Control Dominated ASIC 설계를 위한 최소 제한조건 스케쥴링 알고리즘

인 치 호†

요 약

본 논문에서는 최적의 control dominated ASIC 설계를 위한 VHDL 중간 표현 그래프 CDDG(Control Dominated Data Graph) 와 최소 제한조건 스케쥴링 알고리즘을 제안한다.

CDDG는 VHDL 동작 기술의 조건 분기 및 반복구조 등을 효과적으로 나타낼 수 있는 제어 흐름 그래프로서 하드웨어 설계의 특성을 지원하기 위한 데이터 종속 관계, 하드웨어 자원 제한 및 시간 제한 조건이 표현된다. 제안된 스케쥴링 알고리즘은 CDDG의 부그래프로 표현된 제한조건에 대해, 부그래프들의 최소화하는 과정과 회로 동작의 허용 시간을 검사하는 최대 시간 제한의 검색 및 각 연산 노드들의 동작시간을 결정하는 과정으로 수행된다.

벤치마크 데이터를 사용하여 실험한 결과, 제안된 알고리즘이 기존의 알고리즘에 비해 우수함을 확인하였다.

A Minimal Constrained Scheduling Algorithm for Control Dominated ASIC Design

Chi-Ho Lin†

ABSTRACT

This thesis presents a new VHDL intermediate format CDDG(Control Dominated Data Graph) and a minimal constrained scheduling algorithm for an optimal control dominated ASIC design.

CDDG is a control flow graph which represents conditional branches and loops efficiently. Also it represents data dependency and such constraints as hardware resource and timing. In the proposed scheduling algorithm, 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 effectiveness of the proposed algorithm has been proven by the experiment with the benchmark examples.

1. 서 론

상위 레벨 합성의 목표는 동작을 기술하는 알고리즘 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, 전력소모 등)을 만족하면서 해를 찾는 것이다 [1-2].

상위 레벨 합성은 스케쥴링과 할당(allocation)으로 크게 구분되어 수행된다. 스케쥴링은 주어진 제한조건에 대해 목적 함수를 최소화하도록 연산이 수행될 시간을 결정하는 것이다. 할당은 구현되는 하드웨어의

※ 이 논문은 1998년도 세명대학교 교내학술연구비 지원에 의해 수행된 연구임.
† 정 회 원 : 세명대학교 컴퓨터과학과 교수
논문접수 : 1998년 10월 19일, 심사완료 : 1999년 5월 3일

면적이 최소가 되도록 연산을 연산자에, 변수를 메모리에 할당하고 메모리와 연산자 사이의 연결구조로 버스나 멀티플렉서를 할당하는 것이다[3-6].

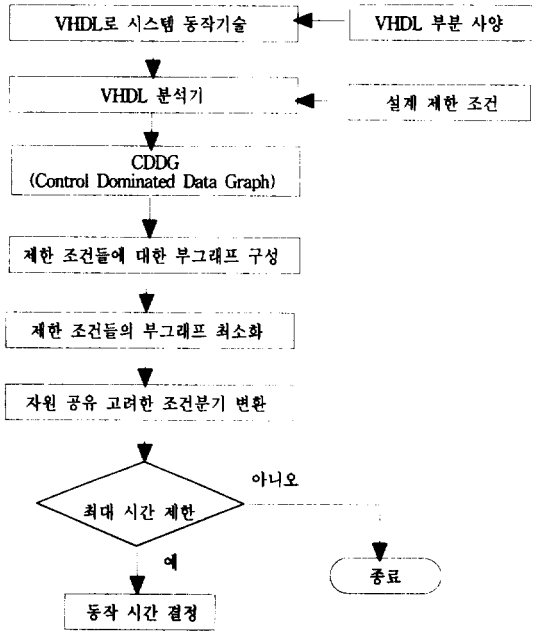
종래의 스케줄링 알고리즘은 ASAP(As Soon As Possible), list 스케줄링, FDS(Force Directed Scheduling)[2], ILP(Integer Linear Programming)[6] 등이 대표적인 알고리즘이며 이 중 ILP는 최적 해를 구할 수 있으나 변수 수의 증가에 따라 수행 시간의 제한을 받는 단점이 있고, FDS의 경우는 단순 조건 분기는 처리가 되나, 시간 제한 등의 제한조건을 처리하는데 있어서 미흡하기 때문에 대부분의 알고리즘은 휴리스틱한 방법을 적용하고 있다. 그러나 위의 방법들은 조건 분기 및 시간 제한 등이 포함된 ASIC 설계를 지원하기에는 부적합하기 때문에 path based scheduling[7], 조건 벡터를 이용한 스케줄링[1], tree based scheduling[8] 등의 알고리즘이 개발되었으나, 설계에 필요한 상대적인 데이터 종속성 및 최대/최소 시간에 대한 처리가 부족하며, 계층적인 구조 및 조건 분기에 대한 처리가 미흡하다[9].

따라서 본 논문에서는 ASIC 설계에 필요한 조건 분기 및 최대/최소 시간 제한을 처리할 수 있는 스케줄링 알고리즘을 제안한다. 제안된 스케줄링 알고리즘은 사용 가능한 자원을 제한 조건으로 하여 동작 시간의 최소화를 목적 함수로 하며 control dominated 회로와 같이 조건분기 및 순차적인 동작을 수행하는 ASIC의 합성에 적용된다. 기존의 스케줄링 알고리즘은 다양한 제한 조건을 동시에 고려하여 각 노드의 동작 시간을 결정하였지만 본 논문의 스케줄링 과정은 제한 조건이 중복되는 것을 제거하여 모든 제한 조건이 중첩되지 않도록 재구성한 결과를 이용하므로 노드의 동작 시간을 결정하는 데 별도의 알고리즘을 적용하지 않고 순차적인 방법에 의해 모든 노드가 빠른 시간에 동작할 수 있도록 시간 할당을 수행한다.

본 논문의 구성은 2장에서는 VHDL 중간 표현 그래프를 기술하고, 또한 본 논문에서 제안한 스케줄링 알고리즘에 대하여 기술한다. 실험 및 결과는 3장 및 4장에서 기술한다.

2. 최소 제한조건 스케줄링 알고리즘

본 논문에서는 조건 분기 및 하드웨어 동작시간을 효율적으로 처리할 수 있는 스케줄링 알고리즘에 대해 설명한다. 제안한 스케줄링 알고리즘은 사용 가능한 자원을 제한조건으로 하여 동작시간의 최소화를 목적 함수로 하며, 그래프의 중첩 관계와 조건 분기 특성을 고려한 자원 공유 개념에 의해 스케줄링이 수행된다. VHDL 기술로부터 구성된 중간 표현 그래프 CDDG는 하드웨어의 특성, 하드웨어 자원의 제한 및 동작시간 제한조건들, 그리고 제어 흐름과 데이터 종속 관계가 에지 및 노드로 구성하는 부그래프로 표현된다. CDDG에 포함된 부그래프간의 중첩 및 포함 관계를 이용하여 제한조건의 수를 최소화하고 조건 분기를 효율적으로 처리함으로써 최적의 스케줄링 결과를 얻는다.



(그림 1) 최소 제한조건 스케줄링 알고리즘

중간 데이터 그래프인 CDDG를 입력으로 하는 본 논문의 스케줄링 알고리즘은 (그림 1)과 같은 과정으로 수행된다.

기존의 스케줄링 알고리즘은 다양한 제한조건을 동시에 고려하여 각 노드의 동작시간을 결정하지만 본 논문의 스케줄링 알고리즘은 제한조건이 중복되는 것을 제거하여 모든 제한조건이 중첩되지 않도록 하며, 또한 조건 분기가 있는 경우도 제한조건을 따로 처리하지 않고 조건 분기가 없는 경우와 같이 재구성한 결과를 이용하여 노드의 동작시간을 결정한다. 또한 반

복적인 노드의 이동에 의한 순차적인 방법에 의하여 모든 노드가 빠른 시간에 동작 할 수 있도록 동작시간을 결정한다. 그리고 본 스케쥴링에서의 최대 시간 제한조건의 검색은 두 노드의 상대적 동작시간이 일정 시간을 초과하지 않도록 하는 최대 시간 제한에 대한 검색을 하는 과정으로서 최대 제한 시간 조건을 만족하지 못하면 스케쥴링을 수행하지 못한다.

2.1 VHDL 중간 표현 그래프

현재까지의 많은 연구에서는 중간 표현으로써 제어 흐름 그래프(Control Flow Grap, CFG)를 주로 사용하고 있는데, 그 이유는 CFG가 설계된 VHDL 구문들의 구조에 맞도록 형성되어 제어신호에 따르는 신호들의 경로를 탐색하는데 매우 유용하기 때문이다. 그러나, 현실적으로 VHDL에 의한 설계를 수행하면 제어 신호에 대한 구문 뿐 아니라 데이터 처리구문에서도 많은 오류가 발생함을 알 수 있다. 데이터 처리를 대상으로 하는 VHDL구문의 변형 형태 중 또 하나는 데이터 흐름(Data Flow Graph, DFG)인데, 이 그래프는 제어구문에 의해 신호의 흐름이 제어되도록 구성되는 것이 아니라, 처리된 데이터가 다음의 어떤 데이터 처리구문에 영향을 주는가를 명시한다. 즉, 실제 VHDL 구문의 제어신호에 따르는 데이터의 흐름을 탐색하기 위해서는 CFG가 사용되어야 하며, 데이터의 상호 연관 관계를 이용하여야 할 때는 DFG가 사용되어야 한다. 그러나, 동일한 설계에 대해 두 그래프를 사용하여야 한다는 것은 비효율적이며 처리시간 또한 많이 소요될 것이다[10-11].

따라서 본 논문에서는 데이터의 흐름과 제어 흐름을 효과적으로 하나로 나타낼 수 있는 중간 데이터 구조 그래프 CDDG를 내부 데이터 구조로 구성한다. CDDG는 하드웨어 설계의 특성을 지원하기 위한 조건 분기, 순차적인 동작 및 시간 제한을 표현 할 수 있도록 구성한다.

VHDL에서 프로세스(process)문 내에 기술되지 않은 동작은 병렬로 이루어진다고 보며 프로세스문 내에 기술된 회로는 순차적(sequential)으로 동작한다고 본다. 따라서 VHDL로 회로의 동작을 기술할 때는 프로세스문에 의한 모듈화가 이루어진다. 프로세스문의 특성이 회로의 동작조건 및 순차적인 동작을 나타내고 있기 때문에 제안된 시스템에서 지원하고자 하는 control dominated 회로의 기술에 프로세스문의 이용이 적합하다. 따라서 CDDG는 제어 흐름 그래프에 데이터 종속 및 제한 조건들을 포함하는 방향성 그래프로서 각 프로세스문 별로 구성되며 정의 1과 같다.

(정의 1) VHDL 기술을 D = (V, E)라 하고, V는 D에 나타나 있는 VHDL 기술문에 대응되는 노드들의 집합이며, E는 D에 나타나 있는 노드들과의 관계를 나타내는 방향성 에지라고 한다.

CDDG는 한 제어 스텝에서 다음과 같은 하드웨어 설계 제한조건을 표현할 수 있도록 구성한다.

첫째, 같은 제어 스텝에서 서로 다른 두 개의 값이 같은 변수에 할당될 수 없다. 같은 제어 스텝에서 한 변수에 2개의 값이 할당되면 데이터간의 충돌이 발생하기 때문에 동작의 오류가 발생한다.

둘째, 입·출력 포트(port)의 경우, 같은 제어 스텝에서 한 포트에 대한 데이터의 읽기와 쓰기를 동시에 할 수 없다. 이 경우도 첫번째와 같은 이유에 의해 한 포트에 대한 입·출력 데이터의 읽기와 쓰기 동작은 최소한 한 개의 제어 스텝 이상 분리되어야 한다.

셋째, 하나의 제어 스텝에서 사용 가능한 하드웨어 자원의 수는 설계자에 의해 지정된 수를 넘길 수 없다. 따라서 제어 흐름상에서 사용자가 정의한 하드웨어 수를 초과하는 경우에는 하드웨어 수가 초과되기 전의 연산과 초과된 직후의 연산간에 동작시간이 분리되어야 한다.

넷째, VHDL의 wait 문은 시간 지연이나 신호의 천이를 지정할 수 있다. 따라서 신호의 천이가 발생되는 시점을 알 수 없기 때문에 wait 문이 나오는 경우는 지금까지의 실행 문과는 다른 제어 스텝에서 동작을 하여야 한다.

VHDL 분석기에 구성된 D의 중간 데이터 구조 그래프인 CDDG는 G_s, G_c, G_t 의 3개의 방향성 부그래프들로 구성되며 부그래프들은 다음과 같이 정의한다.

(정의 2) 순차 그래프(sequential graph) G_s = (V, E_s)는 VHDL 기술문에 대응되는 노드들의 집합 V와 VHDL 기술의 선행 및 후행 관계를 나타내는 방향성 에지의 집합인 E_s로 구성된다.

(정의 3) 하드웨어 제한조건 그래프(hardware constrint graph) G_c = (V, E_c)는 VHDL 기술문에 대응되는

노드들의 집합 V와 4가지 하드웨어 설계 조건을 나타내는 방향성 에지의 집합인 E_c로 구성된다.

(정의 4) 시간 관계 그래프(timing relation graph) $G_t = (V, E_t)$는 VHDL 기술문에 대응되는 노드들의 집합 V와 시간 제한조건을 구성하는 방향성 에지의 집합인 E_t로 구성된다.

(정의 5) 부그래프 G_c, G_t의 각 에지에 대한 가중치를 다음과 같이 정의한다.

E_c : 가중치가 1이며 두 노드가 동작하는 제어스텝의 분리가 필요하다는 것을 의미한다.

E_t : 가중치가 음수이며(역그래프) 두 노드가 동작하는 제어스텝의 차는 가중치의 절대값보다 클 수 없다.

3개의 부그래프들 중 VHDL의 실행문 기술 순서에 따른 제어 흐름을 나타내는 부그래프 G_s는 클럭 부분(clock part) 이나 시간 지연 부분(waiting part)에서는 존재하지 않으며, 일반적으로 CDDG는 순차적으로 구성된 부그래프 G_s에 하드웨어 제한조건 부그래프 G_c와 시간 제약 부그래프 G_t가 추가로 구성된다.

그리고 CDDG의 VHDL 기술문에 대응되는 노드들의 갖는 그래프의 범위를 다음과 같이 정의한다.

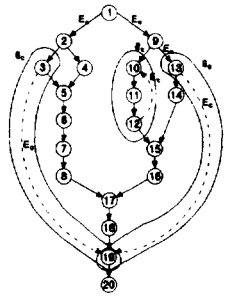
(정의 6) 에지로 연결된 그래프의 범위가 (N1, N2) 일때 N1이 시작(start) 노드이고, N2가 종료(end) 노드인 부그래프라 한다.

VHDL 분석기에 의해 중간 표현 그래프인 CDDG로 생성된 노드들은 (그림 2)와 같이 3개의 부그래프들로 구성된다. (그림 2)(b)의 각 노드 번호와 (그림 2)(a)의 VHDL 기술문과는 1대 1로 대응한다. 즉 (그림 2)(a)의 VHDL 기술문에 붙여진 번호와 (그림 2)(b)의 그래프의 노드 번호가 대응한다. 그리고 (그림 2)(a)의 VHDL 기술문의 (13), (14)의 문장은 스택(stack)의 주소를 지정하여 자료를 후입 선출하는 스택 함수의 하드웨어 모듈과 스택으로부터 후입 선출된 자료를 받는 연산의 복합 문장이기 때문에 (그림 2)(b)의 그래프 노드 번호 13과 14에 대응한다. (그림 2)(b)에 있는 에지들은 정의 2, 정의 3과 정의 4에 기술되어 있는 방향성 에지를 표현하고 있다. (그림 2)(b)에서 에지(3,19)는 한 제어 스텝에서 한 변수 Y_temp에 대한 데이터의 읽기와 쓰기를 동시에 할 수 없다는 제한조건 1을 표시한다. 이 경우 입·출력 데이터의 읽기와 쓰기 동작은 최소한 한 개의 제어 스텝 이상 분리되어야 한다. 또한 assert문을 이용하여 최대 시간 제한이 표시된 (그림 2)(a)와 대응되는 (그림 2)(b)의 에지(10, 12)의 경우 assert문에 의해 10번째 기술문이 수행된 후 12번째 기술문이 200ns이내에 동작하여야 한다는 것을 표시한다. 한 제어스텝이 100ns이면 10번째와 12번째 기술문에 있는 동작은 두 개의 제어스텝 이내에서 동작하여야 한다. 따라서 이 최대 시간 제한을 나타내는 에지 E_t의 가중치는 -2가 부여된다.

CDDG에 구성된 데이터 종속 그래프 G_c에 대해 각 경로별로 부그래프의 에지 E_c의 포함 및 중첩 관계를 다음과 같이 고려한다.

```
entity sub_2910 is
 port (
  RLD_BAR : in BIT;
          CI : in BIT;
         clk : in clock;
           D : in BIT;
           Y : out BIT; );
end sub_2910;
architecture sub_2910 of sub_2910 is
begin
process
begin
 if RE = "000000000000" then                          (1)
  if fail = '1' then                                  (2)
   Y_temp  := D;                                      (3)
  else
   Y_temp1 := uPC;                                    (4)
  end if;                                             (5)
  if (SP /= 0) then                                   (6)
   SP := SP - 1;                                      (7)
  end if;                                             (8)
 else
  if (FAIL = '0') then                                (9)
   if (SP /= 0)) then                                (10)
    SP := SP - 1;                                    (11)
   end if;                                           (12)
   assert (NOW - SP'event <= 200ns)
    report "set up violation"
  else
   Y_temp := stack(sp);                          (13)(14)
  end if;                                           (15)
  RE : RE - "000000000001";                         (16)
 end if                                             (17)
 if ( RLD_BAR = '0' ) then                          (18)
   Y := Y_temp;                                     (19)
 end if;                                            (20)
end process;
end sub_2910;
```

(a) VHDL 기술

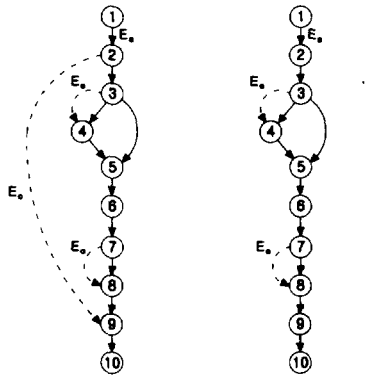


(b) CDDG

(그림 2) VHDL 기술 및 CDDG

2.2 포함 관계를 갖는 부그래프의 제거

제한조건을 나타내는 에지 수를 감소시키는 것은 제한조건에 의해 다른 시간 영역에서 동작하는 노드들이 감소될 수 있다는 것을 의미한다. 따라서 전체 그래프에 대한 제한조건이 없다면 하나의 시간 영역, 즉 하나의 제어 스텝에서 회로가 동작할 수 있다. 부그래프에 대해 포함 관계를 갖는 부그래프를 제거의 예로서 (그림 3)과 같이 부그래프의 에지(2,9)가 다른 부그래프들의 에지(3,4)와 에지(7,8)를 포함하면 에지 (2,9)를 제거한다. 즉, 노드 3과 4가 동시에 같은 제어 스텝에서 동작할 수 없기 때문에 노드 2와 9는 같은 시간 영역에서 동작할 수 없다는 제한조건은 제거해도 된다.



(a) (b)

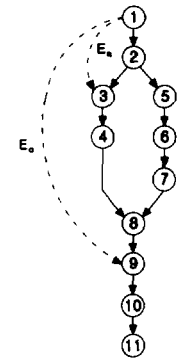
(a) 포함 관계를 갖는 부그래프

(b) (a)의 수정된 표시

(그림 3) 포함 관계를 갖는 부그래프의 수정

(정의 7) 조건 분기에 관계없이 부그래프의 에지 E_c들이 서로 포함 관계에 있으면, 포함하고 있는 에지는 제거한다

포함 관계를 갖는 부그래프의 제거가 가능한 지를 보이기 위해 (그림 4)를 이용하면 부그래프(1,9)의 경우, 노드 1과 9는 같은 제어 스텝에서 동작을 할 수가 없다. 따라서 노드 1이 제어 스텝 C(i)에서 동작하면 노드 9는 제어 스텝 C(i+1)에서 동작한다고 하자. 이때 에지(1,9)가 에지(1,3)를 포함하고 있기 때문에 에지(1,9)를 제거하여 노드 1과 9가 같은 제어 스텝 C(i)에 할당되었다고 가정하자. 노드 9는 제어 스텝 C(i)에 할당되고 노드 3이 제어 스텝 C(i+1)에 할당되기 위해서는 두 노드 사이에 아무 관계가 없거나 노드 9가 노드 3의 선행 노드가 되어야 한다. 그러나 노드 9가 노드 3의 후행 노드가 되므로 노드 9는 노드 3과 같은 제어 스텝 C(i+1)에 할당되거나 C(i+1)이후의 제어 스텝에 할당되어야 한다. 따라서 노드 9가 제어 스텝 C(i)에 할당 될 수 없다.



(그림 4) 조건 분기가 있는 경우의 부그래프 제거

2.3 중첩 관계를 갖는 부그래프의 수정

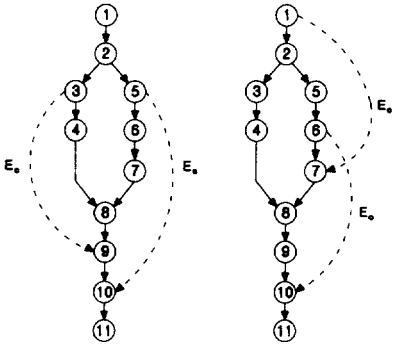
포함 관계를 갖는 부그래프들이 제거된 후, 다수의 부그래프들을 하나의 새로운 부그래프로 대치함으로서 전체적인 동작시간을 최소화하기 위해 부그래프간의 중첩 관계를 검색한다. 부그래프간에 중첩 관계가 있으면 기존의 부그래프들을 제거하고 중첩되는 노드들로 구성된 새로운 부그래프를 생성한다. 그러나 조건 분기가 있는 경우 다양한 형태의 부그래프간의 중첩 관계가 존재하게 된다.

따라서 스케쥴링에 의해 가능한 CDDG의 노드가 빠른 제어 스텝에서 동작하고 전체적인 동작시간의 최소화를 위하여 아래와 같은 우선 순위에 의해 부그래프

의 중첩 관계를 찾아서 부그래프 수를 최소화한다.

1) 부그래프의 시작 노드와 종료 노드가 같은 조건 경로에 있는 경우.
2) 한 부그래프의 종료 노드와 다른 부그래프의 시작 노드가 같은 조건 경로에서 중첩되어 있는 경우.

우선 순위 1)에 대한 예는 (그림 5)(a)와 같다. (그림 5)(a)의 경우 두 에지(3,9)와 에지(5,10)의 종료 노드 9와 10은 같은 조건의 경로에 존재한다. 우선 순위 2)의 예로는 (그림 5)(b)이다. (그림 5)(b)에서 에지(1,7)의 종료 노드(노드 7)와 에지(6,10)의 시작 노드(노드 6)가 같은 조건(거짓 조건)의 경로에 위치하고 노드 6이 부그래프(1,7)에 포함된다.



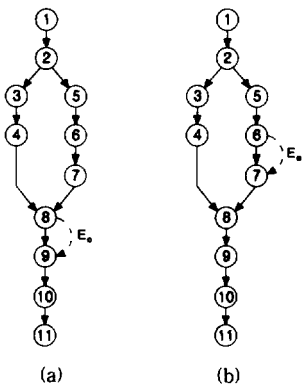
(a) (b)

(그림 5) 조건 분기들이 있는 경우의 중첩된 부그래프들

따라서 (그림 5)의 부그래프들은 (그림 6)과 같이 중첩되는 노드로만 구성된 새로운 부그래프로 대체한다. 즉 (그림 5)와 같이 부그래프의 에지 E_c들이 서로 중첩 관계에 있으면, 중첩되는 부분을 선택하여 새로운 에지 E_c로 생성하고 기존의 에지 E_c들은 제거한다. 그리고 중첩 관계를 갖는 부그래프들이 존재하지 않을 때까지 반복한다.

2.4 자원 공유를 고려한 조건 분기의 변환

동작시간을 최소화하기 위해 포함 관계를 갖는 부그래프들이 제거되고 부그래프간의 중첩 관계를 수정한 후, 중간 표현 그래프 CDDG를 조건 분기 특성에 근거한 자원 공유 개념을 고려하여 변환한다.



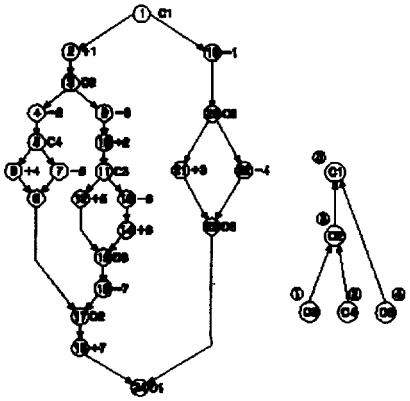
(a) (b)

(그림 6) 그림 5의 수정 결과

변환된 CDDG에서 각 연산들에 대해 하드웨어 자원의 공유를 결정하고 빠른 제어 스텝을 할당하기 위하여 다음과 같은 과정을 거쳐 변환한다.

2.4.1 조건 우선 순위의 결정

CDDG에 포함된 조건 분기가 포함된 경로들의 하드웨어 자원의 공유 가능성을 찾기 위해서 조건 분기를 포함된 경로들을 조건 분기가 포함되지 않은 경로들로 변환한다. 중첩된 조건 분기들을 고려하여 (그림 7)과 같이 조건 그래프를 구성한다. 조건 그래프의 노드는 각 조건 분기들로 구성되며 가장 깊은 부분의 조건 분기들로부터 가장 바깥의 조건 분기들로 상향 방식에 의해 조건 그래프를 구성한다. 조건 그래프의 에지는 그래프의 리프(leaf)들로 시작하여 각 자식 노드(child node)로 부터 부모 노드(parent node)들로 방향성 에지



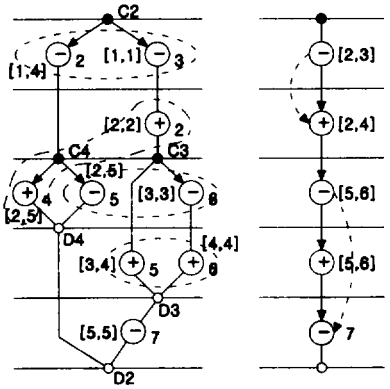
a) CDDG b) 조건 그래프

(그림 7) 조건 그래프의 구성

로 구성된 방향성 그래프(directed graph)이며, 방향은 각 조건 분기의 수행 과정의 진행 방향을 의미한다. (그림 7)(b)와 같이 그래프의 깊이에 따라 조건 우선 순위가 결정되는데, C3 노드와 C4 노드 같이 깊이가 같은 노드는 CDDG의 최대 길이 및 조건 분기들의 중첩의 수를 고려한 의사 제어 스텝(pseudo longest path)에 걸쳐 있는 조건 분기의 노드가 우선 순위가 높도록 하여 C3을 우선 순위 1로 설정하였다. 또한 노드 C2와 C5는 그래프의 깊이가 같으나 리프 노드의 존재에 의해 C2 노드의 우선 순위를 C5 보다 높게 설정하였다.

2.4.2 조건 분기들이 포함된 CDDG의 변환

조건 분기들의 하드웨어 자원의 공유 가능성을 찾기 위해서 조건 분기가 포함된 조건 경로들은 조건 그래프에서 생성된 조건 우선 순위에 따라 (그림 8)과 같이 조건 분기가 포함되지 않는 경로들로 변환한다. 우선 (그림 8)(a)와 같이 각 조건 분기들의 연산 노드들에 대해 시간 간격(time frame)을 설정한다. 시간 간격은 각 연산이 각 의사 제어 스텝에 한 연산자를 할당되도록 정의하였으며, 조건 그래프의 노드들의 우선 순위에 따라 조건 분기들이 포함된 경로들을 조건 분기가 포함되지 않은 경로들로 변환을 반복적으로 진행한다.



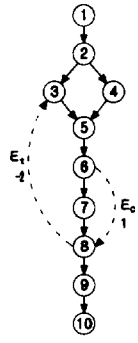
a) 변환 과정 b) 변환된 CDDG

(그림 8) 조건 분기들이 포함된 CDDG의 변환

(그림 8)(a)와 같이 우선 순위가 가장 높은 조건 분기 C3에서는 연산 +5와 +6이 제어 스텝 4에서 시간 간격이 중첩되므로 +5와 +6은 한 기능 연산자로 자원 공유 될 수 있다. 때문에 연산 +5와 +6은 한 연산자 +(5,6)으로 통합하여 조건 분기 C3가 포함된 경로를 조건 분기가 포함되지 않은 경로로 구성할 수 있다. 또한 조건 분기 C2에서도 자원이 공유 될 수 있는 -(2,3), +(2,4), -(5,6)의 통합된 연산을 생성하여 조건 분기가 포함된 경로를 조건 분기가 포함되지 않은 경로인 (그림 8)(b)와 같이 변환 할 수 있다.

2.5 최대 시간 제한조건의 검색

데이터 종속성 부그래프의 에지 E_c의 포함 및 중첩 관계를 갖는 에지들을 제거하고, 조건 분기를 포함된 경로들을 하드웨어 자원 공유를 고려하여 조건 분기가 포함되지 않은 경로들로 변환한 후, 시간 제한조건을 만족하는 지를 검색한다. 최대 제한 시간이라 함은 두 연산 노드의 동작시간의 차가 제한 시간을 초과하지 못한다는 것을 의미한다. (그림 9)와 같이 최대 시간 제한의 에지(6,8)를 고려하면 노드 6과 8사이에 가중치의 합이 음수의 사이클인지를 판단한다. 가중치의 합이 음인 사이클이 존재하면 최대 제한 시간을 만족하므로 스케줄링을 계속 수행한다. 그러나 (그림 10)과 같이 제한 조건을 나타내는 부그래프의 일부가 최대 시간 제한을 검색하는 범위에 중첩되는 경우가 존재한다. 즉 E_c의 종료 노드 9가 최대 시간 제한의 에지(5,9)내에 존재한다. 이 때는 (그림 10)(b)와 같이 에지(3,9)를 에지(5,9)로 대체한 후, 최대 시간 제한을 만족하는 지를 검색한다. 이것은 에지(3,9)에 의해 에지(5,9)내에서 노드들의 동작 시간이 분리되어서 최대 시간 제한을 만족시키지 못할 수도 있기 때문이다.



(그림 9) 최대 시간 제한의 표시

2.6 각 노드들의 제어 시간 결정

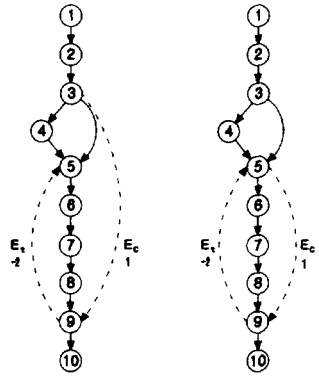
CDDG를 최소의 제한 조건을 나타내는 부그래프를 최소화한 후 각 노드의 동작시간을 결정하여야 한다.

각 노드의 동작 시간은 1로 초기화한다. 동작 시간을 결정하기 위해 BFS(Breath First Search) 방법을 이용하여 CDDG의 근 노드로부터 순차적으로 탐색한다. 이때, 탐색중인 노드들 중 하드웨어 설계 조건을 나타내는 에지 E_c의 종료 노드에 해당되는 노드들과 선행 및 후행 관계를 나타내는 에지 E_s에만 연결되어 있는 노드들은 구분되어 동작 시간이 결정된다. 즉 에지 E_s에만 연결되어 경우 인접된 에지의 가중치를 고려하지 않고 선행노드들이 갖는 동작시간의 최대값이 부여되며, 에지 E_c의 종료 노드에 해당되는 노드들은 선행노드들이 갖는 동작시간 중 가장 큰 값에 가중치의 값을 더한 동작시간이 식(1)과 같이 결정된다.

즉, 노드 N의 동작시간을 T(N)이라 하면

$$T(N) = \text{인접된 에지의 가중치} + MAX(\text{선행 노드들의 동작시간}) \quad (1)$$

으로 표현할 수 있다.



(a) 포함 관계를 갖는 부그래프 (b) (a)의 수정된 표시

(그림 10) 최대 시간에 의한 부그래프 수정

3. 실험 및 고찰

본 논문에서 제안한 스케쥴링 알고리즘을 SPARC 1+에서 C 언어로 구현하였다. 예제 회로는 92년에 발표된 벤치마크 HLSynth92를 사용하였다. <표 1>은 본 논문에서 제안한 스케쥴링 알고리즘을 벤치마크 회로인 AM2910 및 AM2901과 i8251 등의 예제에 적용한 실험 결과이다. 예제 회로로 AM2910 및 AM2901과 i8251 등을 선택한 것은 본 논문에서 지원하고자 하는 control dominated 회로이기 때문이다.

1992년에 발표된 상위 레벨 합성의 벤치마크 HLSynth92 AM2910 및 AM2901은 순서적 기술인 경우 각 1개의 프로세스문으로 구성되어 있고, 벤치마크 HLSynth92 i8251은 3개의 프로세스문으로 구성되어 있다. 또한 VHDL로 기술된 벤치마크 GCD (Greatest Common Divisor), DIFFEQ (DIFFerential EQuation solver), TLC (Traffic Light Controller)는 1개의 프로세스문으로 구성되어 있으며, ARMS_COUNTER(Controlled Counter)는 4개의 프로세스문으로 구성되어 있다. 각 프로세스문에 대한 스케쥴링을 위해 주어진 하드웨어 자원의 제한조건에서 덧셈기와 뺄셈기는 1개의 제어 스텝에서 동작한다고 가정하였다.

본 논문에서 제안된 스케쥴링 알고리듬은 VHDL의 입력으로부터 하드웨어 설계의 특성을 지원하기 위한 조건 분기, 동작 및 시간 제한을 표현 할 수 있도록 구성된 중간 데이터 구조 CDDG를 입력으로 받는다. 스케쥴링 방법은 중첩되는 제한 조건들을 하나의 제한 조건으로 표현함으로서 제한 조건들을 단순화시킨 후 연산의 동작 시간을 결정하고 있다. 그러나 control dominated 회로의 특성에 의해 조건 분기가 많이 나타난다. 따라서 제한 조건들이 중첩되는지를 알기 위해서는 조건 분기에 의해 생성되는 모든 경로를 검색하여야 한다. 이러한 문제점을 해결하고자 조건 분기에 의해 생성되는 모든 경로를 검색하지 않기 위해 제한 조건들을 그래프로 처리함으로서 조건 분기에 의한 경로를 탐색하지 않고 그래프간의 중첩이 있는 지를 찾는다. 따라서 제한 조건을 단순화시키는 과정의 수행 시간이 짧다는 장점을 갖으며 <표 1>에서 보여주는 CPU 처리 시간은 이를 입증하고 있다.

본 논문에서는 회로가 FSM(Finite State Machine)의 형태로 동작한다고 보기 때문에 <표 1>에서 상태는 논리 합성에서 사용하고 있는 상태와 같은 개념이다. <표 1>에서 제어스텝은 회로를 구동하는 상태 중 가장 긴 상태의 수를 나타내며 상태 수는 회로가 동작하기 위한 전체 상태 수를 나타낸다. 본 논문의 예제 회로는 VHDL로 기술되어 있고 [4]의 예제 회로는 ISPS로 기술되어 있기 때문에 각 예제 회로의 제어스텝 및 연산자의 형이 다르게 표시되고 있다. <표 1>에서 스케쥴링 알고리즘은 벤치마크 HLSynth92 예제 회로에 대해 최적의 근사 해를 보이면서 빠른 동작시간을 보이고 있다. 그러나, VHDL 설계 사양의 기술 방법 및 분할하는 방법 등에 따라 스케쥴링 결과가 달라

질 수가 있으므로 <표 1>에서의 [4], [5]의 결과는 본 논문과의 직접적인 비교보다는 본 논문에서 제안한 합성 방법의 효용성을 보이기 위해 제시하였다.

〈표 1〉 실험 결과

	예제 회로 및 설계형		제어 스텝	FU	CPU (sec)
본논문	AM2910	no constraints	3	+(1) 12bits -(1) 12bits	8.9
		constraints : [+,-] ALUs	3	[+,-] 12bits	8.9
	AM2901	no constraints	3	+(1) 5bits	12.7
		constraints : [+,-] ALUs	3	[+,-] 5bits	13.2
	i8251 main	-	5	+(1) 8bit CNT(1) 8bit	0.2
	i8251 Tx	-	9	CNT(1) 8bit CNT(1) 4bit	0.1
	i8251 Rx	-	17	+(1) 8bit CNT(1) 8bit CNT(1) 4bit	1.2
	ARMS_ COUNT ER	-	1	+(1) 4bits -(1) 4bits	0.053
	DIFFEQ	-	4	*(2) integer +(1) integer -(1) integer	0.052
	GCD	-	1	-(1) 8bits	0.049
	TLC	-	1	0	0.061
[5]*	AM2910	no constraints	-	+(1) 12bits +(1) 3bits -(1) 12bits -(1) 3bits	27
		constraints : [+,-] ALUs	-	[+,-] 12bits [+,-] 3bits	26
	AM2901	no constraints	-	+(2) 5bits	279
		constraints : [+,-] ALUs	-	[+,-](2) 5bits	178
[4]*	i8251	-	5		2.0
	i8251 Tx	-	8	-(1)	8.8
	i8251 Rx	-	13	-(1)	2.0
	i8251 HUNT	-	7	-(1)	0.1
	COUNT-ER	-	1	+(1) -(1)	0.1
	DIFFEQ	-	4	*(2) +(1) -(1)	0.1
	GCD	-	2	-(1)	0.1
	TLC	-	6	0	0.1

* IBM RISC System/6000 Model 520 workstation

본 논문의 예와 비교한 [4], [5]는 스케줄링과 할당을 수행한 결과이므로 비교 예제의 AM2910과 AM2901은 제어 스텝을 참고 할 수가 없었다. 또한 i8251에 대한 스케줄링은 <표 1>에서 보인 것과 같이 1개의 덧셈기와 카운터를 연산자로써 사용된다. 본 논문에서는 스케줄링시 제어 스텝 및 연산자 할당에 하드웨어 공유 개념이 적용된 결과이다. 특히 본 논문에서 적용한 i8251 Rx는 [4]의 i8251 Rx와 i8251 HUNT가 하나의 회로로 기술된 것이다. 따라서 두 개의 회로가 하나의 회로로 기술될 때 조건 분기에 의해 발생될 상태들을 고려하면 상대적으로 본 논문의 스케줄링 결과가 우수함을 알 수 있다.

4. 결 론

본 논문에서는 행위 기술 영역에서의 control dominated ASIC 설계를 지원하기 위한 VHDL 번역 방법 및 그를 이용한 최소 제한조건 스케줄링 알고리즘을 제안 및 구현하였다.

본 논문의 VHDL 부분사양 및 분석기는 동시문과 순서문들의 계층적인 구조의 처리가 가능하며, 생성된 중간 표현 그래프 CDDG는 변수의 데이터 흐름 처리, 입·출력 포트의 사용, 자원의 이용 제한조건들을 효과적으로 부그래프로 처리할 수 있다. 또한 조건 분기에 따른 제어 흐름 그래프를 분리하지 않고 조건 분기가 없는 경우와 같이 부그래프로 처리 할 수 있다는 장점을 가지고 있다. 따라서 제한 조건을 단순화시키는 과정의 수행 시간이 짧다는 장점을 갖으며 실험 결과에서의 CPU 처리 시간은 이를 입증하고 있다. 본 논문의 스케줄링 알고리듬을 벤치마크 회로인 AM2910 및 AM2901 수행한 결과 기존의 방법에 비해 최적의 결과를 얻을 수 있었으며 이는 스케줄링시 제어 스텝 및 연산자 할당에 하드웨어 공유 개념이 적용된 결과이다. 또한 본 논문에서 적용한 i8251 Rx는 기존의 i8251 Rx와 i8251 HUNT가 하나의 회로로 기술된 것이다. 따라서 두 개의 회로가 하나의 회로로 기술될 때 조건 분기에 의해 발생될 상태들을 고려하면 상대적으로 본 논문의 스케줄링 결과가 우수함을 알 수 있다. 특히 제안된 알고리즘은 조건 분기뿐만 아니라 최대/최소 시간 제한조건을 처리할 수 있기 때문에 다양한 메모리 및 마이크로 프로세서 등과 같은 제어 회로를 합성하는데 적합하다.

본 논문은 계층적인 다양한 형태의 기술 시스템에 대한 스케쥴링 방법을 제시 하였으나 논리 합성과 연계될 수 있는 VHDL 하드웨어 모델링 방법에 대한 연구가 더욱 필요하다. 또한 앞으로는 저전력 소비 ASIC 설계를 위한 스케쥴링 방법에 대한 연구를 진행하고자 한다.

참 고 문 헌

[1] K. Wakabayashi and H. Tanaka, "Global scheduling independent of control dependencies based on condition vectors," in *Proc. Design Automation Conf.*, pp.112-115, June 1991.

[2] D. D. Gajski, N. Dutt, A. Wu, and S. Lin, *High-Level Synthesis : Introduction to chip and System Design*, Kluwer Academic Pub., Boston, 1992.

[3] R. Camposano, R. A. Bergamaschi, C. E. Haynes, and S. M. Wu, *Trends in High-Level Synthesis*, Kluwer Academic Pub., Norwell. 1991.

[4] R. Camposano and W. Wolf, *High-Level VLSI Synthesis*, Kluwer Academic Pub., pp.55-78, 1990.

[5] R. A. Bwergamaschi, "A system for production use of high-level synthesis," *IEEE Trans. on Computer Aided Design*, Vol.1, No.3, pp.233-243, Sep. 1993.

[6] C. T. Hwang, Y. C. Hsu, and Y. L. Lin, "Optimum and heuristic data path scheduling under resource constrains," *Proc. the 27th Design Automation Conf.*, pp.65-70, 1990.

[7] R. Compasano and R. A. Bergamaschi, "A synthesis using path-based scheduling : algorithm and exercises," *Proc. the 27th Design Automation Conf.*, pp.450-455, 1990.

[8] 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 the 30th Design Automation Conf.*, pp.578-582, 1993.

[9] D. C. Ku and G. De. Micheli, "Relative scheduling under timing constraints," *Proc. the 27th Design Automation Conf.*, pp.59-64, 1990.

[10] VHDL *System Simulation Workshop*, Synopsys Inc., 1995.

[11] G. Lakshminarayana, S. Khouri, and N. K Jha, "Wavesched : A novel scheduling technique for control-flow intensive behavioral descriptions", in *Proc. Int. Conf. Computer-Aided Design*, pp. 244-251, Nov., 1997.

인 치 호

e-mail : ich410@venus.semyung.ac.kr

1985년 한양대학교 전자공학과(학사)

1987년 한양대학교 대학원(공학석사)

1996년 한양대학교 대학원(공학박사)

1992년~현재 세명대학교 컴퓨터과학과 부교수

관심분야 : VLSI CAD, ASIC 설계, CAD 알고리듬 등