재구성형 시스템을 위한 하드웨어/소프트웨어 분할 기법

김 준 용[†] · 안 성 용[††] · 이 정 아[†††]

요 약

본 논문에서는 재구성 가능한 시스템에서 Y-chart 설계공간 탐색 기법을 응용하여 하드웨어 소프트웨어 분할문제를 해결하기 위한 방법론을 제시하고 이 방법에 기초하여 성능분석 도구를 개발하였다. 이 방법론은 어플리케이션모델의 각 Task들로부터 범용프로세서나 FPGA와 같은 하드웨어 요소들로의 사상의 경우들을 생성하고 각각의 사상의 경우에 대한 시뮬레이션을 수행하여 시스템의 성능을 평가한다. 시뮬레이션 결과로 산출된 처리율에 기초하여 가장 좋은 성능을 산출하여 사상의 경우를 선택할 수 있다. 본문에서는 또한 시뮬레이션 속도를 향상시키기 위하여 작업량과 병렬성과의 관계에 기초하여 사상집합의 크기를 줄이는 휴리스틱 알고리즘을 제안한다. 제안된 사상집합 축소 휴리스틱을 적용한 시뮬레이션 결과 사상집합의 크기를 80%가량 줄일 수 있었다.

Hardware/Software Partitioning Methodology for Reconfigurable System

Jun Yong Kim[†] · Seong Yong Ahn[††] · Jeong A Lee[†††]

ABSTRACT

In this paper, we propose a methodology solving the problem of the hardware-software partitioning in reconfigurable systems using a Y-chart design space exploration and implement a simulator according to the methodology. The methodology generates a mapping set between tasks and hardware elements using the hardware element model and the application model. We evaluate the throughput by simulating cases in each mapping set. With the throughput evaluation result, we can select the mapping case with the highest throughput. We also propose an heuristic improving the simulation time by reducing the mapping set on the basis of the relationship between workload and parallelism. Simulation results show that we can reduce the size of mapping set which poses difficulties on hardware-software partitioning by up to 80%.

키워드 : 재구성 가능 시스템(Reconfigurable System), 내장형 시스템(Embedded System), FPGA, 하드웨어-소프트웨어 분할(Hardware-Software Partitioning), 설계공간탐색(Design Space Exploration), Y-chart

1. 서 론

특정 계산을 수행하는데 가장 높은 성능(속도, 처리율)을 내도록 하려면 그에 특화된 하드웨어를 제작하는 것이며 이를 위해 지금까지는 주로 ASIC(Application Specific Integrated Circuit)을 사용하였다. 그런데 FPGA(Field Programmable Gateway Array)와 같이 재구성 가능한 장치(reconfigurable device)들이 발전하여 예전에는 ASIC으로 구현해야 했던 기능(function)들을 재구성 가능한 장치들에서도 구현할 수 있게 되었으며 같은 기능을 구현하였을 때 ASIC만큼은 아니지만 범용 프로세서들보다는 높은 성능을 낼 수 있게 되었다. 그리고 재구성 가능한 장치들은 고정된 기능을 수행해야 했던 ASIC과는 달리 다양한 기능을 구성(configure)하여 사용할 수 있기 때문에 범용 구조(general-purpose architecture)가 가지는 유연성을 지닌다[11]. 특히 FPGA는 셀 크기 제한 내에서 다양한 기능들을 동시에 구성해서 사용할 수 있다는 장점을 지니며 재구성 가능한 시스템은 주로 이러한 FPGA를 여러 개 사용해서 구성한다[1].

이러한 성능과 유연성을 동시에 지닌 재구성 가능한 시스템에 대한 연구가 근래에 들어 활발히 진행되고 있으나 현재로서는 재구성 가능한 시스템을 설계하고 분석하는 방법이나 도구들이 충분하지 않으며 재구성 가능한 시스템을 설계하고 하드웨어-소프트웨어 분할을 하는 것은 아직까지 어려운 문제로 남아 있다[2].

재구성 가능한 시스템은 재구성 가능한 장치들을 응용 프로그램에 맞게 구성하여 시스템의 성능을 높이는 것을 목적으로 하는 정적 재구성 가능 시스템과, 시스템의 실행 환경이나 상태에 따라 시스템 자체를 재구성하는 동적 재구성 가능 시스템으로 나누어진다[3]. 정적 재구성 가능 시스템은

† 정 회 원 : 한국마이크로소프트 개발팀
†† 준 회 원 : 조선대학교 대학원 전자계산학과(교신저자)
††† 정 회 원 : 조선대학교 컴퓨터공학과 교수
논문접수 : 2004년 5월 10일, 심사완료 : 2004년 8월 24일

응용 프로그램을 하위 작업(subtask)들로 나누어 계산을 수행하는데, 나누어진 하부 작업들과 시스템 구성 요소들과의 대응 시점에 따라 두 가지 시스템 구성 정책을 가진다. 수행 전 시스템 구성 정책은 응용 프로그램의 수행 전에 하부 작업과 하드웨어 구조에 대한 관계를 분석하여 시스템을 구성한 다음 프로그램을 수행한다. 이와는 반대로 수행 중 시스템 구성 정책은 응용 프로그램의 수행 도중에 하부 작업이 필요로 하는 기능을 재구성 가능한 장치들에 구성하여 프로그램을 수행한다.

본 논문에서는 설계 공간 탐색 방법의 하나인 Y-chart 설계 공간 탐색 기법[4]을 응용하여 수행 전 시스템 구성 정책을 가지는 정적 재구성 가능 시스템에서 발생하는 하드웨어-소프트웨어 분할 문제를 푸는 방법을 제시하고 이 방법을 사용하여 도구를 제작하였다. 이 방법은 먼저 하드웨어 요소와 응용 프로그램을 모델링한 결과를 이용하여 응용 프로그램의 기능들이 하드웨어 요소에 속하는 주 프로세서나 FPGA들에 어떻게 사상(mapping)될 수 있는지를 나타내는 사상 집합(mapping set)을 생성한다. 사상 집합을 생성할 때에 응용 프로그램의 여러 부분이 동일한 기능을 수행하게 될 경우에 발생할 수 있는 자원 요구 충돌 문제까지 고려한다. 그 다음 생성된 사상의 경우(mapping case)들을 시뮬레이션 하여 각각의 사상의 경우에 시스템이 얼마의 처리율을 가지는지를 보여준다. 각 사상의 경우에 대한 처리율을 비교해봄으로서 주어진 시스템에서 응용 프로그램이 가장 높은 처리율을 낼 수 있도록 하기 위해 응용 프로그램의 어떠한 기능들이 FPGA들로 구성해야 하는가를 알 수 있다.

추가적으로 본 논문에서는 이 방법을 사용할 경우에 발생하는 사상 집합의 크기 문제를 해결하기 위해 사상 집합을 축소하는 휴리스틱 알고리즘을 제시한다. 응용 프로그램이 수행하는 기능 중에서 범용 프로세서와 FPGA들 중에서 선택할 수 있는 기능의 개수가 i이고 각 기능을 실행할 수 있는 프로세서의 수가 f_i일 때 사상 집합의 크기는 Πf_i로 주어진다. 예를 들어, 응용 프로그램의 기능 중에 8개가 FPGA에서 동작할 수 있고, 시스템이 1개의 범용 프로세서와 3개의 FPGA를 가질 때 나타나는 사상 집합의 크기는 최대 $8^4=4,096$이 된다. 이러한 사상 집합의 크기는 시스템의 복잡도에 따라 급수로 증가하여 시뮬레이션 시간에 심각한 영향을 미치기 때문에 시뮬레이션을 하지 않아도 되는 사상 집합을 제외할 필요가 있다. 그렇게 하기 위해, 시뮬레이션 이전에 각 사상의 경우에서 식별할 수 있는 두 가지 요소인 작업량과 병렬성이 FPGA를 사용하는 재구성 가능한 시스템의 처리율과 가지는 관계를 근거로 사상 집합을 축소하고, 그 결과 최적의 시스템 구성을 찾아내기 위해 Y-chart 설계 공간 탐색으로 소요되는 시뮬레이션 시간이 줄어든다.

본 논문의 구성은 다음과 같다. 2장에서는 기존의 관련 연구를 요약하였으며 3장에서는 Y-chart 설계 공간 탐색 기법을 응용한 시뮬레이터에 대해서 설명한다. 4장에서는 사상 집합을 어떻게 축소하여 시뮬레이션 시간을 줄이는지를 기술한다. 5장에서는 구현된 분할 도구에서 사상 집합 축소를 수행하여 축소 방법으로 적절한가와 사상 집합을 얼마나 줄일 수 있었는가를 보이고 6장에서 결론을 내린다.

2. 관련 연구

기존의 하드웨어-소프트웨어 분할에 대한 연구는 기존의 ASIC 설계나 병렬 처리 시스템 설계 분야에서 이루어져 왔다. 여기서는 하나의 응용 프로그램의 기능을 주어진 시스템의 하드웨어 자원에 적절히 배분하여 수행하도록 나누는 기법들과 비용이나 처리율과 같은 제약 조건들을 만족하는 설계 방법을 찾는 기법들이 연구되어 재구성 가능한 시스템을 설계하는데 참고할 수 있지만 FPGA와 같이 설계 자유도가 높은 재구성 가능한 장치를 적절히 이용하기 위해서는 발전된 연구가 필요하다. 따라서 최근에 재구성 가능한 시스템에 대한 하드웨어-소프트웨어 통합 설계에 대한 연구가 많이 수행되었다.

Lee등은 다양한 프로그램의 수행이 요구되는 PDA(Personal Digital Assistant)와 IMT2000(International Mobile Telecommunication) 터미널과 같은 다중 응용 내장형 시스템을 설계하는 데 적합한 재구성 가능한 시스템의 하부 구조와 하나의 FPGA에 여러 기능이 수행 시간에 구성되는 모델을 제시하였다[5].

Keinhuis는 Y-chart 설계 공간 탐색 방법을 제안하였다[4]. 이 방법은 설계자가 응용 프로그램이 사상될 시스템 구성 요소에 대한 중요한 성능 척도를 정량적으로 정하도록 한다. 그는 이 방법을 사용하여 ORAS라고 하는 재구성 가능한 시뮬레이터를 만들었다[6]. ORAS는 시뮬레이션을 통해 시스템 구성 요소들을 이용해서 만들 수 있는 시스템의 구성에 대한 성능 수치(performance number)를 얻는다. 하지만 ORAS는 SBF라고 불리는 특정한 계산 모델에 한정되어 있기 때문에 재구성 가능한 시스템을 시뮬레이션하기 위해서는 ORAS와는 다른 하부 구조가 필요하다.

Bakshi 등은 시스템 수준의 명세와 처리율이 제한된 경우에 하드웨어-소프트웨어를 분할하기 위해 하드웨어와 소프트웨어를 모델링하고 파이프라인의 단계 형태로 나누어 시뮬레이션 하는 방법을 사용했다[7].

Kalavade등은 다양한 응용을 수행하는 내장형 시스템을 설계하는 문제를 지적했다[8]. 하지만 이 연구에서는 FPGA에서의 분할은 고려되지 않았다.

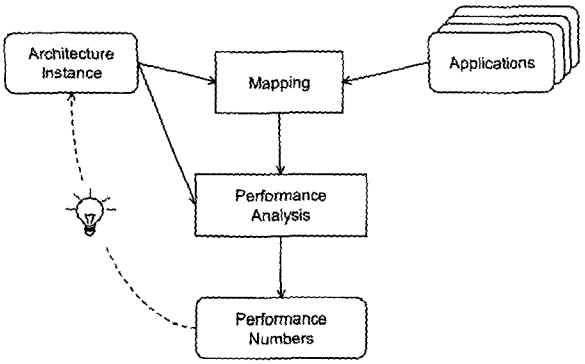
Pruna등은 재구성 가능한 시스템에서 응용 프로그램을 재구성 가능한 장치들의 크기에 맞게 하위 작업들로 분할하고

자료 흐름 그래프(data flow graph)를 스케줄 한다[9]. 하지만 이 연구에서는 하나의 FPGA에서 여러 기능이 수행되는 경우는 고려하지 않고 있다.

3. 분할 도구

3.1 Y-chart 기법

분할 도구에서 사용한 Y-chart 설계 공간 탐색 방법은 Kienhuis등이 1997년에 제안한 방법이다[4]. 이 방법은 주어진 응용 프로그램과 시스템 구성 요소들로 구성할 수 있는 시스템에 대한 정량적인 성능 평가를 할 수 있는 방법이다. Y-chart 방법은 (그림 1)에서와 같이 수행된다. 먼저 설계자는 특정한 시스템 구성 요소(architecture instance)들을 식별하고 성능 평가(performance analysis)를 수행하여 성능 모델을 만든다. 이 성능 모델과 주어진 응용 프로그램을 사상(mapping)하여 성능 분석을 수행함으로 그 사상의 경우에 얼마의 성능을 가지는지에 대한 성능 수치(performance number)를 얻는다. 이 과정은 주어진 응용 프로그램에 대해 만족할 만한 성능 수치를 얻을 수 있을 때까지 반복할 수 있다. Y-chart 기법을 이용하면 나타날 수 있는 다양한 시스템 구성 방법에 대해서 정량적인 평가를 할 수 있기 때문에 재구성 가능한 시스템에서 응용 프로그램이 수행될 경우 하위 작업들이 수행하는 다양한 기능들이 어떤 하드웨어 자원을 사용할 때 가장 좋은 성능을 낼 수 있는가를 판단하기가 용이하다.



(그림 1) Y-chart 설계 공간 탐색 방법

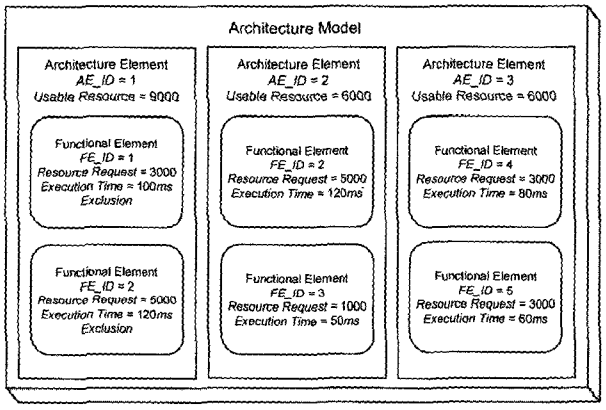
3.2 하드웨어-소프트웨어 분할 도구의 설계

재구성 가능한 시스템에서 하드웨어-소프트웨어 분할을 수행하기 위해 Y-chart 기법을 응용하여 도구를 설계하였다. 이 도구는 수행 전에 응용 프로그램을 하위 작업들로 나누고 각 작업들이 요구하는 기능과 이를 수행할 재구성 가능한 시스템에서 어떻게 구성할 것인지를 분석하고 구성 방법에 따른 처리율을 보임으로써 설계자가 최적의 구성 방법을 결정하게 한다.

〈표 1〉 하드웨어 모델과 응용 프로그램 모델

Architecture Model	Application Model
Architecture := List_of_AE List_of_AE := AE | List_of_AE AE AE := AE_ID Usable_Resources List_of_FE List_of_FE := FE | List_of_FE FE FE := FE_ID Resource_Request Execution_Time Exclusion	Application := List_of_PE List_of_PE := PE | List_of_PE PE PE := PE_ID Next_PE_list FEQ_list Next_PE_list := PE_ID | Next_PE_list PE_ID FEQ_list := FEQ | FEQ_list FEQ

Y-chart 기법에서와 같이 하드웨어 모델과 응용 프로그램의 모델을 사용하였으며 각각에 대한 모델은 <표 1>과 같이 주어진다. 하드웨어 모델(Architecture Model)은 시스템을 구성하는 하드웨어 구성 요소(Architecture Element ; AE)들로 이루어지며 이는 범용 프로세서나 FPGA와 같은 프로세서 단위를 나타낸다.

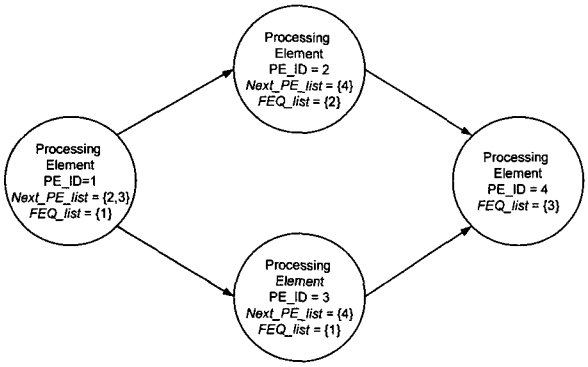


(그림 2) 하드웨어 모델의 예

하드웨어 구성 요소는 각각의 식별자(AE_ID)를 가지고 있고 FPGA의 경우에는 최대로 사용 가능한 셀의 개수(Usable_Resource)를 명시한다. 하드웨어 구성 요소는 기능 요소(Function Element ; FE)들을 가진다. 기능 요소는 FPGA에서 수행될 때 요구하는 FPGA의 셀 개수인 자원 요구량(Resource_Request)과 자신이 속한 하드웨어 구성 요소에서 수행될 때 데이터 단위(data unit)당 소요되는 수행 시간(Execution_Time)으로 구성된다. 한 FPGA에 여러 기능이 구성되어도 이들은 서로의 수행 시간에 영향을 미치지 않지만 범용 프로세서의 경우는 영향을 미친다. 이에 따라 하부 작업을 모델링하기 위해 각 데이터 단위는 범용 프로세서가 수행하는 각 기능이 preemption되지 않고 처리하는 데이터의 크기로 정하고 범용 프로세서의 각 기능 요소에 대해 배제(exclusion)를 넣도록 했다. (그림 2)에서는 각각 2개의 기능 요소와 3개의 구조 요소를 가지는 하드웨어 모델의 예를 보여준다.

응용 프로그램(Application)의 모델은 응용 프로그램을 구성하는 하위 작업(Process Element ; PE)들과 그 작업들간

의 자료 흐름의 방향(Next_PE_list)으로 표현한다. 이 때 작업의 흐름은 루프나 궤환(Feedback)이 없도록 모델링한다. 하위 작업들은 각각의 식별자(PE_ID)를 가지고 어떤 기능을 수행해야 하는지(FEQ_list)를 명시한다. (그림 3)에서는 4개의 하위 작업을 가지는 응용 프로그램 모델의 예를 보여준다.



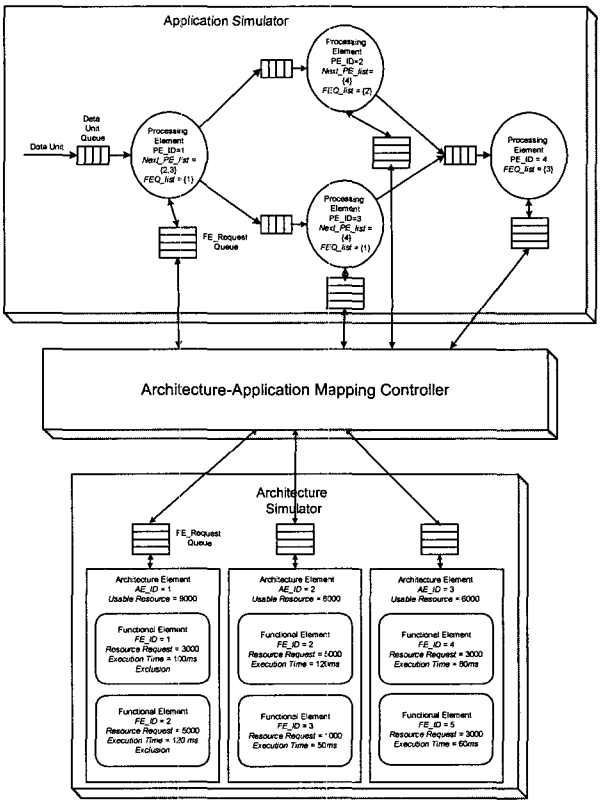
(그림 3) 응용 프로그램 모델의 예

Y-chart 기법에서의 사상은 이 도구에서는 이렇게 주어진 하드웨어 모델과 응용 프로그램의 모델을 이용하여 사상 집합(mapping set)을 구함으로 구현된다. 사상 집합은 응용 프로그램의 하위 작업들이 어떤 기능들을 사용해야 하고 어떤 하드웨어 구성 요소가 그 기능을 수행할 수 있는지를 식별한 다음 그에 따른 사상의 경우(mapping case)들을 생성함으로 구한다.

이 도구는 이렇게 구한 사상 집합에 속하는 각 사상의 경우에 대해 응용 프로그램의 여러 부분이 동일한 기능을 수행할 것을 동일한 하드웨어 요소에 요청하게 될 경우에 발생할 수 있는 처리 요구 충돌 문제까지 고려하여 시뮬레이션 한다. 시뮬레이션은 하드웨어 구성 요소의 처리 시간을 계산하는데 사용된 데이터 단위(data unit)들을 응용 프로그램의 시작에서 끝까지 흘려 보냄으로 이루어진다. 설계자가 정한만큼의 데이터 단위를 처리한 뒤 각각의 사상에 대한 시뮬레이션의 결과로 그 사상의 경우에 대한 성능 수치를 낸다. 성능 수치는 처리율, 각 데이터 단위가 시스템 내에서 머무른 시간, 처리 요구의 충돌로 대기해야 하는 시간, 병렬성 그리고 각 하드웨어 구성 요소들의 사용 율로 나타낸다. 이러한 성능 수치에 따라 설계자는 어떤 사상의 경우가 적합한지 판단한다.

3.3 하드웨어 소프트웨어 분할 도구의 구조

분할 도구는 응용 프로그램의 모델을 받아 구성하고 데이터 단위의 흐름을 시뮬레이션 하는 응용 시뮬레이터(Application Simulator)와 하드웨어에 데이터 단위에 대한 기능 수행 요구가 들어 왔을 때에 대한 수행을 시뮬레이션 하는 하드웨어 구조 시뮬레이터(Architecture Simulator) 그리고 응용 프로그램과 하드웨어 구성 요소가 어떻게 사상되는지에 대한 정보를 담고 각 사상의 방법에 대해 응용 프로그램이 보내는 데이터 단위 처리 요구를 해당하는 하드웨어 구성 요소로 보내 주는 하드웨어-소프트웨어 사상 제어기(Architecture-Application Mapping Controller)로 구성된다. (그림 4)에서는 앞에서 예로 보인 하드웨어 모델과 응용 프로그램 모델의 예를 가지고 하드웨어-소프트웨어 분할을 수행하는 시뮬레이터를 보여준다.



(그림 4) 하드웨어-소프트웨어 분할 도구의 구조

응용 시뮬레이터는 데이터 단위의 흐름을 시뮬레이션하며 주어진 응용 프로그램 모델의 하위 작업들간에 데이터 단위가 이동하도록 한다. 응용 시뮬레이터 내에서 하위 작업들은 프로세스(process)로 표현되며 이들은 전달된 데이터 단위에 대해 자신이 수행하는 기능 요구를 더하여 응용 시뮬레이터를 통해 하드웨어-소프트웨어 사상 제어기로 데이터 단위를 전달한다. 하드웨어 구조 시뮬레이터에서 처리가 끝나서 하드웨어-소프트웨어 사상 제어기를 통해 응용 시뮬레이터로 되돌아온 자료 단위는 처리를 요청한 프로세스에게 되돌려 주고 이를 받은 프로세스는 다음 프로세스에게 이 데이터 단위를 넘겨준다. 응용 시뮬레이터는 프로세스가 처리할 자료 단위를 생성해서 처음 프로세스에 넣어주고 마지막 프로세스까지 거쳐서 나온 데이터 단위를 처리하는데 걸린 시간과 시스템 내에서의 대기 시간 등에 대한 자료를 모아 각 사상의 경

우에 대한 시뮬레이션이 끝났을 때 성능 수치를 계산하는 근거로 사용한다. 이러한 데이터 단위의 이동과 처리 요청이 동시에 여러 개가 발생할 경우 선입선출 대기 행렬(queue)를 사용하여 처리 순서가 될 때까지 대기하도록 한다.

하드웨어 구조 시뮬레이터는 데이터 단위의 처리를 시뮬레이션하며 각 데이터 단위에 대하여 처리하는데 얼마의 시간이 걸렸는지를 계산한다. 프로세스에서 처리를 요구한 자료 단위가 하드웨어-소프트웨어 사상 제어기를 통해서 어떤 하드웨어 구성 요소의 기능을 사용해야 하는지가 정해져서 전달되면 이를 받아서 기능 요소에서 처리하는 시간 동안 이 기능 요소는 다른 처리 요청을 받지 않도록 한다. 또한 기능 요소에 배제가 설정되어 있을 경우에는 해당 하드웨어 구성 요소가 처리하고 있는 처리 요구가 모두 끝난 다음에 하드웨어 구성 요소를 점유하도록 한다. 이렇게 하여 여러 하위 작업에서 동일한 하드웨어 구성 요소로 그리고 동일한 하드웨어 구성 요소의 동일한 기능 요소로 사상된 경우에 발생하는 하드웨어 자원 처리 요구 충돌까지 고려한 시뮬레이션을 수행하게 된다. 예를 들어 (그림 3)의 응용 프로그램 모델에서는 하위 작업 1과 3이 동일한 기능 요청을 하기 때문에 사상 방법에 따라 두 하위 작업의 처리 요구 충돌이 발생할 수 있으며 이것은 시뮬레이션을 통해 고려된다. 처리가 모두 끝난 뒤에는 얼마의 시간이 걸렸는지 자료 단위에 기록한 뒤에 처리 요청을 한 하위 작업에 결과를 돌려준다. 그리고 각 하드웨어 요소가 사용된 총 시간을 기록하여 각 사상의 경우에 대한 시뮬레이션이 끝났을 때 하드웨어 요소의 사용 율을 구하는 근거로 사용한다.

하드웨어 소프트웨어 사상 제어기는 시뮬레이션 수행 전에 하드웨어 요소와 각 하드웨어 요소가 수행할 수 있는 기능 요소에 대한 정보, 그리고 응용 프로그램의 하위 작업들이 요청하게 될 처리 요구에 대한 정보를 이용하여 어떤 하위 작업을 어떤 하드웨어 요소의 기능 요소에 사상할 수 있는지를 판단한 다음 가능한 사상의 경우를 모두 찾는다. 이렇게 구한 가능한 사상의 경우의 집합의 크기는 응용 프로그램의 하위 작업들에서 수행하는 기능의 총 수가 i이고 각각의 기능이 선택 가능한 경우의 수가 f일 때 Πf_i로 주어진다. (그림 4)의 예에서는 하위 작업 2번이 요구하는 기능 2가 하드웨어 요소 1과 2에서 수행이 가능하기 때문에 두 가지의 사상의 경우가 발생한다. 시뮬레이션은 각 사상의 경우에 대해 반복해서 수행되며 하드웨어-소프트웨어 사상 제어기는 선택된 사상의 경우에 따라 응용 시뮬레이터의 하위 작업들이 보내는 처리 요구를 이를 처리할 수 있는 하드웨어 구조 시뮬레이터의 하드웨어 구성 요소로 보낸다. 매 시뮬레이션마다 선택된 사상의 경우가 가지는 성능 값을 모아서 모든 사상의 경우에 대한 시뮬레이션이 끝났을 때 적합한 사상의 경우를 판단하는 근거로 사용한다.

4. 시뮬레이션 시간 향상 기법

하드웨어-소프트웨어 사상 제어기에서 구한 가능한 사상의 경우의 개수가 Πf_i로 주어지기 때문에 응용프로그램의 모델에 속하는 하위 작업의 수와 그 기능 요구가 많아짐에 따라 그리고 하드웨어 모델에서 여러 하드웨어 요소들이 속하고 각 하드웨어 요소들이 같은 기능 요소들을 가지는 경우가 많아짐에 따라 사상 집합의 크기는 매우 커지게 된다. 이러한 사상 집합의 크기는 시뮬레이션 시간에 심각한 영향을 미치기 때문에 가능하다면 시뮬레이션 이전에 사상 집합의 크기를 줄임으로 하드웨어-소프트웨어 분할을 위한 시뮬레이션과 성능 평가에 걸리는 시간을 줄일 수 있다. 그 방법으로는 먼저 FPGA의 사용 가능한 자원 양(FPGA 셀 개수)과 기능 요소들을 FPGA에 구성하기 위해 소요되는 자원 요구 량을 고려하여 실제로 시스템이 구성할 수 없는 사상의 경우를 배제하는 방법이 있다. 그리고 각 사상의 경우에서 식별할 수 있는 두 가지 요소인 작업량과 병렬성이 FPGA를 사용하는 재구성 가능한 시스템의 처리율과 가지는 관계를 근거로 사상 집합을 축소함으로 시뮬레이션 시간을 줄일 수 있다.

4.1 자원 요구 량에 근거한 사상 집합의 축소

하나의 FPGA에 구성된 여러 가지의 기능을 동시에 수행하는 것은 가능하지만 여러 기능을 하나의 FPGA에 구성하려면 각 기능이 FPGA에서 차지하는 셀의 개수의 합이 FPGA에서 허용하는 셀의 개수를 넘지 않아야 한다. 예를 들어 Xilinx 사의 XC5202는 256개의 논리 셀(logic cell)을 가지고 있고 여기에 구현 가능한 논리 회로의 크기는 3,000개의 게이트(Gate)이하이다. XC5210의 경우에는 1,296개의 논리 셀을 가지고 있고 최대 16,000개의 게이트까지를 구현할 수 있으며 보통 10,000개에서 16,000개의 게이트 사이의 크기를 가지는 논리 회로를 구성할 수 있다[10]. 이러한 하드웨어의 제약에 따라 하드웨어-소프트웨어 사상 제어기에서 생성된 사상 집합의 일부 사상의 경우는 실제로 구성될 수 없을 수 있다. 이것은 각 사상의 경우에 하드웨어 요소에 어떤 기능이 구성되는지와 그 때 소요되는 FPGA의 셀 개수를 계산하여 하드웨어 요소가 실제로 그만큼의 셀 개수를 지원해 줄 수 있는지를 비교함으로 이루어진다.

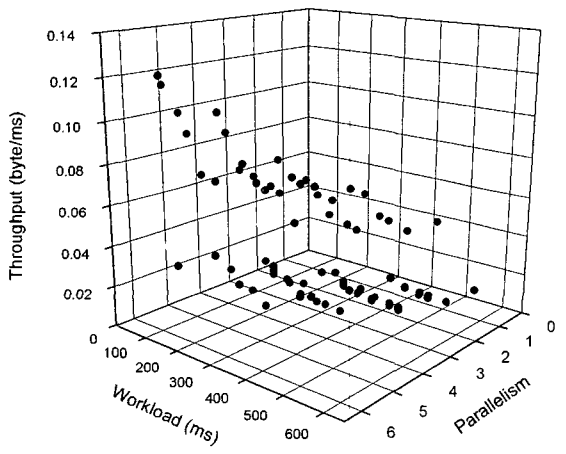
4.2 작업량과 병렬성에 근거한 사상 집합의 축소

재구성 가능한 시스템은 시스템의 성능을 높이기 위해 보조 프로세서나 하드웨어 가속기를 FPGA와 같이 재구성 가능한 장치에 구성하며 같은 기능을 수행할 경우 클럭 속도의 차이가 많이 날 때에도 범용 프로세서에서보다 재구성 가능한 장치에서 수행할 때에 더 높은 성능을 보인다[1, 3]. 또한 FPGA에서 수행되는 기능은 범용 프로세서와 병렬로 수행

될 수 있고 FPGA내에 구성된 여러 기능 역시 서로간에 간섭이 없이 병렬로 수행될 수 있기 때문에 병렬성이 높아지게 된다. 이에 따라 FPGA를 사용하는 시스템의 경우에서는 어떤 사상 집합 M내의 사상의 경우를 $m1$과 $m2$라고 할 때 각각의 사상의 경우에 응용 프로그램의 하위 작업 각각에 소요되는 수행 시간의 합인 작업량(workload) $W1$, $W2$와 하위 작업의 기능들이 최대로 동시에 수행될 수 있는 수인 병렬성(parallelism) $P1$, $P2$의 관계는 각각의 처리율 $T1$, $T2$에 대해 대체적으로 다음을 만족한다.

$$\begin{aligned} &\text{If } W1 > W2 \text{ and } P1 < P2 \rightarrow T1 < T2 \\ &\text{If } W1 < W2 \text{ and } P1 > P2 \rightarrow T1 > T2 \end{aligned} \quad (1)$$

즉, FPGA를 사용함으로서 작업량을 줄이면서 병렬성을 높인 경우의 처리율은 그렇지 않은 경우보다 대체적으로 더 높다. (그림 5)에서는 H.263 해석기(decoder)를 하나의 범용 프로세서와 세 개의 FPGA에서 수행할 때를 시뮬레이션한 결과로부터 병렬성에 따른 작업량과 자료 단위 당 수행시간의 관계를 구한 그래프를 보이고 있으며 위의 식을 만족함을 볼 수 있다. 이러한 관계에 근거하여 사상 집합 축소 휴리스틱 알고리즘을 만들 수 있다.



(그림 5) H.263 해석기 응용 프로그램을 하나의 범용 프로세서와 세 개의 FPGA를 사용한 시스템에서 구현할 때 처리량의 병렬성과 작업량 관계

본 논문에서는 사상 집합 내의 각 사상의 경우들에 대해 다른 사상들과 작업량과 병렬성에 대한 비교를 하여 그 사상의 경우보다 높은 작업량과 낮은 병렬성을 가지는 사상의 경우들은 사상 집합에서 배제하는 알고리즘을 제안하고 실험하였다.

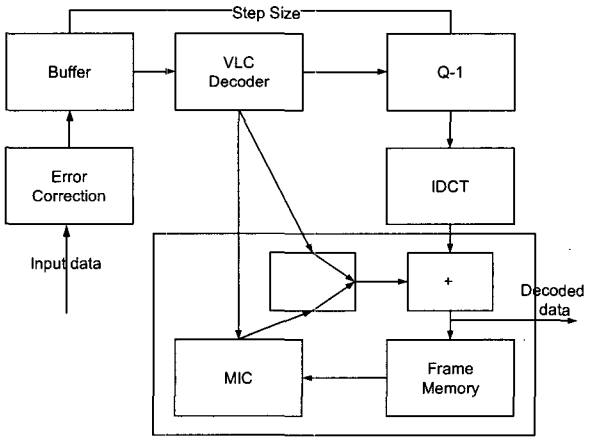
5. 실 험

본 논문에서 구현한 하드웨어-소프트웨어 분할 도구에서 사상 집합 축소 휴리스틱 알고리즘을 수행하였을 때 사상 집합을 얼마나 축소할 수 있는가와 높은 성능을 내는 시스템 구성을 선택해 내는가를 실험하였다.

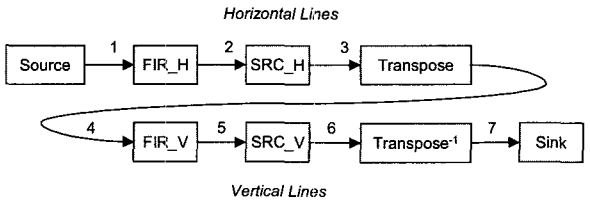
5.1 시뮬레이션 환경

하드웨어 구조 모델로서 일반적으로 재구성 가능한 내장형 시스템에서 주로 사용하는 하나의 범용 프로세서와 여러 개의 FPGA를 가진 시스템 모델을 사용하였다. 응용 프로그램 모델로서는 H.263 해석기(decoder)(그림 6)와 이미지크기를 변경하여 두 개의 이미지를 하나의 TV화면에 보여주는 Picture in Picture 알고리즘(그림 7)을 택하였다. H.263 해석기의 경우 궤환이 있는 부분을 하나의 하위 작업으로 묶어서 모델링함으로 데이터가 한 방향으로 흘러가도록 재모델링하였다. H.263은 널리 쓰이는 비디오 신호 처리 모델로서 데이터의 흐름을 나타내기에 적합하고 Picture in Picture는 여러 하위 작업들이 같은 기능을 사용하기 때문에 이 때 발생하게 되는 처리 요구 충돌 문제가 시스템에 어떤 영향을 미칠 수 있는지 고려할 수 있기 때문에 이 두 응용 프로그램을 선택하였다. 각 프로그램의 모델에 대해 일반 프로세서에서 수행될 경우와 FPGA에서 수행될 경우에 소요되는 시간 그리고 실험을 위해 데이터를 보내기 위해 사용하는 데이터 패킷의 간격은 <표 2>와 <표 3>에서 볼 수 있다.

이러한 각각의 응용 프로그램 모델에 대하여 하드웨어에서 사용하는 FPGA의 크기와 개수를 달리하는 재구성 가능한 시스템에서 어떻게 사상될 수 있는지를 시뮬레이션하였다.



(그림 6) H.263 해석기 모델



(그림 7) Picture in Picture 처리기 모델

〈표 2〉 H.263 해석기의 파라메터

PE Name	PE_ID	Execution time on a processor(ms)	Execution time on a FPGA(ms)	FPGA resource request	System Parameters	
Error Correction	1	50	10	1000	Input Period	1ms
Buffer	2	10	2	500	Buffering Time	500ms
VLC Decoder	3	200	40	2000	Scheduling Method	FCFS
Q-1	4	70	14	700	Packets processed	1000
IDCT	5	100	20	1000	Number of Processor	1
MIC	6	150	30	1500	Number of FPGA	3

〈표 3〉 PIP 처리기의 파라메터

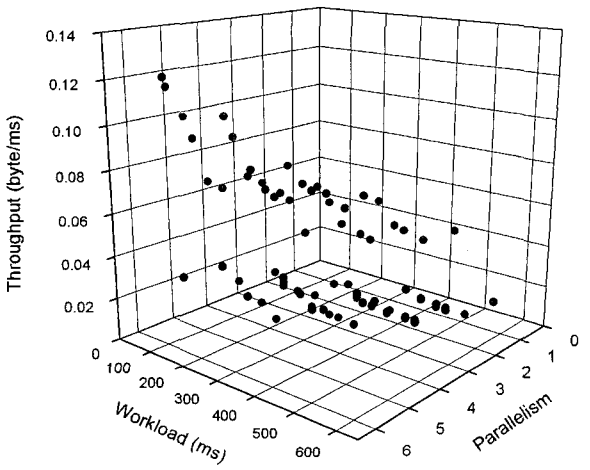
PE Name	PE_ID	Execution time on a processor(ms)	Execution time on a FPGA(ms)	FPGA resource request	System Parameters	
Reader(Source)	1	6	6	200	Input Period	1ms
FIR_H	2	108	22	1000	Buffering Time	500ms
SRC_H	3	36	8	500	Scheduling Method	FCFS
Transpose	4, 7	36	8	500	Packets processed	1000
FIR_V	5	108	22	1000	Number of Processor	1
SRC_V	6	36	8	500	Number of FPGA	3

5.2 사상 집합 축소 휴리스틱 알고리즘의 정확성 조사

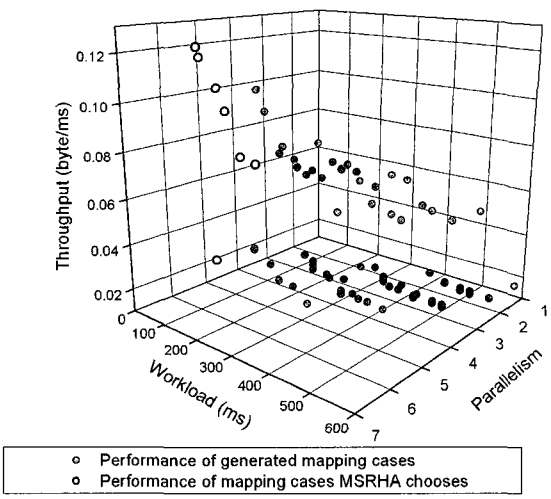
사상 집합 축소 휴리스틱 알고리즘의 정확성을 증명하기 위해 알고리즘을 통하여 배제된 사상의 경우는 선택된 사상의 경우보다 항상 낮은 수행 성능을 가지는 것임을 보였다. 하나의 범용 프로세서와 세 개의 FPGA에서 H.263과 Picture in Picture가 수행되는 경우에 있어서 먼저 전체 사상 집합에 대한 시뮬레이션 결과를 보이고, 축소했을 때 남은 사상 집합을 이에 대비하여 높은 처리율을 보이는 사상의 경우가 배제되지 않음을 보임으로 알고리즘이 정확한지를 보였다.

사상 집합 축소 휴리스틱 알고리즘의 수행전과 수행 후 결과 비교 (그림 8)은 H.263에서 사상 집합이 자원 제약으로 인해 축소된 후의 그래프를 보여 준다. 사상 집합의 초기 크기와 축소된 크기는 <표 4>의 H.263(5)에서 볼 수 있다. 초기 사상 집합의 크기는 4,096이었으며 자원 제약으로 인해 12개의 사상의 경우가 배제된 후의 4084개의 사상의 경우를 시뮬레이션한 결과이다. 이 집합에 대해 사상 집합 축소 휴리스틱 알고리즘을 수행하여 선택된 사상의 경우들을 (그림 9)에서 보여준다.

이 두 그래프를 비교한 (그림 10)의 그래프에서 볼 수 있듯이 최종적으로 축소된 사상 집합은 축소되기 전의 사상 집합에서 가장 높은 처리율을 가지는 사상의 경우를 포함한다. 이 경우에 축소 휴리스틱 알고리즘으로 축소하였을 때에는

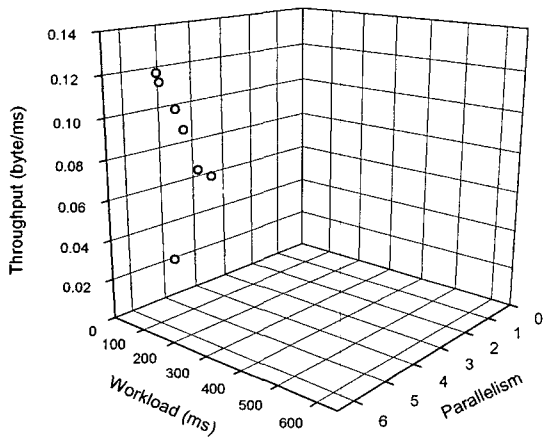


(그림 8) H.263의 축소 휴리스틱 알고리즘 수행 전 사상 집합



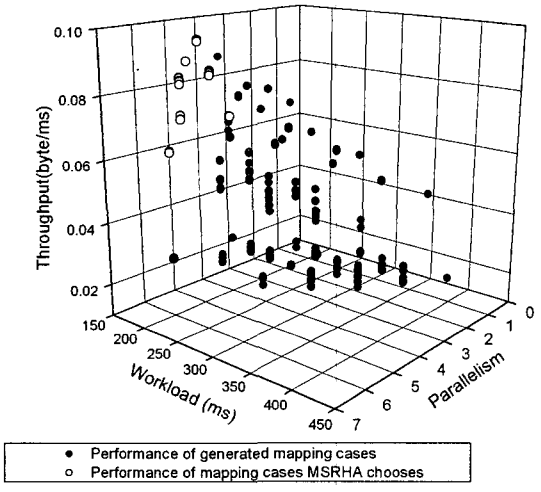
(그림 9) H.263에서 축소되기 전의 사상 집합과 축소 휴리스틱 알고리즘으로 축소한 사상 집합의 비교

4,084개의 사상의 경우에서 2,176로 줄어서 사상 집합의 크기는 원래 사상 집합의 53%로 줄어들었다.



(그림 10) H.263의 축소 휴리스틱 알고리즘 수행 결과로 선택된 사상의 경우들

(그림 11)은 Picture in Picure의 경우를 보여주고, 사상 집합의 초기 크기와 축소된 크기는 <표 4>의 Pip(2)에서 볼 수 있다. Picture in Picture의 사상 집합의 초기 크기는 16,384개이고 이것을 자원 량에 근거한 사상 집합의 축소를 통해 1,648개로 줄었다. 사상 집합 축소 휴리스틱 알고리즘을 적용하면 다시 240개로 줄어들게 된다. 줄어든 결과는 원래의 사상 집합에서 가장 좋은 처리율을 가지는 사상의 경우를 포함함을 볼 수 있다.



(그림 11) Picture in Picture에서 축소되기 전의 사상 집합과 축소후의 사상 집합의 비교

이렇게 시뮬레이션 결과에서 보는 것과 같이 일반적으로 휴리스틱 알고리즘 모두 가장 좋은 처리율을 가지는 사상의 경우를 배제하지 않으면서 사상 집합의 크기를 축소할 수 있다.

5.3 사상 집합 축소 휴리스틱 알고리즘의 효율

축소 휴리스틱 알고리즘을 사용하여 사상 집합을 축소하였을 때 그 크기가 얼마나 줄어들었는지를 구하였다. Y-chart 방법은 시뮬레이터를 사용하여 사상 집합에 속한 모든 사상의 경우를 시뮬레이션하고, 그에 따른 성능 수치를 구한 다음, 그 중에서 가장 높은 성능을 나타내는 사상의 경우를 하드웨어 소프트웨어가 최적으로 분할된 시스템 구성(system configuration)으로 채택한다. 그러므로 사상 집합의 크기가 줄어들게 되면 설계 공간 탐색에 소요되는 시간도 그에 비례하여 줄어들게 된다.

<표 4>에서는 H.263 해석기 모델에 대하여 다섯 가지 시스템 구성을 대응하여 실험한 결과와 Picture in Picture에 대하여 두 가지 시스템 구성을 대응하여 실험한 결과를 보였다. H.263(1)은 두 개의 범용 프로세서와 두 개의 FPGA를 사용한 시스템을 가정하였고 H.263(2)는 하나의 범용 프로세서와 하나의 FPGA를 사용한 시스템을 가정하였다. H.263(3)과 H.263(4)는 하나의 범용 프로세서와 하나의 FPGA를 사용하지만 H.26(3)에서 사용하는 FPGA의 크기가 더 크다. H.263(5)는 하나의 범용 프로세서와 세 개의 FPGA를 사용하는 시스템을 가정하였다. Pip(1)은 하나의 범용 프로세서와 하나의 FPGA를 사용하는 시스템을 그리고 Pip(2)는 하나의 범용 프로세서와 세 개의 FPGA를 사용하는 시스템으로 구성된 경우이다. 설계 공간 탐색에 소요되는 시간은 사상 집합의 크기에 비례하므로, 사상 집합 축소 휴리스틱 알고리즘의 효율은 축소 휴리스틱 알고리즘으로 인해 자원 요구 량에 따른 축소 후에 얼마나 더 축소되었는지를 비교하여 구하며, 다음과 같이 백분율로 계산하였다.

$$\text{효율} = \frac{(\text{자원 요구량에 따른 축소후 크기} - \text{축소 알고리즘 수행 후 크기})}{\text{자원 요구량에 따른 축소후 크기}} \times 100$$

〈표 4〉 사상 집합 축소 휴리스틱 알고리즘의 효율

	사상 집합의 원래 크기	자원 요구 량에 따른 축소 결과	병렬성과 작업량에 따른 축소 결과	
			크 기	효 율(%)
H.263(1)	4,096	4,096	928	77.3
H.263(2)	729	624	186	70.2
H.263(3)	64	40	7	82.5
H.263(4)	64	64	8	87.5
H.263(5)	4,096	4,084	2,175	46.7
Pip(1)	64	13	12	7.7
Pip(2)	16,384	1,648	240	85.4

<표 4>에서 볼 수 있는 바와 같이 작업량과 병렬성에 근거한 사상 집합 축소 휴리스틱 알고리즘을 수행한 결과 자원 요구량에 따른 축소 후에 약 80%를 더 줄일 수 있었다. H.263(5)의 경우에는 같은 크기의 세 개의 FPGA에서 같은

기능들이 수행될 수 있기 때문에 사상 집합에서는 다른 사상의 경우로 나타나지만 같은 사상의 경우인 것이 발생하게 되므로 사상 집합 축소 휴리스틱 알고리즘을 사용한 결과가 다른 결과들에 비해 비교적 낮은 46.7% 정도의 효율을 나타내게 된다. Pip(1)의 경우에는 자원 요구량에 따라 축소한 결과가 충분히 작기 때문에 사상 집합 축소 휴리스틱 알고리즘의 수행 결과에 따른 축소 효율이 7.7%로 매우 낮게 나타난다.

6. 결 론

본 논문에서는 재구성 가능한 시스템에 대한 하드웨어 모델과 이 시스템이 수행하고자 하는 응용 프로그램 모델을 받아 가장 높은 처리율을 가지는 시스템 구성을 할 수 있도록 하는 최적의 사상의 경우를 찾는 하드웨어-소프트웨어 도구를 제안하였고 이를 구현하였다. 이 도구는 Y-chart 기법을 응용한 것으로 하드웨어 모델과 응용 프로그램의 모델간에 가능한 사상의 경우들을 모두 찾아 각 사상의 경우에 대해 시뮬레이션 하여 처리율과 같은 성능 수치를 구하여 최적의 사상을 찾는다. 이 도구는 FPGA와 같은 재구성 가능한 장치에 동시에 여러 개의 기능이 구성될 수 있을 때 고려해야 하는 FPGA의 크기 제약 문제와 시스템이 응용 프로그램을 수행할 때에 같은 하드웨어 요소의 기능을 응용 프로그램의 여러 하위 작업이 이용함으로 발생하는 처리 요구 충돌 문제를 고려하였다.

여러 하드웨어 요소들이 동일한 기능을 제공하여 하위 작업들이 하드웨어 요소들 중에서 선택하여 수행할 수 있는 경우가 많아질 경우 하드웨어-응용 프로그램의 사상 집합의 크기가 매우 크게 되며 이것은 시뮬레이션하고 결과를 평가하는데 드는 시간에 심각한 영향을 미친다. FPGA를 사용하는 재구성 가능한 시스템의 경우 처리량이 대체적으로 작업량과 병렬성과 양의 상관 관계를 가지며 이러한 관계를 이용하여 시뮬레이션 이전에 하드웨어 모델과 응용 프로그램의 모델을 분석하여 작업량과 병렬성에 따라 사상 집합을 줄였다.

본 논문에서는 제안된 사상 집합 축소 휴리스틱 기법이 시뮬레이션 해야 하는 사상 집합을 축소함으로 Y-chart 기법으로 설계 공간을 탐색하여 최적의 사상의 경우를 찾아내는 데 소요되는 시간을 줄일 수 있음을 보였다. 실험한 결과, 제안된 사상 집합 축소 기법을 사용하여 H.263과 Picture in Picture 알고리즘을 다양한 구성의 재구성 가능한 시스템에서 나타나는 사상의 경우를 작업량과 병렬성에 따라 축소하여 약 80% 정도의 사상의 집합을 줄일 수 있었다.

참 고 문 헌

[1] O. T. Albahama, P. Cheung and T. J. Clarke, "On the Viability of FPGA-Based Integrated Coprocessors," In Proceedings of IEEE Symposium of FPGAs for Custom Computing Machines, pp.206-215, Apr., 1996.

[2] J-L, Gaudiots, "Guest Editors' Introduction," IEEE Transactions on Computers, Vol.48, No.6, June, 1999.

[3] E. Sanchez, M. Sipper, J.-O. Haenni, J.-L. Beuchat, A. Stauffer and A. Perez-Uribe, "Static and Dynamic Configurable Systems," IEEE Transactions on Computers Vol.48, No.6, June, 1999.

[4] B. Kienhuis, E. Deprettere, K. A. Vissers and P. Wolf, "An approach for quantitative analysis of application-specific dataflow architectures," In Proceedings of 11th Intl. Conference of Applications-specific Systems, Architectures and Processors (ASAP'97), Zurich, Switzerland, pp. 338-349, 1997.

[5] S. Lee, K. Yun, Choi, Kiyoung, S. Hong, S. Moon and J. Lee, "Java-based programmable networked embedded system architecture with multiple application support," International Conference on Chip Design Automation, pp. 448-451, Aug., 2000

[6] A. C. J. Kienhuis, "Design Space Exploration of Stream-based Dataflow Architectures," PhD thesis, Delft University of Technology, Netherlands, 1998.

[7] S. Bakshi, D. D. Gajaski, "Hardware/Software Partitioning and Pipelining," In Proceedings of the 34th annual conference on Design Automation Conference, pp.713-716, 1997.

[8] A. Kalavade, P. A. Subrahmanyam, "Hardware/Software Partitioning for Multifunction Systems," In Proceedings of International Conference on Computer Aided Design, pp. 516-521, Nov., 1997.

[9] K. M. Gajjala Purna and D. Bhatia, "Temporal Partitioning and Scheduling Data Flow Graphs for Reconfigurable Computers," IEEE Transactions on Computers, Vol.48, No.6, June, 1999.

[10] XC5200 Series Field Programmable Gate Arrays Databook, ver 5.2, Xilinx Inc, Nov., 1998.

[11] Katherine Compton, Scott Hauck, "Reconfigurable Computing : A survey of Systems and Software," ACM Computing Surveys, Vol.34, No.2, pp.171-210, June, 2002.

김 준 용

e-mail : tiecmdr@hanafos.com

1998년 서울대학교 컴퓨터공학과(공학사)

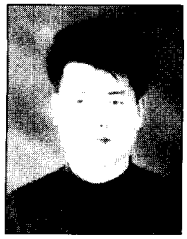
2000년 서울대학교 컴퓨터공학과 석사과정 (공학석사)

2000년~2003년 Digital Solution 개발팀

2003년~2004년 CICA 무역 전산팀

2004년~현재 한국마이크로소프트 개발팀

관심분야 : 내장형 시스템, 시스템 수준 설계, 재구성 가능한 시스템, 실시간 시스템



안 성 용

e-mail : dis@chosun.ac.kr

1996년 조선대학교 전자계산학과(이학사)

1998년 조선대학교 대학원 전자계산학과 석사과정(이학석사)

1999년~현재 조선대학교 대학원 전자계산학과 박사과정 재학 중

관심분야 : 내장형 시스템, 시스템 수준 설계, 재구성 가능한 시스템, 실시간 시스템

이 정 아

e-mail : jalee@chosun.ac.kr

1982년 서울대학교 컴퓨터공학과(공학사)

1985년 Indiana University 컴퓨터과학과 석사과정(공학석사)

1990년 UCLA 컴퓨터공학과 박사과정 (공학박사)

1990년~1995년 미국 텍사스주 휴스턴주립대학 전기전산공학과 조교수

2000년~2002년 Stanford University Visitng Professor

1995년~현재 조선대학교 컴퓨터공학과 교수

관심분야 : 재구성 가능한 시스템, 고속 연산기 설계, CORDIC, 적응형 시스템, 실시간 시스템