論文96-33A-2-22

조건부 자원 공유를 고려한 스케쥴링 알고리듬

(A Scheduling Algorithm for Conditional Resources Sharing Consideration)

印 致 虎 *, 鄭 正 和 *

(Chi-Ho Lin and Jong-Wha Chong)

요 약

본 논문에서는 상위 레벨 합성에서 가장 중요한 과제인 스케쥴링 문제를 해결하기 위한 새로운 알고리듬을 제안한다. 제안된 알고리듬에서는 VHDL로 기술된 설계 사양으로 부터 생성된 중간 데이타 구조 CDFG(Control Data Flow Graph)를 입력으로 하여 조건 분기 특성에 근거한 자원 공유 개념을 고려한 스케쥴링을 행한다.

본 알고리듬은 ASAP와 ALAP을 이용하여 계산된 연산의 time frame에 대해 조건 그래프(condition graph)를 구성한다. 각 조건 경로들에 의해 구성된 조건 그래프로부터 조건 우선순위(condition priority)를 구한다. 조건 우선순위는 조건 분기를 가진 경로를 조건 분기가 없는 경로로 변환 할때의 순서를 의미한다. 최소의 자원 비용(resource cost)을 위해 조건 분기가 있는 경로들을 조건 분기가 없는 경로들로 조건 우선 순위에 따라 변환을 수행 한다. 변환된 경로들에 데이타 종속 및 제시된 제한 조건을 고려 하여 연산이 각 제어 스텝에 할당되도록 한다. 이때 연산을 제어 스텝에 할당 하는것은 반복적인 연산의 이동에 의해 스케쥴링을 수행한다. 벤치 마크 예의 실험을 통하여 본 알고리듬의 효용성을 보인다.

Abstract

This paper presents a new scheduling algorithm, which is the most important subtask in the high level synthesis. The proposed algorithm performs scheduling in consideration of resource sharing concept based on characteristics of conditional branches in the intermediate data structure, CDFG(Control Data Flow Graph) generated by a VHDL analyzer.

This algorithm constructs a condition graph based on time frame of each operation using both the ASAP and the ALAP scheduling algorithm. The condition priority is obtained from the condition graph constructed from each conditional branch. The determined condition priority implies the sequential order of transforming the CDFG with conditional branches into the CDFG without conditional branches. To minimize resource cost, the CDFG with conditional branches are transformed into the CDFG without conditional branches according to the condition priority. Considering the data dependency, the hardware constraints, and the data execution time constraints, each operation in the transformed CDFG is assigned to control steps. Such assigning of unscheduled operations into control steps implies the performance of the scheduling in the consecutive movement of operations. The effectiveness of this algorithm is shown by the experiment for the benchmark circuits.

I. 서 론

VLSI 집적 기술과 설계 기술의 발전으로 인하여 대규모의 복잡한 시스템도 하나의 칩내에 구현이 가능하게 됨에 따라 CAD 분야에서는 1980년대 중반 이후부

* 正會員, 漢陽大學校 電子工學科

(Dept. of Elec. Eng., Hanyang Univ.)

接受日字:1995年10月7日, 수정완료일:1996年2月3日

터 상위 레벨 합성(high level synthesis)은 많은 주목을 받았으며, 이에 대한 연구도 활발히 진행되어 많은 발전을 하고 있다. 상위 레벨 합성은 동작 기술로부터 목적 함수와 제한 조건들을 만족하는 RT(Register Transfer) 수준의 구조적인 설계를 자동적으로 생성하는 것이다.

상위 레벨 합성은 크게 스케쥴링과 할당(allocation)으로 구분할 수 있다. 스케쥴링은 동작시간(면적)의 제한 조건을 만족하면서 목적 함수인 면적 (동작시간)이 최소화되도록 각 연산이 수행될 시간을 결정하는 것이다. 할당은 구현되는 하드웨어의 면적이 최소가 되도록 연산을 연산자에, 변수를 메모리에 할당하고 메모리와 연산자 사이의 연결 구조로 버스나 멀티플렉서를 할당하는 것이다[1][5][7][9][11][12].

종래의 스케쥴링 알고리듬은 ASAP(As Soon As Possible), list 스케쥴링[1][2][9], FDS(Force Directed Scheduling)[3], ILP(Integer Linear Programming)[4]등의 다양한 알고리듬이 제안되었으며, 특히 ILP의 경우 최적의 해를 구할수 있지만 수행 시간이 오래 걸린다는 단점을 가지고 있어서 대부분의 알고리듬은 휴리스틱(heuristic)한 방법을 적용하고 있다. 그러나 위의 알고리듬은 조건 분기등이 포함된 ASIC 설계를 지원하기에는 부 적합하기 때문에 path based scheduling[6], condition vector를 이용한 스케쥴링[8], tree based scheduling[10]등의 알고리듬이 개발되었으나 ASIC 설계에 필요한 동작시간의 제한 조건등에 대한 처리를 하지 못하고 있으며 relative scheduling[13] 알고리듬은 동작시간의 지연등에 대한 처리가 가능하지만 ASIC 설계에 필요한 조건 분기에 대한 처리가 단점으로 지적된다. 따라서 본 논문에서는 ASIC 설계에 필요한 동작시간의 최소화 및 조건 분기를 효율적으로 처리할 수 있는 스케쥴링 알고리듬을 제안 한다. 제안된 스케쥴링 알고리듬은 사용 가능한 자원을 제한 조건으로 하여 동작 시간의 최소화를 목적 함수로 한다. 또한 다구간(multi cycle), 체이닝(chainging) 연산이 하나의 모델에 의해서 쉽게 스케쥴링이 가능하다.

본 논문의 구성은 2장에서는 VHDL 부분사양(subset) 및 조건 분기 경로들에 대한 처리 방법을 기술하고 3장에서는 본 논문에서 제안한 스케쥴링 알고리듬에 대하여 기술한다. 실험 및 결과는 4장에서 기술한다.

Ⅱ. VHDL 부분사양 및 중간 데이타 구조

본 알고리듬의 입력은 제안된 VHDL 부분사양으로 기술된 시스템의 순차적인 설계 사양이며, 이를 VHDL 분석기에 의하여 중간 데이타 구조인 CDFG를 생성한다[15].

1. VHDL 부분사양

본 논문에서 지원되는 VHDL 부분사양의 전체적인 구조는 그림 1과 같으며 이는 동기 및 비동기 회로의 효과적인 동작기술의 설계 지원 측면 뿐 아니라 시뮬레이션 지원 측면을 고려함으로써 하드웨어의 병렬성을 효과적으로 지원 할수 있으며 합성에 적합한 정보를 효과적으로 기술할수 있다[15][16].

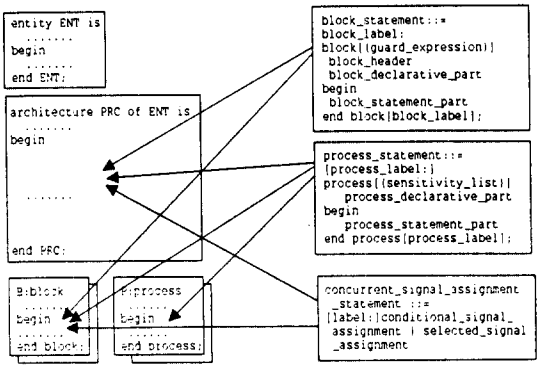


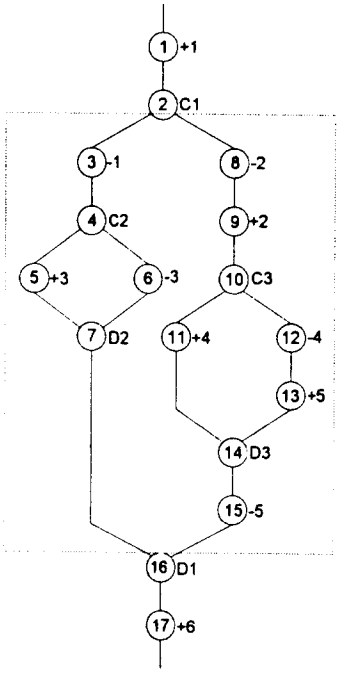
그림 1. VHDL 부분사양의 구조
Fig. 1. VHDL subset structure.

2. 중간 데이타 구조(Intermediate Data Structure)

본 논문의 중간 데이타 구조 CDFG는 3 종류의 노드 즉 fork 노드,join 노드, 연산(operation) 노드들이 있다. fork 노드은 2개 이상의 분리된 출력 분기 경로들을 가지고 있다. 실행 경우에는 조건의 배타적인 성질에 의해 그들 중 하나만 실행한다. fork 노드로부터 나온 출력 분기들은 join 노드에서 합쳐진다. 그러므로 조건 분기들의 경로는 각 조건 분기의 fork 노드와 대응되는 join 노드에서 끝을 맺는다. fork 노드와 그와 대응하는 join 노드 사이의 조건 분기 경로들의 집합을 조건 블럭(condition block)이라 한다. 또한 조건 분기 경로들이 포함되지 않는 CDFG의 부분을 비 조건 블럭(non condition block)이라 한다.

```
entity EXAMPLE is
   port(
     IN1,IN2,IN3,IN4:in bit_vector(7downto0);
     COM1,COM2,COM3:in bit_vector(7downto0);
     output : out bit_vector(7downto0));
end EXAMPLE;
architecture BODY of EXAMPLE is
begin
process (IN1, IN2, IN3, IN4)
variable A,B,C,D,E,F:bit_vector(7downto0);
variable setvar : bit;
begin
  .....
   A := IN2 + IN3;                          (1)
   if ( A /= "00000000" ) then              (2)
      B := IN1 - A;                         (3)
      if ( B /= "00000000" ) then           (4)
         D := A + B;                        (5)
      else
         D := A - "00000001";  (6)
      end if;                               (7)
   else
      C := IN1 - IN4;                       (8)
      setvar := setvar + '1';   (9)
      if ( setvar = '1' ) then  (10)
          E := C + IN4;                     (11)
      else
         E := C - IN4;                      (12)
         setvar := setvar + '1';            (13)
      end if;                               (14)
      F := E - "00000001";                  (15)
   end if;                                  (16)
   output= COM1 + COM2;                     (17)
   .....
end process;
end BODY;
```

(a)



(b)

그림 2. VHDL 기술 및 CDFG a) VHDL 기술 b) CDFG
Fig. 2. VHDL description and CDFG. a) VHDL description b) CDFG

그림 2a의 VHDL 기술 사양으로 부터 VHDL 분석기에 의해 생성된 중간 데이타 구조 CDFG는 그림 2b와 같으며, 그림 2b에서 각 노드의 번호는 그림 2a의 VHDL 기술문들의 번호와 대응된다. 그림 2b에서 노드 1과 노드 3의 +1과 -1은 그림 2a의 (1) 과 (3)에 대응되는 덧셈기와 뺄셈기들의 연산노드이고, 그림 2b에서의 노드 2와 노드 16의 C1 과 D1은 그림 2a의 (2)의 fork 노드와 그와 대응되는 (16)의 join노드 사이의 조건 분기 경로들의 집합인 조건 블럭을 나타낸다.

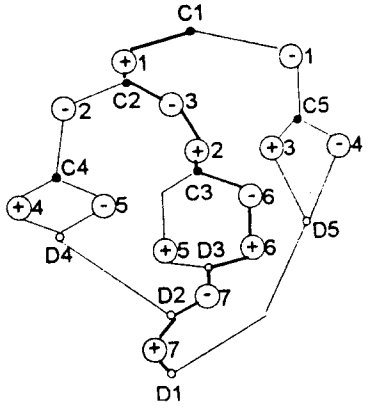
Ⅲ. 조건부 자원 공유를 고려한 스케줄링 알고리듬

본 논문에서 제안하는 스케줄링 알고리듬은 control-dominated 회로와 같이 조건 분기 및 순차적인 동작을 수행하는 IC의 합성에 적용된다. 따라서 VHDL 분석기에 의해 생성된 중간 데이타 구조 CDFG를 입력으로 하여 조건 분기 특성에 근거한 자원 공유 개념을 고려하여 각 연산들의 동작 제어 스텝을 결정하기 위하여 다음과 같은 과정을 거쳐 스케줄링이 수행된다.

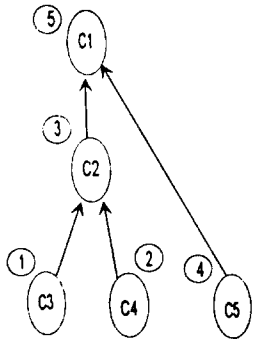
1. 조건 우선 순위의 결정

생성된 CDFG에 포함된 조건 분기 경로들의 하드웨어 자원들의 공유 가능성을 찾기 위해서 조건 분기를 포함된 경로들의 조건 블럭들을 조건 분기가 포함되지 않은 경로들의 비 조건 블럭(non condition block)으로 변환한다. 중첩된(nested) 조건 블럭들을 고려하여 그림 3과 같이 조건 그래프를 구성한다. 조건 그래프의 노드는 각 조건 블럭들로 구성되며 가장 깊은 부분의 조건 블럭들로 부터 가장 바깥의 조건 블럭들로 bottom - up 방식에 의해 조건 그래프를 구성한다. 조건 그래프의 에지(edge)는 그래프의 리프(leaf)들로

시작 하여 각 자식 노드(child node)로 부터 부모 노드(parent node)들로 방향성 에지(directed edge)로 구성된 방향성 그래프(directed graph)이며, 방향은 각 조건 블럭의 수행 과정의 진행 방향을 의미한다. 그림 3과 같이 그래프의 깊이(depth)에 따라 조건 우선순위가 결정되는데, C3 노드와 C4 노드 같이 깊이가 같은 노드는 CDFG의 최대길이 및 조건분기들의 중첩의 수를 고려한 의사(pseudo) longest path에 걸쳐 있는 조건 블럭의 노드가 우선 순위가 높도록 하여 C3를 우선순위 1로 설정 하였다. 또한 노드 C2와 C5는 그래프의 깊이가 같으나 리프 노드의 존재에 의해 C2 노드의 우선 순위를 C5 보다 높게 설정 하였다.



(a)



(b)

그림 3. 조건 그래프의 구성
a) CDFG b) 조건 그래프
Fig. 3. Construction of condition graph.
a) CDFG b) condition graph

2. 조건 분기들이 포함된 CDFG의 변환

조건 분기 경로들의 하드웨어 자원들의 공유 가능성을 찾기 위해서 조건 분기가 포함된 조건 블럭들은 조건 그래프에서 생성된 조건 우선 순위에 따라 그림 4와 같이 비 조건 블럭들로 변환 한다. 우선 그림 4a와 같이 각 조건 블럭 연산 노드들에 대해 time frame을 설정한다. time frame은 각 연산이 각 의사(pseudo) 제어 스텝에 한 연산자를 할당 되도록 정의하였으며, 조건 그래프의 노드 우선 순위에 따라 조건 블럭을 비 조건 블럭으로 변환을 반복적으로 진행한다.

그림 4a와 같이 우선 순위가 가장 높은 조건 블럭 C4에서는 연산 +5와 +4가 제어 스텝 4에서 time frame이 중첩되므로 +5 와 +6은 한 기능 연산자로 자원 공유 될수 있기 때문에 연산자 +(5,6)로 통합하여 조건 블럭을 비 조건 블럭으로 구성한다. 또한 조건 블럭 C2에서도 자원이 공유 될수 있는 -(2,3), +(2,4), -(5,6)의 통합된 연산을 생성 하여 조건 블럭을 비 조건 블럭으로 그림 4b.와 같이 변환 할 수 있다.

3. 변환된 CDFG의 스케쥴링 알고리듬

2.에서 변환된 조건 분기가 포함되지 않은 CDFG에 데이타 종속 및 제한 조건을 고려하여 각 연산 노드들을 각 제어 스텝에 할당하는 스케쥴링을 수행하는 알고리듬은 다음과 같은 과정을 거쳐 수행된다.

1) 함수 정의

스케쥴링 알고리듬을 설명하기 전에 알고리듬에서 사용되는 함수들에 대해 정의한다. 데이타 종속 및 제시된 조건들을 고려하여 그림 5와 같이 조건 분기 및 순차적인 동작을 수행하는 회로의 합성에 적용 될수 있도록 각 연산 노드의 동작 시간을 결정 하기 위해 연산자의 가중치를 계산한다.

a) 가중치의 계산

같은 형의 두 연산이 한 제어 스텝에서 동시에 수행될 수 있는 확률로써 가중치를 계산한다. 이때 두 연산간에는 데이타 종속 관계 및 제약조건이 존재 하지 않아야 하며 time frame이 중복되어야 한다. 같은 형의 연산 O_i와 O_j에 대한 각 연산 별로의 연산자 가중치 (operator weight) OW(i)를 계산 하고 각 연산들의 가중치 Weight(i,j)를 각 제어 스텝 별로 다음과 같이 계산 한다.

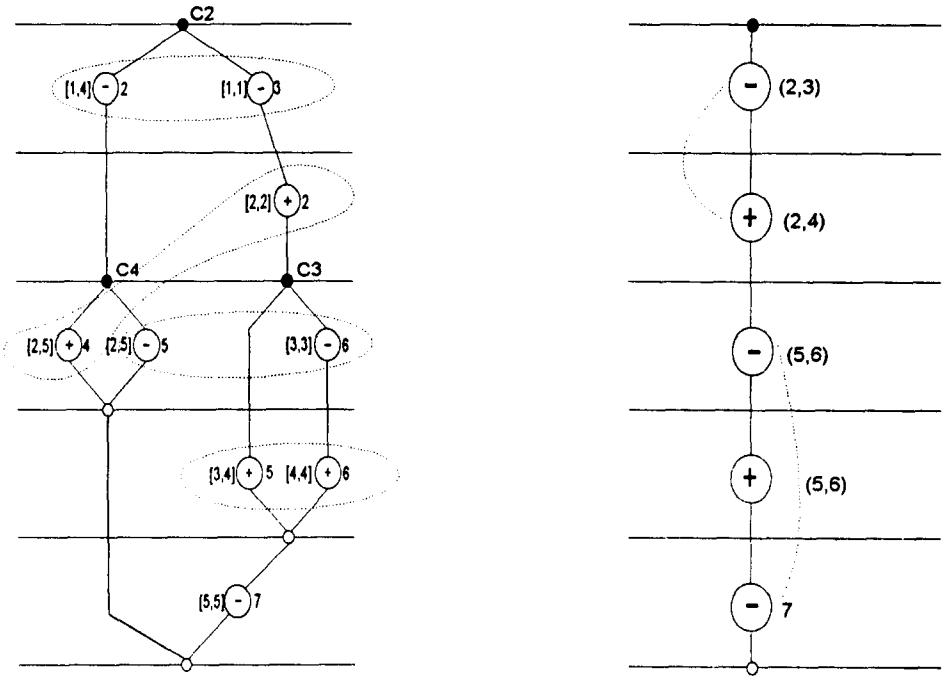


그림 4. 조건 분기들이 포함된 CDFG의 변환 a) 변환 과정 b) 변환된 CDFG
Fig. 4. Transformation of a CDFG with conditional branches.
a) Transformation process b) Transformed CDFG.

$$OW(i) = \frac{1}{ALAP[i] - ASAP[i] + 1} \quad (1)$$

$$Weight(i, j) = \sum_{j} \frac{OVER\_LAP(i,j)}{OW(i) + OW(j)} \quad (2)$$

(단 Weight(i,j) 〉 0)

i, j : 같은 형의 연산의 번호를 표시하는 양의 정수

OVER_LAP(i,j) : 연산 O_i와 O_j의 time frame이 중복되는 시간

그림 5의 CDFG에 대한 time frame이 그림 6과 같이 구해진 경우 연산 *1과 *4간의 가중치는 1/(1+0.5) = 0.67이고, 연산 *2와 *4간의 가중치는 1/(1+0.5) = 0.67이 되어 *4 에 대한 첫번째 제어 스텝에 대한 가중치는 1.34가 된다. 또한 *4의 또 다른 제어스텝인 두번째 제어스텝의 연산 *3과 *4의 계산된 가중치는 1/(1+0.5)=0.67이 된다. 이와 같이 식(1)과 식(2)를 이용하여 각 연산의 형(type)별로 연산간의 가중치를 각 time frame의 제어 스텝에따라 반복해서 계산하여 최소의 가중치 값으로 수행될 연산의 제어 스텝을 결정 선택한 후, 연산의 가중치 및 time frame을 수정 및 반복하여, 데이타 종속 및 시간 종속에 의하여 스케쥴링을 수행한다.

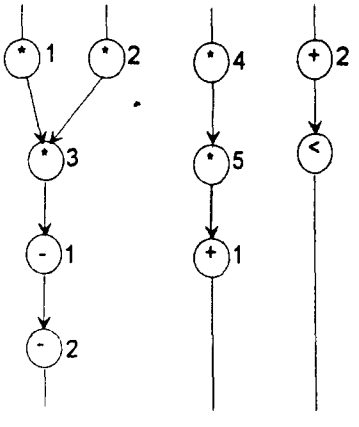


그림 5. 조건 블럭이 포함되지 않은 CDFG
Fig. 5. A CDFG without condition block.

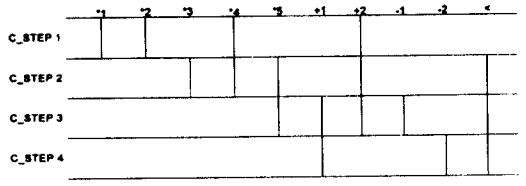
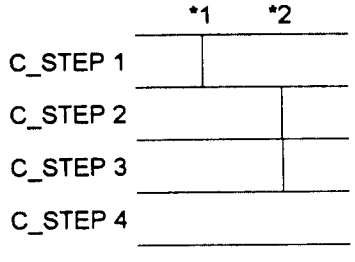
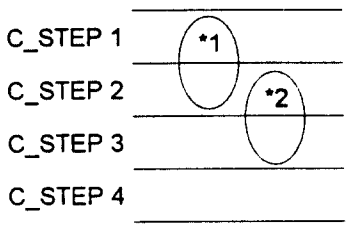


그림 6. 그림 5의 연산에 대한 Time frame
Fig. 6. Time frames of operations in Fig. 5.

체이닝 연산의 경우 가중치는 식 (1),(2)에 의해 계산되지만 연산의 동작 시간이 1개의 제어 스텝을 넘어서 동작하는 다주기 연산의 경우 두 연산의 time frame이 중복되지 않으면 식(1),(2)의 계산에 의해 가중치는 0 이 되어, 두 연산은 동시에 수행되지 않아야 한다. 그러나 그림 7과 같이 식(1),(2)의 적용으로 두 연산이 동시에 수행되지 않는 것처럼 보이지만 실제적으로 두 연산이 동시에 수행되는 경우가 발생할 수 있다. 예를 들면 곱셈 연산을 수행하는 데 2개의 제어 스텝이 필요하다고 하자. 식(1),(2)에 의한 가중치에 의해 본 논문에서 제안한 스케줄링 알고리듬을 수행하면 그림 7a와 같이 연산 *1이 제어 스텝 1에 할당되고 연산 *2는 제어스텝 2 또는 3에 할당될 수 있다. 이때 연산 *2는 제어 스텝 2에 할당되고 두 연산간의 가중치는 0 이 되어 1개의 * 연산만이 필요하다고 계산되지만 실제로는 연산의 수행시간이 중복되어 그림 7b와 같이 2개의 * 연산이 제어 스텝 2에서 동시에 수행된다. 따라서 본 논문에서 제안하는 스케줄링 알고리듬에서 다주기 연산을 처리하고자 식(1),(2)을 식(3),(4)와 같이 수정한다. 식(3),(4)에서는 α는 그림 7과 같은 경우 1의 값을 갖는다. 따라서 그림 7a의 경우 *1과 *2간에는 가중치가 1/(2+3) = 0.2가 된다.

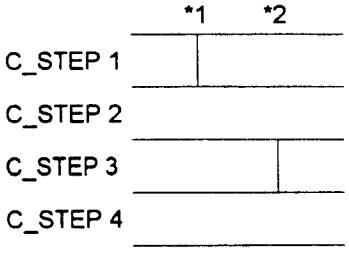


(a)

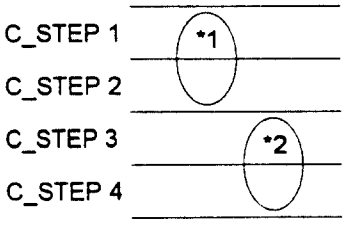


(b)

그림 7. 식(1),(2)에 의한 가중치를 적용한 스케줄링 (a) Time frame (b) 연산의 수행
Fig. 7. A scheduling result using weights of equation (1),(2). (a) Time frame (b) Operation execution



(a)



(b)

그림 8. 식(3),(4)의 가중치를 적용한 스케줄링 (a) Time frame (b) 연산의 수행
Fig. 8. A scheduling result using weights of equation (3),(4). (a) Time frame (b) Operation execution

그림 8과 같이 *2 연산을 제어 스텝 3에 할당하여 1개의 * 연산이 수행되도록 본 논문에서 제안한 스케줄링 알고리듬이 적용된다. 다주기가 아닌 경우, α는 0이 되지만 다주기의 경우는 (다주기 - 1)의 값을 갖게

된다.

$$OW(i) = \frac{1}{ALAP[i] - ASAP[i] + 1} + \alpha \quad (3)$$

$$Weight(i, j) = \sum_{j} \frac{OVER\_MUL(i,j)}{OW(i) + OW(j)} \quad (4)$$

(단 Weight(i,j) 〉 0)

i, j : 같은 형의 연산의 번호를 표시하는 양의 정수

OVER_MUL(i,j) : 다주기를 고려하여 연산 O_i와 O_j의 time frame이 중복되는 시간

α : 다주기 연산의 (수행 시간 - 1)

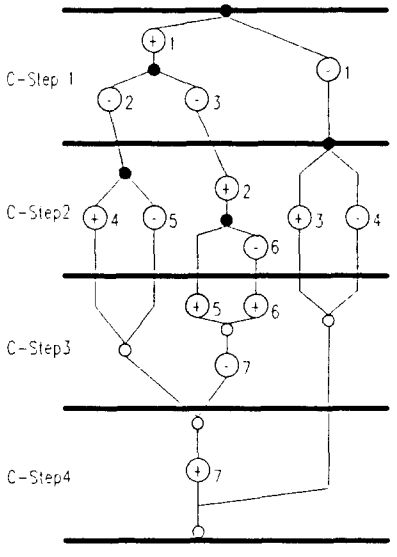


그림 9. 그림 3의 스케줄링 결과
Fig. 9. A scheduling result of Fig. 3.

2) 스케줄링 알고리듬

1)에서 설명한 과정을 변환된 CDFG에 데이타 종속 및 제한 조건을 고려하여 합성을 하기 위한 스케줄링 알고리듬에 대해 기술하면 다음과 같다.

```
ASAP와 ALAP로 각 연산의 time frame을 설정
while(unscheduled each path or operation) {
  while(unscheduled operation){
    - 식(1)과 이용하여 각 연산의 가중치를 계산
    - 식(2)를 이용하여 각 연산의 형(type)별로 각
      제어 스텝에 대한 가중치를 계산
    - 식(1)(2)를 계산결과 최소의 가중치로서 연산
      의 제어 스텝을 결정
    - 선택된 연산의 operator weight 및 time
      frame을 수정
  }
}
```

위와 같은 알고리듬을 수행하면 최적의 스케줄링 결과를 얻을수 있으며 그림 3의 최종 스케줄링 결과는 그림 9와 같다.

Ⅳ. 실험 결과

본 논문에서 제안된 알고리듬은 SUN SPARC 4/60에서 C 언어로 구현되었다. 본 절에서는 여러 예를 통하여 알고리듬의 효용성을 보이고자 한다. 각 예제는 참고 문헌 [4] 및 [5]에서 인용하였다.

표 1의 경우 조건 분기 경로들이 포함된 MAHA 예제를 가지고 resource sharing을 고려하여 덧셈기와 뺄셈기를 가지고 스케줄링한 결과를 보여준다. 본 논문의 결과는 1 제어 스텝에 1 연산만을 할당시 제어 스텝이 8 일 경우 1개의 덧셈기와 1개의 뺄셈기가 필요하고 1 제어 스텝에 2 연산을 할당시 제어 스텝이 4 일 경우 2개의 덧셈기와 2개의 뺄셈기가 필요함을 보여 준다. KIM[14]의 경우 1 제어스템에 3 연산을 할당시 제어 스텝이 3 일 경우 2개의 덧셈기와 3개의 뺄셈기가 필요로 하여 본 논문의 결과는 1 개의 뺄셈기의 향상됨을 보인다.

표 2의 경우 1988년 high-level synthesis workshop에서 benchmark로 채택된 4개의 예중 가장 많이 인용되고 있는 fifth-order elliptic filter 예제이다. 이 예는 26개의 덧셈 연산과 8개의 곱셈 연산으로 구성 되어 있으며 곱셈의 수행주기는 2이며 덧셈의 수행 주기는 1로 하였다. 제어 스텝이 17일 경우 3개의 곱셈기와 3개의 덧셈기가 필요하여 FDS와는 같은 결과를 보여주나 ALPS와는 1개의 곱셈기의 향상됨을 보여준다. 또한 제어 스텝 18의 경우 2개의 곱셈기와 2개의 덧셈기가 필요하며 ALPS와 비교할때 같은 결과를 보여주나 FDS와의 결과와 비교할때 1개의 덧셈기가 절약됨을 보여준다. 표 3에서는 2차 미분 방정식의 예제이다. 덧셈, 뺄셈 및 비교 연산은 모두 ALU에 의해 수행되며 곱셈 연산은 곱셈기에 의해 수행됨을 가정 하였다. 또한 곱셈의 수행주기는 2이며, 나머지 연산의 수행주기는 1로 하였다. 표 3에 나타난

바와 같이 이 예에 대해서는 전체 제어스텝의 수에 관계 없이 본 논문의 알고리듬을 포함한 모든 알고리듬이 최적의 결과를 산출 한다. 그러나 본 논문 에서는 연산들의 수행 시간 및 자원의 비용에 있어서 최적의 해를 구하였지만 스케쥴링시에 이동할 연산의 선택 및 이동 방향이 최종 결과에 대한 정확한 예측을 할 수 없기 때문에 부분적인 최소화에 빠지는 문제점을 가지고 있다.

표 1. MAHA 예에 대한 실험 결과
Table 1. The experimental result for the example of MAHA.

	(+)	(-)	opers/ c_step	Control step	XPU time(s)
MAHA	1	1	1(2)	8	160
	2	3	3	4	160
TASS	1	1	1	8	50
	1	1	2	6	25
PATH	1	1	2	9	0.26
	2	3	5	4	0.28
KIM[14]	1	1	1	8	0.26
	2	3	3	3	0.08
본논문	1	1	1	8	0.20
	2	2	2(3)	4(3)	0.06

표 2. Fifth-order elliptic filter에 대한 실험 결과
Table 2. The experimental result for fifth-order elliptic filter.

	(*)	(+)	Control step	CPU (time(s)
FDS*	3	3	17	60
	2	3	18	180
	2	2	19	420
ALPS**	4	3	17	0.26
	2	2	18	3.1
	2	2	19	10.5
본논문	3	3	17	0.020
	2	2	18	0.048
	2	2	19	0.039

* : Xerox 1108 Lisp machine

** : VAX 11/8800

표 3. 2차 미분 방정식에 대한 실험 결과
Table 3. The experimental result for 2nd differential equation.

	(*)	(+)	Control step	CPU (time(s)
FDS	3	2	6	15
	2	2	7	35
ALPS	3	2	6	0.17
	2	2	7	0.28
본논문	3	2	6	0.01
	2	2	7	0.08

V. 결 론

본 논문에서는 사용 가능한 자원을 제한 조건으로 하여 ASIC 설계에 필요한 동작 시간의 최소화 및 자원 공유 개념을 이용하여 조건 분기들을 효율적으로 처리할 수 있는 스케쥴링 알고리듬을 제안하였다. 본 알고리듬의 입력은 VHDL 부분 사양으로 기술된 시스템의 순차적인 설계 사양이며, 이를 VHDL 분석기에 의하여 중간 데이타 구조인 CDFG를 생성한다.

제안된 스케쥴링 알고리듬은 조건부 자원 공유를 고려하여 조건부 구문들(conditional constructs)를 효율적으로 처리할 수 있으며, 특히 조건 분기가 있는 경우도 제한 조건들을 따로 처리 하지 않고 조건 분기가 없는 경우와 같이 처리할 수 있기 때문에 CRT 콘트롤러, 마이크로 아키텍쳐등과 같은 제어 회로를 합성하는데 하드웨어 자원 및 동작 시간의 최적에 근사한 해를 얻을수 있다.

본 논문은 계층적인 다양한 형태의 기술 시스템에 대한 상위 수준 합성 방법을 제시 하였으나 논리 합성과 연계 될수 있는 VHDL 하드웨어 모델링 방법에 연구가 더욱 필요하다.

참 고 문 헌

[1] M.C. McFarland, A.C. Parker and Raul Camposano," Tutorial on High-Level Synthesis," Proc. 25th DAC, 1988, pp. 330-336.

[2] Barry M. Pangrle and Daniel D. Gajski," Slicer : A State Synthesis for

Intelligent Silicon Compilation," ICCD, 1987, pp. 42-45.

[3] P.G. Paulin, J.P. Knight and E.F. Girczyn," HAL : A MULTIPLE-PARADIGM APPROACH TO AUTOMATIC DATA PATH SYNTHESIS," Proc. 23rd DAC, 1986, pp. 263-270.

[4] Cheng-Tsung Hwang, Yu-Chin Hsu and Yong-Long Lin," Optimum and Heuristic Data Path Scheduling Under Resource Constraints," Proc. 27th DAC, 1990, pp. 65-70.

[5] Raul Camposano and Wayne Wolf, High-Level VLSI Synthesis, Kluwer Academic Pub. 1990, pp. 55-78.

[6] Raul Camposano, Reinaldo A. Bergamaschi, "SYNTHESIS USING PATH-BASED SCHEDULING : ALGORITHMS AND EXERCISES," Proc. 27th DAC, 1990, pp. 450-455.

[7] Miodrag M. Potkonjak, ALGORITHM FOR HIGH LEVEL SYNTHESIS : RESOURCE UTILZATION BASED APPROACH, PhD thesis, U.C. Berkeley, 1992.

[8] K. Wakabayashi and T. Yoshimura," A Resource sharing and Control Synthesis Method for Conditional Branches," ICCAD, 1989, pp.62-65.

[9] A.C. Parker, T. "T" Pizzaro, M. Mliner, "MAHA : A Program for Datapath Synthesis, " Proc. 23rd DAC, 1986, pp. 461-465.

[10] S. H. Huang, Y. L. Jeang, C. T. Hwang, Y. C. Hsu, and J. F. Wang," A Tree-Based Scheduling Algorithm for Control-Dominated Circuits," Proc 30th DAC, 1993, pp. 578 - 582.

[11] F. Brewer and D. Gajski, " Chippe : A System for Constraints Driven Behavioral Synthesis, " IEEE Trans. CAD, Vol. 9, No. 7, Jul. 1990, pp. 681-695.

[12] M. Potkonjak and J. Rabaey," Optimizing resource utilization using transformations," Proc. ICCAD, 1991, pp. 88 - 91.

[13] D.C. Ku and G. De. Micheli,"Realtive scheduling under timing constraints," Proc. 27th DAC, 1990, pp. 59 - 64.

[14] T. Kim and C.L.Liu et al. "A Scheduling For Conditional Resource Sharing," Proc. ICCAD, 1991, pp. 84 - 87.

[15] 인 치 호, 정 정 화, " 마이크로 아키텍처 설계를 위한 VHDL 스케쥴링 알고리듬", 정보과학회 논문지 96년 2월호 게재

[16] C. Wang, K. Parhi," High Level DSP Synthesis Using Concurrent Transformation, Scheduling, and Allocation, " IEEE Trans. CAD, Vol.14, No. 3, March. 1995, pp. 274-295.

저 자 소 개

印 致 虎(正會員) 第 33卷 A編 第 1號 參照
현재 세명대학교 전자계산학과 조교수

鄭 正 和(正會員) 第 33卷 A編 第 1號 參照
현재 한양대학교 전자공학과 교수