퍼지 및 지능시스템학회 논문지 2005, Vol. 15, No. 2, pp. 224-229

유전 알고리즘 처리속도 향상을 위한 강화 프로세서 구조

Enhanced Processor-Architecture for the Faster Processing of Genetic Algorithm

윤한얼 · 심귀보

Han-Ul Yoon and Kwee-Bo Sim

중앙대학교 전자전기공학부

요 약

일반적으로 유전 알고리즘은 전형적인 프로세서에서 수행할 경우 매우 큰 시간·공간 복잡도를 가진다. 따라서 유전 알고리즘 처리를 위해서는 고성능·고가격의 프로세서를 필요로 하게 된다. 또한 이것은 유전 알고리즘을 소형 이동 로봇과 같이 비교적 간단한 룰을 필요로 하는 실제 하드웨어에 적용하는데 있어 큰 장벽으로 작용한다. 이러한 문제의 해결을 위해, 본 논문에서는 유전 알고리즘의 신속한 처리를 위해 강화된 프로세서 구조를 보인다. 정렬 네트워크와 residue number system (RNS)를 이용하여 일반적인 프로세서의 구조를 유전 알고리즘의 처리에 효율적이도록 강화할 수 있다. 정렬 네트워크는 유전 알고리즘에 필수적인 해들의 품질 비교를 하드웨어적으로 처리할 수 있게 하여 수행에 요구되는 시간을 줄일 수 있다. RNS는 산술 연산의 속도를 좌우하는 bit 사이즈를 줄여 전체적인 로직의 사이즈를 줄이고, 산술 연산의 처리 속도를 빠르게 할 수 있다.

Abstract

Generally, genetic algorithm (GA) has too much time and space complexity when it is running in the typical processor. Therefore, we are forced to use the high-performance and expensive processor by this reason. It also works as a barrier to implement real device, such a small mobile robot, which is required only simple rules. To solve this problem, this paper presents and proposes enhanced processor-architecture for the faster GA processing. A typical processor architecture can be enhanced and specialized by two approaches: one is a sorting network, the other is a residue number system (RNS). A sorting network can improve the time complexity of which needs to compare the populations' fitness. An RNS can reduce the magnitude of the largest bit that dictates the speed of arithmetic operation. Consequently, it can make the total logic size smaller and innovate arithmetic operation speed faster.

Key Words : sorting network, residue number system, elevator group control, enhanced processor-architecture

1. 서 론

유전 알고리즘의 매력은 기계 스스로 복잡한 문제들을 해결한다는데 있을 것이다. 과거에는 문제들을 해결하는데 정확한 초기 모델링과 수많은 시행착오를 거치는 실험이 요구되었으나 (human doing), 유전 알고리즘을 통해 기계에게 문제의 해결을 (machine doing) 맡길 수 있게 된 것이다. 이러한 강점은 NP-Hard 문제, 함수 최적화, 복잡한 제어기의 파라미터 값 추적, 신경망 최적화, 퍼지 시스템 최적화, 차량 라우팅, 그래프 분할, VLSI 회로 배치 등 광범위한 분야에 걸쳐 유전 알고리즘이 사용되는 배경이 되었다 [1].

일반적인 유전 알고리즘은 적합도 함수를 통해 해들의 품질을 결정하고, 해들의 품질에 따라 선택 연산을 거쳐, 교차나 돌연변이를 통해 우수한 품질의 해를 탐색하는 전형적인 과정을 갖는다 [2]. 따라서 일반적인 유전 알고리즘은 해들의 품질 비교를 위해 적어도 하나 이상의 정렬 알고리즘을 그 내부에 가지게 된다. 유전 알고리즘은 여러 세대를 반복하여 해를 생성하므로, 만약 해집단의 크기가 매우 큰 경우 해들의 품질 비교를 위해 정렬 알고리즘 내부에서 소요되는 시간 복잡도는 크게 증가 할 것이다. 이것은 간단한 룰 셋(rule set)으로 소형 로봇을 제어하려는 경우와 같은 간단한 문제에도, 전체 시스템은 고성능의 프로세서를 요구 하게 된다. 또한 유전 알고리즘을 C/C++이나 Matlab과 같은 소프트웨어에서의 구현에만 묶어두게 되며, 이 높은 소프트웨어 의존성은 유전 알고리즘을 적용하는 하드웨어를 구현할 때에 장벽으로 작용한다.

시스템을 설계할 때 고려해야 할 중요한 문제 중 또 다른 하나는 전체 시스템 로직의 사이즈를 줄이는 것이다. 처음 핸드폰이 시장에 나왔을 때, 대부분의 모델이 집에서 사용하는 무선 전화기와 거의 비슷한 사이즈를 가지고 있었다. 그러나 최근의 모델들은 담뱃갑보다도 작은 크기를 가지고 있으며, 음악이나 동영상 재생기능이나 인터넷 접속기능 까지

접수일자 : 2005년 2월 2일
완료일자 : 2005년 3월 18일
감사의 글 : 본 연구는 과학기술부의 뇌신경정보학연구사업의 '뇌정보처리에 기반한 감각정보 융합 및 인간행위 모델 개발'의 연구비 지원으로 수행되었습니다. 연구비 지원에 감사드립니다.

갖추고 있어 더욱 작은 사이즈와 여러 기능이 집적된 프로세서를 필요로 한다 [3]. 유전 알고리즘의 경우 초기에 유전자의 표현형과 그 길이를 정해 주어야 하는데, 이것이 적용하고자 하는 시스템의 로직 사이즈를 결정하는 중요한 요인이 된다. 예를 들어 임의의 문제에서 유전자의 표현형을 이진 표현 11bit으로 정의하였다고 가정해보자. 일반적으로 생산되는 프로세서들은 8bit, 16bit, 32bit으로 구분이 되므로 선택의 여지없이 16bit 프로세서를 사용해야 한다. 여기서 5bit의 낭비되는 로직이 발생하게 된다. 만약 표현형의 길이를 17bit으로 하였다면 32bit 프로세서를 채택해야 하고, 15bit의 낭비되는 로직이 발생할 것이다. Bit의 크기는 시스템의 산술/논리 연산에 수행되는 시간을 결정하는 요인이므로, 표현형의 길이를 17bit으로 하였지만 전체 시스템은 32bit의 연산을 수행하는데 걸리는 속도를 갖게 된다. 즉, 15bit의 처리시간을 낭비하게 된다. 이것은 시간 복잡도와 공간 복잡도를 모두 증가시키고, 유전 알고리즘의 소프트웨어 의존성을 높이는 또 하나의 요인이다.

본 논문에서는 위의 두 가지 문제의 해결을 위한 방법으로, 정렬 네트워크와 RNS를 적용한 프로세서 구조를 제안한다. 정렬 네트워크는 유전 알고리즘에서 해의 품질비교를 하드웨어 적으로 처리하므로, 시스템의 시간 복잡도를 크게 감소시킨다. RNS는 수를 표현하는데 요구되는 가장 큰 bit의 크기를 줄여주므로 시간·공간 복잡도를 모두 감소시킬 수 있다. 본문의 2장에서는 정렬 네트워크의 개념과 실제적 구현을 나타내고, 3장에서는 RNS의 개념, RNS adder 설계, RNS 교차 연산부의 구현에 대해 설명한다. 4장에서는 정렬 네트워크와 RNS를 적용한 유전 알고리즘 처리속도 강화 프로세서를 가지고 엘리베이터 군 제어 시뮬레이션을 보인다. 마지막으로 5장에서는 결론 및 향후과제에 대해 논한다.

2. 정렬 네트워크의 모델과 구현

2.1. 정렬 네트워크 (Sorting Network)

정렬 네트워크(sorting network)란 N개의 모든 입력값을 정렬해 낼 수 있는 k개의 비교기(comparator) 블록으로 구성된 네트워크를 말한다. 1969년 Green은 16개의 입력을 정렬하는 네트워크의 최적 구성은 60개의 비교기와 10회의 병렬 스텝(parallel step)과 같다는 것을 증명하였다. Hillis는 공진화 알고리즘을 사용하여 61개의 비교기와 11회의 병렬 스텝으로 이루어진 정렬 네트워크 모델을 찾았다. 이것은 Green이 증명한 최적 모델 보다는 각각 1개씩 많은 비교기와 병렬 프로세싱 스텝으로 이루어져 있지만, 유전 알고리즘을 이용하여 머신 스스로 모델을 찾아내도록 했다는데 의미가 있다 [1][4].

정렬 네트워크는 다음과 같은 장점을 가진다. 첫째, 구성이 단순한 비교기로 되어있어 하드웨어적 구성이 쉽다. 둘째, 어떠한 정렬 알고리즘이 진행되는 과정을 눈으로 직접 보여주므로 그 알고리즘을 쉽게 이해할 수 있다. 마지막으로, 최대/최소값을 요구하는 어떠한 프로그램의 과정에 있어, 매우 짧은 시간(4 machine ticks)내에 최대/최소값을 구할 수 있다 [4]. 이러한 장점들은 데이터 정렬에 많은 수행시간을 소비하는 프로그램의 처리 속도를 크게 향상시킬 수 있다.

2.2. 정렬 네트워크 설계

정렬 네트워크를 구성하는 하나의 블록은 2개의 Input data bus, 1개의 comparator cell, 그리고 2개의 Output data bus로 구성된다. 그림 1은 정렬 네트워크를 구성하는 1개 블록의 구조를 나타낸다. 2개의 input data는 bus A와 B를 통해 comparator cell로 들어간다. comparator cell은 1개의 comparator와 2개의 2×1 MUX로 이루어져 있어, 만약 A>B 라면 output bus의 data는 위쪽이 B, 아래쪽이 A가 되어 두 값이 서로 위치를 바꾸도록 데이터 흐름을 제어한다. A≤B 의 경우에는 output bus data는 그대로 위쪽이 A, 아래쪽이 B가 된다. 즉, 두 값 중에서 큰 값은 아래쪽 bus로, 작은 값은 위쪽 bus로 나가게 된다.

본 논문에서는 Green의 16-input 정렬 네트워크(그림2)를 사용하고 이를 통한 효율성을 최대화하기 위해 이 프로세서에서 수행할 유전알고리즘의 선택 연산에 순위 기반 선택을 사용한다. 순위 기반 선택은 해집단 내의 해들을 품질 순으로 '순위'를 매긴 다음 가장 좋은 해부터 일차 함수적으로 적합도를 배정하는 방법이므로 정렬 네트워크의 효율성을 가장 잘 보여줄 수 있다 [1]. 여기서 품질에 따른 해들의 순위를 정하기 위해서는 해들을 정렬 네트워크를 통과 시킨 후, 저장된 메모리에 위치에 위에서부터 그대로 순위를 배정한다.

Green의 16-input 정렬 네트워크는 16개의 입력만 정렬해 낼 수 있으므로 16개 이상의 데이터를 정렬하기 위해서는 다른 방법이 요구된다. Input data의 개수를 확장하려 할 때는 cross sorting을 이용한다. cross sorting이란 4개의 N-input local 정렬 네트워크와 cross sorting network을 이용하여 1개의 2N-input data 정렬 네트워크를 구성하는 것이다 [5]. 그림 3은 4개의 16-input 정렬 네트워크와 1개의 cross sorting network을 이용하여 32-input 정렬 네트워크의 구성을 보여준다.

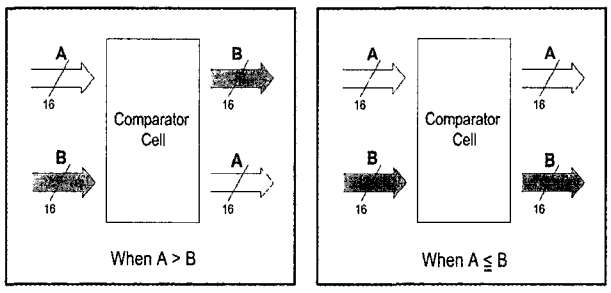

그림 1. Comparator cell을 통한 데이터 입/출력 흐름
Fig. 1. Data I/O flow through a comparator cell

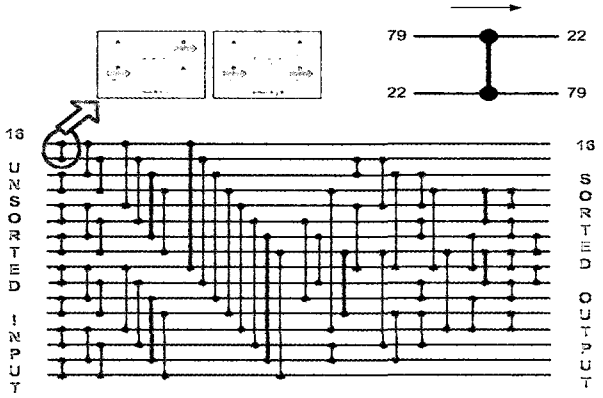

그림 2. Green model 16-input 정렬 네트워크
Fig. 2. Green model 16-input sorting network

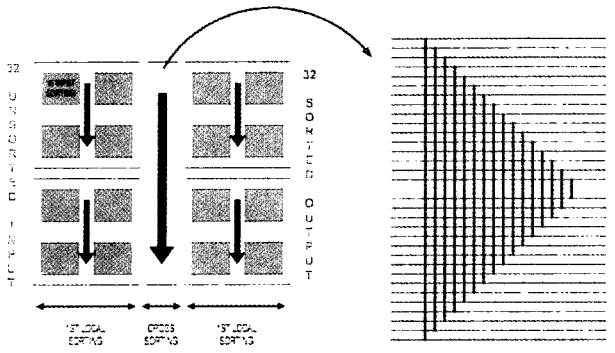

그림 3. 데이터 입력 개수 확장을 위한 cross sorting 방법
Fig. 3. Cross sorting method for input data extension

2.3. 정렬 네트워크 시뮬레이션

Input data sequence를 최상위 16부터 최하위 1까지의 Worst case로 두고 정렬 네트워크를 테스트 하였다. 본 논문의 모든 설계에는 Altera사의 Max-plus II를 사용하였다. 그림 4는 16-input 정렬 네트워크가 10 machine clocks에 모든 입력 데이터를 1부터 16으로 정렬해 내는 것을 Max-plus II 시뮬레이터 환경으로 보여준다.

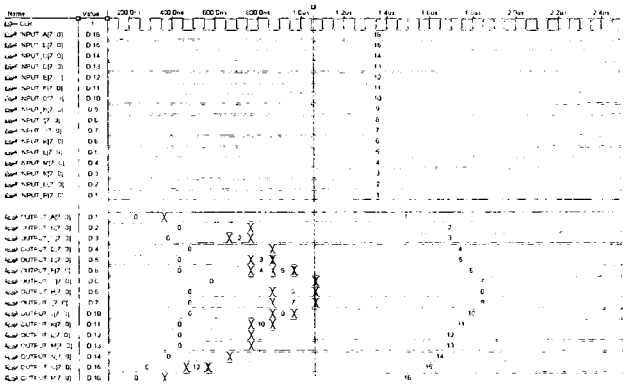

그림 4. 정렬 네트워크를 통한 16-input data의 정렬
Fig. 4. 16-input data sorting by sorting network

3. Residue Number System의 개념과 설계로의 응용

3.1. Residue Number System

약 1500년 전 중국의 학자 Sun Tsu는□□2로 나누었을 때 7이 남고, 3으로 나누었을 때 5가 남고, 2로 나누었을 때 3이 남는 수들 중, 가장 작은 것은 무엇인가?□□라는 운문 형태의 수수께끼를 썼는데, 이것이 바로 residue number system (RNS)의 기본 개념을 내포하고 있다. 즉 RNS는 어떤 수의 각 자리의 숫자를, 각 자리에 할당된 제수로 나누어 졌을 때의 나머지로 표현한다.

이를 바탕으로 Behrooz Parhami는 RNS라는 새로운 수 체계를 제안하였다. Parhami의 제안에 따르면, RNS에서 어떤 수 x는, k개의 서로 소 관계의 제수인 prime moduli들 $m_{k-1} > \cdots > m_1 > m_0$로 나누었을 때의 나머지들의 나열로 표현된다[6]. 따라서 k자리수의 각 자릿수 x_i는 그 자리에 해당된 $m_i\ (0 \le i \le k-1)$로 나누었을 때의 나머지들을 의미한다. 따라서 x_i의 표현 범위는 $[0, m_i - 1]$이 되고, 이를 기호로,

$$x_i = x \bmod m_i = \langle x \rangle_{m_i}$$

와 같이 나타낸다. 앞에 나온 Sun Tsu의 수수께끼에서, 수수께끼의 답이 되는 수는 $(2|3|2)_{RNS(7|5|3)}$과 같이 표현할 수 있다. 어떤 수 x의 모든 m_i들의 곱을 *dynamic range*라 하고, 이것은 해당 RNS수로 표현할 수 있는 수들의 개수를 의미한다. 즉, M은

$$M = m_{k-1} \times \cdots \times m_1 \times m_0$$

이다. 음수는 다음과 같이 표현된다.

$$\langle -x \rangle_{m_i} = \langle M - x \rangle_{m_i}$$

한 가지 예외적인 것은 0은 바뀜 없이 그대로 나타낸다. 따라서 $(5|0|1|0)_{RNS(8|7|5|3)}$은 21을 나타내고, 음수 표현 규칙에 따라 $(8-5|0|5-1|0)_{RNS(8|7|5|3)} = (3|0|4|0)_{RNS(8|7|5|3)}$은 -21을 나타낸다.

RNS에서 덧셈, 뺄셈, 곱셈은 각 자리끼리의(digit by digit) 연산으로 간단히 이루어진다. 예로 $m_3 = 8$, $m_2 = 7$, $m_1 = 5$, $m_0 = 3$으로 표현된 어떤 RNS수 $x = (6|6|1|0)_{RNS\ (8|7|5|3)}$은 +6을 나타내고, $y = (6|5|3|1)_{RNS(8|7|5|3)}$은 -2의 연산은 다음과 같다.

$$x = (6|6|1|0)_{RNS(8|7|5|3)} = +5$$
$$y = (6|5|3|1)_{RNS(8|7|5|3)} = -1$$

$$x + y: \langle 6+6 \rangle_8 = 4,\ \langle 6+5 \rangle_7 = 4,\ etc.$$
$$= (4|4|4|1)_{RNS(8|7|5|3)} = +4$$

$$x + y: \langle 6-6 \rangle_8 = 0,\ \langle 6-5 \rangle_7 = 1,\ etc.$$
$$= (0|1|3|2)_{RNS(8|7|5|3)} = +8$$

$$x \times y: \langle 6 \times 6 \rangle_8 = 4,\ \langle 6 \times 5 \rangle_7 = 2,\ etc.$$
$$= (4|2|3|0)_{RNS(8|7|5|3)} = -12$$

RNS의 정의에 따라, 가장 큰 제수 m_{k-1}는 Arithmetic/Logic Unit (ALU)의 adder 크기를 결정한다. Adder의 크기에 의해 산술 연산의 속도가 결정되므로, 적절한 m_{k-1}을 정해주어야 한다. 또한 각 자리들은 이진 코드로 표현되므로 연산에 있어서 1's complement adder를 그대로 이용할 수 있도록, 가능하면 $2^a - 1$ 형태의 m_i를 정해주는 것이 좋다. 만약 다른 임의의 m_i를 정하면 0부터 $m_{k-1} - 1$ 까지 범위를 가지는 새로운 adder를 설계해야 하기 때문이다. 이런 이유로 $2^a - 1$ 형태의 m_i들을 *low-cost moduli*라고 한다.

3.2. RNS Adder 설계

RNS adder의 설계에서 가장 우선시 되는 것은 binary수를 RNS수로 바꾸어 주는 것이다. 먼저 Prime moduli들을 $m_3 = 2^5$, $m_2 = 2^5 - 1$, $m_1 = 2^4 - 1$, $m_0 = 2^3 - 1$ 과 같이 정하였고, 이것은 각각 십진수로 32, 31, 15, 7에 해당한다. 16bits 은 unsigned라고 가정했을 때 0부터 65535까지 표현되므로, dynamic range M은 65535보다 커야 한다. 위에서 정한 prime moduli들에 의해 $M = 32 \times 31 \times 15 \times 7 = 104160$이므로 조건을 만족한다. 그림 5는 앞에서 정한 RNS수를 표현하는데 필요한 총 bit의 개수를 보여준다. 이 RNS 포맷은 5bits, 5bits, 4bits, 3bits 모두 17bits으로

16bits binary input 보다는 1bit이 더 많다. 그러나 위의 포맷으로 수를 표현함에 따라, 산술 연산을 처리하는데 요구되는 가장 큰 adder의 사이즈는 5bit adder가 된다 (산술 연산 속도는 가장 상위 bit에 할당된 제수 m_{k-1} 에 의해 결정되므로). 결론적으로 16bits 산술 연산을 5bit 산술 연산의 속도로 처리 한다.

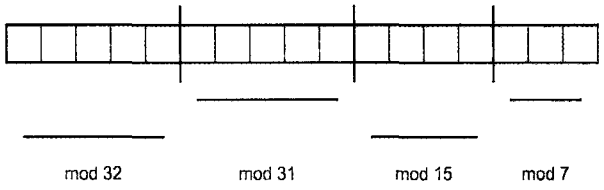

그림 5. RNS(32|31|15|7)의 이진 코드 포맷
Fig. 5. Binary coded format for RNS(32|31|15|7)

RNS로의 변환에 있어서 *representational efficiency*의 정의는 다음과 같다.

dynamic range M / the number of value that k bits can represent

RNS(32|31|15|7)은 4자리를 표현하는데 17bits를 필요로 하므로, 위의 정의에 따르면

$$104160 / 2^{17} \cong 79\%$$

가 된다. 양변에 log를 취하면 log2(104160) − 16.668 이므로 17 − 16.668 = 0.332가 되고, 이것은 0.332bit이 낭비됨을 뜻한다. 그러나 0.332bit이란 의미 없는 값이므로, 낭비되는 bit은 실질적으로 없다고 할 수 있다.

이와 같이 Prime moduli들을 정한 후에 binary수로 부터 정해준 포맷의 RNS수로의 변환은 다음의 등식에 의해 이루어진다 [6].

$$\langle (y_{k-1} \cdots y_1 y_0)_{two} \rangle_m = \langle \langle 2^{k-1} y_{k-1} \rangle_m + \cdots \langle 2 y_1 \rangle_m + \langle y_0 \rangle_m \rangle_m \quad (1)$$

따라서 mod32의 나머지(x_3)는 자연적으로 y의 LSB(least significant bit) 5bits이 된다. mod31, mod15, mod7의 나머지는 31, 15, 7이 모두 2^a-1 형태의 low cost moduli이므로 a bit 씩으로 y를 나눈 다음, 모든 값을 2^a-1 자리에 더해주면 된다. 예를 들어 y가

$$y = 0110010100010100_{(2)}$$

일 때, mod31은 2^5-1 이므로 하위 bit부터 5자리씩 끊어 읽으면,

$$y = 0\ 11001\ 01000\ 10100_{(2)}$$
$$\therefore x_2 = 0+25+8+20 = \langle 51 \rangle_{31} = \langle 20 \rangle$$

이므로 x_2는 20이 된다. x_1과 x_0도 같은 원리로 변환된다.

RNS에서 binary로의 역변환에서는 Chinese Remainder Theorem (CRT)에 의해 결정된다[6]. CRT는 다음과 같다.

$$x = (x_{k-1}|\cdots|x_1|x_0)_{RNS} = \left\langle \sum_0^{k-1} M_i \langle \alpha_i x_i \rangle_{m_i} \right\rangle_M \quad (2)$$

여기서 $M_i = M/m_i$, 그리고 $\alpha_i = \langle M_i^{-1} \rangle_{m_i}$ 로서, 식(1)과 관련된 곱셈에 대한 역원을 의미한다. 이것은 역변환의 과정에 곱셈 연산이 요구됨을 의미하는데, 곱셈 연산은 반복적인 덧셈 수행 시간이 요구되므로 시스템의 효율을 떨어뜨린다. 따라서 미리 정해진 값을 Look up table을 통해 읽어오는 방식을 택하는 것이 전체 시스템의 성능을 고려할 때 더 효율적이다. 그림 6에 전체 RNS adder의 구조를 나타내었다. 그림 7은 2개의 16-bit binary input에 대한 RNS로의 변환과 덧셈연산의 수행 결과를 Max-Plus II 시뮬레이터로 보여준다. 결과를 보면 십진수 42320은 $(16|5|5|5)_{RNS(32|31|15|7)}$로, 10523은 $(27|14|8|2)_{RNS(32|31|15|7)}$로 각각 변환된다. 두 RNS수의 연산 결과는 $(11|19|13|0)_{RNS(32|31|15|7)}$가 되고, 십진수로 바꾸면 52843이므로 동일한 결과를 보이고 있다.

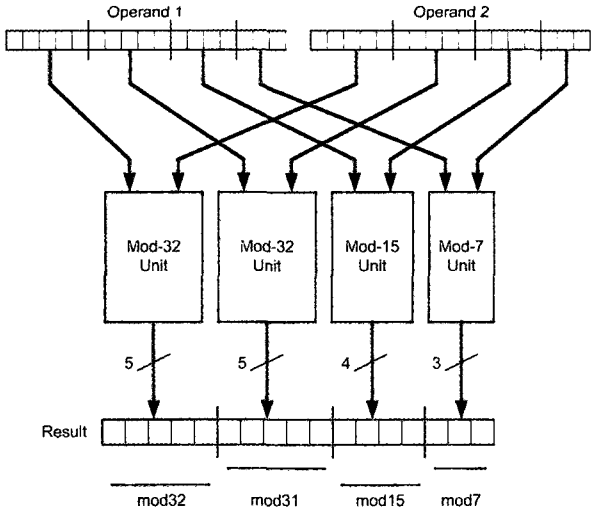

그림 6. RNS(32|31|15|7)의 포맷 RNS adder 구조
Fig. 6. The structure of an adder for RNS(32|31|15|7)

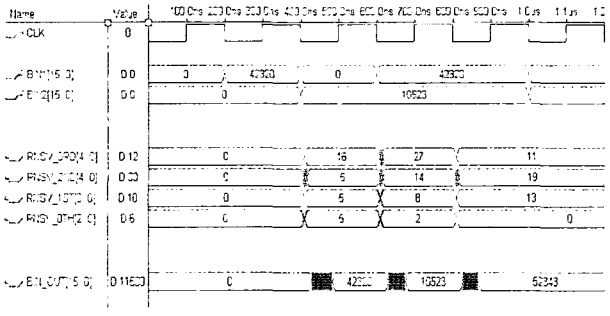

그림 7. 십진수 42320과 10523의 RNS 변환 및 덧셈 연산
Fig. 7. RNS addition with decimal 42320 and 10523

3.3. RNS 교차 연산부 설계

RNS의 장점은 낭비되는 bit을 줄이고, 산술 연산의 속도를 빠르게 한다는데 있다. 이것은 유전 알고리즘의 교차연산에도 강력한 이점을 가지는데, 첫째로 적절한 문제의 해에 대한 표현형에 있어 bit의 낭비가 없으며, 둘째로 교차 연산에서 자름선의 기준을 두는 문제에 있어 RNS는 그 표현법 자체에서 명확한 자름선의 기준을 제공한다는 것이다. 그림 8에 RNS 표현을 이용한 일점 교차 (one point crossover) 연산의 예를 보였다.

일반적으로 교차 연산은 소프트웨어적 bitwise연산을 통

해 이루어진다. 그러나 본 논문에서는 교차 연산에 소요되는 시간 역시 줄이기 위해, 교차 연산을 하드웨어 적으로 처리하도록 한다. 그림 9는 RNS 변환 후의 하드웨어적 교차 연산이다. $x=(6|6|1|0)_{RNS(8|7|5|3)}$, $y=(6|5|3|1)_{RNS(8|7|5|3)}$의 두 부모해로 부터 가운데 자름선을 기준으로 (6|6)과 (3|1)을 가지고 일점 교차를 수행하면 $(6|6|3|1)_{RNS(8|7|5|3)}$로 이것은 +118이 된다. 교차 연산은 버스 renaming을 통해 쉽게 이루어질 수 있다.

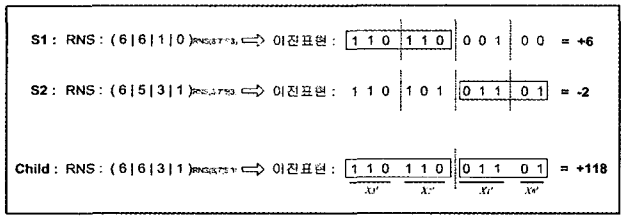

그림 8. RNS 표현상의 일점 교차 연산
Fig. 8. One-point crossover operation with RNS

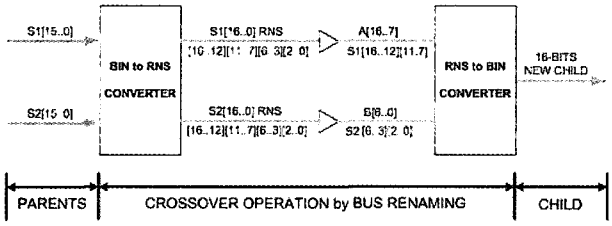

그림 9. 하드웨어적 일점 교차 연산
Fig. 9. One-point crossover by hardware bus renaming

4. 유전 알고리즘을 위한 강화 프로세서

4.1. 유전 알고리즘의 빠른 처리를 위한 프로세서 구조

유전 알고리즘의 빠른 처리를 위한 강화 프로세서 구조는 다음과 같이 이루어진다. 32×32bits 범용 레지스터 내부의 8개(R24, …, R30, R31)는 정렬 네트워크와 연결되어 있다. 프로세서는 64k words의 데이터 메모리(Data MEM)를 가지고 있는데, 이 중 0x854h번지부터 정렬 네트워크를 통해 정렬된 값이 저장이 되도록 하였다. 따라서 앞에서 말한 범용 레지스터 R24, … , R30, R31에 데이터를 보내고, 데이터 메모리의 0x854h부터 순차적으로 8개의 값을 로드하면 정렬된 8개의 데이터를 얻을 수 있다. RNS 교차 연산부는 뒤에 나올 엘리베이터 매력 함수에 속한 각 함수들의 웨이트를 조정하기 위한 간단한 유전 알고리즘 수행에 이용된다. 그림 10은 프로세서의 전체 구조를 보여준다.

4.2. 강화 프로세서를 이용한 엘리베이터 군 제어 시뮬레이션

엘리베이터 군 제어 시스템이란 임의의 층에서의 호출이 있을 때, 복수개의 엘리베이터로부터 현재위치·서도록 예약되어있는 층의 수 등의 정보를 받아 매력함수를 산출한 후, 각 값들을 비교하여 매력 함수 값이 가장 높은 엘리베이터를 호출 층에 서도록 제어하는 시스템이다. 이 매력 함수 값 산출을 위해 어떤 층에서 호출이 오면 각 엘리베이터 들은 자신이 가지고 있는 정보들을 보낸다. 각 엘리베이터는 시뮬레이션을 위해 실제 건물의 엘리베이터와 동일한 작용을 하도록 VHDL로 모델링 하였다. 그림 11은 모델링한 엘리베이터 객체와 프로세서간의 인터페이스를 나타낸다.

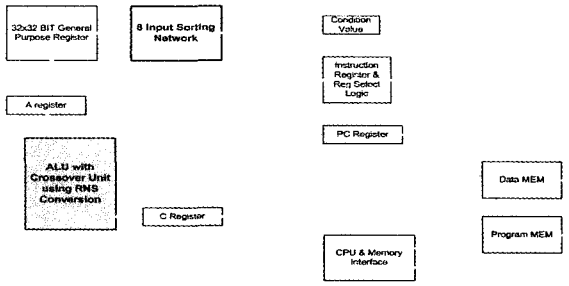

그림 10. 유전 알고리즘 처리 강화 프로세서 구조
Fig. 10. Processor architecture for faster GA processing

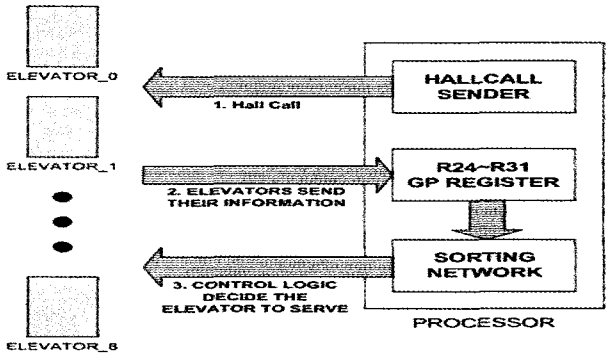

그림 11. 엘리베이터와 프로세서간의 I/O 인터페이스
Fig. 11. I/O interface between elevator and processor

엘리베이터의 매력 함수 모델에는 여러 종류들이 있지만 본 논문에서는 다음의 식을 사용하였다.

$$f(i,j) = w_1 h(i,j) + w_2 g(i,j) + w_3 c_i + w_4 t(i) \quad (3)$$

여기서 $h(i,j)$는 엘리베이터 i의 현재 위치에서 j층까지의 물리적 거리를 반영하는 함수 이다. $g(i,j)$는 엘리베이터 i가 j층 근방에 있는 층에 정지 계획이 있을 때 보상을 해주는 함수이다. c_i는 엘리베이터 i의 탑승인원 수를 %로 나타낸다. $t(i)$는 엘리베이터 i의 현재 위치와 j층 사이의 층들 중 엘리베이터 i가 정지하기로 계획되어 있는 층들의 수이다. w_1, w_2, w_3, w_4는 각 함수들의 weight들을 나타낸다[1][7]. 본 논문에서는 $h(i,j)$와 $g(i,j)$가 매우 유사한 형태임을 이용, 두 함수를 $w_H H(i,j)$로 합성하여 적용하였다 ($w_H = w_1 + \alpha w_2$, $\alpha = g(i,j)/h(i,j)$). 표 1은 8대 엘리베이터의 초기 설정을 보여준다.

표 1. 8대 엘리베이터의 초기 환경 설정
Table 1. Initial conditions for 8-elevators

E_ID#	E0	E1	E2	E3	E4	E5	E6	E7
현재층	5	12	17	23	11	30	18	18
정지예약	7, 10	10	25	12,17	5	22	22	19

그림 12는 표 1 상황의 시뮬레이션 결과를 보여준다. 먼저 호출은 20층에서 발생하도록 하였다. 매력 함수 산출 식을 바탕으로 예측을 해보면 20층에서 가장 가까운 18층에 있고, 그 사이 정지하도록 예약 된 층이 없는, 6번 엘리베이터의 매력 함수 값이 가장 높을 것임을 알 수 있다. 만약 예측과 다른 엘리베이터가 선택이 되었다면 RNS 교차 연산부를 통해 엘리베이터 매력 함수의 파라미터 값을 다시 결정한다. 결과를 보면 예측대로 6번 엘리베이터가 선택되었음을 확인할 수 있다. 최적 엘리베이터 선택까지의 시간은 시뮬레이터 환경상의 값으로 700us이 소요되었다.

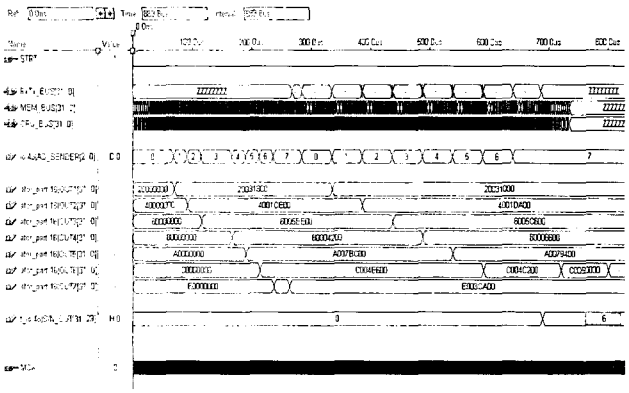

그림 12. 엘리베이터 군 제어 시뮬레이션
Fig. 12. Elevator group control simulation

5. 결론 및 향후과제

본 논문에서는 유전 알고리즘의 신속한 처리를 위한 프로세서 구조를 설계하였다. 정렬 네트워크를 이용한 해들의 품질 비교에 소요되는 시간 단축과 RNS를 이용한 빠른 산술연산 및 교차 연산에의 적용을 엘리베이터 군 제어 시뮬레이션을 통하여 보였다. 그러나 어셈블러로만 프로그래밍이 가능하도록 설계되어 있어, 단위시간 내 엘리베이터간의 경쟁과 같은 보다 복잡한 상황을 실험해 볼 수 없었다. 또한 이 프로세서 구조를 적용하기 전 해결하고자 하는 문제가 이 구조에 적절한지를 먼저 판단해야 하며, 비교적 간단한 문제에만 적용될 수 있다.

향후 과제로는 C/C++ 등을 이용한 보다 쉬운 프로그래밍 인터페이스 구축과, 보다 넓은 문제에 적용하기 위한 부동소수점 처리부의 적용 등이 뒤따라야 할 것이다.

참 고 문 헌

[1] 문병로, *유전 알고리즘*, 두양사, 2003.
[2] 강훈, 심귀보, *지능정보시스템*, 브레인코리아, 2003.
[3] Han-Ul Yoon, Kyoung-Taik Park, and Kwee-Bo Sim, "Improvement of processing time using residue number system and sorting network in controller design," *IECON 2004 Computer and control system*, pp. 1-6, 2004.
[4] H. W. Lang. (2002, Dec 12). Sorting Network. [Online]. Available: http://www.iti.fe-flensburg.de/lang/algorithmen/sortieren/ sortineren.htm.
[5] S. Olariu, M. C. Pinotti, and W. Q. Zheng, "How to sort n Items using a sorting network of fixed I/O size," *IEEE Trans. On Parallel and Distributed Syst.*, vol. 10, no. 5, pp. 1-12, May, 1999
[6] Behrooz Parhami, *Computer Arithmetic*, Oxford Newyork, 2000.
[7] Young Cheol Cho, Zavarin Gagov, and Wook Hyun Kwon, "Elavator Group Control with Accurate Estimation of Hall Call Waiting Times", *IEEE International Conference on Robotics & Automation*, pp. 1-3, 1999.

저 자 소 개

윤한일(Han-Ul Yoon)
2004년 : 중앙대학교 전자전기공학부 공학사
2004년~현재 : 동대학원 전자전기공학부 석사과정

관심분야 : Processor architecture, DARS, Reinforcement learning

Phone : +82-2-820-5319
E-mail : huyoon@wm.cau.ac.kr

심귀보(Kwee-Bo Sim)
1984년 : 중앙대학교 전자공학과 공학사
1986년 : 동대학원 전자공학과 공학석사
1990년 : The University of Tokyo 전자공학과 공학박사
1991년 ~ 현재 : 중앙대학교 전자전기공학부 교수
2003년 ~ 2004년 : 일본계측자동제어학회(SICE) 이사
2000년 ~ 2004년 : 제어자동화시스템공학회 이사 및 (현) 지능시스템연구회 회장
2003년 ~ 2004년 : 한국퍼지 및 지능시스템학회 부회장 (현) 수석부회장
2002년 ~ 현재 : 중앙대학교 산학연컨소시엄센터 센터장 및 기술이전센터 소장
2005년 ~ 현재 : 한국퍼지 및 지능시스템학회 수석부회장
관심분야 : 인공생명, 지능로봇, 지능시스템, 다개체시스템, 학습 및 적응알고리즘, 소프트 컴퓨팅(신경망, 퍼지, 진화연산), 인공면역시스템, 침입탐지시스템, 진화하드웨어, 인공두뇌, 지능형 홈 및 홈네트워킹, 유비쿼터스 컴퓨팅 등

Phone : +82-2-820-5319
Fax : +82-2-817-0553
E-mail : kbsim@cau.ac.kr
Homepage URL : http://alife.cau.ac.kr