시뮬레이션 기반 최적화에서 유전자 알고리즘 이용한 후보 모델 생성 기법

김 호 영, 김 준 경, 김 영 걸, 김 탁 곤
한국과학기술원 전자전산학과
시스템 모델링 시뮬레이션 연구실
E-mail : hykim@smslab.kaist.ac.kr

GA-instrumented Candidate Model Generation Method for Simulation-based Optimization

Ho Young Kim, Jun Kyung Kim, Yeong Geol Kim, Tag Gon Kim
Systems Modeling & Simulation Laboratory
Department of EECS, KAIST

본 논문에서는 시뮬레이션 기반 최적화에서 유전자 알고리즘을 이용하여 후보 모델을 자동으로 생성하는 기법을 제안하였다. 이 방법론은 잘 알려진 계획-생성-평가의 틀을 기반으로 구축되었다. 계획은 확장된 AND-OR 트리(AND, OR, Multiple AND 노드를 갖는 트리)를 이용하여 가능한 모든 후보 모델을 표현하였고, 이러한 트리상에서 후보 모델을 자동생성하기 위하여 유전자 알고리즘을 사용하였다. 마지막으로 생성된 후보 모델을 평가하기 위하여, 시뮬레이션을 수행하였다. 시뮬레이션을 이용한 평가를 통하여 목적에 맞는 후보 모델을 찾을 수 있게 된다. 본 논문에서 제시한 방법론의 효율성은 DSP 프로세서 설계 예제를 통하여 보여주었다.

1. 서론

가능한 후보 모델로부터 요구 조건을 만족하는 후보를 찾는 일은 여러 분야에서 사용되는 일이다. 차량 운송 모델이나 네트웍 라우팅 또는 DSP(Digital Signal Processing)[2][3] 프로세서 설계를 위한 명령어 선택과 같은 경우에 이러한 절차가 필요하게 된다. 후보 모델의 수가 적은 경우에는 사람이 직접 후보 모델을 하나씩 평가해 보며 원하는 모델을 찾을 수 있으나, 위의 나열한 문제와 같은 경우에는 후보가 무수히 많기 때문에 이들을 하나씩 평가하는 것이 불가능하다. 따라서 후보해들을 잘 표현한 후에, 여러 가지 알고리즘을 이용하여 원하는 모델을 찾는 것이 필요하다.

후보 모델을 평가하기 위해서는 분석적인 방법과 시뮬레이션을 이용한 방법이 사용된다. 분석적인 방법은 수식을 푸는 것으로 모델을 평가할 수 있으나, 적용 가능한 범위가 모델을 수식으로 표현한 경우로 한정되며, 또한 매우 복잡한 수식은 해결 할 수 없다. 따라서 분석적인 방법을 사용할 수

없을 경우에는 시뮬레이션을 이용하게 된다.

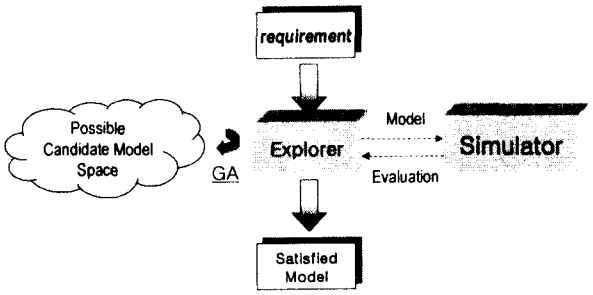

그림 57 전체 틀

본 논문에서는 확장된 AND-OR 트리를 이용하여 가능한 모든 후보 모델을 표현하고, 유전자 알고리즘을 사용하여 후보 모델을 생성하고, 시뮬레이션을 이용하여 후보 모델을 평가하고 다시 새로운 모델을 생성하는 방법으로 원하는 모델을 찾는 방법론을 제안하였다.(그림1) 2장에서 후보 모델 표현법을 기술하였고, 3장에서는 후보군 모델에서 최적화 후보모델 생성법을 나타내었다. 4장에서 시뮬레이션기반의 최적화를 설명하였고 5장에서 실제 예제로 DSP 프로세서 설계를 제안된 방법론을 통하여 보였고, 6장에서 결론 및 추후과제를 논한다.

2. 후보 모델 표현법

2.1 확장된 AND/OR 트리(EAT)

EAT는 최적화 문제에서 가능한 모든 후보 모델을 표현하기 위하여 도입된 도표이다. 이 도표는 세가지 관계와 각 노드의 특성을 표현하는 변수들로 구성된다.

$$EAT = <NS, AS, F_{NA}, R_{AND}, R_{OR}, R_{MAND}>,$$

각각은 다음과 같이 정의된다.

NS(Node Set) : 구성 요소를 나타내는 노드들의 집합

AS(Attribute Set) : 구성 요소의 속성들의 집합

F_{NA} : NS -> AS : 구성 요소에서 속성들로의 함수

$R_{AND} \subset NS \times NS$: 필요한 구성 요소들로 이루어진 AND 관계

$R_{OR} \subset NS \times NS$: 대안인 구성 요소들로 이루어진 OR 관계

$R_{MAND} \subset NS \times NS$: 같은 구성 요소로 이루어진 다중의 AND(MAND) 관계

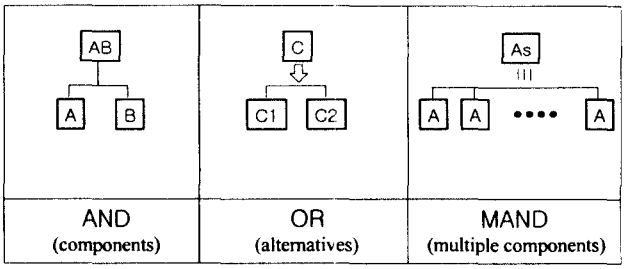

그림 58 EAT 관계들의 그래픽 표현

그림 2는 EAT의 관계들을 쉽게 알아 볼 수 있는 그래픽 표현이다.

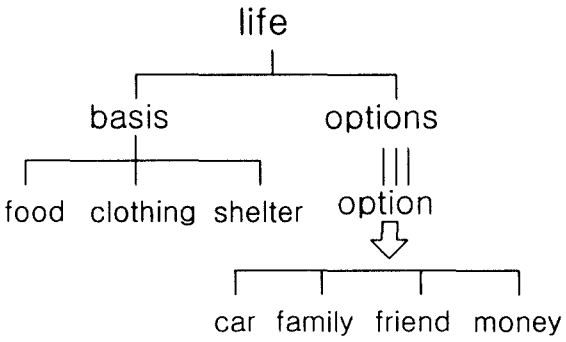

그림 59 EAT로 표현한 후보 모델 계획 예제

그림 3은 인생에서 필요한 것에 대한 예제이다. 가능한 모든 후보 모델을 표현한 것으로, 의식주는 반드시 필요하고, 차, 가족, 친구, 돈을 선택적으로 포함하는 경우를 나타낸다. 먼저 인생은 기본적인 것과 선택적인 것을 AND관계로 가지고 있어서, 두 가지를 반드시 포함한다. 다시 기본적인 것에는 의, 식, 주가 AND관계로 있다. 따라서 인생은 의, 식, 주 세 가지 모두를 반드시 포함하게 된다. 선택적인 것에는 MAND관계를 가지고 있어서, 수개의 선택적인 것을 갖을 수 있다. 그리고 각각은 차, 가족, 친구, 돈을 OR관계로 가지고 있어서 각각 하나씩 중복되지 않도록 선택할 수 있다. 따라

서, 이러한 EAT에서 가능한 총 후보수는 선택적인 것이 0개일 때(1), 1개일 때(4), 2개일 때(6), 3개일 때(4), 4개일 때(1)로 총 16가지가 된다.

3. EAT상에서 최적화 후보 모델 생성법

3.1 후보 모델 생성법

EAT는 가능한 모든 후보 모델을 표현하는 도구이다. 따라서, 모든 후보 중에서 하나의 후보를 생성해 내기 위해서는 각 관계에서 필요한 선택을 해주어야 한다. AND관계는 AND관계의 하위 노드에 있는 모든 성분을 다 가져야 하므로 선택이 따로 필요하지 않다. OR관계는 OR관계의 하위 노드에 있는 모든 성분 중에서 한가지가 선택되어야 한다. MAND관계는 MAND관계의 하위 노드의 개수를 정해준다. 이렇게 세 가지 관계에 대하여 처리하면, 하나의 후보 해가 생성된다.

그림 4는 그림 3의 인생에서 필요한 것에 대한 예제의 선택된 하나의 후보 모델을 보여준다. 그림 3의 options의 MAND관계는 2개의 하위 노드를 갖도록 되었고, 각각의 option은 family와 friend를 선택하여 나온 하나의 후보 모델이다.

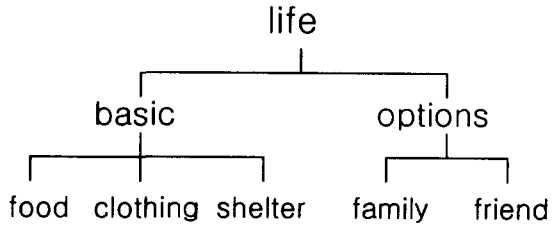

그림 60 선택된 하나의 후보 모델

3.2 유전자 알고리즘을 사용한 EAT 상에서의 최적화 후보 모델 생성법

유전자 알고리즘[4]은 자연선택과 정보상속에 기초한 탐색 방법 중의 하나이다. 유전자 알고리즘은 세 가지 연산자에 의하여 구현되는데, 각각은 재생산, 교차, 변이이다. 본 논문에서는 유전자 알고리즘을 EAT로 표현된 후보군에 적용하여 조건에 맞는 후보를 찾기 위하여, EAT상에서 재생산(reproduction), 교차(crossover), 변이(mutation)를 정의하였다.

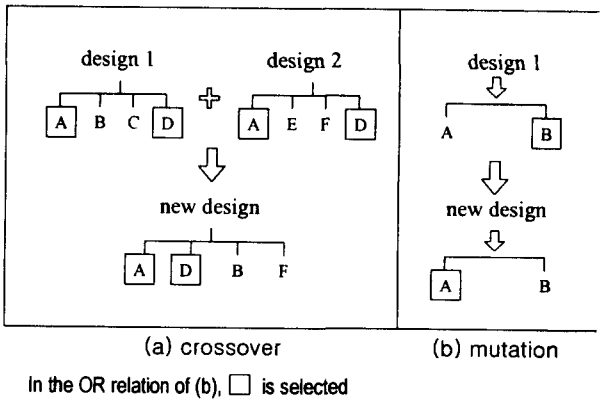

그림 61 EAT에 적용된 유전자 연산자

재생산은 먼저 구한 해의 집합들로부터 적합도가 높은 해를 다시 뽑아내는 것이다. 각 설계들을 시뮬레이션을 통해 평가하여 적합도를 구하고, 전 세대의 설계 중에서 적합도의 크기에 따라 확률적으로 설계를 뽑을 수 있도록 하였다. 교차는 두 개의 해를 합쳐서 새로운 하나의 해를 만들어 내는 것이다. 그림 5의 (a)에서 설계 1과 설계 2가 공통으로 갖는 구성 요소인 A와 D는 새로운 설계에서도 그대로 갖도록 하고 두 설계에서 서로 다른 부분에 대해서는 번갈아 가며 갖도록 하였다. 그 결과로 새로운 설계는 A, D, B, F를 구성 요소로 갖게 된다. 변이는 교차만으로 얻을 수 없는 추가적인 성질 변화를 주기 위하여 필요하다. 교차만을 사용한다면, 같은 성질만이 계속 유지되기 때문이다. 변이는 EAT상에서 OR관계를 통하여 구현되었다. 설계로부터 임의의 OR 관계를 선택하고 OR 관계의 선택된 구성요소를 변경한다. (그림 5(b))

4. 유전자 알고리즘을 사용한 시뮬레이션 기반의 최적화

최적화를 수행하기 위해서는 후보 모델을 평가하는 것이 필요하다. 시뮬레이션을 이용한 평가 방법은 시뮬레이션을 하는데 소요되는 시간으로 인해 분석적인 방법으로 계산을 하는 것보다 느리

지만, 분석적인 방법으로 계산하지 못하는 복잡한 경우에 적용할 수 있다. 따라서, 적용할 문제에 따라 분석적인 방법 또는 시뮬레이션 기반의 최적화를 수행하는 것이 필요하다.

본 논문에서 제안한 전체 틀은 그림 6과 같이 시뮬레이션을 기반으로 한 최적화를 수행한다. EAT를 이용하여 전체 후보 모델들을 계획(plan)하고, 유전자 알고리즘(GA)을 이용하여 후보 모델을 생성(generation)한다. 그 후, 분석적인 방법 또는 시뮬레이터를 이용하여 그 모델을 평가(evaluation)하고 새로운 후보 모델 생성에 참고가 되게 한다.

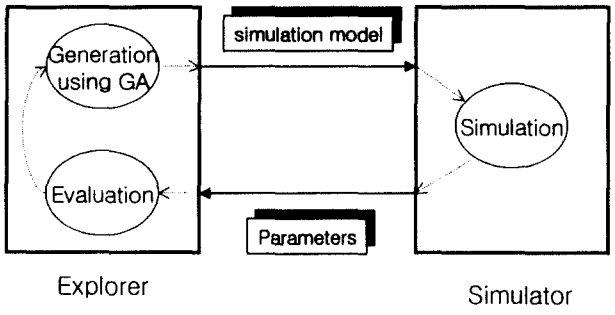

그림 62 시뮬레이션 기반의 최적화

5. 예제 : DSP 프로세서 설계

본 논문에서 제안한 방법론의 유용성을 입증하기 위하여, DSP 프로세서 설계에 제안된 방법론을 적용하였다. DSP 프로세서 설계 및 후보 설계에 대한 평가를 위해서는 프로세서 구조에 따라 컴파일 및 시뮬레이션을 할 수 있는 재표적화 컴파일러/시뮬레이터(Retargetable Compiler/Simulator)가 사용된다.[5][6]

5.1 DSP 프로세서의 후보 모델 표현

DSP 프로세서를 추상화하면, 각각의 물리적인 부분들을 명령어 집합, 파이프라인 구조로서 나타낼 수 있다. DSP 프로세서의 명령어 집합을 EAT 형태로 나타내면 그림 7과 같이 나타낼 수 있다. 명령어 집합은 크게 기본 명령어들과 선택 명령어들로 나뉜다. 기본 명령어는 명령어 집합에 반드시 포함되는 명령어들이고, 선택 명령어들은 EAT의 관계에 따라서 선택적으로 포함되는 명령어들이다. 기본 명령어는 다시, 산술논리연산(alu) 명령어, 제어(control) 명령어, 이동(move) 명령어들로 AND 관계로 이루어져 있고, 각각은 다시 세부 명령어들로 AND관계로 이루어져 있다. 따라서 기본 명령어들은 모두 반드시 포함된다. 선택명령어는 MAND관계를 가지고 있어서 수 개의 선택 명령어

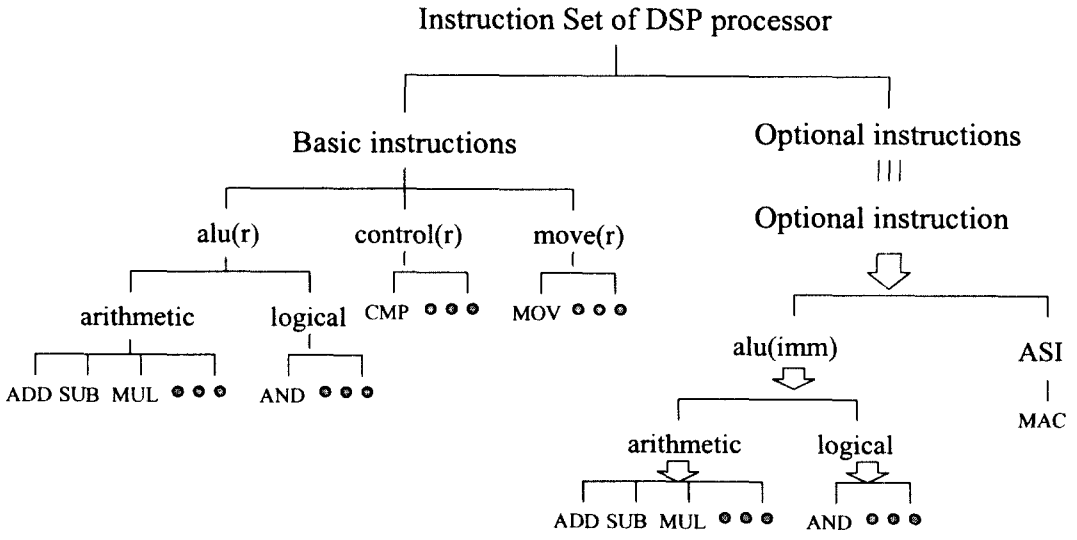

그림 63 명령어 집합의 후보에 대한 EAT 표현

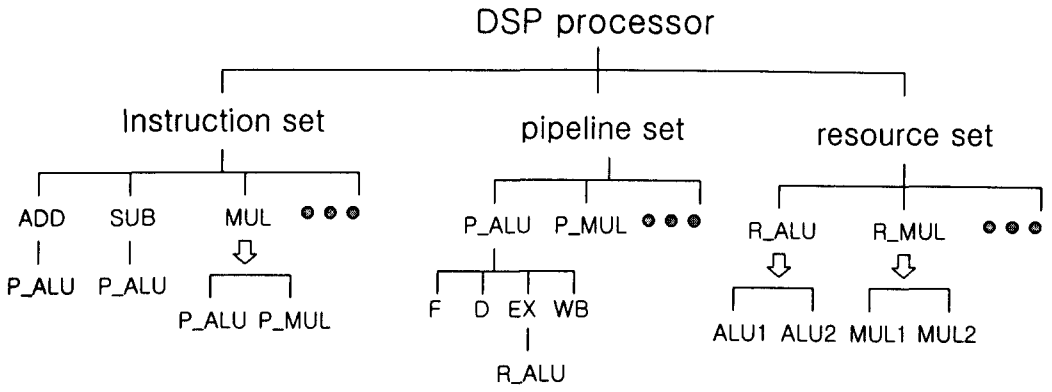

그림 64 전체 DSP 프로세서의 후보에 대한 EAT 표현

들을 갖는다. 그리고 각각의 선택 명령어는 OR관계로 산술논리연산 명령어와 특정응용분야 명령어를 가지고 있어서 둘 중 하나를 선택할 수 있게 하였다. 다시, 산술논리연산 명령어는 세부 명령어들로 OR관계를 가지고 있어서 세부 명령어들 중 하나가 선택될 수 있게 하였다. 결국 선택 명령어는 선택된 수 개의 세부명령어가 된다. 각 세부 명령어들은 수행되는 데 필요한 싸이클 정보, 명령어 크기 등을 속성으로 갖는다. 가능한 총 후보 모델을 수는 2^(선택 명령어) = 2^{13} ≒ 8000 가지이다.

파이프라인 구조를 포함한 DSP 프로세서의 EAT는 그림 8과 같다.

DSP 프로세서는 명령어 집합과 파이프라인 구조, 자원의 집합들을 AND관계로 갖는다. 명령어 집합은 앞에서 선택된 명령어들을 AND관계로 가지고 있고, 각 명령어들은 사용할 수 있는 파이프라인을 OR관계로 가지고 있어서 선택할 수 있게 하였다. 파이프라인 구조는 파이프라인들(ALU 파이프라인, Multiplier 파이프라인 등)을 AND관계로 가지고 있으며, 각 파이프라인은 스테이지와 해당 자원을 AND관계로 갖는다. 자원 집합은 다른 함수성을 갖는 자원들을 가지고 있으며, 각 자원들은 서로 다른 면적, 싸이클, 파워 속성들을 갖는 자원들을 OR관계로 가지고 있어서 선택되도록 하였다. 따라서, 가능한 총 후보 모델의 수는 2^{10} ≒ 1000 가지이다.

5.2 후보모델 생성 및 평가

후보모델 생성 및 평가는 두 단계에 걸쳐서 이루어진다. 명령어 집합에 대한 EAT상에서 먼저 최적화가 이루어진 후에, 전체 DSP 프로세서 구조에 대하여 최적화가 이루어진다.

명령어 집합에서는 명령어들을 결정한다. 그림에서 EAT는 이러한 설계 공간을 나타내고 있다. AND 관계를 통하여, 명령어 집합은 기본 명령어들과 선택 명령어로 나뉘어지고 각각은 다시 여러 세부 명령어들로 나타내어진다. 선택 명령어는 다중 AND 관계와 OR 관계를 통하여 수 개의 선택 명령어들이 선택될 수 있도록 하였다.

설계 결정은 명령어 결정에서 주로 사용되는 Holmer의 k% 법칙[7]을 사용하였다. 추가된 명령어는 k%의 성능향상을 가져와야 하고 그렇지 않으면 그 명령어는 거부된다. 이 법칙은 다음과 같은 수식으로 나타난다.

$$100\Delta S/S + k*I<0$$

여기서, S는 컴파일된 코드 크기이며, I는 명령어의 개수가 된다. 유전자 알고리즘에 사용할 목적함수(cost function)는 이 수식을 극한으로 정적분한 형태가 된다.

$$100\ln S + k*I$$

설계 평가를 하기 위하여 재표적화 컴파일러가 사용되었다. 재표적화 컴파일러는 설계 공간으로부터

선택된 하나의 설계에 따른 컴파일러를 만들어 내고, 벤치마크 알고리즘을 컴파일하여 어셈블리 코드를 만들어 준다. 이렇게 생성된 어셈블리 코드로부터 목적 함수를 계산할 수 있게 된다. 설계 생성 및 평가는 코드 크기와 명령어의 개수를 만족할 때까지 반복하여, 원하는 메모리 사용량을 갖는 설계를 찾을 수 있다.

전체 DSP 프로세서에서는 선택된 명령어 집합을 기초로 하여, 그림 8의 후보 모델로부터 파이프라인 구조, 스테이지와 자원의 할당을 결정한다. 고려될 성능지수들은 다음과 같다.

싸이클 횟수(C) : 벤치마크 알고리즘을 수행하는데 걸리는 총 싸이클 수로서 시뮬레이션을 통하여 알아낸다.

면적(A) : 파이프라인 구조를 구축하는데 사용되는 모든 자원의 면적의 합이다. 전체 면적은 모든 자원들의 면적을 합쳐서 구한다.

$$A \quad \oint_{RS} Area_R$$

파워 소모(P) : 벤치마크 알고리즘을 수행하는데 요구되는 총 파워 소모이다. 이 값은 각 자원들의 명령어별 평균 파워소모가 주어져 있는 경우에 계산 가능하다.

$$P \quad \oint_{RS} Power_R \cup Usage_R$$

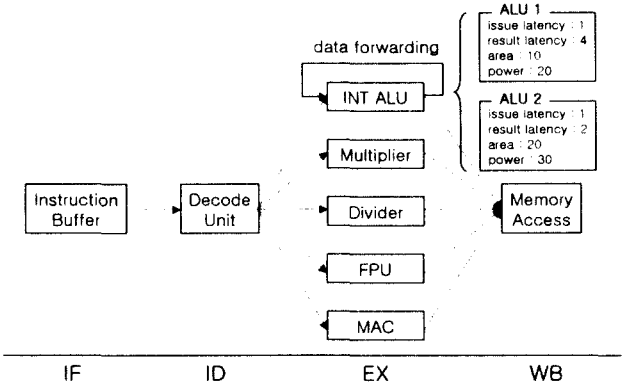

그림 9 파이프라인 구조

그림 9는 파이프라인 구조와 자원에 대한 한 가지 예를 보여준다. 파이프라인 구조는 스테이지로 구성되어 있으며, 각 스테이지는 수 개의 자원들을 가질 수 있다. 자원을 많이 사용되면, 파이프라인 지연을 막아서 수행시간을 단축시킬 수 있는 반면에 면적이 커지는 단점이 있다. 같은 작용을 하는 자원라 하더라도 그림 9의 ALU1과 ALU2와 같이 레이턴시, 면적, 파워소모가 다를 수 있어서, 이중에 적절한 것을 선택해야 한다.

k% 법칙을 이용하여 목적함수를 구하면, 다음과 같이 된다.

$$100\Delta C/C + k\Delta A/A + l\Delta P/P<0$$
$$cost\ function = 100\ln C + k\ln A + l\ln P$$

k와 l 값을 조절하여 싸이클 횟수, 면적, 파워소모 사이의 가중치를 부여할 수 있다. 벤치마크 알고리즘을 수행하는데 걸리는 총 싸이클 수와 파워소모를 계산하기 위하여 재표적화 컴파일러/시뮬레이터가 사용되었다. 컴파일된 어셈블리 코드를 이용하여 시뮬레이터는 총 수행 싸이클과 각 명령어별 사용빈도를 알려준다.

5.3 실험 결과

DSP 프로세서의 모델을 가능한 후보 모델 중에서 찾기 위해서는 설계 목적이 주어진다. 본 실험에서 사용한 설계 목적은 표 1과 같다. 또한 DSP 프로세서 상에서 수행될 벤치마크 알고리즘은 이차원 영상 저대역 통과 필터링 알고리즘을 사용하였다.

표 24 예제의 설계 목적

# of instruction	Code size	Area	Power	Total Cycle
≤34	≤5500	≤45	≤30000	≤75000

그림 10은 예제의 실험 결과이다. 그림 10의 (a)와 (b)는 명령어 개수와 코드 크기를 고려한 명령어 집합에서의 시뮬레이션 기반 최적화의 결과이고 (c), (d), (e)는 면적, 파워소모, 총 싸이클을 고려한 전체 DSP 프로세서에서의 시뮬레이션 기반 최적화의 결과이다. 실험을 위하여 사용된 파라미

터는 표 2와 같이 설정해 주었다.

표 25 실험에 사용된 파라미터

k	l	population	mutation rate
20	5	8	0.3

k와 l 값으로부터 총 싸이클, 면적, 파워 소모에 대하여 가중치를 주었고, 유전자 알고리즘의 한 세대의 크기는 8로 하였다. 또한 변이가 일어날 수 있는 비율을 0.3으로 주었다.

명령어 집합의 후보 모델 탐색은 21세대(21*8=168 반복)만에 8000개의 설계 중에서 원하는 설계를 찾았고, DSP 프로세서의 후보 모델 탐색은 9세대(9*8=72 반복)만에 1000개의 설계 중에서 원하는 설계를 찾을 수 있었다. 소요된 CPU 시간은 펜티엄2 400MHz에서 6분이었다.

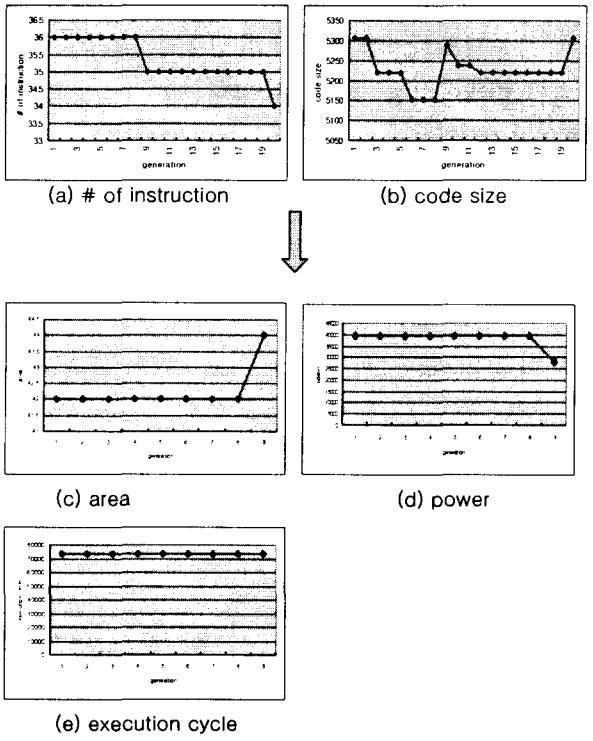

그림 10 DSP 프로세서 예제의 실험 결과

5 결론

많은 후보 해를 가지는 문제에서 효율적으로 후보 해들을 계획하고 탐색하는 것은 시간과 노력을 줄일 수 있다는 점에서 매우 중요하다. 본 논문에서는 효율적인 후보 모델 공간을 탐색하기 위한 모델 생성 기법을 제안하였다. 계획-생성-평가 틀을 기반으로 EAT를 이용한 후보 모델 공간을 구축하였고, 유전자 알고리즘을 사용하여 생성된 후보 모델을 시뮬레이션을 이용하여 평가하였다. 또한 DSP 프로세서 예제를 통하여 제안된 방법론의 유용성을 보여주었다.

앞으로 제안된 방법론을 차량 운송 경로 문제 또는 네트웍 라우팅과 같은 다른 분야에 적용할 수 있을 것이다.

참고 문헌

[1] B. P. Zeigler, H. Praehofer, and T. G. Kim, *Theory of Modeling and Simulation*, ACADEMIC PRESS, 2000.

[2] E. A. Lee, "Programmable DSPs: A brief overview," *IEEE Micro*, vol. 10, pp.14-16, 1990.

[3] P. Lapsley, J. Bier, A. Shoham, and E. A. Lee, *DSP processor Fundamentals*, IEEE PRESS, 1997

[4] D. E. Goldberg, *Genetic Algorithms*, Addison-Wesley, 1989.

[5] Target Compiler Technologies, *CHESS/CHECKERS: A Retargetable DSP compilation Environment*, Target Compiler Technologies, Dec. 1999.

[6] J. Yang *et al.*, "MetaCore: An Application-Specific Programmable DSP Development System," *IEEE Trans. VLSI System*, vol. 8, no. 2, pp.173-183, April 2000

[7] B. Holmer and A. M. Despain, "Viewing instruction set design as an optimization problem," in *Proc. 24th Int. Conf. Microarchitecture*, 1991.