디지털 시스템 설계를 위한 분할 알고리즘의 분석

최정필, 한강룡, *황인재, 송기용
충북대학교 컴퓨터공학과, *컴퓨터교육과

An Analysis of the Partition Algorithm for Digital System Design

Jeong-Pil Choi, Gang-Ryong Han, *In-Jae Hwang, Gi-Yong Song
Dept. of Computer {Engineering, *Education}, Chungbuk National University
{welcomerain, hankang}@archi.chungbuk.ac.kr, {*ihwang, gysong}@chungbuk.ac.kr

요 약

High-level synthesis는 주어진 동작과 면적, 성능, 전력 소모, 패키징, 테스팅 등의 주어진 제한을 만족하게 구현된 구조적 디자인을 생성한다. 즉 high-level syntehesis란 디지털 시스템의 알고리즘 레벨 서술로부터 레지스터 전달구조의 구현에 이르는 과정을 의미한다. 이러한 high-level synthesis의 과정은 컴파일, 분할, 스케줄링 등의 단계를 거쳐 디지털 시스템을 설계할 수 있다.

본 논문에서는 high-level synthesis의 단계중 분할 과정을 연구하고, 분할 알고리즘 중에서 min-cut 알고리즘과 simulated annealing 알고리즘을 사용하여 비교 분석하였다

Abstract

High-level synthesis generates a structural design that implements the given behavior and satisfies design constraints for area, performance, power consumption, packaging, testing and other criteria. Thus, high-level synthesis generates that register-transfer(RT) level structure from algorithm level description. High-level syntehsis consist of compiling, partitioning, scheduling.

This paper we study the partitioning process, and analysis the min-cut algorithm and simulated annealing algorithm.

I. 서론

VLSI 기술은 단일칩에 수백만개 이상의 트랜지스터를 집적하고 있는 가운데 computer-aided design(CAD) 연구는 설계 과정상 회로 시뮬레이션, 배치, floorplanning, 라우팅 등 하위 레벨에서의 여러 문제들을 성공적으로 해결해 오고 있다.

설계 복잡도가 증가하고 판매시장으로 전달되는 시간이 짧아지면서 CAD 연구는 고 레벨로 그 초점이 옮겨지고 있는 바, 게이트 또는 트랜지스터가 아닌 기억소자, 레지스터 ALU, 버스 등이 구성단위가 되는 고 레벨은 다루어야 할 설계 객체가 줄어드는 반면 설계 범위는 확대된다. 즉 다양한 설계가 존재하게 되며 최적구현을 위한 설계 범위의 빠른 탐색이 필요하게 된다.

Design synthesis는 고 레벨에서의 기능을 목적 레벨에서의 구조로 변환하는 과정으로 변환된 구조는 면적, 전파지연 등의 제한조건을 만족해야 하며 CAD 연구는 회로, 논리 등의 하위 레벨에서 고 레벨로 그 연구중심이 이동하고 있다 하겠다.

High-level synthesis(HLS)는 디지털 시스템의 동작(behavior)을 레지스터 전달 레벨 구현으로 바꾸어주는 변환으로, 동작은 입력 순서와 출력 순서 사이의 매핑을 명시하며 레지스터 전달 레벨 구조는 데이터패스와 유한 상태 제어기로 구성된다. 데이터패스는 통상 레지스터 컴포넌트 라이브러리로부터 선택된 소자들로 구성된다.

본 논문에서는 high-level synthesis의 단계중 분할 과정을 연구하고, 분할 알고리즘 중에서 min-cut 알고리즘과 simulated annealing 알고리즘을 사용하여 비교 분석해 보기로 한다.

II. High-level Synthesis

High-level synthesis는 주어진 동작과 면적, 성능, 전력 소모, 패키징, 테스팅 등의 주어진 제한을 만족하

게 구현된 구조적 디자인을 생성한다. 즉 high-level syntehesis란 디지털 시스템의 알고리즘 레벨 서술로부터 레지스터 전달구조의 구현에 이르는 과정을 의미한다. 동작은 입력에서 출력으로의 매핑을 의미하며 구조는 netlist등으로 상호연결된 소자들을 의미한다. 주어진 동작은 보통 여러 가지 다른 구조로 구현될 수 있다.

합성(Synthesis)은 설계 대상의 서술방식에 따라 여러 레벨에서 수행할 수 있으며 설계 계층구조(hierarchy)는 시스템, 알고리즘, 레지스터전달(RT), 논리, 회로의 레벨을 규정한다.

레지스터전달 레벨 구조는 레지스터, 기능 장치(functional unit), 멀티플렉서, 버스 등으로 구성되는 데이터패스와 데이터패스에서의 데이터 처리 및 전달을 제어하는 제어부 하드웨어를 포함한다. 제어부가 데이터패스에 결합되어 있지 않을 경우 - 일반적으로 데이터패스와 제어부는 구분구성 됨 - 합성은 제어부에 요구되는 기능을 수행하는 유한 상태 머신(FSM) 등을 구성한다.

설계 계층구조에서 high-level synthesis는 논리 수식으로 표현된 시스템에 최적화를 적용한 후 주어진 기술(technology)에 매핑하는 논리 합성(logic synthesis), 알고리즘을 병렬 또는 파이프라인 방식처리 다중프로세스로 분할(partition)하는 구조적 합성(architectural synthesis)과 대조된다. 논리 합성은 high-level synthesis의 결과를 가정하므로 설계 과정에서 high-level synthesis 수행 후에 적용된다.

1. 컴파일링

High-level synthesis의 입력은 hardware-description language(HDL)로 작성된 동작이며 high-level synthesis의 첫 단계는 HDL 서술로부터 중간 표현으로의 컴파일이다.

중간표현의 두 가지 형태는 parse tree와 그래프이며 데이터-흐름과 제어-흐름을 나타내는 그래프가 많이 이용된다. 제어 그래프는 HDL의 순서명시로부터 유도되며 데이터 그래프는 연산관계를 나타낸다. 데이터-흐름 그래프(DFG)는 연산에 의해 생산되고 다른 연산에 의해 소비되는 값을 호(arc)로 나타내므로 내부변수 사용방식에서의 의존성 제거에도 이용된다. 이러한 변수 재배정 기능은 연산순서조정이나 데이터패스 간략화 등에 중요하다.

중간표현은 어느 정도의 최적화를 필요로 하며 이에는 죽은 코드 제거, 상수 전파, 프로시저의 inline 확장, 루프 unrolling 등이 포함된다.

2. 분할

분할은 기능의 최적화를 위한 대상의 그루핑으로 layout에서 면적과 전파지연 최소화를 위해 함께 위치할 소자를 결정하거나 패키징 제한 만족을 위해 대형 설계에서 나타날 수 있는 여러 칩으로의 분리 등에서 그 예를 볼 수 있다.

High-level synthesis에서는 스케줄링, 할당, unit 선택 등에 적용되며 저장소자, 기능 장치, 연결망으로 매핑될 변수나 연산의 그루핑을 수행한다.

분할은 대규모 동작을 다수의 소규모 동작으로 분해하여 상호연결 프로세스로 변환하는 경우에도 적용된다.

분할은 그래프 모델에서 수행되며 한 개 또는 몇 개의 씨앗(seed) 노드에서 시작하여 한번에 한 개의 노드를 추가하면서 그래프를 분할하는 구조적(constructive) 방식과 임의의 초기 분할에서 시작하여 노드를 이동시키면서 계속적으로 개선해나가는 반복-개선방식이 있다.

구조적 방식에는 랜덤 선택, 클러스터 성장, 계층적 클러스터링이 있고, 반복-개선방식에는 min-cut 분할과 simulated annealing이 있다.

3. 스케줄링

동작은 구현된 회로에서 수행되는 내용을 나타내며 제어/데이터 흐름 그래프(CDFG) 등의 중간 표현을 거쳐 구현된다. 스케줄링은 CDFG를 하나의 제어스텝에서 수행되는 부그래프로 구분하는 기능이며 각각의 제어스텝은 제어부 유한 상태 머신의 한 상태에 대응한다. 하나의 제어스텝 내에서 할당된 연산수행을 위해 별도의 기능 장치가 필요하며 여러 연산이 한 제어스텝에 스케줄 되면 그 만큼의 기능 장치가 필요하게 되고, 각 제어스텝에 스케줄된 연산을 적게 하면 보다 많은 제어스텝이 필요하게 된다.

즉, 스케줄링의 목적은 주어진 제한조건 내에서 수행에 필요한 소요시간 또는 제어스텝 수를 최소화하도록 각 연산을 시간 스텝에 할당하는 것이다.

4. 데이터패스 할당

데이터패스 할당 또는 데이터패스 합성은 실리콘 면적과 임계 경로 지연의 제한 조건 내에서 소요 자원의 형태와 개수를 결정하는 장치 선택(unit selection)과 연산을 가산기, 쉬프터, ALU 등의 기능 장치에 매핑하고 값을 레지스터, 레지스터 파일, RAM, ROM 등 저장장치(storage unit)에 할당하며 버스, 멀티플렉서 등으로 이 장치를 연결하는 장치 바인딩(unit binding)으로 구

성된다.

III. 분할 알고리즘

1. Min-cut Algorithm

Min-Cut 알고리즘은 Kernighan-Lin 알고리즘이라고도 하며, 최대로 개선된 분할 결과를 얻기 위해 같은 크기의 두 개의 분할 사이의 두 개의 노드를 교환하며 분할을 개선해 나가는 알고리즘이다. Min-cut 알고리즘은 또한 일반적인 n-way partition 문제를 위한 기반으로 사용할 수 있다. 이 알고리즘은 짧은 CPU 시간에서 좋은 결과를 나타내기 때문에 많은 응용분야에서 넓게 사용되고 있다. Min-cut 알고리즘은 n개의 노드의 경우 $O(n^2 \log n)$의 복잡도를 가진다. 아래에 알고리즘을 보인다.

```
G1 = CLUSTERGROWTH(G, n)
G2 = G - G1
bestGAIN = ∞
while bestGAIN > 0 do
  bestGAIN = 0
  for i=1 to 2n do
    Locked(i) = 0
  endfor
  for k=1 to n do
    gain(k) = 0  GAIN(k) = 0;
    for all v_i ∈G1 AND Locked(i)=0 do
      for all v_j∈G2 AND Locked(j)=0 do
        if D_i + D_j -2c_ij> gain(k) then
          gain(k) = D_i + D_j - 2c_ij
          max1(k) = i
          max2(k) = j
        endif
      endfor
    endfor
    Locked(max1(k)) = Locked(max2(k))=1

    for i=1 to 2n do
      if v_i∈G1 AND Locked(i)=0 then
        D_i = D_i - c_i,max2(k) + c_i,max2(k)
      endif
      if v_i∈G2 AND Locked(i)=0 then
        D_i = D_i - c_i,max1(k) + c_i,max2(k)
      endif
    endfor
    GAIN(k) = GAIN(k-1) + gain(k)
    if GAIN(k) > bestGAIN then
      bestk = k
      bestGAIN= GAIN(k)
    endif
  endfor
  for k=1 to bestk do
  (G1, G2) = EXCHANGE(V1, V2, v_max1(k), v_max2(k))
  endfor
endwhile
```

2. Simulated Annealing Algorithm

Simulated annealing 알고리즘은 물질이 녹았을 경우 최소 에너지 상태는 서서히 온도를 낮추면서 관찰되는 물리학에서의 annealing process와 combinatorial optimization의 상사성에 기인한다. 이 알고리즘은 임의의 초기 분할에서 출발하여 임의의 이동으로 진행한다. 각 이동시 새로운 cost가 계산되어 온도 T에 기반 한 수용기준에 만족하는지가 조사된다. 이동의 수용여부는 현재의 simulated 온도에 비례하며 이 온도가 내려가면 이동의 수용확률도 내려가게 된다.

위와 같은 개념에 의거한 simulated annealing 분할 알고리즘은 지역적인 최적의 해답이 아닌 전체적인 최적의 해답을 구해낼 수 있는 방법을 제시하는 것이다. 아래에 알고리즘을 보인다.

```
T = initial_temperature
S_old = initial_partition
c_old = cost(S_old)
while "stopping criterion" is not satisfied do
  while "inner loop criterion" is not satisfied do
    S_new = GENERATE(S_old)
    c_new = cost(S_new)
    Δc = c_new - c_old
    x = F(Δc, T)
    r = RANDOM(0, 1)
    if r < x then
      S_old = S_new
      c_old = c_new
    endif
  endwhile
  T = UPDATE(T)
endwhile
```

Ⅳ. 분할 알고리즘의 분석

두 알고리즘을 비교 분석하기 위해서 20개의 노드를 가지는 그래프를 임의로 생성하고, min-cut 알고리즘의 경우에는 초기 분할 상태를 랜덤하게 생성하여 1000회를 반복하며 최적의 분할을 생성하는지를 검토하였고, simulated annealing에서는 임의의 그래프를 동일하게 적용하고, 초기 상태가 알고리즘의 전체적인 진행에 큰 영향을 주지 않으므로 초기 분할 상태를 하나를 선정하여 1000회의 반복을 통하여 전역 최적 분할을 생성하는지 검토하였다. 그에 대한 결과를 <표 1>에 보였다.

<표 1> 두 알고리즘의 최적해 생성 비율

알고리즘 / 그래프 생성횟수	min-cut	simulated annealing
1	92.1%	99.9%
2	0%	71.5%
3	5%	100%
. . .	. . .	. . .
99	71.5%	95.5%
100	0.1%	54.3%
통 계	33.74%	84.24%

위의 결과에서 보이는 바와 같이 min-cut에 비해 simulated anneal 알고리즘이 최적해를 생성하는 비율이 높게 나타남을 알 수 있다. 특이하게도 임의의 그래프에서 min-cut 알고리즘은 전혀 최적해를 생성하지 못하였다. 이는 min-cut의 알고리즘의 특성상 짧은 계산 시간을 가지기 때문에 지역 최적해에 도달하기에 쉽다는 것을 보여준다.

Ⅴ. 결론

두 가지 알고리즘에서 최적의 해를 생성하는 방법은 다르지만, 최적의 분할을 하는 것이 목적이다. 통계로 나타나는 결과를 보면 simulated annealing이 높은 최적해 생성을 보였지만, 항상 정확한 최적해를 생성하지는 못한다. 따라서 어느 한쪽의 알고리즘만을 사용하는 것보다는 두 알고리즘을 여러번 반복하여 공통되는 해를 구한다면 최적해에 가깝게 접근 할 수 있다.

참고문헌

[1] Daniel D.Gajski, Nikil D.Dutt, *High-Level Synthesis Introduction to Chip and System Design*, Kluwer Academic Publishers

[2] K.H. Kernighan and S. Lin, "An Efficient Heuristic Procedure for Partitioning Graph", Bell System Technical Journal, vol. 49, no. 2, pp. 291-307, February, 1970.

[3] S. kirkpatrick, C.D. Gelatt and M.P. Vecchi. "Optimization by Simulated Annealing", Science, vol. 220, no. 4598, pp.671-680, 1983.

[3] C.M Fiduccia and R.M. Mattheyses, "A Linear-Time Heuristic for Improving Network Partitions", Proceedings of the 19th Design Automatic Conference, pp. 175-181, 1982

[5] A.E. Dunlop and B.W. Kernighan, "A Procedure for Placement of Standard-Cell VLSI Circuits", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. CAD-4, no. 1, pp. 92-98, January, 1985.