031-290-7154
Fax : 031-290-7211
E-mail : jhlee@ece.skku.ac.kr

전재욱(Jae Wook Jeon)
1984년 : 서울대학교 전자공학과 졸업
1986년 : 서울대학교 대학원 전자공학과 석사
1990년 : Purdue University, Ph.D
1990년~1994년 : 삼성전자생산기술센터 선임연구원
1994년~현재 : 성균관대학교 정보통신 공학부 교수

관심분야 : 로봇공학, 내장형 시스템, 공장자동화
Phone : 031-290-7129
Fax : 031-290-7231
E-mail : jwjeon@yurim.skku.ac.kr