論文95-32A-7-14

ASIC 설계를 위한 스케쥴링 알고리듬

(A Scheduling Algorithm for ASIC design)

金 基 鉉 *, 鄭 正 和 *

(Ki-Hyun Kim, Jong-Wha Chong)

요 약

본 논문에서는 control-dominated ASIC 설계를 위해 중간 형태인 HSFG(Hanyang Sequential Flow Graph)와 스케쥴링 알고리듬을 제안한다.

HSFG는 하드웨어 자원의 제한 및 동작 시간 제한과 같은 제한 조건들뿐만 아니라 제어 흐름과 데이타 종속관계를 표현하고 있다. 제안된 스케쥴링 알고리듬은 조건 분기에 의해 발생되는 모든 경로중, 일부 경로만을 탐색하여 제한 조건들을 최대한 제거함으로서 전체적인 동작 시간을 최소화 한다. 즉, 제한 조건을 부그래프로 표현한 후, 부그래프간의 중첩 및 포함관계를 이용하여 부그래프의 수 (제한 조건의 수)를 최소화한다.

벤치마크 데이타를 사용하여 실험한 결과, 제안된 알고리듬이 기존의 알고리듬에 비해 우수함을 확인하였다.

Abstract

In this paper, an intermediate representation HSFG(Hanyang Sequential Flow Graph) and a new scheduling algorithm for the control-dominated ASIC design is presented.

The HSFG represents control flow, data dependency and such constraints as resource constraints and timing constraints.

The scheduling algorithm minimizes the total operating time by reducing the number of the constraints as maximal as possible, searching a few paths among all the paths produced by conditional branches. The constraints are substituted by subgraphs, and then the number of subgraphs (that is the number of the constraints) is minimized by using the inclusion and overlap relation among subgraphs.

The proposed algorithm has achieved the better results than the previous ones on the benchmark data.

I. 서 론

상위 레벨 합성의 목표는 동작을 기술하는 알고리듬 레벨에서 레지스터 전송 레벨의 회로로 변환하는 것이다. 시스템의 동작은 입력과 출력사이의 관계만 기술하므로, 시스템의 동작을 만족하는 다양한 구조의 레지스터 전송 레벨 회로가 실현될 수 있다. 따라서 사용자가 회로에 필요로 하는 조건들(동작시간,면적,전력소모등)을 만족하면서 해를 찾는 것이다.

상위 레벨 합성에 대한 연구는 1960년대에 시작되었으나 1970년대 말에 CMUDA[1]와 MIMOLA[2]가 발표됨으로서 본격적인 연구가 시작되었다. 1980년대 중반에 접어들면서 상위 레벨 합성은 CAD 분야에 있어서 주연구과제로 각광을 받기 시작하면서 많은 발

* 正會員, 漢陽大學校 電子工學科

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

接受日字:1994年 9月10日, 수정완료일: 1995年7月2日

전을 하게 되었다. 1980년대 중반까지의 연구 분야는 데이타 패스의 합성에 대한 연구가 주류를 이루었다. 데이타 패스 합성에 대한 연구는 그 응용분야가 확대되면서 특정부분의 전용 툴이 개발되기 시작했다. 대표적인 것으로 DSP 전용 합성 툴인 CATHEDRAL-III을 들 수 있다. 데이타패스 합성은 SEWHA[3]에서 파이프라인 개념이 처음으로 발표된 이후 이에 대한 연구도 활발히 진행되고 있으며[4][5] 또한 ASIC 설계를 지원하는 상위 레벨 합성에 대한 연구도 활발히 진행되고 있다.[6-10]

상위레벨 합성은 스케쥴링과 할당(allocation)으로 크게 구분되어 수행된다. 스케쥴링은 동작 시간(면적)의 제한 조건을 만족하면서 목적 함수인 면적(동작 시간)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. 상위 레벨 합성을 수행함에 있어서 할당이 먼저 수행되고 스케쥴링이 나중에 수행되면 각 제어 스텝에 할당될 하드웨어가 부분적으로 결정되기 때문에 스케쥴링의 결과에 영향을 준다. 한편 스케쥴링이 할당보다 먼저 수행되면 하나의 제어 스텝에서 필요한 연산자 및 메모리의 수가 결정되기 때문에 할당의 결과에 영향을 준다. 따라서 스케쥴링과 할당이 동시에 수행되는 것이 바람직하나 이는 레이아웃의 배치와 배선을 동시에 수행하는 것과 같이 NP-complete 문제이기 때문에 최적해를 구하는 데 heuristic한 알고리듬들이 제안되고 있다.

종래의 스케쥴링 알고리듬은 ASAP(As Soon As Possible), list scheduling[11][12], ILP(Integer Linear Programming)[4], FDS(Force Directed Scheduling)[13]등이 있으며 이 중 ILP는 최적해를 구할 수 있으나 수행 시간이 오래 걸린다는 단점을 가지고 있어서 대부분의 알고리듬은 heuristic한 방법을 적용하고 있다. 그러나 위의 알고리듬은 조건 분기 및 시간 제한등이 포함된 ASIC 설계를 지원하기에는 부적합하기 때문에 path based scheduling[10], condition vector를 이용한 스케쥴링[7], tree based scheduling[14]등의 알고리듬이 개발되었으나 최대 및 최소 시간 제한[15]에 대한 처리를 하지 못하고 있으며, relative scheduling[16]은 최대 및 최소 시간 지연에 대한 처리가 가능하지만 조건 분기에 대한 처리가 단점으로 지적된다. 따라서 본 논문에서는 ASIC 설계에 필요한 조건 분기 및 최대/최소 시간 제한을 처리할 수 있는 스케쥴링 알고리듬을 제안한다. 제안된 스케쥴링 알고리듬은 사용 가능한 자원을 제한 조건으로 하여 동작 시간의 최소화를 목적 함수로 한다.

본 논문의 구성은 다음과 같다. 2장에서는 스케쥴링의 입력이 되는 제어흐름 그래프인 HSFG에 대한 정의를 한 후, 제한 조건의 부그래프 표현 , 부그래프의 최소화 및 연산 노드의 동작 시간 할당등에 의한 스케쥴링 방법에 대해 기술한다. 3장에서는 실험 결과에 대해 설명하고, 본 논문의 결론은 4장에서 논한다.

II. 제한 조건의 최소화를 이용한 스케쥴링 알고리듬

본 논문에서 제안하는 스케쥴링 알고리듬은 control-dominated 회로와 같이 조건 분기 및 순차적인 동작을 수행하는 IC의 합성에 적용된다. 따라서 제어 흐름 그래프를 입력으로 하여 각 노드의 동작 시간을 결정하기 위해 그림 1과 같은 흐름에 의해 스케쥴링을 수행한다.

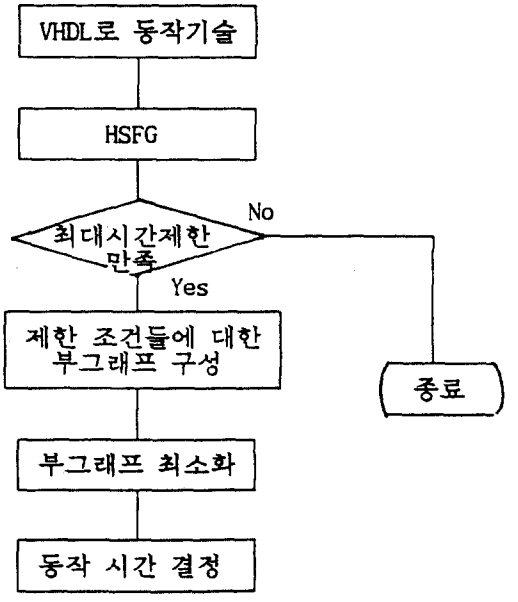


그림 1. 스케쥴링 흐름
Fig. 1. The scheduling flow.

기존의 스케쥴링 알고리듬은 다양한 제한 조건을 동

시에 고려하여 각 노드의 동작 시간을 결정하였지만 본 논문의 스케쥴링 과정은 제한 조건이 중복되는 것을 제거하여 모든 제한 조건이 중첩되지 않도록 재 구성한 결과를 이용하므로 노드의 동작 시간을 결정하는데 별도의 알고리듬을 적용하지 않고 순차적인 방법에 의해 모든 노드가 빠른 시간에 동작할 수 있도록 시간 할당을 수행한다.

HSFG를 입력으로 하는 스케쥴링은 4개의 과정으로 나누어져 수행된다.

- 최대 시간 제한 만족의 검색
- 제한 조건에 대한 부그래프 구성
- 부그래프의 최소화
- 각 노드의 동작 시간 결정

최대 시간 제한은 두 노드의 상대적 동작 시간이 일정 시간을 초과하지 않도록 하는 최대 시간 제한에 대한 검색을 하는 과정으로 Floyd-Warshell 알고리듬[17]을 적용한다. Floyd-Warshell 알고리듬은 그래프에서 에지의 가중치 합이 음 또는 양의 값을 갖는 cycle이 존재하는지를 검색하는 알고리듬이다.

최대 제한 시간의 검색은 스케쥴링 중에도 수행될 수 있다. 만일 최대 시간 제한을 만족하지 못하면 스케쥴링은 수행되지 못한다.

제한 조건에 대한 부그래프의 구성은 제한 조건이 두 노드의 상대적인 동작 시간을 나타내기 때문에 그래프의 terminal 노드는 선행노드들이 갖는 제한 조건들에 의해 많은 동작 시간의 제한을 가져올 수 있다. 따라서 중첩관계를 갖는 제한 조건들을 모두 제거함으로서 후행노드들에게 미치는 영향을 제거하여야 한다. 제한 조건은 두 노드간의 관계만을 의미하므로 에지에 의해 표현되고 있다. 따라서 에지간의 중첩을 아는 것은 어렵기 때문에 에지의 두 노드에 대한 부그래프를 구성한다. 이때 메모리 관리 및 부그래프 수정등을 고려하여 bit vector로 노드들을 표현한다.

부그래프의 최소화 과정은 제한 조건 수를 최소로 한다는 것을 의미한다. 그래프의 start 노드에 근접한 노드들에서 많은 제한 조건들이 존재한다면 이것은 결국 후행 노드들의 동작 시간 결정에 영향을 미치게 된다. 따라서 중첩되는 제한 조건의 수는 적을 수록 좋다. 중첩된 부그래프의 제거를 위하여 정렬 및 노드의 depth를 이용하며 2개의 중첩된 부그래프는 1개의 부그래프로 대체함으로서 제한 조건의 수를 줄인다.

각 노드의 동작 시간 결정은 중첩되는 제한 조건들이 제거되었으므로 HSFG를 BFS(Breadth First Search)로 각 노드를 탐색하여 ASAP와 같이 각 노드에 가장 빠른 시간 할당을 한다. 이와 같은 일련의 과정에 대해 2.1절 부터 상세하게 설명한다.

1. HSFG(Hanyang Sequential Flow Graph)

스케쥴링의 입력인 제어흐름을 표현하기 위해 HSFG를 정의하며 아래와 같은 하드웨어 설계 제한 조건을 표현할 수 있도록 한다.

1) 같은 제어스텝에서 서로 다른 2개의 값이 같은 변수에 할당될 수 없다.

2) 입출력 port의 경우, 같은 제어스텝에서 데이타의 읽기와 쓰기를 동시에 할 수 없다.

3) 한 제어스텝내에서 사용 가능한 하드웨어 자원의 수는 설계자에 의해 지정된 수를 넘길 수 없다.

4) VHDL의 wait문을 이용한 최소시간제한

상기의 제한 조건이 성립되는 두 노드는 같은 제어스텝에서 동작할 수 없다. 이러한 제한 조건들을 포함하는 HSFG에 대한 정의는 다음과 같다.

(정의 1) HSFG = (N,E)는 VHDL 기술문에 대응하는 노드들의 집합 N과 노드들간의 관계를 나타내는 방향성 에지의 집합인 E로 구성된다.

(정의 2) 에지 E = { ef, ed, ec, em }는 아래와 같은 에지로 구성된다.

여기서 ef : VHDL의 실행문 기술순서에 따른 제어흐름을 나타냄
ed : 데이타 종속 관계를 나타냄
ec : 4가지 하드웨어 설계 제한조건을 나타냄
em : 최대 시간 제한을 나타냄

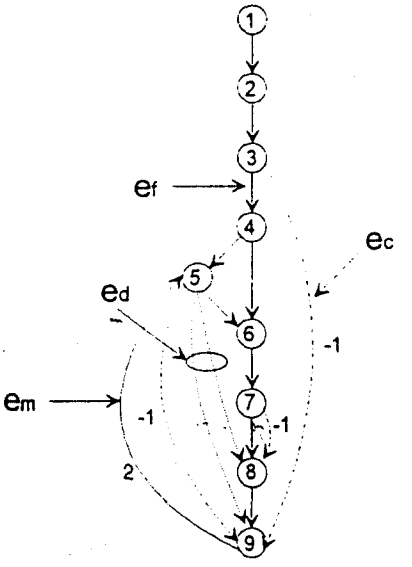
(정의 3) 노드 n1을 initial 노드로 하고 n2를 terminal 노드로 하는 방향성 에지에 대한 표현은 ei(n1,n2) (단 i = f,d,c,m)로 한다. 이 때, 노드 n1과 n2를 에지 ei의 인접 노드라 한다.

(정의 4) 아래와 같이 각 에지에 대한 가중치를 정의한다.

ef : 가중치가 0이며 HSFG에는 표시하지 않음.
ec : 가중치가 음의 정수이다.
이 에지에 인접된 두 노드가 동작하는 제어 스텝은 최소한 가중치의 절대값만큼 차이가 있어야 함.

```
entity example is
  port(branchpc,ibus : in BIT_VECTOR(3 downto 0);
    branch,ire : in BIT;
    ppc, popc, obus : out BIT_VECTOR(3 downto 0);
 );
end example;
architecture behavior of example is
begin
  process
    variable pc, oldpc : BIT_VECTOR(3 downto 0);
    begin
      ppc <= pc;                                    (1)
      popc <= oldpc;                                (2)
      obus <= ibus + "0100";                        (3)
      if(branch = '1')                              (4)
      pc := branchpc;                               (5)
      end if;                                       (6)
      wait until (ire = '1')                        (7)
      oldpc := pc;                                  (8)
      pc := pc + "0100"                             (9)
      assert NOW - pc'EVENT <= 200 ns
        report " Max. Time violation";
  end process;
end behavior;
```

(a) VHDL 기술



(b) HSFG

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

em : 가중치가 양의 정수이다.
이 에지에 인접된 두 노드가 동작하는 제어 스텝의 차는 가중치보다 클 수 없다.

(정의 5) HSFG의 노드 n1이 갖는 depth는 다음과 같이 정의한다.
HSFG의 root 노드를 0으로 하고 root노드에서 노드 n1까지 에지 ef만으로 구성된 경로중 가장 긴 경로의 길이를 그 노드의 depth라 한다.

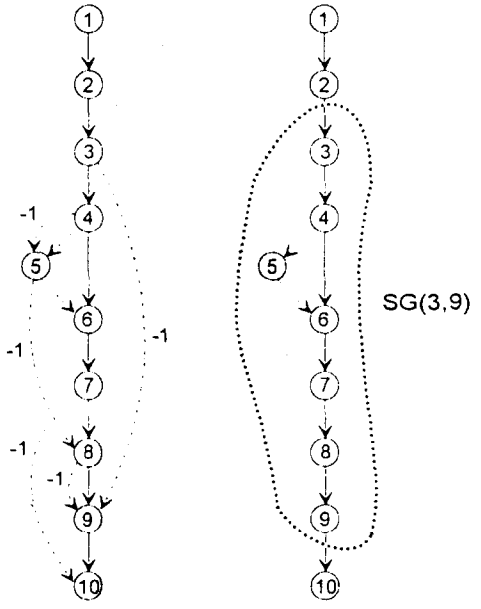
(정의 6) 본 논문에서 사용하는 용어를 아래와 같이 정의한다.
SG(n1,n2) : HSFG에서 n1이 source 노드이고 n2가 sink 노드인 부그래프 (그림 3 참조)

그림 2에서 최대시간제한의 예를 보이고자 assert 문을 삽입한 VHDL기술 예를 보여준다. 그림 2 (a)의 VHDL 기술에 대한 HSFG는 그림 2(b)와 같다. (1 제어 스텝 시간은 100ns라 가정) 그림 1에서 에지 ec(5,9)는 변수 pc에 덧셈의 결과와 변수 branchpc의 값이 같은 제어스텝에 할당될 수 없다는 제한 조건 1을 표시한다. 에지 ec(3,9)는 1개의 덧셈기만을 사용할 경우, 제한 조건 3에 해당한다. 에지 ec(3,9)와 ec(5,9)의 가중치는 인접된 두 노드의 동작 시간이 1 제어스텝만큼 차이가 나도 제한 조건을 만족하므로 -1을 부여한다. 그러나 초기에 설정된 에지 ec들이 모두 스케줄링에 적용될 필요는 없다. 예를 들면 ec(5,9)에 의한 노드 5와 노드 9가 다른 제어 스텝에 할당된다. 따라서 노드 3과 노드 9가 다른 제어스텝에 할당되어야 하는 제한조건 ec(3,9)는 ec(5,9)에 의해 만족된다. 그러므로 ec(3,9)와 같이 다른 제한 조건에 의해 대체될 수 있는 제한 조건들을 제거함으로서 효율적인 스케줄링을 수행할 수 있다. 그러나 단순히 에지 ec들만으로 제한 조건이 다른 제한 조건에 의해 대체될 수 있다는 것을 찾기가 어렵다. 따라서 본 논문에서는 에지 ec의 인접 노드가 source 노드와 sink 노드인 부그래프에 의해 에지 ec를 대체한 후(그림 3), 부그래프간의 포함 및 중첩관계를 이용하여 제한 조건을 나

타내는 부그래프의 수를 최소로 함으로서 최소 동작 시간에 스케쥴링을 하고자 한다. 그림 3의 예를 이용하여 스케쥴링 과정을 설명하기 전에 부그래프간의 관계를 정의하면 아래와 같다.

(정의 7) 두 부그래프 SG(n1,n2) 와 SG(n3,n4)에 대해 SG(n1,n2) ⊇ SG(n3,n4)이면 SG(n1,n2)와 SG(n3,n4)는 포함관계가 있다고 하며 SG(n1,n2)가 SG(n3,n4)를 포함한다고 한다.

(정의 8) 두 부그래프 SG(n1,n2) 와 SG(n3,n4)가 포함관계가 아니면서 SG(n1,n2)∩SG(n3,n4)={ni|SG(n1,n2)∋ni and SG(n3,n4)∋ni}인 노드 ni가 존재하면 두 부그래프는 중첩관계에 있다고 한다.



(a) 제한조건들 (b) 부그래프 SG(3,9)의 표시

그림 3. 제한 조건들의 표시와 부그래프
Fig. 3. The constraints and a subgraph SG(3,9)
(a) Constraints
(b) A subgraph SG(3,9)

2. 포함 관계를 갖는 부그래프의 제거

제한 조건들에 의해 동작 시간이 분리되는 노드들의 집합을 나타내는 부그래프의 수를 최소화하기 위해서는 부그래프를 최대한 제거하여야 한다. 부그래프의 제거는 다른 부그래프를 포함하는 부그래프를 찾아야 한다. 따라서 빠른 시간에 포함관계를 갖는 부그래프를 제거하기 위해서 다음과 같은 우선 순위로 부그래프들을 정렬한다.

1) source와 sink 노드의 depth차가 작은 값
2) 부그래프내의 노드수가 큰 것
3) source 노드의 depth가 작은 것

그림 3 (a)의 ec들에 대한 부그래프들을 그림 3에서 보여준다. 그림 4의 부그래프에 대한 정렬을 하기 위한 예로서 SG(4,5) 와 SG(8,9)는 정렬 우선 순위 1)과 2)의 항목인 depth 차와 노드들의 수가 같기 때문에 source 노드의 depth에 의해 정렬된다. 즉, 노드 4의 depth는 3이고 노드 8의 depth는 7이므로 SG(4,5)가 먼저 선택된다. 이상과 같은 방법에 의해 그림 3 (b)의 부그래프를 정렬하면 SG(4,5), SG(8,9), SG(5,8), SG(7,10), SG(3,9)의 순서로 된다.

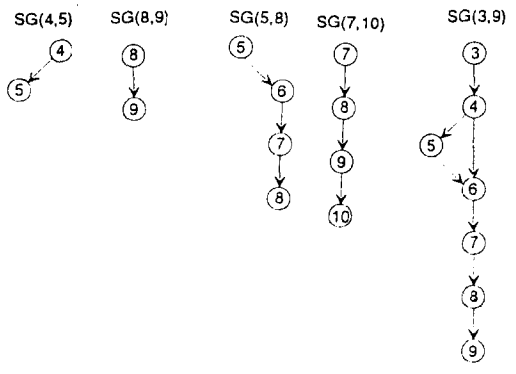


그림 4. 그림 3의 에지 ec들에 대한 부그래프
Fig. 4. Subgraphs for ecs in fig. 3.

정렬된 부그래프에 대해 포함관계를 갖는 부그래프를 제거의 예로서 그림 5와 같이 부그래프 SG(3,9)가 다른 부그래프 SG(4,5), SG(8,9)를 포함하면 부그래프 SG(3,9)를 제거한다. 즉, 노드 4와 5가 동시에 같은 제어 스텝에서 동작할 수 없기 때문에 노드 3과 9는 같은 시간 영역에서 동작할 수 없다는 제한 조건은 제거해도 된다.

그러나 조건 분기가 있는 경우에도 포함관계를 갖는 부그래프의 제거가 가능한 지를 보이기 위해 그림 6을 고려하자. 그림 6에서 부그래프 SG(1,8)이 SG(1,5)를 포함하고 있다.

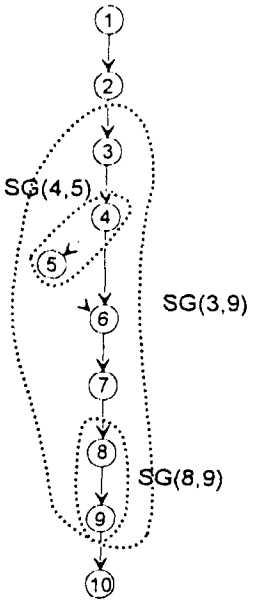


그림 5. 포함 관계를 갖는 부그래프의 예
Fig. 5. The example of the inclusion relation between some SGs

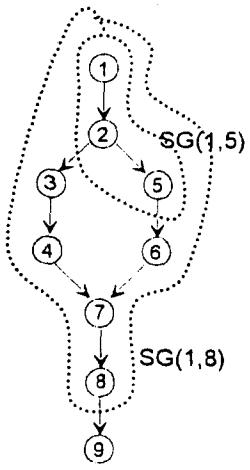


그림 6. 조건 분기가 있는 경우의 부그래프 SG 제거
Fig. 6. The elimination of a subgraph SG for the conditional branch

SG(1,5)에 의해 노드 5는 노드 1과 다른 제어 스텝에서 동작해야한다. 이 때 노드 8이 노드 5의 후행 노드이므로 SG(1,8)이 제거되어도 노드 8의 동작 시간은 노드 5와 같거나 또는 그 이후의 제어 스텝에서 동작하게 된다. 따라서 SG(1,8)이 갖는 의미인 노드 1과 8이 다른 제어 스텝에 동작해야 한다는 조건을 SG(1,5)에 의해 만족되므로 SG(1,8)을 제거할 수 있다. 따라서 포함관계를 갖는 부그래프의 제거에 대해 다음과 같이 정리할 수 있다.

[정리 1] 조건 분기에 관계없이 부그래프 SG1이 SG2를 포함하면 부그래프 SG2를 포함하고 있는 부그래프 SG1을 제거한다.

프로그램에서 부그래프의 처리를 위해 부그래프에 포함된 노드를 bit vector에 의해 표현한다. 즉 각 부그래프를 구성하는 노드 번호에 대응하는 bit위치에 1의 값을 부여함으로서 그 노드가 부그래프에 포함되었음을 나타낸다. 따라서 부그래프간의 포함관계는 Bitwise and 연산에 의해 다른 부그래프의 source와 sink 노드가 존재하는 지를 검색함으로서 알 수 있다. 그림 5에서 SG(3,9)의 경우 SG(3,4)의 source와 sink 노드가 존재하므로 부그래프 SG(3,9)는 제거된다. 그림 3에 대해 포함 관계를 갖는 부그래프들이 제거되면 그림 7과 같이 HSFG가 구성된다.

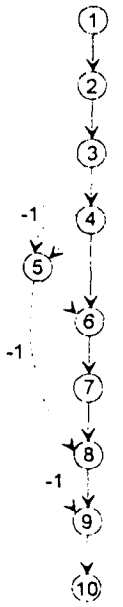


그림 7. 그림 3(a)의 ec 제거
Fig. 7. The result of the elimination of ec in figure 3(a).

3. 중첩되는 부그래프에 대한 수정

포함관계를 갖는 부그래프들이 제거된 후, 다수의 부그래프들을 하나의 새로운 부그래프로 대치함으로서 전체적인 동작 시간을 최소화하기 위해 부그래프간의 중첩관계를 검색한다. 부그래프간에 중첩관계가 있으면 기존의 부그래프들을 제거하고 중첩되는 노드들로 구성된 새로운 부그래프를 생성한다. 그러나 조건 분기가 있는 경우 그림 8과 같이 다양한 형태의 부그래프간의 중첩관계가 존재하게 된다. 따라서 스케쥴링에 의해 가능한 HSFG의 노드가 빠른 제어스텝에서 동작하고 전체적인 동작 시간의 최소화를 위하여 아래와 같은 우선순위에 의해 부그래프의 중첩관계를 찾아서 부그래프 수를 최소화 한다.

1) 두 sink 노드가 같은 조건의 경로에 존재한다.
2) 한 부그래프의 source 노드와 다른 부그래프의 sink 노드가 같은 조건의 경로에 존재하면서 source 노드가 다른 부그래프에 포함되어야 한다.
3) 두 source 노드가 같은 조건의 경로에 존재한다.

우선 순위 1)에 대한 예는 그림 8(b)와 (d)이다. 그림 8(b)의 경우 두 SG(5,13)과 SG(9,14)의 sink 노드 13과 14가 같은 조건의 경로에 존재한다. 우선 순위 2)의 예로는 그림 8(a)이다. 그림 8(a)에서 SG(2,6)의 sink 노드(노드 6)과 SG(5,13)의 source 노드(노드 5)가 같은 조건(COND가 false)의 경로에 위치하고 노드 5가 SG(2,6)에 포함된다. 우선 순위 3)에 대한 예는 그림 8(c)이다. 따라서 그림 8의 부그래프들은 그림 9와 같이 중첩되는 노드로만 구성된 새로운 부그래프로 대체한다. 포함관계 제거시에 효율적인 부그래프 제거를 위해 부그래프들을 정렬하였듯이 그림 9와 같은 중첩 제거를 위해 각 부그래프를 다음과 같은 우선 순위에 의해 재정렬한다.

1) Source 노드의 depth가 작다.
2) Source 노드와 sink 노드의 depth차가 작다.

정렬된 부그래프들중 중첩관계를 갖는 부그래프들에 대해 중첩되는 노드를 찾아 새로운 부그래프를 생성한다. 생성된 부그래프에 대응하는 에지 ec를 만들고 기존의 부그래프 및 ec들을 제거한다. 중첩관계를 갖는 부그래프들이 존재하지 않을 때까지 반복한다.

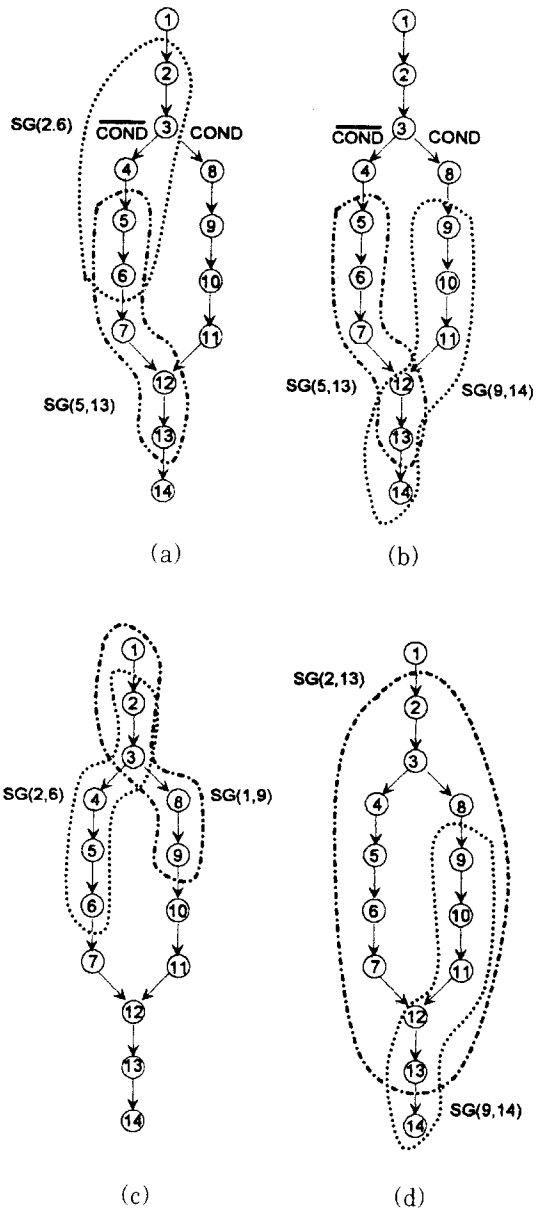


그림 8. 조건 분기가 있는 경우의 중첩
Fig. 8. Overlap graphs in a conditional branch

4. 최대 시간 제한의 검색

부그래프를 최소화 한 후, 그림 10과 같이 에지 em이 존재하면 현재의 부그래프를 이용한 스케쥴링 결과가 최대 시간 제한을 만족할 수 있는 지를 검색한다. (최대 시간 제한의 검색은 부그래프를 최소화하지 않고도 수행할 수 있다.) 최대 시간 제한의 검색은 그림

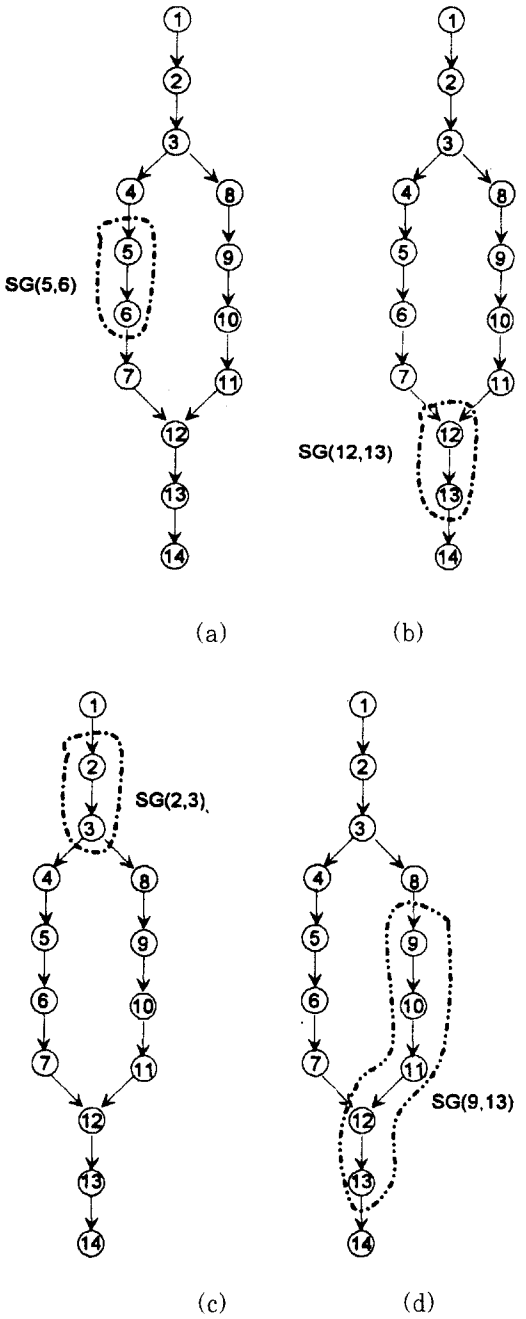


그림 9. 그림 8의 수정 결과
Fig. 9. The modified result of Fig. 8.

10과 같이 em(8,3)이 있는 경우 노드 3과 8사이에 가중치의 합이 음수가 되는 사이클이 있는 지를 찾는 것이다. 가중치 합이 음수가 됨은 ec에 의한 두 노드의 시간 분리가 최대 허용 시간을 벗어나서 요구되는 시간에 동작할 수 없다는 것을 나타낸다. Floyd-Warshell 알고리듬[17]을 이용하여 음의 가중치 합을 갖는 사이클이 존재하는 지를 검색한다. 사이클의 가중치 합이 음의 값을 갖으면 최대 시간 제한을 만족하지 못하는 것이고, 가중치 합이 음인 사이클이 존재하지 않으면 최대 제한 시간을 만족하므로 스케줄링을 계속 수행한다.

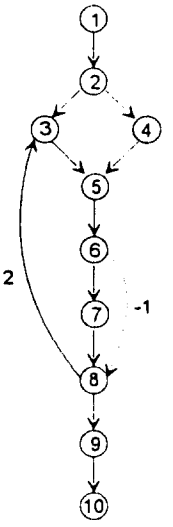


그림 10. 최대 시간 제한의 예
Fig. 10. A example of A maximum time constraint

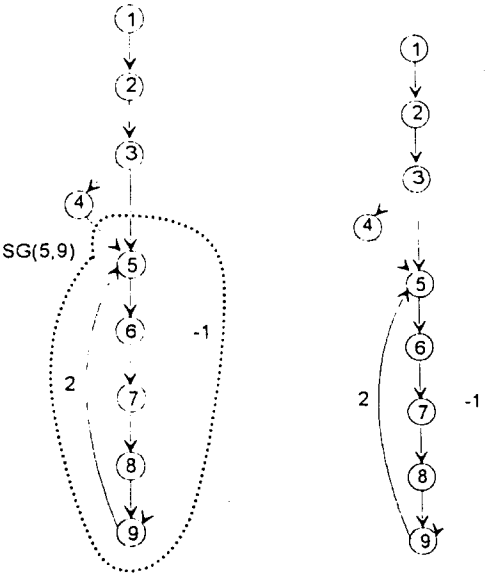
그림 10의 경우 가중치의 합이 음인 사이클이 존재하지 않으므로 최대 시간 제한을 만족한다. 그러나 그림 11(a)와 같이 부그래프의 일부가 중첩되는 경우,

즉 ec의 종료 노드 9가 최대 시간 제한에 의한 부그래프 SG(5,9)내에 존재하는 경우에는 그림 11(b)와 같이 ec(3,9)를 ec(5,9)로 대체한 후 최대 시간 제한을 만족하는 지를 검색한다. 이것은 ec(3,9)에 의해 SG(5,9)내에서 노드들의 동작 시간이 분리되어서 최대 시간 제한을 만족시키지 못할 수도 있기 때문이다.

5. 각 노드의 동작시간 결정

그림 7과 같이 HSFG에서 제한 조건을 나타내는 부그래프를 최소화 한 후, 각 노드의 동작 시간을 결정하여야 한다. 각 노드의 동작 시간은 1로 초기화한다. 노드의 동작 시간을 정하기 위해 BFS 방법을 적용하여

HSFG의 root 노드부터 에지 ef를 따라 탐색한다. 이 때, 탐색중인 노드들 중 에지 ec의 terminal 노드에 해당되는 노드들과 에지 ef에만 연결되어 있는 노드들을 구분하여 동작 시간을 결정한다. 즉 에지 ef만의 terminal 노드인 경우, 선행노드들이 갖는 동작 시간의 최대값을 부여한다. 에지 ec의 terminal 노드인 경우 선행노드의 동작 시간뿐만 아니라 이 노드에 관련된 모든 ec에 대해 initial 노드의 동작 시간과 ec의 가중치를 고려하여야 한다. Ec의 terminal 노드에 대한 동작 시간은 식 1과 같다.



(a) 수정 전 (b) 수정 후

그림 11. ec의 수정 예
Fig. 11. A example of the modification for ec
(a) Before the modification
(b) After the modification

$$T = MAX\{MAX(\text{ec의 initial 노드의 동작시간} - \text{ec의 가중치}),\ \text{선행노드의 동작시간})\} \quad (1)$$

그림 12는 부그래프가 최소화된 HSFG에 대해 스케줄링 과정을 보여준다. 노드 12의 동작 시간 T(12)가 3이고 노드 31의 동작 시간 T(31)이 1이라고 하자. 노드 34가 ec(12,34)와 ec(31,34)의 terminal 노드가 되기 때문에 queue에 저장된다. 현재 queue에 노드 34만 존재하므로 노드 34에 대한 동작 시간을 계산하면 T(34) = MAX{(MAX(3 - (-1), 1 - (-1)),3} = 4가 된다. 이때 노드 23, 32, 33의 동작 시간은 BFS에 의해 노드를 탐색하면서 선행노드들이 갖는 동작 시간 중 최대의 시간을 갖는다. (즉, T(23) = MAX(T(12),T(22)) = 3 T(32) = 1, T(33) = MAX(T(23), T(32)) = 3)

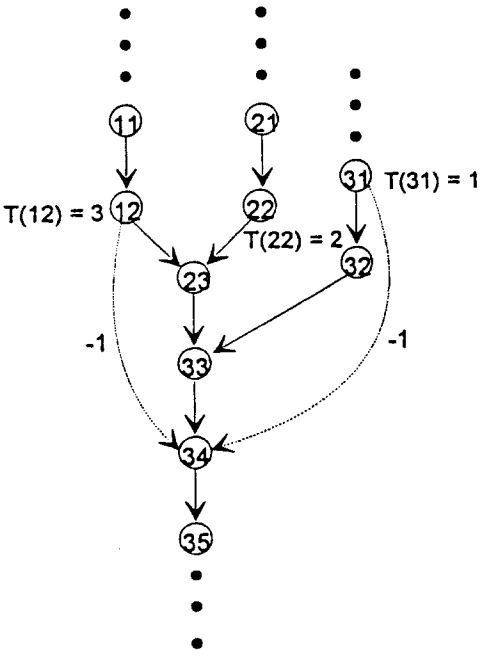


그림 12. 노드의 동작시간 결정의 예
Fig. 12. An example of the operation time assignment for a node.

Ⅲ. 실험결과

본 논문에서 제안한 알고리듬을 SPARC 1+에서 C 언어로 구현하였다. 예제 회로는 92년에 발표된 benchmark HLSynth92를 사용하였다. 표 1은 본 논문에서 제안한 스케줄링 알고리듬을 benchmark 회로인 i8251에 적용한 실험 결과이다. 본 논문의 제안된 스케줄링 과정은 VHDL의 입력으로 부터 직접 하드웨어에 대한 매칭이 수행된 결과인 카운터등이 존재하는 HSFG를 입력으로 받는다. 따라서 표1에서 사용된 하드웨어 자원을 보면 4 bit 카운터와 8 bit 카운터가 사용되고 있다.

표 1. 실험 결과
Table 1. The experimental result.

	예제 회로	노드 수	제어스텝	상태수	FU-(1)	CPU시간(Sec)
본 논문	i8251 main	168	5	12	+(1) CNT8(1)	0.2
	i8251 Tx	98	9	24	CNT8(1) CNT4(1)	1.2
	i8251 Rx	245	17	39	+(1) CNT8(1) CNT4(1)	1.2
[8]*	i8251	100	5	15		2.0
	i8251 Tx	93	8	22	-(1)	8.8
	i8251 Rx	84	13	23	-(1)	2.0
	i8251 HUNTER	93	7	12	-(1)	0.1

* IBM Risc System/6000 Model 520 workstation
**CNT8 : 8 bit binary counter CNT4 : 4 bit binary counter

92년에 발표된 benchmark HLSynth92 I8251은 3개의 process문으로 기술되어 있다. 각 process에 대한 스케쥴링을 위해 주어진 자원 이용의 제한 조건은 1개의 덧셈기만을 이용하도록 하였으며 덧셈기는 1개의 제어 스텝에서 동작한다고 가정하였다. 표 1에서 제어스텝은 회로를 구동하는 상태 중 가장 긴 상태의 수를 나타내며 상태수는 전체 상태 수를 나타낸다. 본 논문의 예제회로는 VHDL로 기술되어 있고 [10]의 예제회로는 ISPS로 기술되어 있기 때문에 노드 수 및 FU의 형(type)이 다른 결과를 보이고 있다. 특히 본 논문에서 적용한 i8251 Rx는 [10]의 i8251 Rx와 i8251 HUNTER가 하나의 회로로 기술된 것이다. 따라서 각 예제회로의 노드 수가 다르게 표시되고 있다. 또한 입력 기술의 방법에 따라 스케쥴링 결과가 달라질 수 있으므로 표 1에서 [10]의 결과는 본 논문과의 직접적인 비교보다는 본 논문에서 제안된 스케쥴링 방법의 효용성을 보이기 위해 제시하였다.

Ⅳ. 결 론

본 논문에서는 control dominated ASIC 설계를 지원하는 새로운 스케쥴링 알고리듬을 제안하였다. 제안한 알고리듬은 다양한 제한 조건들 (입출력 핀의 사용, 변수의 데이타 천이, 자원의 이용등)을 부그래프로 처리하면서, 조건 분기가 있는 경우도 제한 조건들을 따로 처리하지 않고 조건 분기가 아닌 경우와 동일하게 부그래프로 처리할 수 있다는 장점을 가지고 있다. 특히 제안된 알고리듬은 조건 분기 뿐만 아니라 최대/최소 시간 제한등을 처리할 수 있기 때문에 CRT 콘트롤러, 메모리 콘트롤러등과 같은 제어 회로를 합성하는데 적합하다. 또한 조건 분기에 따른 제어 흐름 그래프를 분리하지 않고 하나의 부그래프로 처리하였기 때문에 벤치마크 회로에 대한 실험 결과는 빠른 스케쥴링 수행 시간을 보이면서 제어 스텝의 수가 최적에 근사한 해를 보여주고 있다.

그러나 제안된 알고리듬은 주어진 그래프에 대해 회로가 최소의 시간에 동작하도록 하고 있지만 기술된 VHDL에 따라 순차적으로 그래프가 구성되므로 VHDL 기술 순서에 의해 동작 시간이 영향을 받는 것은 해결하지 못하고 있다. 또한 앞으로는 저전력 소비 ASIC 설계를 위한 스케쥴링 방법에 대한 연구를 진행하고자 한다.

참 고 문 헌

[1] S.W. Director, A.C. Parker, D.P. Siewiorek, and, D.E. Thomas, Jr., "A Design Methodology and Computer Aids for Digital VLSI Systems," IEEE trans. Circuits and System , pp 634-635, July, 1981.

[2] Peter Marewdel, "The MIMOLA Design System: Tools for the Design of Digital Processors", Proc. 21st DAC, pp. 587-593, 1984.

[3] N. Park and A.C. Parker, "SEHWA : A PROGRAM FOR SYNTHESIS OF PIPELINES," Proc. 23rd DAC, 1986, pp. 454-460.

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

[5] K.S. Hwang, A.E. Casvant, C.T. Chang and M.A. d'Abreu, "SCHEDULING AND HARDWARE SHARING IN PIPELINED DATA PATHS," ICCAD-89, 1989, pp. 24-27.

[6] Giovanni De Michelli and David C. Ku, "HERCULES-A System for High-level Synthesis," Proc 25th DAC, 1988, pp. 483-488.

[7] T. Kim, J. W. S. Liu, C. L. Liu, "A Scheduling Algorithm for Conditional Resource Sharing," ICCAD, 1991, pp. 84 -87.

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

[9] W. Wolf, A. Takach, C. Huang, and R. Manno, "The Princeton University Behavioral Synthesis System," Proc. 29th DAC, 1992, pp. 182 - 187.

[10] Raul Camposano et al, "High-level ULSI Synthesis," Kluwer academic press, 1991.

[11] Barry M. Pangrle and Daniel D. Gajski, "Slicer : A State Synthesis for Intelligent Silicon Compilation," ICCD, 1987, pp.42-45.

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

[13] 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.

[14] 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.

[15] D.E. Thomas et al., "The System Architect's Workbench," Proc. 25th DAC, 1988, pp. 337-343.

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

[17] C.H. PAPADIMITRIOU and K. STEIGLITZ, COMBINATORIAL OPTIMIZATION: ALGORITHM AND COMPLEXITY, Prentice-Hall press, 1982.

저 자 소 개

金 基 鉉(正會員) 제31권 A편 1호 참조
현재 한양대학교 전자공학과 박사과정

鄭 正 和(正會員) 제31권 A편 1호 참조
현재 한양대학교 전자공학과 교수