論文95-32A-10-10

메모리 비용 최소화를 위한 데이타패스 합성 시스템의 설계

(Design of a Datapath Synthesis System for Minimization of Multiport Memory Cost)

李 海 東 *, 黃 善 泳 *

(Hae Dong Lee and Sun Young Hwang)

요 약

본 논문은 면적 측면에서 효율적인 레지스터 전송 수준의 데이타패스 생성을 위한 상위수준 합성 시스템에 대하여 기술한다. 제안된 스케쥴링 알고리듬은 주어진 제약 조건을 만족시키면서 최소의 메모리 비용으로 연산기를 최대한으로 공유할 수 있도록 연산을 특정 제어구간에 할당한다. 메모리 비용의 척도로서 각 제어구간에서 참조되는 변수인 MAV를 제안하였으며, 전체 제어구간에 대하여 MAV를 균등히 배분하므로써 전체적인 메모리 비용이 감소하였다. 실험결과를 통하여 제안된 알고리듬의 효율성을 보였다. 벤치마크 프로그램에 대하여 이전의 연구 결과와 비교할 때, 제안된 알고리듬이 스케쥴링 과정에서 메모리의 비용을 고려함으로써 적은 비용의 연결구조 및 메모리를 가지는 데이타패스를 생성함을 보인다.

Abstract

In this paper, we present a high-level synthesis system that generates area-efficient RT-level datapaths with multiport memories. The proposed scheduling algorithm assigns an operation to a specific control step such that maximal sharing of functional units can be achieved with minimal number of memory ports, while satisfying given constraints. We propose a measure of multiport memory cost, MAV (Multiple Access Variable) which is defined as a variable accessed at several control steps , and overall memory cost is reduced by equally distributing the MAVs throughout all the control steps. Experimental results show the effectiveness of the proposed algorithm. When compared with previous approaches for several benchmarks available from literature, the proposed algorithm generates the datapaths with less memory modules and interconnection structures by reflecting the memory cost in the scheduling process.

I. 서 론

상위수준 합성은 하드웨어 기술 언어 (Hardware Description Language)를 이용하여 설계하고자 하는 하드웨어를 행위 수준에서 기술한 형태를 입력으로 받아들여 여러 단계의 최적화 과정을 거쳐 레지스터 전송 수준 (Register Transfer Level)의 데이타패스를 자동적으로 생성하는 과정이다[1-2]. 레지스터 전송 수준의 데이타패스는 기본적으로 데이타의 저장을 위한 기억소자, 데이타를 이용하여 특정 연산을 수행하기 위한 연산기, 그리고 기억소자와 연산기 간의 데이타 전달을 위한 연결구조 소자로 구성된다. 대표적인 기억소자로는 레지스터 및 메모리를 들 수 있으며, 덧셈기, 곱셈기 등이 연산기에 해당된다. 연결구조 소자로는 버스 및 멀티플렉서가 사용된다. 일반적으로 상위수준 합성은 크게 스케쥴링과 모듈할당 과정으로 분리되어 수

* 正會員, 西江大學校 電子工學科 CAD 및 Computer Systems 硏究室

(CAD & Computer Systems Lab., Dept. of Electronic Engineering, Sogang University)

接受日字:1994年12月12日, 수정완료일:1995年10月2日

행되며, 각 과정은 상호 보완적인 관계를 가진다. 스케줄링 과정에서는 입력행위 기술에 존재하는 연산이 수행될 제어구간 (control step)을 결정하며, 최적의 연산기 사용을 목적으로 한다. 이 과정에서 시간 또는 면적 등이 설계 제약조건으로 주어질 수 있으며, 주어진 제약조건을 만족하면서 최적의 결과를 생성한다. 모듈 할당 과정은 스케줄링 결과를 이용하여 실질적인 하드웨어의 형태를 구성하는 단계로서, 레지스터 및 연산기의 할당, 연결구조의 구성 과정으로 세분화된다. 각 과정에서 전체적인 하드웨어의 비용을 감소시키기 위한 최적화 과정을 거친 후 최소 비용의 레지스터 전송 수준의 데이타패스를 생성함을 목적으로 한다.

상위수준 합성에 관한 연구는 1990년대에 들어서면서 급진적인 발전을 거듭하였으며, 현재는 연구 수준에서 벗어나 상용화를 위한 노력이 계속되고 있다. 이러한 추세와 함께 다양한 하드웨어 모듈을 지원함으로써 설계의 융통성을 제공하는 연구도 함께 진행되고 있으며, 기억소자로서 다중포트 메모리를 사용하는 설계 방식을 대표적인 예로 들 수 있다. 간단한 구조의 데이타패스, 테스트의 용이성 등의 다중포트 메모리가 제공하는 여러 가지 장점으로 인하여 이를 지원하는 여러 합성 시스템이 발표된 바 있다[3-8]. 이들 시스템은 스케줄링 과정이 완료된 후의 결과를 입력으로 하여 각 시스템 고유의 목적함수에 바탕을 둔 모듈할당 과정을 수행한다. 그러나, 스케줄링의 결과가 모듈할당 과정에 영향을 미친다는 점을 감안할 때, 메모리의 비용이 모듈할당 과정에서만 고려되어 비효율적인 결과를 초래할 가능성이 있다. 따라서, 스케줄링 과정에서 메모리의 비용을 고려하는 합성 방식이 요구되며, 메모리의 개수 및 포트 수와 같은 제약조건 하의 합성이 가능해야 한다.

본 논문에서는 메모리의 비용을 스케줄링 과정에서 고려함으로써 칩 면적 측면에서 효율적인 데이타패스를 자동적으로 생성하는 상위수준 합성 알고리듬에 대하여 기술한다. 메모리 모듈의 개수는 각 제어구간에서 동시에 참조되는 변수의 수에 의해 결정되므로, 스케줄링 과정에서 이들 변수의 수를 줄이는 방향으로 연산을 특정 제어구간에 할당하므로써 메모리의 비용을 감소시킬 수 있다. 스케줄링 과정에서는 입력 행위기술에 존재하는 연산을 각 제어구간에 균등히 분포시키므로써 연산기의 개수를 최소화하며, 이 과정에서 메모리 포트의 비용을 목적함수에 함께 고려하여 메모리의 비용을 감소시킨다. 스케줄링 과정의 설계 제약조건으로는 시간 및 면적 제약조건이 있다. 시간 제약조건으로서 입력 행위기술을 수행시키기 위한 제어구간의 수가 주어지며, 시간 제약조건하의 스케줄링은 주어진 제약조건을 만족하면서 연산기와 메모리의 비용을 최소화시킨다. 일반적으로 면적 제약조건으로는 합성에 사용될 수 있는 연산기 모듈의 개수로서 주어지나, 제안된 시스템은 연산기 제약조건과 함께 메모리 모듈의 개수 및 포트의 개수를 면적 제약조건으로 받아들인다. 면적 제약조건하의 스케줄링은 주어진 하드웨어 모듈을 사용하면서 최소의 제어구간으로 입력 행위기술을 수행하는 방향으로 연산을 특정 제어구간에 할당한다. 모듈 할당 과정에서는 스케줄링의 결과를 이용하여 입력 행위기술 내의 변수를 메모리 모듈에 할당한다. 할당 과정은 각 메모리 모듈의 크기 및 연결구조의 비용을 최소화하는 방향으로 수행되며, 제약조건으로서 하나의 메모리가 가질 수 있는 최대 포트 수가 주어진다.

본 논문의 구성은 다음과 같다. 본 논문에서 제시한 시스템의 개관 및 타겟 아키텍춰를 2 장에서 기술하며, 3 장에서 메모리의 비용을 고려하는 스케줄링 및 모듈 할당 알고리듬을 제시한다. 시스템의 성능 평가를 위한 실험 결과 및 타 시스템과의 비교를 4 장에서 보이고, 끝으로 결론을 5 장에서 제시한다.

II. 시스템 개관

본 논문에서 제시한 시스템 SODAS(SOgang Design Automation System)의 구성도를 그림 1에 보였다. VHDL[9]로 기술된 입력 행위기술은 시뮬레이터에 의한 검증 단계를 거쳐 중간형태로서 C/DFG (Control/Data Flow Graph)를 생성한다. 상위수준 합성 과정은 검증된 C/DFG를 바탕으로 하여 수행되고, 합성 과정에서 필요한 하드웨어 모듈의 사양에 대한 정보를 시스템 라이브러리에서 제공받는다. 그리고, 사용자가 정의한 시간/면적 측면의 설계 제약조건이 합성기의 입력으로 주어질 수 있다. 주어진 설계 제약조건은 합성 과정에 적합한 합성 제약조건으로 변환되어 반영되며, 합성 결과로서 데이타패스와 콘트롤러가 생성된다. 데이타패스는 VHDL 구조 기술로 출력되고, 이의 구동을 위한 콘트롤러는 상태전이표의 형태로 출력된다. SODAS의 타겟 아키텍춰는 기억 소자로서 다중포트 메모리를 사용하는 linear topology 형태[10]

의 레지스터 전송 수준의 데이타패스로서, 2 phase 클럭에 의하여 구동된다.

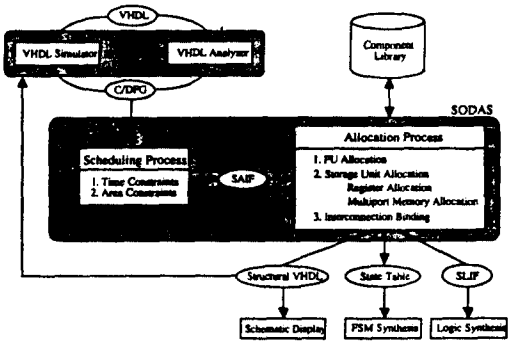


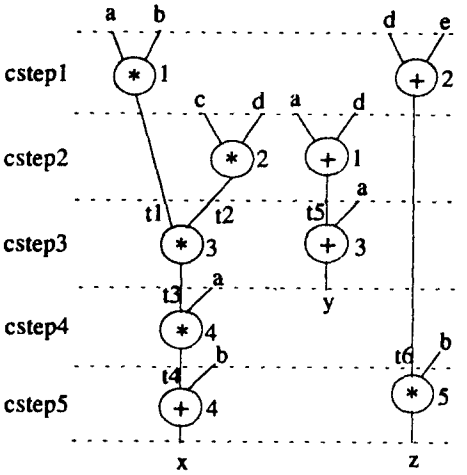
그림 1. SODAS 시스템의 구성도
Fig. 1. Overall system flow of SODAS.

Ⅲ. 메모리의 비용을 고려한 데이타패스 합성

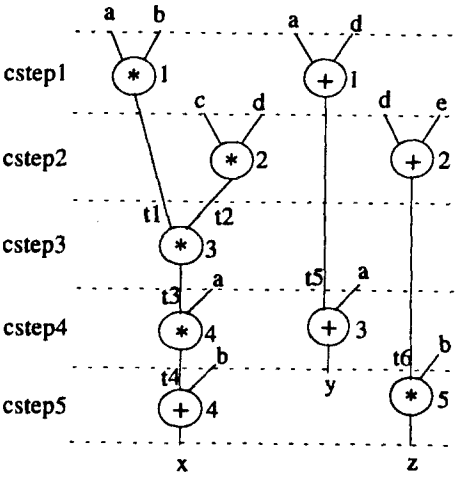
1. 스케쥴링 과정

스케쥴링 과정에서 적절한 목적함수의 설정은 전체 합성 시스템의 성능을 결정하는 중요한 요소이다. 연산기의 효율적인 사용은 입력 행위기술 내의 연산을 전체 제어구간에 균등히 분배하므로써 얻을 수 있으며[14], 대부분의 스케쥴링 알고리듬이 연산기의 개수를 최소화하기 위한 목적함수를 사용하고 있다. 제안된 스케쥴링 알고리듬은 연산기의 개수와 함께 메모리 포트의 개수를 최소화하여 전체적인 메모리의 비용을 감소시키는 방향으로 스케쥴링 과정을 수행한다. 제어구간에서 수행되는 각 연산은 피연산자 적재, 연산수행, 그리고 결과 저장의 부작업으로 분할된다. 메모리 모듈 내의 데이타는 피연산자의 적재 과정에서 연산기의 입력단에 주어지며, 결과 저장 과정에서 연산수행의 결과가 메모리에 저장된다. 이러한 적재 및 저장의 부작업을 데이타 전달 연산으로 정의하며, 이는 2-phase 클럭에 의해 수행된다. 메모리 모듈의 포트 개수는 데이타 전달 연산의 개수에 비례하므로, 스케쥴링 과정에서 데이타 전달 연산의 개수를 고려하여 전체적으로 요구되는 메모리 포트의 개수를 감소시킬 수 있다. 시간 제약조건으로 5 개의 제어구간을 주었을 때 가능한 두가지의 스케쥴링을 그림 2에 보였다.

두 스케쥴링 결과에서 연산기의 개수는 각 1 개의 덧셈기와 곱셈기를 사용하지만, 그림 2 (b)의 스케쥴링에서 데이타 전달 연산의 개수가 그림 2 (a)에 비해 감소함을 알 수 있다.



(a)



(b)

그림 2. 시간 제약조건 하의 스케쥴링 예
(a) 데이타 전달 연산을 고려하지 않은 경우
(b) 데이타 전달 연산을 고려한 경우
Fig. 2. Two different schedules
(a) not considering data transfers
(b) considering data transfers.

그림 2 (a)의 경우 최대 데이타 적재의 개수는 제어구간 1의 변수 a, b, d, e에 의하여 4 개로 결정된다. 이에 비하여 그림 2 (b)의 스케쥴링에서는 모든 제어구간에 대하여 3 개의 데이타 적재로 수행이 가능하다. 2-phase 클럭하에서 한개의 read-only 포트와 두개의 read/write 포트를 가진 메모리 모듈을 사용한다고 가정할 때, 그림 2 (a)의 스케쥴링에서는 2 개의

메모리 모듈이 요구되나 그림 2 (b)의 경우에는 한개의 메모리 모듈로 수행 가능함을 알 수 있다. 제안된 스케쥴링 알고리듬에서는 이러한 데이타 전달 연산을 목적함수에 반영하여 전체적인 메모리의 비용을 감소시킨다.

2. 목적함수

C/DFG에 존재하는 각 연산의 시간 프레임 구간(time frame interval)은 ASAP (As Soon As Possible) 및 ALAP (As Long As Possible) 스케쥴링에 의해 결정된다. 연산 opn의 ASAP와 ALAP 스케쥴링의 결과를 각각 b_{opn}과 e_{opn}으로 정의한다. 제안된 스케쥴링 과정의 목적 함수는 크게 두가지의 요소로 구성된다. 첫번째 요소는 제어구간에 대한 연산의 균등한 분포를 나타내는 척도로서, 각 연산의 시간 구간에 대한 함수로서 결정된다. OP 타입의 연산이 제어구간 i에 할당될 확률 $P_{OP}(i)$를 식 (1)에 보였으며, 분포 그래프 (distribution graph [4])의 값을 OP 타입의 연산의 개수로 나눈 형태로 주어진다. 식 (1)에서 N_{OP}는 OP 타입 연산의 개수를 나타내며, Prob(opn,i)는 연산 opn이 제어구간 i에 스케쥴될 확률을 의미한다.

$$P_{op}(i) = \sum_{opn \in OP} Prob(opn, i) / N_{OP} \quad (1)$$

$$\text{where } Prob(opn, i) = \begin{cases} 1/(e_{opn} - b_{opn} + 1), & \text{if } b_{opn} \le i \le e_{opn} \\ 0, & \text{otherwise} \end{cases}$$

각 연산 타입에 대하여 연산의 균등한 분포도를 나타내는 척도로서 엔트로피 함수[11]를 정의하였고, 이를 식 (2)에 나타냈다. H(OP)는 0과 1 사이의 값을 가지며, OP 타입의 연산이 모든 제어구간에서 동일한 확률 분포를 가질 때 최대값인 1이 된다. 반대로 모든 연산이 특정 제어구간에 집중되어 분포할 경우 최소값인 0을 가지게 된다.

$$H(OP) = -\sum_{i=1}^{\#csteps} P_{OP}(i) \cdot \log(P_{OP}(i)) / \log(\#csteps) \quad (2)$$

목적 함수의 두번째 요소는 전체 제어구간에 대한 데이타 전달 연산의 분포 상태를 나타낸다. 메모리 포트의 개수는 각 제어구간에서 수행되는 데이타 전달 연산의 수에 의해 결정된다. 따라서, 제안된 알고리듬은 데이타 전달 연산을 전체 제어구간에 대하여 균등히 분포시키므로써 메모리 포트의 수를 최소화한다. 입력 행위기술에서 한 변수가 여러 제어구간에서 참조될 수 있으며, 이를 MAV (Multiple Access Variable)이라 정의한다. 예로서, 그림 3 (a)의 변수 a는 cstep1, cstep2, cstep3, cstep4의 4 개의 제어구간에서 참조되나, 그림 3 (b)의 경우에는 cstep1과 cstep4의 2 개의 제어구간에서만 참조된다. 이는 그림 3 (b)의 스케쥴링 결과에서 메모리 포트의 수가 감소될 수 있음을 의미한다. 따라서, MAV를 전체 제어구간에 균등히 분포시키므로써 메모리의 비용을 감소시킬 수가 있다. 연산 opn에 의해 참조되는 변수의 집합을 *VarSet(opn)*으로 정의하자. 예로서, 그림 3에서 *VarSet* (*1)은 {a, b, t1}으로 결정된다. 하나의 변수 v가 제어구간 i에서 참조될 확률 Prob(v,i)를 식 (3)에 보였으며, V를 MAV 변수의 집합으로 정의할 때 MAV 변수가 제어구간 i에서 동시에 참조되는 확률을 식 (4)에 보였다. 그리고, 전체 제어구간에 대한 MAV 변수의 확률 분포를 나타내는 엔트로피 함수 H(V)를 식 (5)에 나타냈다. H(V)의 값이 높을수록 각 제어구간 i에 대해 확률 $P_V(i)$가 균등히 분포되어 있음을 나타내며, 이는 적은 개수의 메모리 포트를 사용함을 의미한다.

$$Prob(v, i) = \prod_{\text{for all opn st. } v \in VarSet(opn)} \frac{1}{e_{opn}(v) - b_{opn}(v) + 1} \quad (3)$$

where $b_{opn}(v)$ and $e_{opn}(v)$ are ASAP and ALAP schedules of operation *opn* which uses an MAV v as operand.

$$P_V(i) = \sum_{v \in V} Prob(v, i) \quad (4)$$

$$H(V) = -\sum_{i=1}^{\#csteps} P_V(i) \cdot \log(P_V(i)) / lob(\#csteps) \quad (5)$$

그림 2의 C/DFG에 대한 초기 스케쥴링 상태를 그림 3에 보였다. 그림 3 (a)는 각 연산의 시간 프레임 구간을 나타내며, MAV 변수의 생존구간(lifetime)을 그림 3 (b)에 보였다. MAV 변수의 생존구간은 그 MAV 변수를 참조하는 연산의 시간 프레임 구간의 집합으로 주어진다. 그림 3 (b)에서 보인 바와 같이 MAV 집합 V는 {a, b, d}로 결정되고, 변수 a의 생존구간은 연산 *1, *4, $+_1$, $+_3$의 시간 프레임 구간의 집합으로 주어진다. 그림 3 (c)는 연산과 MAV 변수에 대한 확률 분포를 보인다. 그림 3 (c)에서 연산과 MAV의 엔트로피 값은 각 제어구간 i에서의 확률 $P_{OP}(i)$와 $P_V(i)$를 식 (2)와 식 (5)에 대입하여 산출된다. 예로서, MAV의 엔트로피 값은 (0.13*log0.13 + 0.19*log0.19 + 0.19*log0.19 + 0.3*log0.3 +

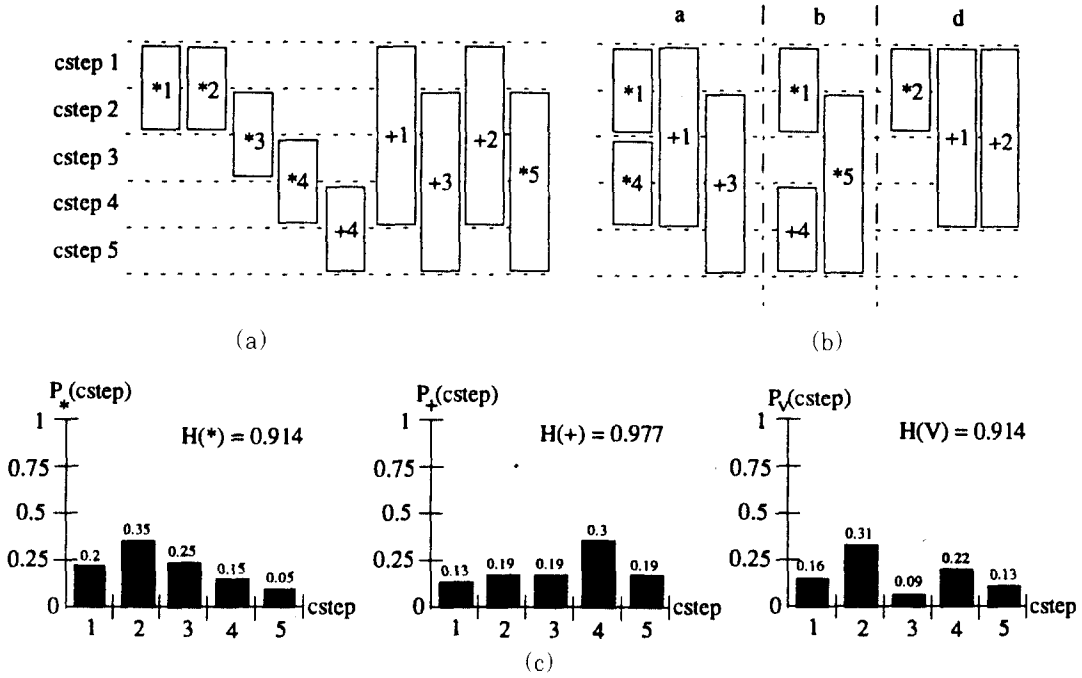


그림 3. 그림 2의 C/DFG에 대한 초기 스케쥴링 상태
(a) 연산에 대한 시간 프레임 구간 (b) MAV 변수에 대한 생존구간 (c) 각 제어구간에 대한 연산 및 MAV 변수의 분포 확률

Fig. 3. Initial scheduling state for the C/ DFG in Figure 2
(a) Time frame intervals for operations (b) Lifetimes for MAVs (c) Probabilities for operations and MAVs in each control step.

0.19*log0.19)/log5 = 0.914로 결정된다. 같은 방식으로 덧셈 연산과 곱셈 연산에 대한 엔트로피 값인 H(+)와 H(*)는 각각 0.914와 0.977로 계산된다. 그림 3 (c)에서 엔트로피의 값이 클수록 각 제어구간에서의 확률이 균등히 분포됨을 알 수 있다.

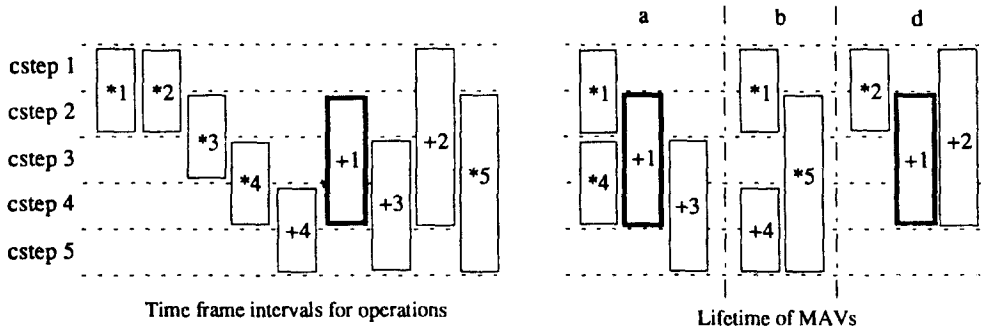
스케쥴링 과정은 그림 3과 같은 초기 상태에서 시작된다. 스케쥴링 과정이 진행되면서 각 연산의 시간 프레임 구간의 길이는 점차 줄어들게 되고, 최종적으로 모든 연산의 시간 프레임 구간이 하나의 제어구간에 고정되므로써 스케쥴링 과정이 완료된다. 이러한 과정의 한 시점에서 전체 연산의 시간 프레임 구간은 하나의 스케쥴링 상태로서 관리되고, 목적 함수에 의하여 상태전이가 수행된다. 따라서, 목적 함수는 각 스케쥴링 상태에 대하여 정의되며, 연산과 MAV 변수의 엔트로피 값의 함수로 표현된다. 식 (6)에 상태전이를 위한 목적 함수를 보였다.

$$OF(S) = \sum_{\text{for all OP types}} H(OP) \cdot W_{OP} + H(V) \cdot W_p \quad (6)$$

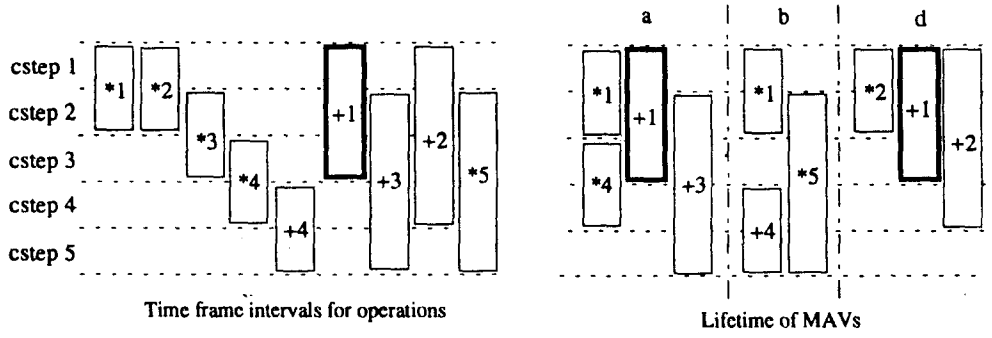
식 (6)에서 WOP는 'OP' 타입의 연산에 대한 가중치로서 라이브러리에 존재하는 해당 하드웨어 모듈의 비용을 의미하며, Wp는 메모리 포트의 비용을 나타낸다. 임의의 스케쥴링 상태에서 상태전이를 수행할 경우, 식 (6)의 목적 함수 값이 최대인 스케쥴링 상태를 다음 상태로 결정하므로써 효율적인 메모리 모듈의 사용과 최적의 연산기 공유를 얻을 수 있다. 임의의 스케쥴링 상태 S에서 연산 opn의 시간 프레임 구간 [bopn, eopn]을 [bopn+1, eopn]와 [bopn, eopn-1]로 변환시킬 때의 스케쥴링 상태를 각각 Sbopn 및 Seopn으로 정의하고, 이들을 상태 S의 이웃 상태로 명명한다. 스케쥴링 상태 S에서 이웃 상태로 상태 전이를 수행할 때의 이득은 이웃 상태에 대한 목적 함수의 미분값에 비례하게 되며, 연산 opn에 대한 우선순위 함수를 식 (7)에 보였다.

$$PF(opn) = \frac{|OF(Sb_{opn}) - OF(Se_{opn})|}{e_{opn} - b_{opn}} \quad (7)$$

그림 3 (a)의 초기상태에서 $+_1$ 의 시간 프레임 구간은 [1,4]로 주어진다. 다음 상태로서 Sb_{+_1}와



(a)



(b)

그림 4. 연산 +1에 대한 이웃 상태 (a) Sb_{+1} , (b) Se_{+1}

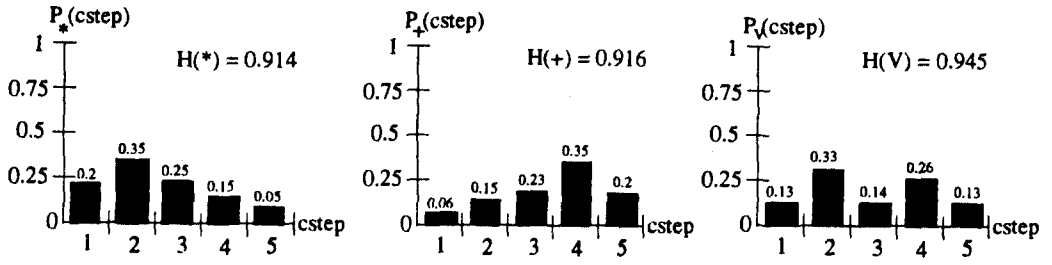
Fig. 4. Two neighbor states for operation $+_1$, (a) Sb_{+1} , (b) Se_{+1}.

Se_{+1}는 각각 시간 프레임 구간을 [2,4]와 [1,3]로 변환한 상태로 얻을 수 있으며, 이를 그림 4에 보였다. 그리고, 각 상태에서 MAV의 생존 구간을 함께 보였으며, 변화된 시간 프레임 구간은 굵은 선으로 표시하였다. Sb_{+1}의 경우 +1의 시간 프레임 구간이 변화함에 따라 연산 +3의 시간 프레임 구간도 함께 변화함을 알 수 있다.

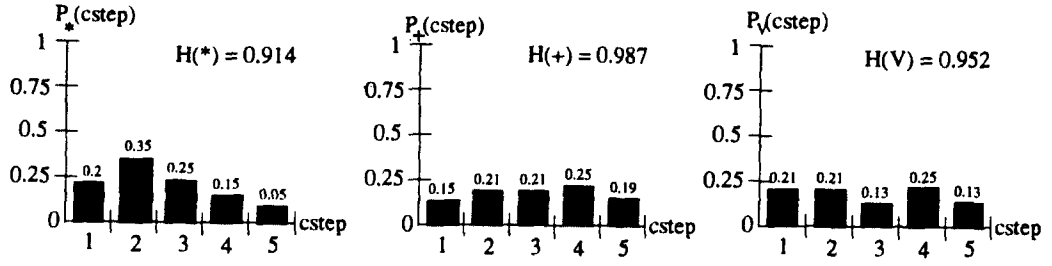
그림 4의 이웃 상태에서 각 제어구간에 대한 연산과 MAV 변수의 확률은 식 (1)과 (4)를 이용하여 산출될 수 있으며, 이를 그림 5에 보였다. 그리고, 각 연산 타입과 MAV의 엔트로피 값도 함께 제시하였다. 이웃 상태의 확률 분포를 그림 3의 초기 상태와 비교할 때, 덧셈 연산과 MAV의 확률이 더욱 균등히 분포되었음을 알 수 있으며, 이는 덧셈기와 메모리 포트의 수가 이웃 상태로의 상태 전이를 통해 감소할 수 있음을 의미한다. 각 이웃 상태의 목적 함수 값은 그림 5에서 제시된 연산과 MAV의 엔트로피 값을 식 (6)에 대입하여 구할 수 있으며, 이들 중에서 최대값을 가지는 상태로 상태 전이가 수행된다. 상태 전이에 의해 곱셈 연산의 시간 프레임 구간은 변하지 않으므로, 이웃 상태에서의 곱셈 연산에 대한 엔트로피 값은 변화가 없다. 그러나, 두 이웃 상태에서의 엔트로피 값을 비교할 때, Se_{+1}로의 전이를 통하여 덧셈 연산과 MAV의 엔트로피 값이 더욱 증가하므로 Se_{+1}를 다음 상태로 선택하게 된다.

3. 시간 제약조건하의 스케쥴링

그림 6에 시간 제약조건하의 스케쥴링 알고리듬을 보였다. 시간 제약조건은 입력 행위기술을 수행하기 위한 최대 제어구간의 수로서 주어진다. ASAP와 ALAP 스케쥴링을 이용하여 초기 스케쥴링 상태를 결정하고, 모든 연산이 스케쥴링될 때까지 각 연산에 대하여 최대의 우선순위 함수 값을 가지는 상태 전이을 수행하므로써 주어진 시간 제약조건내에서 연산기와 메모리의 비용을 최소화한다.



(a)



(b)

그림 5. 연산과 MAV에 대한 확률 분포 (a) Sb_{+1}, (b) Se_{+1}
Fig. 5. Probabilities for operations and MAVs (a) in Sb_{+1}, (b) in Se_{+1}.

단계 1 : ASAP와 ALAP 스케쥴링을 수행하여 초기 스케쥴링 상태를 구한다.
단계 2 : 모든 연산이 스케쥴링되었으면 중지한다.
단계 3 : 스케쥴링되지 않은 각 연산 opn에 대하여 다음을 수행한다.
  단계 3.1 : 식 (6)을 이용하여 목적함수 OF (Sb_{opn})와 OF (Se_{opn})를 구한다.
  단계 3.2 : 식 (7)을 이용하여 우선순위 함수 PF(opn)을 구한다.
단계 4 : PF(opn)이 최대가 되는 상태로 전이하고 단계 2로 간다.

그림 6. 시간 제약조건하의 스케쥴링 알고리듬
Fig. 6. Scheduling algorithm under time constraints.

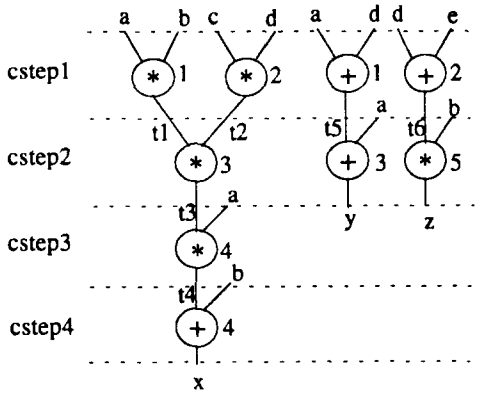
4. 면적 제약조건하의 스케쥴링

면적 제약조건하의 스케쥴링 방식에서 가장 보편적으로 사용되는 알고리듬으로서 리스트 스케쥴링을 들 수 있으며, 면적 제약조건으로는 사용 가능한 연산기의 개수로 주어진다[4]. 제안된 시스템에서는 면적 제약조건으로 연산기의 개수와 함께 메모리에 대한 제약조건을 받아들여 다양한 설계 공간을 제공한다. 면적 제약조건하의 스케쥴링 과정의 목적은 주어진 면적 제약조건을 만족하면서 최소의 메모리 비용과 함께 연산기를 최대한으로 공유하는 데 있다. 면적 제약조건하의 스케쥴링 알고리듬을 그림 7에 보였다. 연산기의 제약조건은 각 연산기 타입에 대하여 사용 가능한 최대 개수로서 주어지며, M_{OP}를 'OP' 타입 연산기의 최대 개수로 정의한다. 메모리 모듈의 제약조건은 설계 제약조건으로 사용 가능한 메모리의 종류, 포트의 개수 등을 사용한다. 이들은 설계 제약조건으로서 실제 합성 과정에 적합한 합성 제약조건으로 변환된다. 한 제어구간에서 동시에 사용될 수 있는 최대의 적재 및 저장의 데이타 전달 연산을 각각 M_r과 M_w로 정의하고, 이들이 합성 제약조건으로 주어진다.

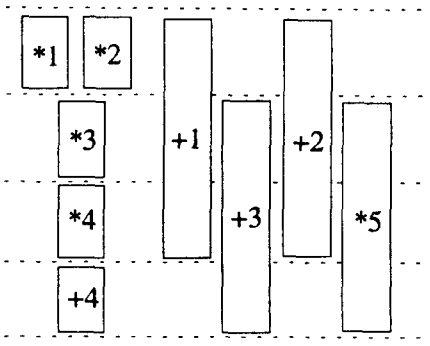
단계 1 : ASAP와 ALAP 스케쥴링을 수행하여 초기 스케쥴링 상태를 구한다.
단계 2 : M_{OP} = 한 제어구간에서의 최대 'OP' 타입 연산의 수
  M_r = 한 제어구간에서의 최대 적재 연산의 수
  M_w = 한 제어구간에서의 최대 저장 연산의 수
단계 3 : cstep = 1
단계 4 : 모든 제어구간에서 면적 제약조건을 만족하면 중지한다.
단계 5 : 현재의 cstep에 대한 레디리스트를 구한다.
단계 6 : N_{OP} = 레디리스트 내의 'OP' 타입 연산의 개수
  N_r = 레디리스트 내의 적재 연산의 개수
  N_w = 레디리스트 내의 저장 연산의 개수
단계 7 : N_{OP}〉M_{OP} 또는 N_r〉M_r 또는 N_w〉M_w일 때 단계 7을 반복한다.
  단계 7.1 : 레디리스트의 모든 연산이 임계경로에 있으면 모든 연산의 e_{opn}을 1씩 증가시키고 단계 8로 간다.
  단계 7.2 : 레디리스트 내의 모든 연산에 대하여 Se_{opn}의 우선순위 함수를 구한다.
  단계 7.3 : 최대의 우선순위 함수값을 가지는 상태로 상태전이를 한다.
단계 8 : cstep을 하나 증가시키고 단계 5로 간다.

그림 7. 면적 제약조건하의 스케쥴링 알고리듬
Fig. 7. Scheduling algorithm under resource constraints.

그림 7의 알고리듬에서 제어구간 cstep에 스케쥴링 될 수 있는 연산의 집합을 레디리스트로 정의한다. 레디리스트 내의 'OP' 타입 연산의 개수를 N_{OP}로 정의한다. 그리고, 레디리스트 내의 적재 및 저장 연산의 개수를 각각 N_r과 N_w로 정의한다. 제어구간 cstep에 대하여 레디리스트 내의 'OP' 타입의 연산 또는 데이타 전달 연산의 개수가 주어진 제약조건을 만족하지 못할 경우 우선순위가 낮은 연산을 다음 제어구간으로 지연시키고, 전체 제어구간에서 면적 제약조건을 모두 만족할 때 스케쥴링을 종료한다. 그림 8에 그림 2의 C/DFG에 대한 ASAP 스케쥴링 결과과 연산에 대한 시간 프레임 구간을 보였다.



(a)



(b)

그림 8. 초기 스케쥴링 상태 (a) ASAP 스케쥴링 (b) 시간 프레임 구간

Fig. 8. Initial scheduling state, (a) ASAP scheduling, (b) time frame intervals.

두 개의 곱셈기와 한 개의 덧셈기를 연산기 제약조건으로 주어지고 네 개의 적재와 세 개의 저장 연산이 메모리 제약조건으로 주어졌다고 가정할 때, 제어구간 1에서의 레디리스트는 ASAP 스케쥴링의 결과가 1인 연산인 {*1, *2, +1, +2}로 구성된다. 연산기 제약조건으로서 두 개의 곱셈기가 사용 가능하므로 연산 *1과 *2는 제어구간 1에 스케쥴링될 수 있다. 그러나, 제약조건으로 주어진 덧셈기의 개수는 한 개이므로 제어구간 1에 연산 $+_1$과 $+_2$가 동시에 스케쥴링될 수는 없다. 따라서, 주어진 제약조건을 만족하기 위해서는 두 덧셈 연산 중의 하나가 다음 제어구간으로 지연되어야 하고, 식 (7)의 우선순위 함수를 이용한 상태 전이를 통해 지연시킬 연산을 결정하게 된다. 그림 9에 연산 $+_1$과 $+_2$에 대한 이웃 상태와 각 상태에서의 MAV에 대한 확률 분포를 보였다.

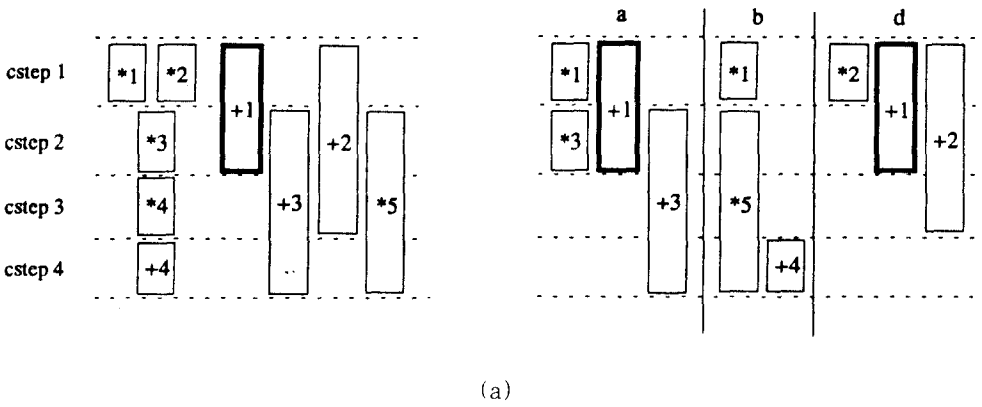
그림 9 (a)와 (b)는 각각 이웃 상태인 Se+1과 Se+2를 나타낸다. 연산에 대한 엔트로피 값은 두 상태에서 동일하나, MAV에 대한 엔트로피 값은 변화된 형태로 나타난다. 상태 Se+1의 경우 엔트로피 값은 0.985로서 상태 Se+2의 0.946에 비하여 큰 값을 보이며, 이는 각 제어구간에 대한 MAV의 확률 분포가 Se+1 상태에서 더 균등히 분포됨을 의미한다. 따라서, Se+1로 상태 전이가 수행되므로써 메모리의 포트 비용이 더 감소될 수 있다.

5. 모듈 할당 과정

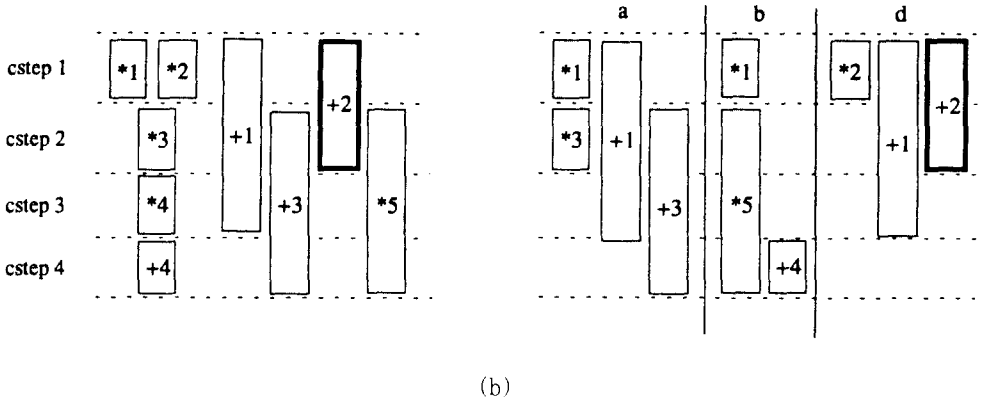
모듈 할당 과정은 스케쥴링 결과를 바탕으로 다중포트 메모리 기반 데이타패스를 구성하는 과정이다. 모듈 할당에서의 메모리 비용 최소화 과정은 각 제어구간에서 참조되는 변수의 메모리 포트 매핑 과정에 의해 좌우된다[12]. SODAS 시스템의 모듈 할당 과정은 메모리의 비용과 연결구조 비용을 합께 고려한 목적함수를 사용한 다중분할 매칭 (bipartite matching) 알고리듬[14]을 사용하여 메모리 비용을 최소화하였으며, 교환법칙 성립이 가능한 연산기에 대하여 그래프 채색 (graph coloring) 알고리듬[14]을 적용하여 연결구조의 비용을 감소시킴로써 전체적으로 하드웨어의 비용을 최소화한다[15].

Ⅳ. 실험 결과

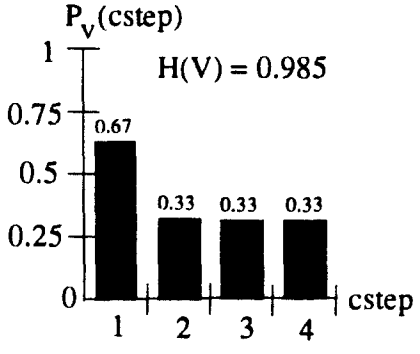
제안된 시스템은 SUN SPARC 워크스테이션 상의 UNIX 운영체제하에서 C 언어를 사용하여 구현되었다. 시스템의 성능을 보이기 위하여 MCNC 벤치마크



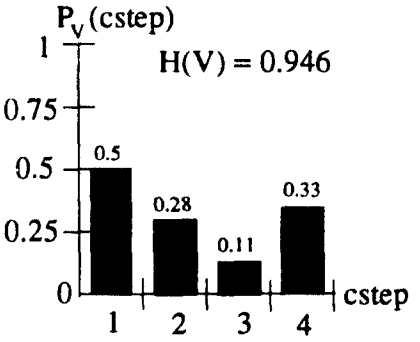
(a)



(b)



(c)



(d)

그림 9. 연산 $+_1$와 $+_2$에 대한 이웃상태
(a) Se_{+1}의 이웃상태, (b) Se_{+2}의 이웃상태, (c) (a)에 대한 P_V, (d) (b)에 대한 P_V
Fig. 9. Neighbor states Se for $+_1$ and $+_2$ in Figure 8
(a) neighbor state Se_{+1}, (b) neighbor state Se_{+2}, (c) P_V for (a), (d) P_V for (b).

프로그램에 대한 실험을 수행하였으며, 기존의 시스템과의 비교를 통하여 제안된 알고리듬의 효율성을 보였다. 미분방정식 연산기, 6차 밴드패스 필터 그리고 5차 엘립틱 웨이브 필터 등의 벤치마크 프로그램이 실험에 사용되었으며, 시간 및 면적 제약조건하의 실험결과도 함께 제시하였다.

1. 미분방정식 연산기에 대한 합성 결과

표 1은 여러 시간 제약조건에 대하여 메모리의 비용을 고려한 스케쥴링의 결과를 보이며, 메모리의 비용을 고려하지 않고 합성한 결과와의 비교도 제시하였다. 메

모리의 비용을 고려하지 않은 합성 결과는 식 (6)의 목적 함수에서 메모리 포트 비용인 W_p를 0으로 설정하여 산출되었다. 두 경우 모두 연산기의 개수 측면에서는 동일한 결과를 보이나, 메모리의 비용을 고려한 스케쥴링의 경우 감소된 데이타 전달 연산을 보인다.

표 1. 시간 제약조건하에서의 미분방정식 연산기에 대한 합성 결과.

Table 1. Synthesis results under several time constraints when reflecting the cost of memory ports for the differential equation solver.

#csteps		4	5	6	7	8	9	10	≥11
Scheduling with port cost considered	#read ops	6	4	4	4	4	3	3	2
	#write ops	4	3	3	2	2	2	2	1
	#RAMs*	2	2	2	2	2	1	1	1
	#ALUs	2	2	2	1	1	1	1	1
	#Mults	2	1	1	1	1	1	1	1
Scheduling without port cost considered	#read ops	6	6	6	4	4	4	4	2
	#write ops	4	3	3	2	2	2	2	1
	#RAMs*	2	2	2	2	2	2	2	1
	#ALUs	2	2	2	1	1	1	1	1
	#Mults	2	1	1	1	1	1	1	1

* a RAM with 1 read only ports and 2 read-write ports

표 2에 다양한 연산기 제약조건하에서의 미분방정식에 대한 합성결과를 보였으며, 스케쥴링 과정에서 메모리의 비용을 고려할 경우와 고려하지 않을 경우의 결과를 비교하였다. 표 2에서 연산기 제약조건은 (곱셈기의 수, ALU의 수)를 나타내며, 실험에 사용된 메모리 모듈은 1 개의 read-only 포트와 2 개의 read/write 포트를 가진다. 각 제약조건에서 전체 제어구간의 수는 두 경우 모두 최적의 결과를 보이며, 연산기의 개수에서도 같은 결과를 생성하였다. 그러나, 스케쥴링 과정에서 메모리의 비용을 고려하므로써 데이타 전달 연산의 수가 감소함을 보이며, 메모리 모듈의 개수 측면에서도 유리한 결과를 생성하였다.

2. 6차 밴드패스 필터에 대한 합성 결과

표 3에서 제어구간 11의 시간 제약조건하에서 6차 밴드패스 필터에 대한 합성 결과를 타 시스템과 비교하였다. 연산기의 개수는 1 개의 곱셈기와 2 개의 ALU를 사용하였고, 모든 시스템에서 동일한 결과를 보인다. 버스의 개수도 5 개로 같은 결과를 보이나 mux 입력 비용에서 SODAS 시스템이 유리한 결과를 보인다. 버스의 구동을 위한 TSB의 개수는 SODAS와 참고문헌 [12]의 시스템이 최소의 결과를 보였다. 메모리의 개수는 SODAS, 참고문헌 [12]의 시스템 그리고 MAP 시스템[10]이 모두 2 개의 메모리를 사용하였으나, 참고문헌 [12]의 시스템은 5개의 read/ write 포트를 사용하는 반면 SODAS 시스템의 경우에는 2 개의 read-only 포트와 1 개의 write-only 포트 그리고 2 개의 rad-write 포트를 사용하므로써 포트 비용 측면에서 SODAS가 가장 유리한 결과를 보인다.

표 2. 면적 제약조건하에서의 미분방정식 연산기에 대한 합성 결과.

Table 2. Synthesis results under various FU constraints with and without memory port cost for the differential equation solver.

FU constraints*		(1,1)	(1,2)	(2,1)	(2,2)
Scheduling with port cost considered	#csteps	7	6	5	4
	#readops	3	5	4	6
	#writeops	2	3	3	4
	#RAMs**	1	2	2	2
	#ALUs	1	2	1	2
	#Mults	1	1	2	2
Scheduling without port cost considered	#csteps	7	6	5	4
	#readops	4	6	6	6
	#writeops	2	3	3	4
	#RAMs**	2	2	2	2
	#ALUs	1	2	1	2
	#Mults	1	1	2	2

* FU constraints : (M_{mult}, M_{ALU})
** RAM with one read-only port and two read/write ports

3. 5차 엘립틱 웨이브 필터에 대한 합성 결과

표 4에서 제어구간 17의 시간 제약조건 하에서의 5차 엘립틱 웨이브 필터에 대한 SODAS의 합성 결과를 타 시스템과 비교하였다. 레지스터의 개수 측면에서는 GMD[16]와 SODAS 시스템이 최소의 결과를 보인다. 이는 두 시스템 모두 메모리 합성 과정에서 레지스터의 비용을 고려하기 때문이다. 연결구조의 비용 측면에서는 SODAS가 가장 유리한 결과를 보인다. 사용된 메모리 모듈의 개수는 SODAS, 참고문헌 [12]의 시스템, MAP 시스템이 최소의 결과를 보였다. SODAS와 참고문헌 [12]의 시스템을 비교할 때, 두 시스템 모두 2 개의 메모리 모듈을 사용하지만 포트 비용은 SODAS가 적은 결과를 보인다. 전체적인 데이타패스의 비용 측면에서 볼 때, SODAS 시스템이 최소 비용의 데이타패스를 가장 빠른 시간 내에 생성함을 알 수

있다.

표 3. 6차 밴드패스 필터에 대한 합성 결과의 비교
Table 3. Comparisons of synthesis results for the 6th order bandpass filter.

	SODAS	Ref [12]	MAP [10]	ADPS [6]
#Csteps	11	11	11	11
#ALUs	2	2	2	2
#Multipliers	1	1	1	1
#Mux Inputs	9	10	12	27
#TSBs	6	6	7	N/A
#Buses	5	5	5	N/A
#Registers	11	11	11	11
#RAMs	2^a	2^b	2^c	N/A
#ROMs	1	1	1	N/A
#CPU Time(sec.)	0.76	N/A	1.01	197*

a : a RAM with 1 read-only port and 1 write-only port, and a RAM with 1 read-only port and 2 read/write ports
b : a RAM with 2 read/write ports, and a RAM with 3 read/write ports
c : Unknown
* include scheduling time

표 4. 5차 엘립틱 웨이브 필터에 대한 합성 결과의 비교
Table 4. Comparison of synthesis results for the 5th order elliptic wave filter.

	SODAS	Ref[12]	MAP	GMD[16]	STAR[5]	SPAID[17]	HAL[4]
#Csteps	19	19	19	19	19	19	19
#Adders	2	2	2	2	2	2	2
#Multipliers	1	1	1	1	1	1	1
#MUX Inputs	8	9	10	N/A	11	17	26
#TSBs	5	5	5	N/A	12	N/A	N/A
#BUSes	5	5	5	N/A	5	5	6
#Registers	11	12	14	11	13	19	12
#RAMs	2a	2b	2c	2c	5d	5e	N/A
#ROMs	1	1	1	1	1	1	1
CPU Time(sec.)	0.97	N/A	1.33	N/A	N/A	N/A	360*

a : a RAM with 2 R/W ports and a RAM with 2 R ports and 1 R/W port
b : a RAM with 2 R/W ports and a RAM with 3 R/W ports
c : Unknown
d : RAMs with 1 R port and 1 W port
e : RAMs with 1 R/W port
* includes scheduling time

V. 결 론

본 논문은 메모리의 비용 측면에서 효율적인 레지스터 전송 수준 데이타패스의 생성을 위한 휴리스틱 스케줄링 알고리듬에 대하여 기술하였다. 제안된 스케줄링 알고리듬은 주어진 시간/면적 제약조건을 만족시키면서 최소의 메모리 비용으로 연산기를 최대한으로 공유할 수 있도록 연산을 특정 제어구간에 할당한다. 메모리 비용의 척도로서 각 제어구간에서 참조되는 변수인 MAV를 제안하였으며, 전체 제어구간에 대하여 MAV를 균등히 배분하여 전체적인 메모리 비용이 감소시킬 수 있었다. 모듈할당 과정에서는 연산기와 메모리의 비용과 연결구조 비용 등의 전반적인 하드웨어 비용을 감소시키는 방향으로 메모리 합성을 수행하였다. 벤치마크 프로그램에 대하여 이전의 연구 결과와 비교할 때, 제안된 알고리듬이 스케줄링 과정에서 메모리의 비용을 고려함으로써 적은 비용의 메모리 및 연결구조를 가지는 데이타패스를 생성함을 보였다.

감사의 글

※ 본 연구는 서울대학교 반도체 공동 연구소의 교육부 반도체 분야 학술연구 조성비 (ISRC 95-E-2007)에 의해 연구되었음.

참 고 문 헌

[1] D. Gajski, 'Silicon Compilation,' Addison-Wesley Pub.: Reading, MA, pp. 3-46, 1988.

[2] M. McFarland, A. Parker, R. Camposano, "The High-Level Synthesis of Digital Systems," Proc. of IEEE, Vol. 78, No. 2, pp. 301-318, Feb. 1990.

[3] C. Tseng, D. Siewiorek, "Automated Synthesis of Data Path in Digital Systems," IEEE Trans. on CAD, Vol. 5, No. 3, pp. 379-395, July 1986.

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

[5] F. Tsai, Y. Hsu, "Data Path Construction and Refinement," in Proc. IEEE International Conference on Computer-Aided Design, ICCAD-90, pp. 308-311, Nov. 1990.

[6] C. 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.

[7] M. Balakrishnan, A. Majmudar, D. Banerji, J. Linders, J. Majithia, "Allocation of Multiport Memories in Data

Path Synthesis," IEEE Trans. on CAD, Vol. 7, No. 4, pp. 536-540, Apr. 1988.

[8] T. Wilson, D. Banerji, J. Majithia, 7A. 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.

[9] "IEEE Standard VHDL Language Reference Manual," IEEE Std 1076-1987, IEEE, NY, Mar. 1988.

[10] I. Ahmad, C. Chen, "Post-Processor for Data Path Synthesis Using Multiport Memories," in Proc. IEEE International Conference on Computer-Aided Design, ICCAD-91, pp. 418-421, Nov. 1991.

[11] H. S. Jun, S. Y. Hwang, "Design of a Pipelined Datapath Synthesis System for Digital Signal Processing," IEEE Trans. VLSI Systems, Vol. 2, No. 3, pp. 292-303, Sept. 1994.

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

[13] C. Huang, Y. Chen, Y. Lin, U. Hsu, "Data Path Allocation Based on Bipartite Weighted Matching," in Proc. 27th ACM/IEEE Design Automation Conference, pp. 499-504, June 1990.

[14] S. Baase, 'Computer Algorithms,' Addison-Wesley Pub.: Reading, MA, pp. 346-354, 1988.

[15] H. D. Lee, S. Y. Hwang, "Design of an Area-Efficient Allocation Algorithm for Datapaths with Multiport Memories," Journal of Microelectronic Systems Integration, Vol. 3, No. 1, pp. 3-17, 1995.

[16] 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.

[17] B. Haroun, M. Elmasry, "Architecture Synthesis for DSP Silicon Compiler," IEEE Trans. on CAD, Vol. 8, No. 4, pp. 431-477, Apr. 1989.

저 자 소 개

李 海 東(正會員)

1990년 2월 서강대학교 전자공학과 졸업. 1992년 2월 서강대학교 전자공학과 석사 취득. 1992년 3월~서강대학교 전자공학과 박사과정 재학중. 주관심 분야는 high-level synthesis, testable synthesis, design for low power 등임.

黃 善 泳(定會員)

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