論文94-31A-7-14

다중포트 메모리를 지원하는 데이타패스 자동 합성 시스템의 설계

(Design of an Automatic Synthesis System for Datapaths Based on Multiport Memories)

李 海 東**, 金 泳 魯*, 黃 善 泳***

(Hae Dong Lee, Young No Kim and Sun Young Hwang)

要 約

본 논문에서는 다중포트 메모리를 지원하는 데이타패스의 합성을 위하여 그래프 이론에 바탕을 둔 모듈 할당 과정에 대하여 기술하였다. 메모리 모듈의 각 포트에 대한 변수 할당 과정을 수행하기 위하여 이중 분할 매칭 알고리듬을 사용한 효율적인 알고리듬을 제시한다. 제안된 알고리듬은 메모리 모듈 자체의 비용과 연결구조 비용을 동시에 고려하여 프로그램 변수를 메모리에 할당하여 최소 비용의 데이타패스를 합성한다. 벤치마크 프로그램에 대한 타 시스템과의 실험결과 비교를 통하여 제안된 알고리듬이 메모리 모듈 내의 레지스터 개수 및 연결구조 비용 측면에서 최소화된 데이타패스를 생성함을 보인다.

Abstract

In this paper, we propose a graph-theoretic approach for solving the allocation problem for the synthesis of datapaths based on multiport memories. An efficient algorithm is devised by using the weighted bipartite matching algorithm to assign variables to each port of memory modules. The proposed algorithm assigns program variables into a minimum number of multiport memory modules, such that usage of memory elements and interconnection cost can be kept minimal. Experimental results show that the proposed algorithm generates the datapaths with fewer registers in memory modules and less interconnection cost for several benchmarks available from the literatures.

Ⅰ. 서 론

VLSI 집적 기술과 설계 기술의 발전으로 인하여 복잡한 시스템도 하나의 칩내에 구현이 가능하게 됨에 따라 CAD 분야에서는 설계 기술자의 생산성 향상을 위하여 보다 상위 수준에서의 설계 자동화에 대한 연구가 활발히 진행되어 왔으며[1], 특히 하드웨어의 행위 기술로부터의 하드웨어 합성에 대한 연구가 많이 소개되었다.[2] 상위 수준 합성은 HDL (Hardware Description Language)을 이용하여 설계하고자 하는 하드웨어의 행위를 기술한 형태를 입력으로 받아들여 여러 단계의 최적화 과정을 통하여 레지스터 전송 수준의 데이타패스를 합성하는 과정이다. 상위수준 합성에 관한 연구는 1970년대 초부터 활발히 진행되어 왔고, 1988년에 VHDL (Very

*準會員, **正會員, ***終身會員, 西江大學校 電子工學科
(Dept. of Elec. Eng., Sogang Univ.)
接受日字 : 1993年 11月 10日

High-speed Integrated Circuit Hardware Description Language) [1]이 IEEE에 의해 공식적으로 표준화되어 설계 데이타의 호환성 문제를 해결하므로써 점차 상위 수준 합성기가 상용화되어 가는 단계에 있다. 상위 수준 합성은 스케쥴링과 모듈할당의 두 과정으로 나뉜다. 스케쥴링 과정에서는 연산기의 개수, 주어진 동작기술의 수행에 필요한 전체 제어구간의 수, 그리고 각 제어구간에서 수행되는 연산을 결정한다. 모듈할당 과정은 연산기 할당, 레지스터 할당, 그리고 연결구조 할당의 부과정으로 구성된다. 연산기 할당에서는 각 제어구간에서 수행되는 연산을 연산기에 할당시키고, 동작기술 상의 변수에 대한 할당이 레지스터 할당 과정에서 수행된다. 연결구조 할당 과정에서는 할당된 레지스터와 연산기 간의 데이타 전달을 위하여 bus 또는 mux를 이용한 연결구조를 구성한다.

최근에 들어서는 상위 수준 합성시 발생하는 방대한 설계 영역 (design space)의 탐색으로 인한 부담을 줄이기 위하여 합성하고자 하는 하드웨어의 아키텍춰를 미리 설정한 상태에서 합성과정을 수행하는 추세이다. 레지스터 전송 수준의 데이타패스는 크게 기억소자, 연산모듈, 그리고 연결구조로 구성된다. 대부분의 상위수준 합성 시스템은 기억소자로서 레지스터를 사용하며, 최적화 과정을 통하여 레지스터와 연산기간의 연결구조 비용을 줄이는 방향으로 목적함수를 설정하였다. [4-7] 그러나, 독립된 레지스터를 사용하여 합성할 경우, 레지스터의 개수에 비례하여 연결구조의 비용이 증가한다는 단점이 있다. 반면 다중포트 메모리를 사용하여 합성한 데이타패스는 연결구조가 간단해지고 설계영역 및 칩면적이 감소한다는 장점으로 인하여 다중포트 메모리를 지원하는 시스템이 발표되고 있다.

Balakrishnan et al. [8]은 전체 메모리 모듈의 개수를 최소화하기 위하여 한번에 하나의 다중포트 메모리를 합성하는 방식을 사용하였다. 할당되지 못한 변수는 독립적인 레지스터로 사용되거나 같은 알고리듬을 반복적으로 적용하여 메모리 모듈을 합성한다. 이러한 방식은 전역적인 설계영역의 탐색이 불가능하므로 메모리 모듈의 비용이나 연결구조 비용 측면에서 최적의 결과를 얻을 수 없다. Wilson et al. [9]은 레지스터를 할당 가능한 메모리 모듈에 하나씩 할당하는 휴리스틱 방법을 사용하였으나, 지엽적인 greedy search로 인하여 전체 메모리 모듈의 개수 및 각 메모리 내의 레지스터의 개수 측면에서 최적의 결과를 구할 수 없었다. Ahmad et al. [10]은 최소의 메모리 모듈 및 메모리 내의 레지스터 개수를 보장하기 위하여 0-1 ILP (Integer Linear Programming)를 사용하였다. 그러나, 목적함수에서 메모리 모듈과 연산기 간의 연결구조 비용을 고려하지 않았기 때문에 생성된 하드웨어의 연결구조 비용이 증가하는 결과를 초래하였다. Kim et al. [11]은 연결구조 비용을 고려한 0-1 ILP 알고리듬을 사용하였으나 메모리 모듈 내의 레지스터의 개수를 최소화하지 못했다.

본 논문에서는 다중포트 메모리를 지원하는 상위 수준 합성 알고리듬에 대하여 기술한다. 제안된 알고리듬은 연결구조 비용과 메모리 모듈 내의 레지스터 개수를 모두 고려하여 하드웨어를 합성하며, 각 제어구간에서 구동되는 변수를 메모리 모듈의 포트에 매핑시키기 위하여 이중분할 매칭 (bipartite matching) 알고리듬을 사용하였다. 제안된 시스템의 개관 및 타겟 아키텍춰를 2장에서 설명하며, 다중포트 메모의 합성 과정과 연결구조의 최소화 과정을 3장에서 자세히 기술한다. 시스템의 성능비교를 위하여 4장에서 실험결과를 타시스템과 비교하였고, 결론을 5장에 제시한다.

Ⅱ. 시스템 개관 및 타겟 아키텍춰

제안된 시스템 SODAS의 개관을 그림 1에 보였다. 입력 동작기술 언어 VHDL은 SODAS의 front-end인 VHDL 분석기에 의해 구문분석 및 의미분석을 수행하고, 데이타 흐름과 콘트롤 흐름 정보를 가진 C/DFG (Control/Data Flow Graph)를 중간형태로 생성한다. 상위 수준 합성 과정은 스케쥴링과

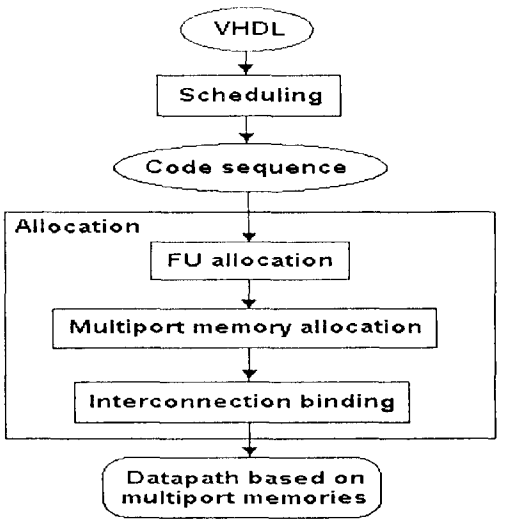


그림 1. SODAS 시스템 개관
Fig. 1. Overall picture of SODAS.

모듈할당으로 구성되며, C/DFG의 정보를 이용하여 수행된다. 스케쥴링의 결과는 순차코드 (code sequence)의 형태로 저장되고, 모듈할당 과정은 순차코드를 이용하여 연산기의 할당, 다중포트 메모리의 할당, 그리고 연결구조의 할당 과정을 통하여 수행된다.

SODAS 시스템의 타겟 아키텍춰를 그림 2에 보였다. 이는 연산기, 메모리 모듈, 그리고 연결구조로 구성된다. 다중포트 메모리가 기억소자로 사용되며, 기억소자와 연산기 사이의 데이타 전달을 위한 연결구조로서 bus와 mux가 사용된다. Bus의 구동을 위하여 TSB (Tri-State Buffer)가 bus의 입력단에 존재하며, 연산기의 입력단에는 입력선택을 위하여 mux가 사용된다. SODAS는 타겟 아키텍춰를 구동하기 위하여 1-phase 및 2-phase 클럭이 지원되며, 2-phase 클럭을 사용할 경우, 데이타의 저장과 적재가 각각 phase 1과 phase 2에서 수행된다.

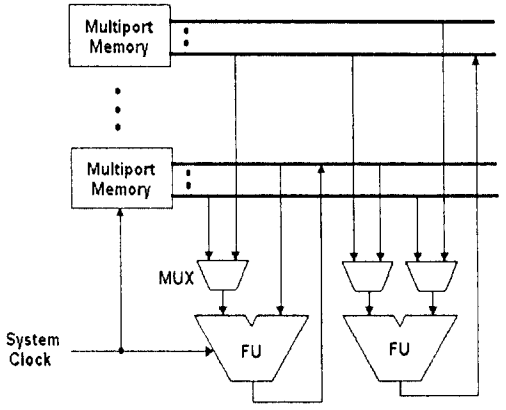


그림 2. 타겟 아키텍춰
Fig. 2. Target Architecture.

Ⅲ. 타겟 아키텍춰의 합성

타겟 아키텍춰의 데이타패스는 스케쥴링과 모듈할당의 과정을 통하여 합성된다. 스케쥴링 과정은 제어구간에 대한 연산기의 면적을 고려하여 균등한 배분을 수행하는 엔트로피에 바탕을 둔 스케쥴링 알고리듬[12]을 사용하였으며, 연산기 할당 과정에서는 similarity measure[13]를 이용하여 C/DFG 상의 각 연산이 연산기에 할당된다.

스케쥴링과 연산기 할당이 완료된 후 다중포트 메모리의 합성과 연결구조의 할당을 수행하므로써 타겟 아키텍춰가 생성된다. 다중포트 메모리의 합성 과정에서는 이중분할 매칭 알고리듬을 이용하여 각 제어구간에서 사용되거나 정의되는 변수가 특정 메모리 포트에 할당되며, 연결구조 할당 과정에서 2-color 그래프 채색 알고리듬을 통한 연결구조 최소화를 수행하여 데이타패스를 합성한다.

1. 다중포트 메모리의 합성

그림 3에 다중포트 메모리 합성의 전체적인 흐름도를 보였다. 합성 알고리듬은 변수와 메모리 포트의 초기화, 메모리 포트에 대한 변수의 할당, 그리고 메모리 모듈의 추출의 세 과정으로 구성된다. 초기화 과정에서는 전체 변수에 대한 생존구간 분석이 수행되고, 전체 메모리 모듈의 포트수가 결정된다. 초기화 과정이 완료되면 각 제어구간에 대하여 변수의 메모리 포트 할당이 수행되고, 할당 결과를 이용하여 메모리 모듈의 추출 과정이 수행된다. 메모리 포트의 할당 과정은 전체 제어구간에 대하여 순차적으로 수행되며, bipartite 그래프의 구성, bipartite 그래프의 간선에 대한 가중치의 계산, 이중분할 매칭 알고리듬의 적용, 그리고 할당결과의 변환 및 저장 과정으로 세분된다. 메모리 모듈의 추출 과정은 메모리 포트 할당의 결과에 의하여 수행된다. 다중포트 메모리 합성의 세부 과정에 대한 자세한 내용은 다음절에서 기술한다.

```
Multi_port_memory_synthesis ()
begin
    /* phase 1 : Initialization */
    Perform lifetime analysis for variables;
    Determine total number of ports;

    /* phase 2 : assignment of variables to memory ports */
    for each control step do
        begin
            Construct the bipartite graph;
            /* one partition for variables, and the other for ports */
            Assign weight to each edge in bipartite graph;
            Perform weighted bipartite matching;
            Update results;
        end;

    /* phase 3 : Memory module extraction */
    Extract memory modules from the results of assignment;
end;
```

그림 3. 다중포트 메모리 합성 흐름도
Fig. 3. Overall flow of multiport memory synthesis.

1) 변수와 메모리 포트의 초기화

초기화 과정에서는 변수에 대한 생존구간과 전체 메모리 포트의 수가 결정된다. 생존구간 분석[4]은 동작기술 상의 모든 변수에 대하여 수행되고, 그 결과는 메모리 모듈 내의 레지스터 수를 줄이기 위한 목

적합수에 반영된다. 메모리 모듈의 포트 수는 전체 제어구간을 탐색하여 각 제어구간에서 구동되는 변수의 개수를 분석하여 결정한다. 변수는 특정 제어구간에서 피연산자로 사용되거나, 연산결과를 저장하는 용도로 사용될 때 구동되며, 메모리 모듈의 전체 포트 수는 최대의 변수가 구동되는 제어구간에서의 변수의 개수로 결정된다. 최대의 변수가 구동되는 제어구간을 cs_{max}, cs_{max}에서 구동되는 변수의 집합을 $VarSet(cs_{max})$로 정의할 때, 메모리 모듈의 전체 포트 수는 집합 $VarSet(cs_{max})$의 크기로 결정된다.

2) 메모리 포트에 대한 변수의 할당

초기화 과정이 완료되면 각 제어구간에서 구동되는 변수를 메모리 포트에 할당하는 작업이 수행된다. 메모리 포트에 대한 변수의 할당 결과에 의하여 최종적으로 생성되는 데이타패스의 연결구조 비용 및 메모리 모듈의 비용이 결정된다. 따라서, 다중포트 메모리의 합성 과정에서 메모리 포트에 대한 변수의 할당은 하드웨어의 비용을 줄이기 위한 중요한 단계이다. SODAS는 가중치를 적용한 이중분할 매칭 알고리듬[14]을 이용하여 메모리 포트에 대한 변수의 할당을 수행한다.

할당 과정은 각 제어구간 cs에서 구동되는 변수의 집합 $VarSet(cs)$에 대하여 수행되고, 각 제어구간 cs에 대한 bipartite 그래프 $G(cs) = (VarSet(cs), PortSet, E)$가 구성된다. Bipartite 그래프는 제어구간 cs에서 구동되는 변수의 집합 ($VarSet(cs)$)과 메모리 포트의 집합 ($PortSet$)으로 두 개의 노드 집합을 구성하고, 노드 집합 사이에 간선의 집합 (E)이 존재한다. $G(cs)$에서 간선은 집합 VarSet(cs)의 각 변수가 할당 가능한 메모리 포트 사이에 연결된다. 간선의 가중치는 배부 및 외부 비용으로 결정된다. 내부 비용은 메모리 모듈 내에 존재하는 레지스터의 수에 의해 결정되고, 연산기의 입력단에 연결되는 mux 입력의 개수가 외부 비용으로 주어진다. 메모리 포트에 대한 변수 할당 과정의 목적은 $VarSet(cs)$에 존재하는 각 변수는 하나의 특정 메모리 포트에 연결되어야 하고, 연결된 전체 간선의 가중치의 합을 최소화하는 것이다.

내부 및 외부 비용은 각 제어구간에서 구성되는 $G(cs)$의 $VarSet(cs)$와 $PortSet$에 의하여 결정된다. 집합 $VarSet(p_j)$를 메모리 포트 p_j에 할당된 변수의 집합이라 정의하고, $VarSet(p_j)$에 존재하는 변수는 각 제어구간에 대한 할당을 수행하면서 갱신된다. 내부 비용은 하나의 변수가 특정 메모리 포트에 할당될 경우 요구되는 레지스터의 개수이다. 변수 v_i를 메모리 포트 p_j에 할당할 경우의 내부 비용을 식 (1)에 보인다.

$$\mathrm{InCost}(v_i, p_j) = \sum_{v_k \in \mathrm{VarSet}(p_j)} \mathrm{LO}(v_i, v_k) \quad (1)$$

$$\text{where } \mathrm{LO}(v_i, v_k) = \begin{cases} 1, & \text{if lifetimes of } v_i \text{ and } v_k \text{ overlap} \\ 0, & \text{otherwise} \end{cases}$$

외부 비용은 변수 v_i와 메모리 포트 p_j 간에 주어지며, 변수 v_i가 메모리 포트 p_j에 할당될 경우에 메모리 모듈과 연산기 사이의 연결구조 비용을 의미한다. Mux는 연산기의 입력 포트와 메모리 모듈 간의 통신 경로가 여러개 존재할 경우, 데이타의 선택을 위하여 연산기의 입력 포트에 존재한다. 변수 vi를 사용하는 연산기의 포트 집합을 $FUPorts(v_i)$로 정의하자. 연산기의 포트 fupk에 메모리 포트 p_i가 연결될 경우 fup_k에 존재하는 mux 입력의 수를 NumMuxInputs(fup_k, p_i)로 정의할 때, 변수 vi를 메모리 포트 p_j에 할당할 경우 발생하는 외부 비용은 식 (2)로 정의된다.

$$\mathrm{ExCost}(v_i, p_j) = \sum_{fup_k \in FUPorts(v_i)} \mathrm{NumMuxInputs}(fup_k, p_j) \quad (2)$$

외부 비용은 $FUPorts(v_i)$의 모든 메모리 포트 pi에 대하여 MuxInputs(fup_k, p_i)를 가산하므로써 얻어진다. 식 (1), (2)의 내부 및 외부 비용을 이용하여 변수 v_i와 메모리 포트 p_j 간의 간선 e_{ij}의 가중치 w_{ij}는 식 (3)으로 정의된다.

$$W_{ij} = \alpha \cdot \mathrm{InCost}(v_i, p_j) + \beta \cdot \mathrm{ExCost}(v_i, p_j) \quad (3)$$

where α and β are impirical parameters set for tuning the cost function 식 (3)에 의하여 bipartite 그래프 G(cs)에서 간선의 가중치가 결정되면 weighted bipartite matching 알고리듬을 적용하여 제어구간 cs에서 구동되는 변수가 특정 메모리 포트에 할당된다. X_{ij} (i = 1, ⋯ , $|VarSet(cs)|$, j = 1, ⋯ , $|PortSet|$)를 0-1 정수형 변수라고 정의하고, X_{ij}의 값은 변수 v_i가 메모리 포트 p_j에 할당될 경우 1, 할당되지 않을 경우 0이다. 집합 PortSet의 크기가 집합 VarSet(cs)의 크기보다 클 경우에는 bipartite 그래프를 구성하기 위하여 $|PortSet| - |VarSet(cs)|$ 개의 dummy node를 첨가한다. 변수의 할당 문제를 식 (4)에 보였다.

$$\text{Minimize} \sum_{i,j} w_{ij} \cdot X_{ij} \quad (4)$$

$$\text{subject to} \sum_{i} X_{ij} = 1 \quad \text{for } i = 1, \ldots, |\mathrm{VarSet(cs)}|$$

$$\sum X_{ij} = 1 \quad \text{for } j = 1, \ldots, |\mathrm{portSet}|$$

식 (4)에서 제약조건은 변수 v_i와 메모리 포트 p_j 간에는 하나의 matching만이 가능함을 의미한다. 변수의 할당 문제는 식 (4)의 bipartite matching 문제로 변형되며, 최소의 비용으로 *VarSet*(*cs*)의 변수를 *PortSet*의 메모리 포트에 할당시킨다.

3) 메모리 모듈의 추출

메모리 모듈은 메모리 포트에 대한 변수의 할당 결과로부터 추출된다. 변수의 할당 과정이 완료되면 각 제어구간에서 구동되는 변수의 데이타를 전달하기 위한 특정 메모리 포트가 결정된다. 메모리 모듈의 추출을 위한 제약조건으로서 동일한 변수가 여러개의 메모리 모듈에 중복적으로 할당되는 것을 방지하기 위하여 하나의 변수는 하나의 특정 메모리 모듈에 할당되어야 한다. 따라서 메모리 포트에 할당된 변수 집합을 탐색하여 같은 변수가 할당된 메모리 포트의 집합으로 하나의 메모리 모듈을 선택한다. 메모리 포트 p_i와 p_j가 주어졌을 때 각 포트에 할당된 변수 집합 중에서 동일한 변수가 존재할 경우 p_i와 p_j를 하나의 메모리 모듈로 할당한다. 전체적인 메모리 모듈의 추출 과정을 그림 4에 보였다. 메모리 포트에 대한 변수의 할당 결과인 메모리 모듈의 포트 집합 PortSet을 입력으로 하여 메모리 모듈에 할당된 변수의 집합 ($VarSet\ (mm_i)$)과 메모리 포트에 할당된 변수의 집합 ($VarSet\ (p_j)$)을 비교하므로써 전체적으로 메모리 모듈이 추출된다. 메모리 모듈의 추출이 완료되면 각 메모리 모듈에 존재하는 특정 변수의 집합이 구성되고, 메모리 모듈 내에 존재하는 레지스터의 개수는 변수에 대하여 clique partitioning 알고리듬[4]을 적용하여 결정된다.

```
Memory_Module_Extraction ()
begin
   /*
      MEM : set of memory modules
      PortSet : set of memory ports
   */
   MEM  ∅ ;
   PortSet  {p1, p2, ... , pnumport};
   i  1 ;
   while PortSet ≠ ∅ do
      begin
         Select an ungrouped port pv ∈ PortSet ;
         Assign pv to memory module i ;
         PortSet  PortSet - pv ;
         for all pj ∈ PortSet do
            /*
               VarSet (mmi) : set of variables assigned to memory module i
               VarSet (pj) : set of variables assigned to port pj
            */
            if VarSet(mmi) and VarSet(pj) have common variables
               begin
                  Assign pj to mmi ;
                  PortSet  PortSet - pj ;
               end
         MEM  MEM ∪ mmi ;
         i  i + 1 ;
      end
end
```

그림 4. 메모리 모듈의 추출 과정

Fig. 4. Multiport memory module extraction.

2. 연결구조의 구성

메모리 모듈의 추출이 완료되면 메모리 모듈과 연산기 간의 데이타 통신을 위하여 bus와 mux를 사용하여 연결구조를 구성한다. 연결구조의 구성 과정은 크게 통신경로의 추출과 통신경로에 대한 버스의 할당으로 구분된다.

통신경로의 추출 과정에서는 연산기와 다중포트 메모리의 합성결과에 의해서 메모리 모듈과 연산기 간에 존재하는 통신경로를 구성한다. 통신경로의 시작점과 종료점은 (module type, module ID, port ID)의 3-tuple로 정의된다. 여기서 module type은 메모리 모듈 또는 연산기의 두 종류로 결정된다. 예로서, 통신경로 {(Mem, 2, 3), (FU, 1, 2)}는 2번 메모리 모듈의 포트 #3와 1번 연산기의 포트 #2 사이의 통신경로를 의미한다. 각 제어구간에서 메모리 모듈과 연산기 간의 데이타 전달은 특정 통신경로의 집합을 통하여 수행되며, 통신경로의 추출 과정에서 전체적인 통신경로의 개수를 줄이기 위한 최소화 과정을 수행한다. 연산기는 크게 교환법칙이 가능한 연산과 가능하지 않은 연산의 두 종류로 나뉜다. 통신경로의 종료점이 덧셈기 또는 곱셈기와 같은 교환법칙이 가능한 연산기에 연결된 경우, 통신경로의 종료점은 연산기의 어느 입력단에 연결되어도 무방하다. 그러나, 뺄셈기와 같이 교환법칙이 적용되지 않는 연산기의 경우에는 통신경로의 종료점이 연산기의 특정 입력단에 연결되어야 한다. 이와 같은 연산기의 특성을 이용하여 교환법칙의 적용이 가능한 연산기에 대하여 입력단에 연결된 통신경로의 쌍을 교환하므로써 전체 통신경로의 개수를 최소화할 수 있다. 통신경로의 최소화 과정을 위하여 2-color 그래프 채색 알고리듬을 사용하였고, 교환법칙 적용의 예를 그림 5에 보였다.

그림 5 (a)는 4개의 통신경로가 교환법칙의 적용이 가능한 연산기에 연결된 연결된 경우를 보이며, 연산기의 입력단에 6개의 mux 입력이 요구된다. 그림 5 (a)에 대한 충돌 그래프 (Conflict Graph : CG)를 그림 5 (b)에 보였다. CG의 노드는 하나의 통신경로를 의미하고, 통신경로 쌍이 같은 제어구간에서 데이타의 전달을 수행할 경우 두 노드 간에 간선이 존재한다. 그림 5 (b)에 대하여 2-color 그래프 채색 알고리듬을 적용한 결과를 그림 5 (c)에 보였으며, 연산기의 입력단에서 요구되는 mux 입력의 수가 3개로 줄었음을 보인다. 그림 5 (d)는 2-color 그래프 채색 알고리듬을 적용한 후의 CG를 나타내며 할당된 색은 연산기의 특정 입력단을 의미한다.

버스의 할당 과정에서는 추출된 통신경로의 집합을

특정 bus에 할당한다. 그림 2의 타겟 아키텍춰에서 보인 바와 같이 bus는 메모리 모듈의 각 포트에 연결되어 있으며, 제어구간에서 구동되는 각 통신경로는 데이타 충돌을 방지하기 위하여 독립적인 bus에 할당된다.

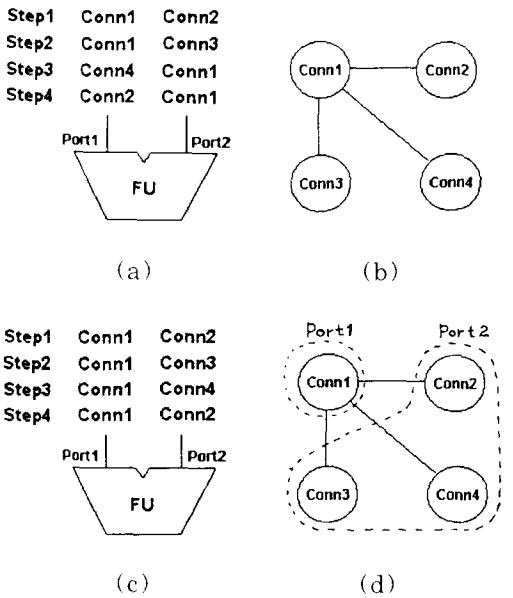


그림 5. 통신경로에 대한 교환법칙 적용의 예
(a) 초기 연결 구조,
(b) (a)에 대한 충돌 그래프,
(c) 교환법칙 적용 후의 감소된 연결구조,
(d) 2-color 채색 알고리듬의 적용 결과

Fig. 5. An example of connection exchange
(a) Initial interconnection,
(b) Conflict graph for (a),
(c) Reduced interconnection after connection exchange,
(d) Result for applying 2-color graph coloring algorithm.

Ⅳ. 실험결과

제안된 시스템 SODAS는 UNIX 운영체제 하에서 C 프로그래밍 언어를 사용하여 구현되었다. 시스템의 성능 비교를 위하여 두 개의 MCNC 벤치마크 프로그램, 6차 밴드패스 필터[7]와 5차 엘립틱 웨이브 필터[5]에 대한 실험결과를 제시한다. 실험결과는 동작 기술 상의 필터 계수가 하나의 계수 ROM에 저장되었다는 가정 하에서 산출되었으며, 이를 기존의 합성 시스템과 비교하였다.

표 1은 제어구간 11의 시간 제약조건 하에서 6차 밴드패스 필터의 합성 결과를 보인다. 합성 결과는 참고문헌[11]에서 제시된 시스템 및 MAP 시스템과 비교하였다. 연산기와 메모리 모듈의 개수는 세 시스템이 동일한 결과를 보이며, 레지스터와 bus의 개수 측면에서도 세 시스템이 같은 결과를 보였으나, mux 입력수와 tri-state 버퍼의 개수는 타시스템에 비해 감소된 결과를 보인다. 이는 SODAS 시스템에서 연산기의 입력단에서의 mux 입력수를 메모리 합성 과정에서 고려하기 때문이다.

표 1. 6차 엘립틱 밴드패스 필터의 합성 결과
Table 1. Synthesis results for the sixth-order bandpass filter.

	SODAS	Ref[11]	MAP
#Csteps	11	11	11
#Adders	2	2	2
#Multipliers	1	1	1
#Mux Inputs	8	10	12
#TSBs	6	6	7
#Buses	5	5	5
#Registers	11	11	11
#RAMs	2	2	2
#ROMs	1	1	1

제어구간 19의 시간 제약조건 하에서 5차 엘립틱 웨이브 필터에 대한 합성 결과의 비교를 표 2에 보였다. 합성결과는 참고문헌[11]의 시스템, GMD[15], MAP, STAR[6], 그리고 SPAID[16]와 비교하였고, 타시스템에 비하여 SODAS가 적은 비용의 데이타패스를 합였음을 보인다. 메모리 포트에 대한 변수의 할당 과정에서 레지스터 개수의 최소화를 고려하는 GMD 시스템과 비교할 때, 레지스터의 개수 측면에서는 같은 결과를 보인다. GMD의 경우 연결구조의 비용을 고려하지 않기 때문에 연결구조 측면의 정확한 비교는 불가능하나, SODAS가 생성한 데이타패스의 연결구조 비용이 더 유리할 것으로 보인다. 참고문헌[11]의 시스템과 MAP과의 비교에서는 적은 개수의 레지스터를 합성하였고, 연결구조 비용인 mux 입력의 개수 측면에서도 유리한 결과를 보인다. 하나의 읽기/쓰기 포트로 구성된 메모리 모듈을 지원하는 STAR 시스템과 SPAID 시스템과의 비교에서는 레지스터의 개수 및 연결구조의 비용이 크게 감소하였다.

표 2. 5차 엘립틱 웨이브 필터의 합성 결과
Table 2. Sythesis results for the fifth-order elliptic wave filter.

	SODAS	Ref[11]	MAP	GMD	STAR	SPAID
#Csteps	19	19	19	19	19	19
#Adders	2	2	2	2	2	2
#Multipliers	1	1	1	1	1	1
#MUX inputs	8	9	10	n/a	17	17
#TSBs	5	5	5	n/a	16	n/a
#Buses	5	5	5	n/a	5	5
#Registers	11	12	14	11	13	19
#RAMs	2	2	2	2	2	2
#ROMs	1	1	1	1	1	1

그림 6은 제어구간 19의 시간 제약조건하에서 5차 엘립틱 웨이브 필터에 대하여 SODAS가 합성한 데이타패스이다. 연산기는 덧셈기 2 개와 곱셈기 1 개를 사용하였고, 메모리 모듈은 2 개의 RAM과 1 개의 ROM을 사용하였다. ROM에는 필터의 특성을 나타내는 계수가 저장되어 있고, 총 11 개의 레지스터를 사용하여 RAM을 합성하였다. 연결구조 측면에서는 5개의 bus를 사용하였다. 각 bus는 메모리 모듈의 포트에 연결되고, 5개의 tri-state 버퍼에 의하여 구동된다. 연산기의 입력단에는 4 개의 mux가 연결되어 있고, 총 8 개의 mux 입력으로 구성된다.

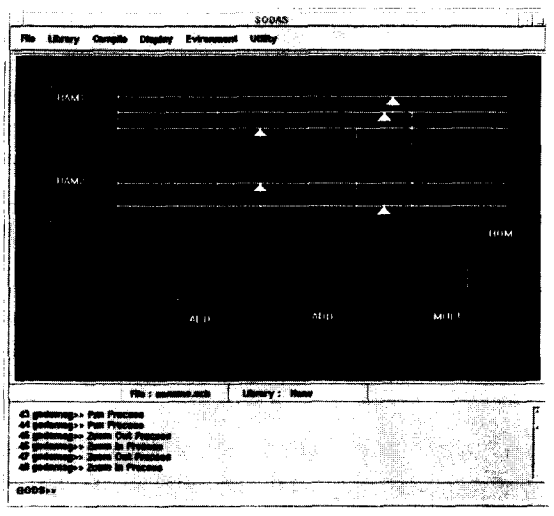


그림 6. SODAS가 생성한 5차 엘립틱 웨이브 필터의 데이타패스
Fig. 6. Datapath for fifth-order elliptic wave filter generated by SODAS.

V. 결 론

본 논문에서는 다중포트 메모리를 지원하는 데이타패스의 합성을 위한 모듈할당 알고리듬에 대하여 기술하였다. 다중포트 메모리의 합성을 위하여 메모리 모듈 자체의 비용 및 연결구조 비용을 최소화하기 위한 목적함수를 설정하였고 이중분할 매칭 알고리듬을 사용하였다. 각 제어구간에 대하여 내부 비용과 외부 비용으로 구성되는 가중치를 적용한 이중분할 그래프를 구성하여 이중분할 매칭 알고리듬을 수행하므로써 프로그램 변수를 메모리 포트에 할당하였다. 메모리 포트에 대한 변수의 할당 결과를 이용하여 메모리 모듈을 추출하고, 연산기와 메모리 모듈 간의 연결구조를 구성하여 전체적인 데이타패스를 합성하였다. 연결구조의 구성 과정에서는 전체 통신경로의 개수를 최소화하기 위하여 2-color 채색 알고리듬을 사용하였다. 통신경로의 최소화 과정은 교환법칙의 적용이 가능한 연산기에 대하여 수행되고, 연산기의 입력단에 연결된 통신경로를 교환하여 연결구조의 비용을 감소시킨다. 2-color 채색 알고리듬의 적용을 위하여 연산기에 연결된 통신경로의 상태를 충돌 그래프로 모델링하였으며, 채색된 통신경로를 해당 포트에 할당하므로써 mux 입력의 수를 최소화하였다.

MCNC 벤치마크 프로그램에 대한 실험결과의 비교를 통하여, 제안된 시스템 SODAS의 효율성을 보였다. SODAS는 변수의 할당 과정에서 메모리 모듈의 내부 비용 및 연결구조의 외부 비용을 동시에 고려하였고, 연결구조의 구성 과정에서는 통신경로를 최소화하므로써 타시스템에 비하여 적은 개수의 레지스터와 감소된 연결구조 비용의 데이타패스를 생성함을 보였다.

參 考 文 獻

[1] D. D. Gajski, 'Silicon Compilation,' Addison Wesley Pub.: Reading, MA, pp. 3-46, 1988.
[2] M. C. McFarland, A. C. Parker, R. Camposano, "The High-Level Synthesis of Digital Systems," Proceedings of IEEE, vol. 78, no. 2, pp. 301-318, Feb. 1990.
[3] "IEEE Standard VHDL Language Reference Manual," IEEE Std 1076-1987, IEEE, N. Y, March 1988.
[4] C. J. Tseng, D. P. Siewiorek,

"Automated Synthesis of Data Path in Digital Systems," IEEE Trans. on CAD, vol. 5, no. 3, pp. 379-395, July 1986.

[5] P. G. Paulin, J. P. Knight, "Scheduling and Binding Algorithms for High-Level Synthesis," in Proc. 23rd ACM/IEEE Design Automation Conference, pp. 1-6, June 1989.

[6] F. S. Tsai, Y. C. Hsu, "Data Path Construction and Refinement," in Proc. IEEE International Conference on Conputer-Aided Design, pp. 308-311, Nov. 1990.

[7] C. A. Papachristou, H. Konuk, "A Linear Program Driven Scheduling and Allocation Method Followed by an Interconnect Optimization Algorithm," in Proc. 27th ACM/IEEE Design Automation Conference, pp. 77-83, June 1990.

[8] M. Balakrishnan, A. K. Majmudar, D. K. Banerji, J. G. Linders, J. C. Majithia, "Allocation of Multiport Memories in Data Path Synthesis," IEEE Trans. on CAD, vol. 7, no. 4, pp. 536-540, April 1988.

[9] T. C. Wilson, D. K. Banerji, J. C. Majithia, A. K. Majumdar, "Optimal Allocation of Multiport Memories in Data Path Synthesis," in Proc. 32nd Midwest Symposium on Circuits and Systems, Urbana, IL, pp. 1070-1073, Aug. 1989.

[10] I. Ahmad, C. Chen "Post-Processor For Data Path Synthesis Using Multiport Memories," in Proc. IEEE International Conference on Conputer-Aided Design, pp. 276-279, Nov. 1991.

[11] T. Kim, C. L. Liu, "Utilization of Multiport Memories in Data Path Synthesis," in Proc. 30th ACM/IEEE Design Automation Conference, pp. 298-302, June 1993.

[12] 전홍신, 황선영, "디지탈 신호처리를 위한 파이프라인 데이타패스 합성 시스템의 설계," 대한전자공학회 논문지, 30권 A편 6호, pp. 49-57, 1993년 6월.

[13] H. D. Lee, H. S. Jun, S. Y. Hwang, "Datapath Synthesis in Sogang Silicon Compiler," in Proc. JTC/CSCC, pp. 430-435, Dec. 1990.

[14] C. Papadimitriou, K. Steiglitz, 'Combinatorial Optimization: Algorithms and Complexity,' Prentice-Hall, pp. 247-266, 1982.

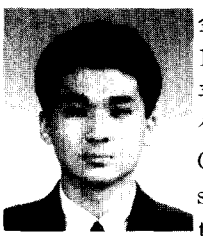
[15] I. Ahmad, C. Chen "Grouping Variables into Multiport Memories for ASIC Data Path Synthesis," in Proc. IEEE International ASIC Conference, pp. 162-165, Sept. 1992.

[16] B. S. Haroun, M. I. Elmasry, "Architecture Synthesis for DSP Silicon Compiler," IEEE Trans. on CAD, vol. 8, no. 4, pp. 431-477, April 1989.

著者紹介

李海東(正會員)

1990年 2월 서강대학교 전자공학과 졸업. 1992年 2월 서강대학교 전자공학과 석사 취득. 1994年 현재 서강대학교 전자공학과 박사과정 재학중. 주관심 분야는 CAD 시스템, high-level synthesis, Testable design/synthesis 등임.



金泳魯(準會員)

1993年 2月 서강대학교 전자공학과 졸업. 서강대학교 전자공학과 석사과정 재학 중. 주관심 분야는 CAD 시스템, high-level synthesis, Computer Architecture 등임.

著 者 紹 介

黃 善 泳(終身會員)

1976年 2月 서울대학교 전자공학과 졸업. 1978年 2月 한국 과학원 전기 및 전자 공학과 공학 석사 취득. 1986年 10月 미국 Stanford대학 공학 박사 학위 취득. 1976年 ~ 1981年 삼성 반도체 주식회사 연구원. 1986年 ~ 1989年 Stanford대학 Center for Integrated Systems 연구소 연구원, Fairchild Semiconductor Palo Alto Research Center 기술 자문. 1989年 3月 ~ 현재 서강 대학교 전자 공학과 교수. 주관심 분야는 CAD 시스템, Computer Architecture 및 Systems Design, VLSI 설계 등임.