論文 90-27-8-20

데이타 경로 합성을 위한 하드웨어 할당 알고리듬

(A Hardware Allocation Algorithm for Data Path Synthesis)

金 洪 植*, 慶 宗 旻*

(Hong Shig Kim and Chong Min Kyung)

要 約

이 논문에서는 디지탈 시스템 데이타 경로 합성을 위한 시스템에 대하여 기술하고 있다. 디지탈 시스템을 자동으로 합성하기 위해서는 scheduling, 레지스터 할당, 연산자 할당, 버스 할당이라는 4가지 작업이 필요하다. 본 논문에서는 scheduling은 기존에 나와 있는 알고리듬을 사용하고, 나머지 세가지 할당과정을 위한 알고리듬들을 제시하였다. MC6502를 포함한 2가지 데이타 경로에 대한 합성결과로부터 이 논문에서 제시한 알고리듬이 기존의 결과들보다 우수함을 확인할 수 있었다.

Abstract

This paper describes the design and implementation of a system for automatic data path synthesis in digital system. There are four subtasks to synthesize a digital system: scheduling, register allocation, functional unit allocation and bus allocation. In this paper, force directed algorithm is used for the scheduling while new algorithms are proposed for the allocation subtasks. Synthesis results of two experimental data paths including MC6502 have shown that our proposed algorithm out goes most of previous works.

I. 서 론

VLSI칩 및 시스템의 설계에 있어서 상위단계의 설계작업은 원하는 동작기능을 레지스터(register), 연산자(functional unit), 버스(bus)등을 사용하여 실현하는 과정으로서 이에 필요한 설계시간을 줄이기 위한 설계자동화연구가 활발하게 진행되고 있는데, 이를 high-level synthesis라고 한다. 즉 high-level synthesis는 원하는 시스템의 behavioral specification을 받아들여서 레지스터, 연산자, 버스등으로 구성된 register transfer level hardware를 합성하는 작업을 말한다. 디지탈 시스템은 데이타 경로와 콘트롤러 부분으로 나눌 수 있는데 본 논문에서는 데이타 경로 합성을 다루고 있다. 디지탈 시스템을 합성하기 위해서는 behavioral specification을 data flow graph(DFG)라는 중간형태로 변형시킨 후에 scheduling, 레지스터 할당(register allocation), 연산자 할당(functional unit allocation), 그리고 버스할당(bus allocation)등 4가지 작업이 필요하다. 기존의 데이타 경로합성 시스템 중에서 몇가지를 보면 아래와 같다.

FACET[3]:이 시스템은 scheduling이 된 상태에서 할당 과정을 담당하는 것으로 할당 문제들을 graph 문제로 modelling하여 clique partitioning algorithm을 사용하여 풀었다.

HAL[2]:HAL은 scheduling 과정은 force directed scheduling algorithm을 사용하였고, 할당 과정은 rule based expert system을 사용하였다. 본 논문에서 사용된 scheduling algorithm은 HAL에서 사용된

*正會員, 韓國科學技術院
(Dept. of Electrical Eng., KAIST)
接受日字 : 1990年 6月 23日

것과 같은 것이다.

DAA[5]:DAA는 knowledge based expert system으로 먼저 storage element allocation을 한 후 data flow graph상의 operation들을 hardware module에 할당하는데, expert analysis phase를 두어서 optimization을 한다.

본 논문의 시스템에서는 scheduling은 HAL[2]에서 사용한 force directed algorithm을 사용하였고, 나머지 하드웨어 할당작업을 위해서는 새로운 알고리듬을 제시하였다. 이 새로운 알고리듬들을 FACET[3]과 같이 하드웨어 할당문제를 graph 문제로 modelling 하였으나, 각 할당단계에서 이후의 할당단계들을 고려한 점이 특색이다. 본 논문에서는 하드웨어 할당 알고리듬들에 대해서만 기술한다.

Ⅱ. 레지스터 할당

레지스터 할당과정에서는 scheduling이 된 상태에서 연산자 할당, 버스 할당과정에 미칠 영향을 고려하면서, 레지스터를 DFG상의 모든 변수들에게 할당시키는 과정이다. DFG의 node는 operation이고 edge는 variable이라고 한다. 본 논문에서는 나중의 할당과정에 미칠 영향까지 고려한 레지스터 할당 알고리듬을 제안하였다. 자세한 알고리듬은 그림1의 DFG를 예로 들어 설명하겠다. 이 DFG는 FACET[3]에서 사용된 것으로 많은 논문에서 예로 사용하고 있다.

1. 생존시간 분석

생존시간 분석(lifetime analysis)과정은 DFG 상에서 각 변수들이 존재하는 시간구간을 알아내는 과정으로, 어떤 변수의 생존시간은 그 변수가 새로 정의된 시간부터 마지막으로 쓰인 시간구간을 말한다. 어떤 두 변수의 생존시간이 서로 겹쳐지는 부분이 없으면 두개의 변수는 합쳐져서 나중에 같은 레지스터를 사용할 수 있다.

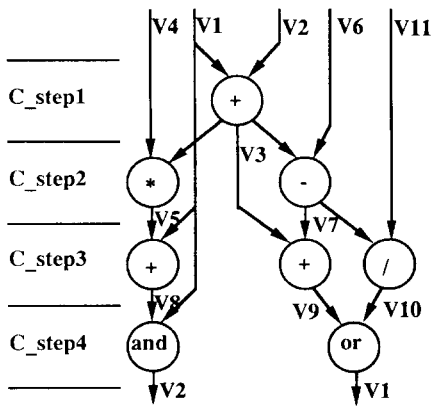


그림 1. 예로 사용된 DFG
Fig. 1. Example DFG.

2. Compatible graph의 생성

Compatible graph는 각 변수들을 node로 하고, 어떤 두개의 변수가 합쳐질 수 있을 때 그에 해당하는 두 node 사이를 edge로 이음으로써 생성되는 graph를 말한다. Compatible graph를 테이블로 표현하면 그림 2와 같이 된다.

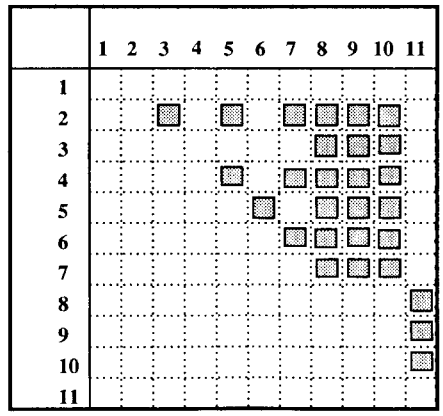


그림 2. Compatible graph
Fig. 2. Compatible graph.

3. Edge의 이득계산

두개의 합쳐질 수 있는 변수를 결합시킴으로써 얻을 수 있는 이득은 두 변수 사이의 관계에 따라 차이가 있게 된다. 두 변수 사이에 존재하는 관계와 그에 따른 이득을 다음과 같이 정하였다.

1) pure transfer:그림 3(a)에서, 변수 v1과 v2를 합치면 연산 자체가 DFG상에서 없어지게 되므로 이득을 5정도로 크게 놓는다.

2) same destination:그림 3(b)에서, v2과 v3는 같은 destination을 갖고 있으므로 이들을 합쳐지게하면 두 transfer operation이 한 종류의 transfer operation이 되므로 이들 사이에 4의 이득을 준다.

3) same source:그림 3(c)에서, v2와 v3의 source가 같은 경우로 이들을 합치면 역시 두 transfer operation이 한 종류의 transfer operaion이 되므로 이들

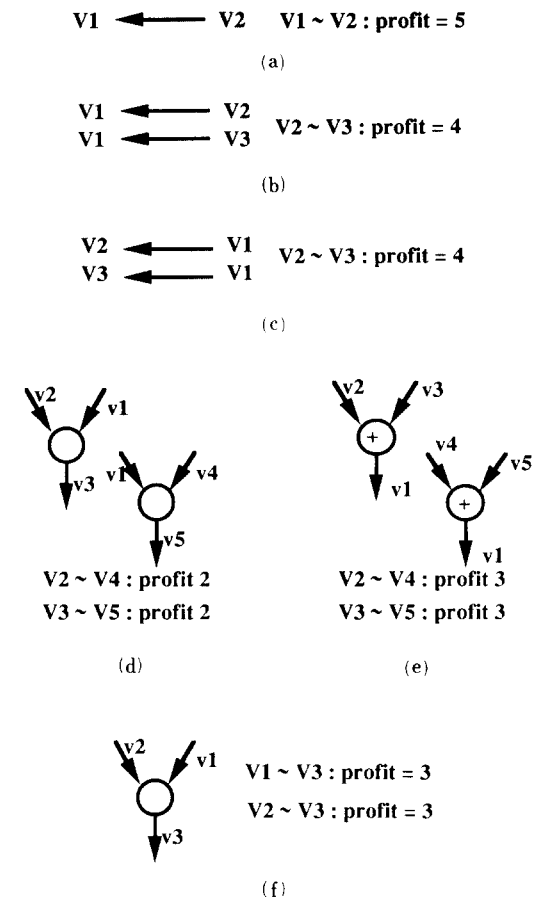


그림 3. 두 변수 사이의 관계
Fig. 3. Relations between two variables.

	1	2	3	4	5	6	7	8	9	10	11
1											
2			3		2		1	3	1	1	
3								2	3	1	
4					3		2	1	1	1	
5						1		3	2	1	
6							3				
7								1	3	3	
8											1
9											1
10											3
11											

그림 4. 이득계산 후의 compatible graph
Fig. 4. Compatible graph after edge profit calculation.

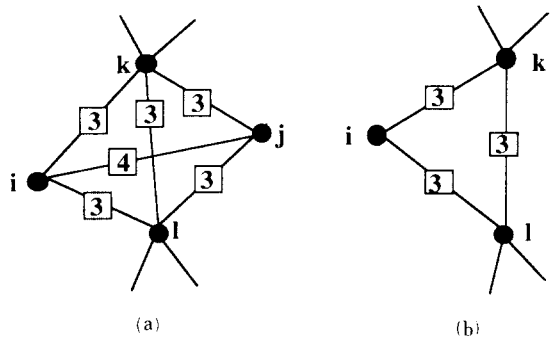


그림 5. 이득의 예측
Fig. 5. Prediciton of profit.

사이에 4의 이득을 준다.

4) 두 변수가 같은 입력, 출력 혹은 기능을 가진 두 operation의 입력이거나 또는 출력인 경우 : 이 경우 두 개의 변수 사이에 이득을 주면 두 operation의 모양이 비슷하게 되어 나중의 연산자 할당과 버스 할당과정에서 유리하게 된다.

5) 두 변수가 한 operation의 입력과 출력인 경우: 그림 3(d)에서는 연산기능은 다르고 하나의 입력또는 출력변수가 같을 때 다른 입력쌍이나 출력쌍 사이의 이득에 2를 더한다. 그림 3(e)처럼 연산기능도 일치할 경우에는 이득값에 1을 더한다.

6) 위의 경우에 속하지 않는 경우 : 이득을 1로 준다.

위와 같은 방법으로 그림 1의 예제에 대하여 edge profit을 정하면 그림 4와 같이 된다.

4. 두 변수의 결합

합쳐지게 되는 두 개의 변수는 compatible graph상에서 가장 결합이득이 큰 edge로 연결된 node에 해당하는 것을 선택하게 되는데, 이 결합이득 값이 가장 큰 edge가 2개 이상인 경우에는 각 쌍의 변수를 결합함으로써 앞으로 얻게 될 이득을 예측하여 이값이 가장 큰 변수 쌍을 결합시킨다. 즉, 현재이득이 같은 경우에는 예측이득이 가장 큰 변수쌍을 결합시킨다. 그림 5에서 변수 i와 j를 결합함으로써 얻을 수 있는 이득의 예측값은, node i와 j를 결합함으로써 생긴 compatible graph에서 node i와 이에 연결된 모든 node 사이의 edge의 이득을 합한 것과, 공히 edge i에 연결되어 있는 두 node간의 edge의 이득을 합한 것이다. 즉, 예측이득항은 다음과 같이 표현된다.

$$\sum_{h\in A} p(i,k) + \frac{1}{2}\sum_{l\in B}\sum_{m\in C} p(l,m)$$

A={k|(i, k)∈E}, E는 compatible graph상의 edge의 집합

B={l|(i, l)∈E}

C={m|(i, m)∈E}

5. 레지스터 할당 알고리듬

레지스터 할당 알고리듬은 앞의 설명과 같이 생존시간분석을 통하여 서로 합쳐질 수 있는 variable 쌍을 나타내는 compatible graph를 얻은 후 한 단계에 하나씩 결합시킬 수 있는 변수 쌍중에서 가장 좋다고 생각되는 쌍을 골라서 결합시키는 과정을 더이상 결합할 변수 쌍이 없을때까지 되풀이한다.

1 단계 : 생존시간을 분석한다;
2 단계 : compatible graph를 만든다;
3 단계 : 각 edge의 이득을 계산한다;
REPEAT(더이상 edge가 없을 때까지)
4 단계 : edge의 이득이 가장 큰 node의 쌍들을 찾는다.
5 단계 : 각 node쌍을 결합했을 때 예측되는 이득을 계산한다;
6 단계 : 예측된 이득이 가장 큰 두 node를 결합한다;
REPEAT END;

앞의 레지스터 할당 알고리듬을 사용하여 레지스터 할당을 한 결과가 표 1이고 그에 따른 DFG가 그림 6에 있다.

표 1. 레지스터 할당의 결과
Table 1. The result of register allocation.

register name	variables assigned
v1	{v1}
v2	{v2, v3}
v4	{v4, v5, v8}
v6	{v6, v7, v9}
v10	{v10, v11}

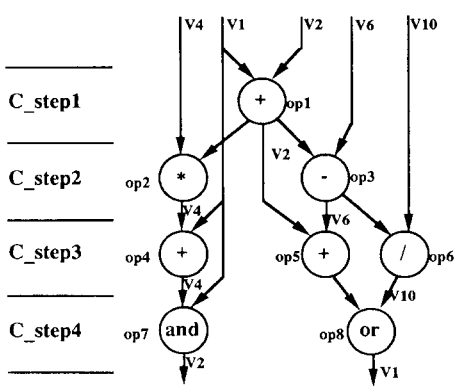


그림 6. 레지스터 할당 후의 DFG
Fig. 6. DFG after register allocation.

Ⅲ. 연산자 할당

연산자 할당과정에서는 레지스터 할당이 끝난 상태에서, 연산자를 DFG상의 operation들에게 해당시키게 된다. 본 논문에서는 버스 할당과정에 미치는 영향까지 고려한 연산자 할당 알고리듬을 제안하였다.

1. Compatible graph의 생성

연산자 할당과정에서 compatible graph는 node를 DFG상의 operation으로 하고, 어떤 두 개의 operation이 수행되는 시간이 달라서 같은 연산자를 사용할 수 있으면 그에 해당하는 node들을 edge로 연결함으로써 생성된다.

2. Edge cost의 계산

연산자 할당과정의 compatible graph에서는 합치고자하는 두 operation의 관계에 따라 edge의 cost가 달라지게 된다. edge cost를 계산하는 방법을 그림 7을 사용하여 설명하겠다.

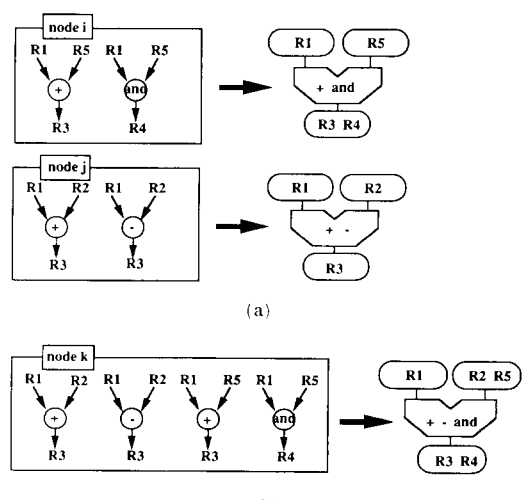


그림 7. Edge cost의 계산
Fig. 7. Calculation of edge cost.

연산자 할당 중간과정에서 그림 7(a)와 같은 node들이 생겼다는 이 두 node를 결합했을 때는 그림 7(b)와 같은 node가 생겨날 것이다. Node j를 node i에 결합시켜서 새로운 node k를 만들어 낸다고 생각하면 이전의 node i에 추가된 것은 하나의 연산기능과 하나의 입력이다. 반면 node j를 node k와 비교하면 하나의 입력과 출력 그리고 연산기능이 추가되었다. cost는 입력이나 출력이 하나 추가되면 2가 증가하고 연산기능이 하나 추가되면 1이 증가하는 것으로 정할 경우에 node i와 node k를 비교해보면 cost는 3이고 node j와 node k를 비교해보면 cost는 5가 되는데 이중 작은 것을 택하므로 두 node를 결합시키는데 드는 cost는 3으로 정한다.

3. 두 operation의 결합

결합하는 두개의 node는 node사이의 cost가 가장 작은 것으로 선택되는데 이러한 node들이 2개 이상일 때는 각 node쌍을 결합함으로써 얻어지는 예측값이 가장 큰 node쌍을 선택하게 된다. 이 예측값을 계산하는 방법은 compatible graph에서 edge들의 cost를 음수로 바꾸고, 레지스터 할당과정에서 사용했던 방법을 적용시키면 된다.

4. 연산자 할당 알고리듬

연산자 할당 알고리듬도 레지스터 할당 알고리듬과 같이 반복적으로 node들을 더이상 결합시킬 수 없을 때까지 결합시켜가는 것으로 전체적인 알고리듬은 레지스터 할당 알고리듬과 비슷하다.

1단계 : compatible graph를 만든다;
2단계 : 각 edge의 cost를 계산한다;
REPEAT(더이상 edge가 없을 때까지)
3단계 : edge의 cost가 가장작은 node쌍들을 찾는다;
4단계 : 4.1 edge의 cost를 음으로하여 이득으로 만든다;
4.2 각 node쌍을 결합시켰을 때 얻을 수 있는 이득을 예측한다;
5단계 : 예측한 이득이 가장 큰 두 node를 결합시킨다;
REPEAT END;

앞의 알고리듬을 적용시켜 연산자 할당을 한 결과가 표 2에 있고 그 결과 생긴 DFG가 그림 8에 있다.

5. 입력 위치 교정

입력 위치 교정과정은 연산자 할당이 끝난 후, 어떤 node에 해당하는 operation들의 묶음에서 operation들에 들어가는 입력들의 위치를 적당히 바꾸어주어, hardware가 적게 들도록 하기 위한 것이다. 그 방법은 noncommutative operation들은 두 입력을 서로 바꿀 수 없으므로 그대로 둔 상태에서 commutative, unary operation 순으로 입력의 위치의 교환여부를 결정하는 것이다.

표 2. 연산자 할당의 결과
Table 2. The result of functional unit allocation.

ALU name	operations assigned
ALU1	{op1, op2, op4, op7}
ALU2	{op3, op5}
ALU3	{op6, op8}

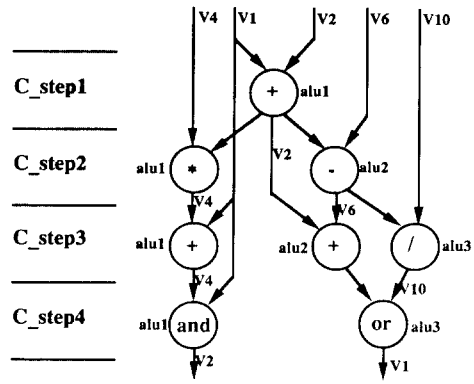


그림 8. 연산자 할당 후의 DFG
Fig. 8. DFG after functional unit allocation.

Ⅳ. 버스 할당

버스 할당은 레지스터 할당과 연산자 할당을 통해서 생성된 레지스터와 연산자들을 연결해주는 연결선을 각 버스에 할당하는 과정이다. 본 논문에서는 새로운 버스 할당 알고리듬을 제시하였다.

1. 연결변수(interconnection variable)의 생성

연결변수는 지금까지 만들어진 레지스터와 연산자들간의 연결을 나타내는 것으로서 source와 destination, 그리고 사용되는 control step으로 표현할 수있다. 예의 DFG로부터 연결변수들을 만들면 표3과 같이 된다.

2. Compatible graph의 생성

버스 할당에서 compatible graph는 node를 연결변수로 하고, 서로 사용하는 시간이 같지 않은 두 연

표 3. 연결변수
Table 3. Interconnection variables.

interconnection variable	source	destination	control steps
con. 1	V2	ALU1-a	[1]
con. 2	V1	ALU1-b	[1]
con. 3	ALU1	V2	[1, 3, 4]
con. 4	V4	ALU1-b	[2, 3, 4]
con. 5	V2	ALU1-b	[1]
con. 6	ALU1	V4	[2, 3]
con. 7	V2	ALU2-a	[2, 3]
con. 8	V6	ALU2-b	[2, 3]
con. 9	ALU2	V6	[2, 3]
con. 10	V6	ALU3-a	[3]
con. 11	V10	ALU3-b	[3, 4]
con. 12	ALU3	V10	[3]
con. 13	ALU3	V1	[4]

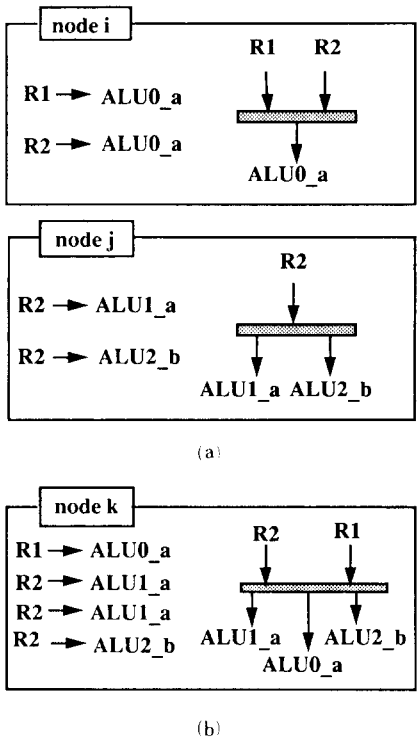


그림 9. Edge cost의 계산
Fig. 9. Calculation of edge cost.

결변수에 해당하는 node를 edge로 이은 것이다.

3. Edge cost의 계산

버스 할당에서의 compatible graph에서도 각 edge의 cost는 두 node사이의 관계에 따라서 달라지게 된다. 버스 할당 중간과정에서 그림 9(a)과 같은 두 개의 node가 있을 때 node i와 j를 결합시켜서 새로운 node를 만들면 그림 9(b)와 같이 되는데 이것은 이전의 node i에 하나의 source와 하나의 destination을 추가한 것이다. edge cost를 하나의 source 또는 destination을 추가할 때마다 2씩 증가하는 것으로 하면 node j를 node i에 결합시키는데 드는 cost는 4가 된다.

4. 두 연결변수의 결합

버스 할당과정에서도 연산자 할당과정과 마찬가지로 결합되는 두 node는 연결하는 edge의 cost가 가장 작은 것인데, 이러한 node쌍이 2개 이상일 때는 연산자 할당에서 적용한 방법과 같은 방법으로 결합시킬 node쌍을 결정한다.

5. 버스 할당 알고리듬

버스 할당 알고리듬도 이전의 할당 알고리듬들과 마찬가지로 더이상 합쳐질 수 없을 때까지 node들을 결합시켜가는 것이다. 버스 할당과정이 끝난 후 레지스터 또는 연산자로 들어가는 버스가 하나 이상이면 그 앞에 MUX를 붙여줘야 한다. 버스할당결과가

표 4. 버스 할당과 결과
Table 4. The result of bus allocation.

BUS	connnections assigned
BUS1	{con. 0, con. 6}
BUS2	{con. 1, con. 4}
BUS3	{con. 2, con. 5}
BUS4	{con. 3}
BUS5	{con. 7, con. 9}
BUS6	{con. 8}
BUS7	{con. 10}
BUS8	{con. 11, con. 12}

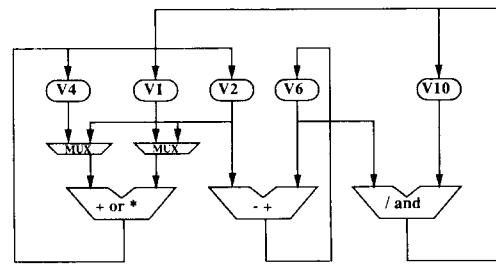


그림10. 합성된 데이타 경로
Fig. 10. Synthesized data path.

표 4 이고 그로부터 그림10과 같은 데이타 경로가 생긴다.

V. 결 과

본 논문의 시스템은 SUN workstation의 UNIX 환경하에서 C language를 사용하여 구현하였다. 본 논문의 시스템이 합성한 데이타 경로를 다른 2개의 시스템에서 합성한 것과 비교하여 보았는데, 본 논문에서 합성한 결과가 FACET[3]에서 합성한 것보다는 좋고, HAL[2]에서 합성한 것과는 비슷한 것(MUX의 수가 하나 적음)으로 나타났다.

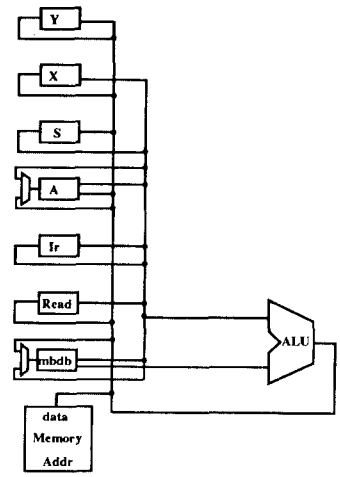
표 5. 다른 시스템의 결과와의 비교
Table 5. Comparison with the results of other systems.

	No. of registers	No. of MUX inputs	No. of connections	Functional unit cost
FACET	8	9	20	+:2 or:1 and:1 −:1 *:1 /:1
HAL	5	5	15	+:2 or:1 and:1 −:1 *:1 /:1
our system	5	4	15	+:2 or:1 and:1 −: *:1 /:1

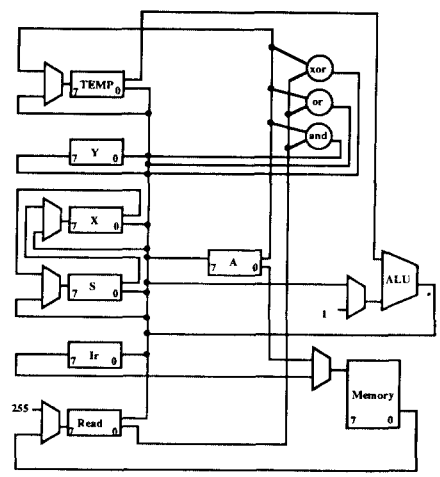
다른 예로 MC6502 마이크로 프로세서가 있는데 그림11(a)는 본 논문의 시스템이 합성한 것이고, 그림11(b)는 DAA[5]가 합성한 것인데, 본 논문의 시스템에서 합성한 것이 더 구조가 간단한 것을 볼 수 있다.

VI. 결 론

본 논문에서는 새로운 데이타 경로 자동합성 시스템에 대하여 기술하였는데, 레지스터 할당, 연산자 할당 그리고 버스할당을 위한 새로운 알고리듬들을 제시하였고, 이 알고리듬들을 사용하여 두가지 예에 대한 데이타 경로를 합성해본 결과 제한조건이 같은 경우 다른 시스템보다 낮거나 비슷한 결과를 얻을수 있었다. 앞으로 해야할 일은 사용자로 하여금 속도와 hardware cost간의 trade off를 가능하게하고, 곱셈과 같은 복잡한 연산도 수행할 수 있도록하고, 레이아웃 시스템과 연계시키는 작업이라고 본다.



(a)



(b)

그림11. MC6502 데이타 경로의 비교
Fig. 11. Comparison of MC6502 data paths.

參 考 文 獻

[1] P.G. Paulin and J.P. Knight, "Force-directed scheduling in automatic data path synthesis," *In Proc. 24th Design Automation Conf,* pp. 195-202, July 1987.

[2] P.G. Paulin, J.P. Knight, and E.F. Girczyc, "HAL: A multi-paradigm approach to automatic data path synthesis," *In Proc. 23rd Design Automation Conf,* pp. 232-237, 1986.

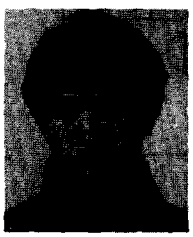
[3] C. Tseng and D.P. Siewiorek, "Automated synthesis of data paths in digital systems," *IEEE Trans. Computer-Aided Design,* vol. CAD5, pp. 379-395, July 1986.

[4] C.Y. Hitchcock and D.E. Thomas, "A method of automatic data path synthesis,"

in Proc. 20th Design Automation Conf, pp. 484-489, July 1983.

[5] D.E. Thomas, C.Y. Hitchcock, T.J. Knowalski, and J.V. Rajan, "Automatic data path synthesis," *IEEE Computer,* vol. 16, pp. 59-73, Dec. 1983.

著 者 紹 介



金 洪 植(正會員)
1965年 10月 25日生. 1988年 2月 연세대학교 전자공학과 졸업. 1990年 2月 한국과학기술원 전기및 전자공학과 졸업(공학석사). 1990年 7月~현재 한국전기통신공사 연구개발단 근무. 주관심분야는 CAD 등임.

慶 宗 旻 (正會員) 第25卷 第10號 參照
현재 한국과학기술원 전기및 전자공학과 부교수