CPLD 조건식을 고려한 RTL 바인딩

김재진*°, 윤충모**, 김희석*
청주대학교 전자공학과*, 서일대학 정보통신과**
충북 청주시 상당구 내덕동 36번지*
서울 중랑구 면목8동 49-3**
kimjj@wslab.chong.ac.kr, , khs8391@alpha94.chongju.ac.kr

A RTL binding Technique with CPLD constraint

Jae-Jin Kim*°, Choong-Mo Youn**, Hi-Seok Kim*
Dep. of Electronic Engineering, ChongJu University*
Dep. of Information and Communication, SeoIl College**
kimjj@wslab.chong.ac.kr, khs8391@alpha94.chongju.ac.kr

요약

본 논문은 HLS에서 CPLD 조건식을 고려한 RTL 바인딩 기술로서 HDL로 기술된 회로의 스케쥴링을 한 후 모듈 연산 간격을 고려하여 합당한 모듈을 선택하고 스케쥴링과 할당을 수행한 후 주어진 조건식에 맞도록 CPLD를 선정한다.

또한 할당된 결과의 모듈을 CPLD 내부의 CLB의 크기를 고려하여 부울식을 분할하고 최적의 CLB를 사용하여 회로를 구현할 수 있도록 binding 알고리즘을 제안하였다.

I. 서 론

HLS(High-level synthesis)에서 HDL로 기술된 회로를 ASIC나 FPGA, CPLD등을 사용하여 널리 회로를 구현하고 있다.

FPGA를 이용하여 회로를 구현하는 방법은 주어진 조건식에 따라서 할당(Allocation)과 스케쥴링(Scheduling), 그리고 바인딩(Binding)을 수행한 후 place와 route를 하게된다. 그러나 이러한 방법은 디바이스의 특성을 고려하지 않고 수행되어 사용자가 원하는 회로의 조건식을 만족할 수 없을 경우, 처음부터 다시 구현해야 하는 단점이 있었다. 따라서 이러한 단점을 보완하기 위해 디바이스의 조건식을 정보로 가지고 있으면서 layout을 고려하여 회로를 구현할 수 있는 layout-driven 방법이 제안되었다.

그러나 CPLD의 경우, 회로 구현용 툴들은 많이 제공되고 있으나 HLS에서 CPLD의 조건식을 고려하여 회로를 구현하는 방법은 아직 제안되지 않았다.

따라서 본 논문에서는 HLS에서 CPLD의 조건식을 고려하여 회로를 구현할 수 있는 RTL 바인딩 알고리즘을 제안하였다.

그림 1은 본 논문에서 제안한 CPLD 조건식을 고려한 바인딩의 전체 순서도 이다.

II. CPLD 조건식을 고려한 RTL 바인딩

CPLD를 이용하여 회로를 구현하는 일반적인 CAD 툴들은 HDL에서 기술된 회로 구성을 스케쥴링과 바인딩을 통해 하나의 CPLD에 맵핑하였다. 그러나 이러한 맵핑은 회로의 지연 시간과 디바이스의 내부 지연시간의 비교가 없고, 입출력 핀의 수와 부울식의 조건식을

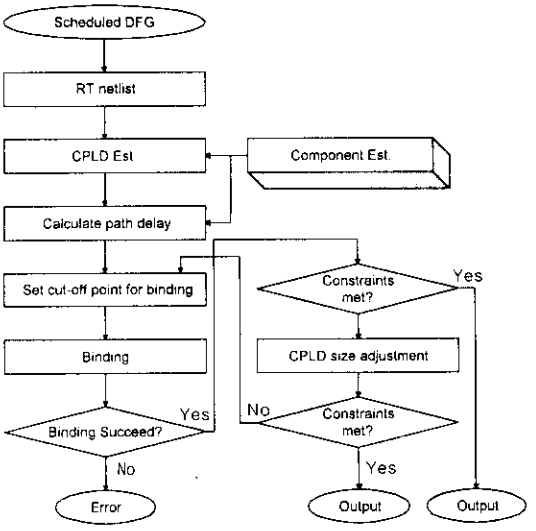

그림 1. CPLD 조건식을 고려한 RTL 바인딩의 전체 순서도

고려하여 수행되질 않았다.

본 논문에서 제안한 CPLD 조건식을 고려한 RTL 바인딩은 HDL로 기술된 회로의 할당과 스케쥴링, 바인딩을 수행한 후 하나의 CPLD로 회로 구현이 가능한가를 측정 후 가능하다면 미리 사용자가 지정한 조건식과 비교하여 일치할 경우 회로를 구현하게 된다.

이때 하나의 CPLD로 회로 구현이 불가능 한 경우는 CPLD의 조건식을 고려하여 스케쥴링과 바인딩을 새로 수행함으로서 두 개 이상의 CPLD로 회로를 구현할 수 있도록 하는 기술이다.

본 논문에서 제안한 CPLD 조건식을 고려한 RTL 바인딩의 예제로 16 비트 FIR 필터를 선정하였다. 그림 2는 16비트 FIR 필터의 입력 파일을 나타내었다.

그림 3은 그림 2의 16 비트 FIR 필터의 입력 파일을 Hyper에 입력하여 얻어낸 스케쥴링 결과이다.

```
#define num16 num<16,10>

#define a0 -0.001953125
#define a1 0.003906250
#define a2 -0.007812500
#define a3 0.01953125
#define a4 -0.06640625
#define a5 0.7500000

func main(In : num16) Out : num16 =
begin
    Acc1=num16(In@10*a0);
    Acc2=num16(In@9*a1)+Acc1;
    Acc3=num16(In@8*a2)+Acc2;
    Acc4=num16(In@7*a3)+Acc3;
    Acc5=num16(In@6*a4)+Acc4;
    Acc6=num16(In@5*a5)+Acc5;
    Acc7=num16(In@4*a4)+Acc6;
    Acc8=num16(In@3*a3)+Acc7;
    Acc9=num16(In@2*a2)+Acc8;
    Acc10=num16(In@1*a1)+Acc9;
    Out=num16(In*a0)+Acc10;
end;
```

그림 2. 16 비트 FIR 필터의 입력 파일

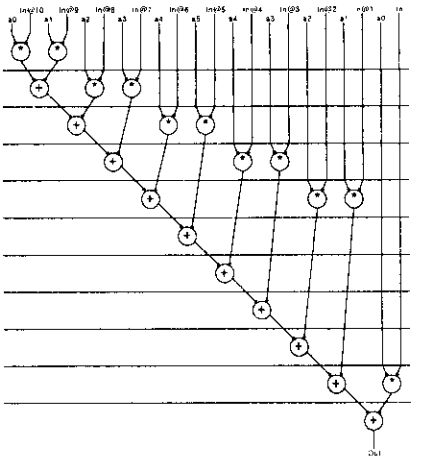

그림 3. 16 비트 FIR 필터의 스케쥴링 결과

스케쥴링 결과 2개의 승산기(Multiplier)와 하나의 가산기(Adder)로 할당되었다.

할당된 결과와 회로의 입출력 개수와 부울식의 조건식을 토대로 CPLD의 조건식과 비교하여 CPLD 내부에 회로가 어떻게 구현되는가에 대한 정보를 갖게 된다.

그림 4는 할당된 결과와 CPLD 내부에서 회로 구현의 예상도이다

그림 4와 같이 2개 이상의 CPLD에 회로를 분할 하기 위해서는 회로의 동작 특성과 CPLD의 조건식과 주어진 조건식을 고려하여 스케쥴링과 바인딩을 수행하여야 한다.

스케쥴링은 회로의 전체 지연 시간과 연산자의 지연 시간, 모듈의 지연 시간을 고려하여 가장 시간적인 요

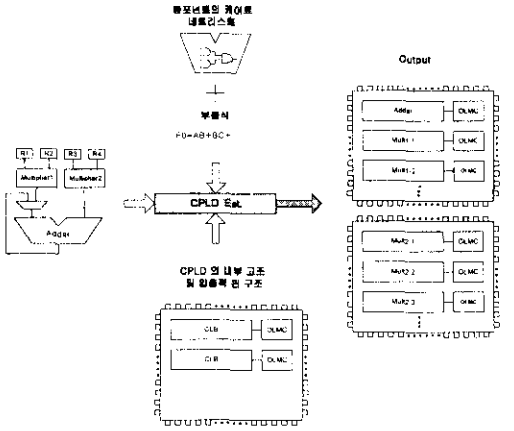

그림 4. 할당된 결과와 CPLD 내부에서 회로 구현의 예상도

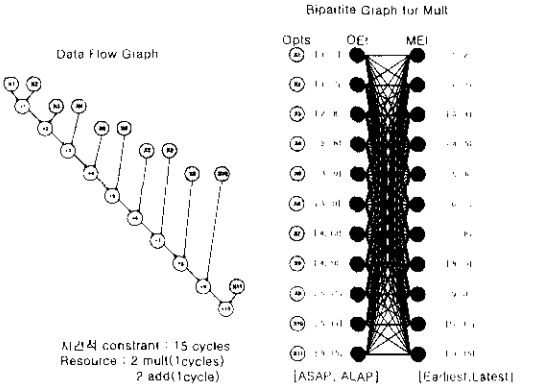

그림 5. 모듈을 찾는 방법

소가 일치되는 모듈을 찾게 된다.

전체 지연 시간의 계산은 식 1에의해 계산된다.

$$t_d = t_{pr} + t_{sr} + t_{im} + t_{fu} + t_{om} + t_{wrm} + t_{wmf} \quad (식\ 1)$$

t_{pr} : *Propagation delay*
t_{sr} : 래지스터의 *setup time*
t_{im} : 입력 *MUX*의 지연시간
t_{fu} : *FU*의 지연시간
t_{om} : 출력 *MUX*의 지연시간
t_{wrm} : 레지스터에서 입력 *MUX*까지의 지연시간
t_{wmf} : 입력 *MUX*에서 *FU*까지의 지연시간

모듈을 찾는 방법은 그래프 형식의 연산자 수행 간격(OEI : Operation Execution Interval)와 모듈 수행 간격(MEI : Module Execution Interval)를 비교하여 적당한 수행 시간을 갖는 모듈을 찾게 된다.

그림 5는 OEI와 MEI를 비교하여 적당한 모듈을 찾는 방법을 나타낸 그림이다. 그림 5에서 굵은 선으로 표현된 MEI의 모듈을 선택하여 사용한다.

식 1을 이용하여 전체의 지연시간을 계산하고 그림 5와 같은 방법으로 사용할 모듈이 결정되면, 주어진 조건식의 시간과 입출력에 맞게끔 분할을 해야 한다.

분할은 CPLD 내부의 CLB는 3개에서 10개 정도의 OR 텀을 가지고 있으므로 하나의 출력을 형성하기위해 2개 이상의 CLB를 사용하여야 하므로 소자의 전체 지연시간이 조건식에서 주어진 시간의 1/2이 되는 소자를 선정한다.

$$t_d = \frac{1}{2} t_{con}$$

t_{con} : *constraint*에서 지정한 시간

소자가 선정되면 선택된 모듈의 부울식을 추출하여 선택된 소자의 CLB 크기에 맞도록 부울식을 분할한다.

분할된 부울식을 선택된 소자에 바인딩(binding)하는 방법은 그림 6과 같은 알고리즘에 의해 수행된다.

```
Binding(S_fu, Ivar, Ovar, N, M, Result_allocation)
  // S_fu : 선택된 FU
  // Ivar : 입력 변수
  // Ovar : 출력 변수
  // N : S_fu의 수
  // M : T_CPLD의 CLB 수

  T_CPLD; // 선정된 CPLD
  CLB; // 선정된 CPLD의 CLB 수
  GBE ← Generate_BE(S_fu);
      // 선정된 모듈의 부울식 생성
begin
  for(j=1 to M)
    if(GBE < CLB_size)
      // CLB_size : CLB의 OR텀수
        VirtualBind(S_fuj, CLBj);
    else if(GBE > CLB_size)
      PGBE ← PARTITION(GBE, CLB_size);
        // PGBE : CLB의 크기에 맞게 분할하여
        //          생성된 부울식
      VirtualBinding(S_fui CLBi);
```

```
        if( i+1 ≤ N)
          success = Binding(S_fu, N, M);
          if(success) then
              return(True);
          else
              UnBinding(S_fu_i, CLB_i);
          end if;
        else
           ActualBinding;
           return(True);
        end if;
      end if;
    end for;
    return(False);
end
```

그림 6. 바인딩 알고리즘

III. 실험 결과

본 논문에서 제안한 CPLD 조건식을 고려한 RTL 바인딩 기술의 예로 16비트 FIR를 선정하였다.

스케쥴링 결과와 할당(allocation) 결과는 앞에서 기술하였다

16 비트 FIR 필터는 2개의 Multiplier와 2개의 가산기로 구성되어 있다.

Multiplier와 가산기를 CPLD로 구현하기 위해 ALTERA사에서 제공되는 CPLD를 사용하였으며 조건식은 제한 시간을 100ns로 지정하였고, 입출력 핀의 수는 250 개로 선정하였다.

그 결과 하나의 multiplier를 구현하기 위한 소자로 FLEX10K30이 선정되었으며, 하나의 가산기를 구현하기 위한 소자로 EPX8160이 선정되었다.

16 비트 multiplier는 입력 32개 출력 33개로 입출력 핀은 65개를 사용하는데 32번째 연산 결과 출력인 CAL31의 경우 OR 팀의 수가 91개로 출력 핀에 연결된 CLB 외에 89개의 CLB를 더 사용하여야 하며 총 1308개의 CLB를 더 사용하여야 한다.

FLEX10K30은 1728개의 CLB로 구성되어 있어 multiplier 구현에 적합하다.

또한 가산기의 경우 입출력 핀은 108핀을 사용하며 32개의 다른 CLB를 사용하여야 한다.

EPX8160은 160개의 CLB로 구성되어 있어 가산기를 구현할 수 있다.

실험 결과는 표 1에 나타내었다.

표 1. 실험 결과

예제명	const-raint	모듈명	알고리즘 미적용		알고리즘 적용	
			선정된 CPLD (총CLB수)	사용된 CLB수	선정된 CPLD	사용된 CLB수
16 비트 FIR 필터	150 ns 250 핀	Mutip-lier	FLEX10K40 (2304)	1832	FLEX10K30 (1728)	1341
		가산기	FLEX10K10 (576)	146	EPX8160 (160)	99

IV. 결론

본 논문은 HLS에서 CPLD 조건식을 고려한 RTL 바인딩 알고리즘을 제안하였다.

제안한 알고리즘은 HDL로 기술된 회로의 스케쥴링을 한 후 모듈 연산 간격을 고려한 모듈을 선택하고, 스케쥴링과 할당을 수행한 후 주어진 조건식에 맞는 CPLD를 선정한다.

또한 할당된 모듈을 CPLD 내부의 CLB의 조건식을 고려하여 부울식을 분할하고 최적의 CLB를 사용하여 회로를 구현하였다.

제안한 알고리즘에 16 비트 FIR 필터를 예제로 선정하여 ALTERA사에서 제공되는 CPLD를 이용하여 회로를 구현한 결과 알고리즘을 적용하기 전보다 작은 CPLD로 회로 구현이 가능하였으며, 회로 구현에 필요한 CLB의 수가 적었고, 가산기의 경우 알고리즘을 적용하지 않았을 때 내부 사용율은 8.45%이었으나 알고리즘 적용한 결과 61.88%로 내부 사용율이 증가되었다.

<참고 문헌>

[1] C. Ramachandran, F. J. Kurdahi, D. Gajski, V. Chaiyakul, A. Wu, "Accurate Layout Area and Delay Modeling for System Level Design" Proc. ICCAD'92, NOV. 1992

[2] M. Xu, F. J. Kurdahi "Chip Level Area and Timing Estimation for Lookup Table Based FPGAs" Tech. Report #95-31, UCI, Aug.1995

[3] N. Dutt, C. Ramachandran, "Benchmarks for the 1992 High Level Synthesis Workshop", UCI, 1992

[4] Ashutosh Mujumdar, Minjoong Rim, Rajiv Jain, Renato De Leone, "BITNET : An Algorithm for solving The Binding Problem" 7th International Conference on VLSI Design, pp. 163-168, 1994