韓國 OA學會 論文誌
第3卷 第4號, 1998. 12.

98-3-4-2-2

콤포넌트 맵핑을 위한 합성 알고리즘에 관한 연구

김 재 진* 이 사 원**

A Study of Synthesis Algorithm for Component Mapping

Kim Jae Jin*, Lee Sa Won**

요 약

본 논문에서는 RT 콤포넌트를 이용한 맵핑 방법으로 HDL로 기술된 연산자들을 RT 콤포넌트에 맞도록 CDFG를 구성한 후 그래프를 최소화하고 cost와 bound를 계산하여 적합한 라이브러리를 선정하여 맵핑할 수 있는 콤포넌트 합성 알고리즘(Component Synthesis Algorithm)을 제안하였다.

Abstract

In this paper proposed Component Synthesis Algorithm(CSA) for mapping described HDL to RT component of given library. CSA transform I/O variables of HDL and relation of operators to control/data flow graph(CDFG) that consists of graph, reduce the size of graph, compute the cost, the bound, and the method that use compatibility graph(CG), and then mapping to component.

Component synthesis used branch-and-bound algorithm.

The result that synthesis using CSA algorithm was proved that this result and the cost of the manual were indentified.

* 청주대학교 전자공학과 대학원
** 청주대학교 전자공학과 대학원
논문접수 : 98.10.22. 심사완료 : 98.11.30.

I. 서 론

HLS(High-level synthesis)에서의 맵핑(mapping)은 기술된 HDL(Hardware description language)을 CDFG(control/data flow graph)로 구성하여 제공되는 RT 콤포넌트(Register transfer component)에 맞도록 RTCDF로 재구성한 후 그래프에 맞는 라이브러리(library)의 검색하여 맵핑하는 방법을 사용한다.

이와같은 맵핑은 제공되는 라이브러리들인 ALU, counter, 비교기(comparater), 레지스터(register), 메모리(memory)등의 storage unit과 bus와 multiplexor등의 interconnect unit을 이용하여 CFG의 연산자들을 맵핑하는것이다.

이러한 맵핑을 하기 위해서는 할당(allocation), 스케쥴링(scheduling), 바인딩(binding)등의 기능을 모두 수행하여 행해진다.

CDFG는 스케쥴링을 행한 후에 생성된 그래프로서 맵핑을 수행할 때는 cost를 연산하여 가장 작은 값의 cost를 같는 라이브러리에 맵핑하는 것이다.

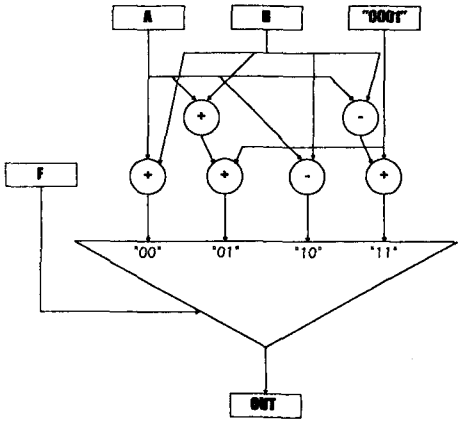
따라서 본 논문에서는 RT 콤포넌트를 이용한 맵핑 방법으로 HDL로 기술된 연산자들을 RT 콤포넌트에 맞도록 CDFG를 구성한 후 그래프를 최소화하고 cost와 bound를 계산하여 적합한 라이브러리를 선정하여 맵핑할 수 있는 콤포넌트 합성 알고리즘(Component Synthesis Algorithm)을 제안하였다.

II. 콤포넌트 합성 알고리즘

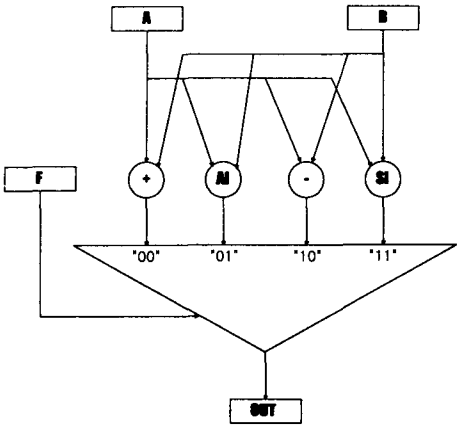
HDL에서 기술된 연산자를 라이브러리로 맵핑하기 위한 방법으로 우선 기술된 연산자들의 연관 관계와 입출력 변수를 탐색하여 그래프형태인 CDFG를 구성한다. 구성된 CDFG의 에지(edge)와 노드(node)들은 기능과 연관관계를 고려하여 최소화한다. 최소화된 CDFG는 주어진 라이브러리에 맞도록 연산자들을 서로 병합(merge)한다. 그림 1은 기술된 HDL을 Adder/Subtractor로 맵핑할 수 있도록 병합도를 만드는 CSA의 흐름도를 나타내었다. 그림 1의 (a)에 기술된 VHDL을 콤포넌트에 맵핑하기위해 (b)번과 같은 CDFG를 생성한다. (c)는 (b)의 CDFG에서 노드와 에지를 감소한 그래프이다. (d)는 감소된 (c)의 그래프를 콤포넌트에 맵핑하기위해 컴포넌트에 맞도록 연산을 합한 결과이다. AI는 Addition-and-Increment를 의미하고, SI는 Subtractionand-Increment를 의미한다.

```
entity design1 is
   port(F : in BIT_VECTOR(1 downto 0);
        A, B : in BIT_VECTOR(3 downto 0);
        OUT : in BIT_VECTOR(1 downto 0));
end design1;
architecture design1-body of design1 is begin
   with F select
   OUT <= A+B          when "00",
          A+B+1        when "01",
          A-B          when "10",
          A-B+1        when "11";
end design1-body;
```

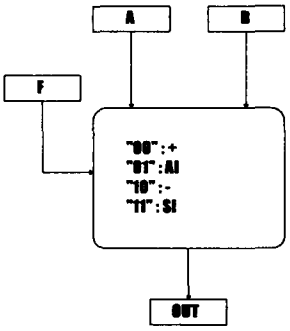
(a) VHDL



(b) CDFG



(c) 감소된 CDFG



(d) 최대로 합병된 CDFG

그림 1. CSA를 사용한 Adder/Subtractor 맵핑

II-1. CDFG 축소

HDL로 기술된 연산자들을 제공되는 라이브러리에 맵핑하기위해서는 라이브러리의 기능에 맞도록 생성된 CDFG를 변환시켜야 한다. 변환 과정은 생성된 CDFG의 형태와 database화 되어있는 라이브러리의 함수표(functionality table)을 검색하여 그래프를 비교 후 라이브러리의 그래프로 변환한다.

[그림 2]는 생성된 CDFG의 연산자를 라이브러리의 연산기능으로 변환한 그림으로 A+B+1의 기능을 라이브러리의 Addition-Increment 기능의 한 노드로 변환된 그림이다.

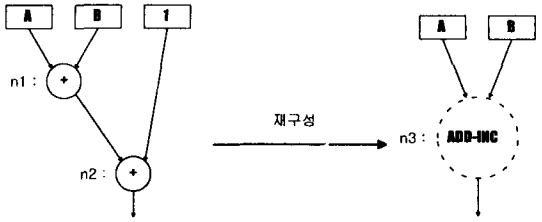


그림 2. CDFG 축소의 예

II.2. 콤포넌트 맵핑

축소된 CDFG를 콤포넌트에 맵핑하는 방법으로 compatibility 그래프(CG : compatibility graph)를 이용하여 그래프를 축소한 후 맵핑하는 방법과, cost와 bound를 계산하여 적합한 크기로 합병하여 맵핑하는 두 가지의 방법이 있다.

II.2.1 CG를 이용한 방법

CG를 이용하여 그래프의 노드를 합병하여 노드의 수를 줄이고, 에지의 상태에 따라서 합하거나 제거하게 된다. 노드의 병합은 서로의 연관성에 따라 연관성이 있고 시간을 공유하지 않는다면 clique의 조건에 합당하기 때문에 병합이 가능하다. 에지의 경우는 다음의 두 가지 경우에 따라서 합하거나 제거된다. 첫 번째 경우로서 노드가 병합된 후 다른 노드와 연결된 에지들의 성질이 같은 경우 하나로서 대치가 가능하며, 두 번째로 에지에 연결된 노드의 성질이 따른 경우는 제거되게 된다.

[그림 3]은 CG를 이용한 노드의 합병과 에지의 변환에 대한 예제를 나타내었다.

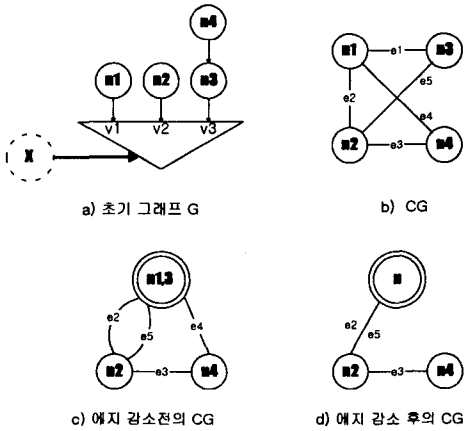


그림 3. CG를 이용한 그래프 축소

II.2.2 Cost와 bound 값을 이용한 방법

축소된 CDFG를 콤포넌트에 맵핑하는 방법으로 연산자에 값지정하여 cost와 지정된 bound의 값을 계산하여 일정한 크기의 노드로 병합하는 방법이다.

Cost의 계산은 식1에 의해 계산되며 bound의 값은 식2에 의해 계산된다.

$$Cost(G) = \sum_{d_i \in D} (Conn - cost(choices(d_i), bw(d_i))) + \sum_{n_i \in N} (op - cost(M(n_i), bw(n_i)) + decoder - cost(n_i))$$

(식 1)

$$bound(CG) = \sum_{\text{allocated node } n_i \in N} (op - cost(M(n_i), bw(n_i))) + Conn - cost(choices(D(n_i), bw(D(n_i)))) + \sum_{c \in conn} (\max_{n \in c} (op - cost(n, bw(n))) + ADDON(c))$$

(식 2)

계산된 cost와 bound값을 가지고 다음의 branch and-bound 알고리즘에 적용하여 각 노드의 값을 계산하고, 에지에 연결된 노드를 병합하여 병합된 노드의 cost와 bound를 다시 계산함으로서 가장 적은 값의 cost를 갖는 병합을 찾아내게 된다.

[그림 4]에 branch-and-bound 알고리즘을 나타내었다.

INPUT : a flow graph, a unit table,
an iteration count.
active-stack={ 0 };
UP = ∞; /* UP = upper bound */
BSF = empty; /* BSF = Best solution found */
while ((active-stack ≠ 0) and (iteration-count > 0)) do begin
(1) Pop branching node b from active-stack
(2) Select edge e with the largest benefit based on a heuristic
(3) Generate children of b with
$child_1 = b+e$, $child_2 = b-e$
(4) Calculate lower bounds of the two children : $bound(child_i)$
(5) for i = 2 downto 1 do
if $bound(child_i) \geq$ UP
then discard $child_i$
else if $child_i$ is complete
(no compatibility edges left)
then $child_i$ replaces the best solution found,
i.e., UP := $bound(child_i)$;
BSF := $child_i$;
else push $child_i$ onto active-stack;
(6) decrement iteration-countend while;

그림 4. branch-and-bound 알고리즘

branch-and-bound 알고리즘을 이용하여 맵핑된 예를 그림 5에 나타내었다. 그림 5의 S1상태의 cost는 + 연산의 1과 - 연산의 1, AI의 2, SI의 2를 합하여 6이 된다. 이와 같은 계산으로 각 상태의 cost값을 계산한다. 알고리즘의 b는 노드를 의미하며, e는 에지를 의미한다. 따라서 b+e는 에지에 연결된 2개의 노드를 합하는 것을 나타내며, b-e는 에지의 제거를 나타낸다. 이러한 알고리즘의 적용 결과로 S4의 상태와 같은 하나의 Adder-/Subtractor로서 맵핑된다.

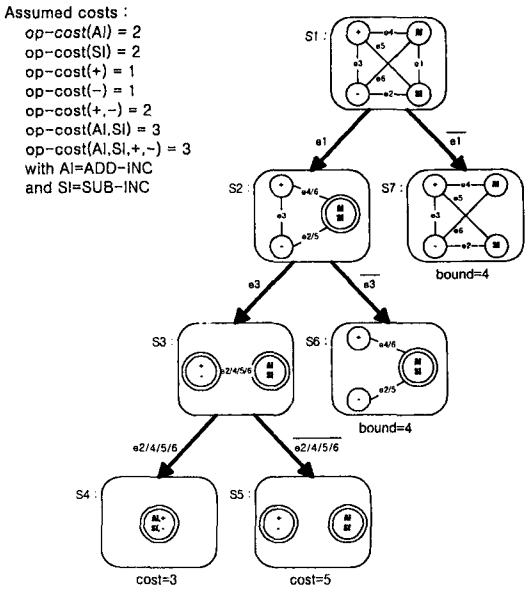


그림 5. branch-and-bound 알고리즘의 적용 예

III. 알고리즘 수행 결과

본 논문에서 제안한 CSA의 적용 예로 6가지 회로를 선정하여 구현하였다. 6가지의 적용 결과는 아래 표 1에 나타내었다.

〈표 1〉의 예제 1은 Adder/Subtractor의 합성결과이며 예제 2와 3은 ALU이다. 예제 4는 TI 74181 ALU이고, 예제 5는 divude-by-3328 counter이고 예제 6은 ALU의 조합 논리 부분만을 합성한 결과이다.

〈표 1〉의 FR은 Functionality recognition 알고리즘으로 CDFG의 노드와 에지를 좀더 줄이기 위해 사용되는 알고리즘이다.

표 1 CSA 적용 예

	CSA(그래프 방식)				CSA(bound 방식)				Manual
	FR부적용		FR적용		FR부적용		FR적용		
Ex	Cost	시간	Cost	시간	Cost	시간	Cost	시간	Cost
1	56	3	28	2	56	11	28	2	28
2	97	3	67	3	97	504	67	5	67
3	128	34	63	3	92	340	63	192	63
4	99	6	67	4	99	720	67	574	67
5	415	7	415	7	415	47	415	47	415
6	74	2	74	3	60	16	60	17	54

IV. 결론

본 논문은 VHDL로 기술된 연산자를 라이브러리에서 제공되는 RT 콤포넌트에 맵핑하는 방법으로 CSA를 제안하였다.

CSA는 기술된 VHDL로 기술된 연산자와 입출력 변수들의 연관 관계를 고려하여 그래프의 형태인 CDFG를 형성한다. 형성된 CDFG는 노드와 에지를 축소하고, 라이브러리의 콤포넌트에 맞도록 연산자들을 합병한다. 콤포넌트 맵핑 방법은 CG를 이용하는 방법과 cost와 bound를 이용하는 branch-and-bound 알고리즘이 있다.

CSA의 적용예로 6가지를 선정하여 수행 결과를 표 1에 나타내었다.

수행결과 manual에서 주어진 cost와 거의 동일함을 입증하였다.

참고문헌

[1] Dutt, N., "GENUS:A Generic ComponentLibrary for High Level Synthesis Tech.", Report 9, Univ. of Cal., Irvine, 1989.

[2] Gajski, D., et al, "Synthesis from VHDL : Rockwell-Counter Case Study, Tech.", Rep. 9, Univ. of Cal., Irvine, 1990.

[3] Mano, M. M., Computer Engineering Hardware Design, Prentice Hall, 1988

[4] Leive, G. W., & D. E. Thomas, "A Technology Relative Logic Synthesis and Module Selection System", DAC, pp. 479-485, 1981.

[5] Lis, J. S., & D. D. Gajski, "Synthesis from VHDL", ICCD, 1988.

[6] McFraland, M. C., Parker, A. C., & Camposano, R., "Tutorial on High Level Synthesis", DAC, 1988.

저자소개

김재진

청주대학교 전자공학과 대학원 박사과정

관심분야 : 컴퓨터 공학, CAD 및 VLSI

이사원

한국 OA학회 논문지 제3권 제3호 참조