ICSC(InCheon Silicon Compiler)를 위한 상태 합성알고리즘에 대한 연구

조 중 휘
인천 대학 전자 공학과

A Study on State Synthesis Algorithm for ICSC(InCheon Silicon Compiler)

Joong Hwee Cho
Dept. of Elec. Eng., Incheon Univ.

Abstract

This paper describes BSDL(Behavioral/Structural Description Language), CDTF(Control Data Text File) and state synthesizer built for use in ICSC(InCheon Silicon Compiler).
BSDL describes structral and behaviral specifications of an ASIC(Application Specific IC) for digital system design. ICSC's paser generates CDTF consists of if-then-else ,arithmetic and data transfer statement according to each BSDL statement. State synthesizer generates CCG(Control Constraint Graph) in considerationof execution of statement and generates VCG (Variable Constraint Graph) in consideration use of variable generation and use of variable. Also, it involves allocating algorithm operation nodes in the data path and the control path to machine states with minimum state number and as small area as possible.

1. 서 론

Digital system의 응용분야가 넓어짐에 따라 ASIC (Application Specific IC)에 대한 빠른 설계가 요구되고 있으며 이를 위해 silicon compiler에 대한 연구가 많이 발표되고 있다.[1]
Silicon compiler는 입력하는 언어의 특성에 따라 구조적 silicon compiler와 동작적 silicon compiler로 구분된다. 구조적 silicon compiler는 사용하는 funtion unit, register 및 연결을 위한 하드웨어 구성요소가 모두 지정되어 이들을 계층적으로 기술한 언어를 입력하며 동작적 silicon compiler는 입.출력 변수들의 동작관계만을 알고리즘적으로 기술한후 [3]에 제시된것 과 같은 여러가지 부분적인 수행에 따라 하드웨어 구성 요소를 합성하는 system을 의미한다. 한편, [2]에 따르면 이상적인 silicon compiler는 위의 두가지 형태의 silicon compiler를 결합한 것인데 최근 설계 요구 조건을 알고리즘 레벨에서 기술한후 이를 register-transfer 레벨의 표현으로 변형하여 이를 구조적 silicon compiler에 입력하기 위한 연구가 data-path generator 형태로 많은 연구가 발표되고 있다.[3-8]

그런데, [3-6]은 target architecture가 centralized processing units이므로 각 module을 aynchonous operation으로 작동할 수가 없으며 control unit의 규모가 증대하게 되는 문제가 있다. 이를 위해 [7-9]에서는 각 module 별로 control unit를 갖는 decentralized system을 구성하기 위한 방법을 제시하고 있다. 이와같은 target architecture에서는 데이타 전송부와 제어부를 함께 고려하여 system의 state를 결정해야 하는데 [7-8]에서는 이에 대한 구체적인 언급이 없으며 [9]에서는 'precedence graph'를 사용하여 system의 state를 합성하는 방법을 제시하고 있다. 이와같은 graph에서는 if, then, else와 같은 reserved word와 연산자 모두를 graph의 vertex로 설정하고 이들 사이에 연산의 흐름을 표시하는 edge를 부여한후 search함으로 state 결정에 많은 시간이 요구된다.

본 논문에서는 이들을 고려하여 decentralized architecture 구조를 갖는 system을 설계하는 ICSC(InCheon Silicon Compiler)를 제안한다.
먼저, 설계하고자 하는 System의 동작 특성은 알고리즘 레벨에서 분석하여 Pascal과 유사한 구조로 기술하고, 구조적 특성은 SDL[10]과 NDL[10]로 함께 기술할수 있는 BSDL(Behavioral/Structural Description Language)을 입력한다. 다음에 BSDL을 VT[4], DDG[11], DAG[12], CDFG[13], DFG[2] 등과 같으나 if-then-else문, 산술문 및 데이타 전송문만을 사용한 CDTF(Control Data Text File)로 변형하는 parser를 제안한다. 변형된 CDFG에 대해 데이타 전송부및 제어부의 연산자 에 대한 상태합성을 함께 행하기 위해 CCG(Control Constraint Graph)와 VCG(Variable Constraint Graph)를 구성하고 ASAP 및 ALAP 개념을 함께 고려하여 최소 상태수에 따라 가능한 한 적은 하드웨어 요소 상태합성을 행하는 새로운 알고리즘을 제안한다. 또한 제안한 알고리즘의 효용성을 보이기 위해

C 언어로 프로그램하여 실제의 예에 적용 실험하여 효용성을 보인다.

2. ICSC (InCheon Silicon Compiler)

본 논문과 함께 본인이 구성하려는 digital system 설계를 위한 Silicon Compiler의 논리 설계 system은 그림 1과 같다. 이 system에의 입력 data는 설계하고자 하는 digital system의 동작 및 구조적 특성을 함께 기술할수 있는 BSDL(Behavioral/Structural Description Language)이며 최종 출력은 1)데이타 전송부/제어부에 대한 random logic 논리회로도 또는 2)데이타 전송부는 random logic 논리회로도로, 제어부는 PLA 회로도이다. 본 논문에서는 그림 1의 paser 와 상태 합성을 위한 알고리즘을 제안 한다.

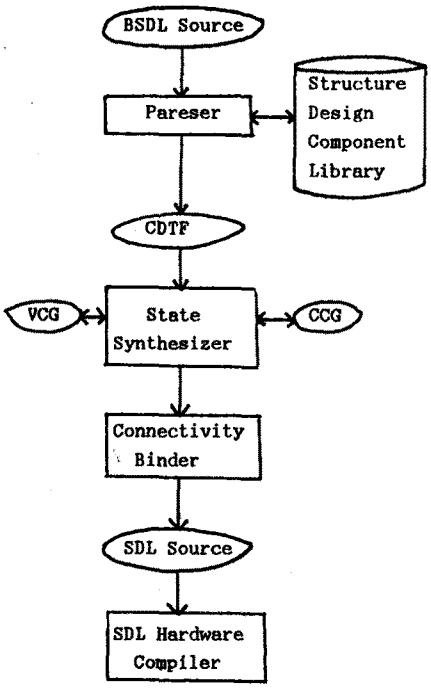


그림 1. ICSC의 논리설계 흐름도

2.1 BSDL

ICSC의 입력언어인 BSDL은 설계하고자 하는 digital system의 동작및 구조적 특성을 함께 기술할 수 있도록 구문이 설정되었다. 구조적 특성으로 기술되는 부분은 이미 설계가 완료되어 Structure Component Digital Library에 Boolean Equation 또는 State Transition Equation으로 저장되어 있는 중간 data를 사용할수 있도록 Component 명과 입/출력 단자명을 기술하거나 새로이 설계되도록 구조적 특성이 주어지면 입/출력 변수와 중간변수를 사용한 Boolean Equation, NDL[10], 또는 SDL [10]을 이용한 Algorithmic State Machine 의 기술이다.

이와같은 구조적 기술부는 ICSC에서 SDL Language Source가 생성될때까지는 그대로 유지된다. 한편, digital system의 동작특성은 알고리즘 레벨에서 분석되어 Pascal과 유사한 구문규칙을 갖는 하드웨어 기술언어로 기술되는데 본 논문에서 제안하는 Parser와 State synthesizer는 이와같은 동작특성 기술부 만을 대상으로 한다. [14]의 Euclidian slow division algorithm에 대한 BSDL의 동작 기술부는 그림 2와같다.

```
procedure_slow-div(A,B,R,Q)
input A,B:integer;
output R,Q:integer;
variable C:integer;
begin
  C:=B;
  while (C<=A) do
    begin C:=C*2; end;
  R:=A; Q:=0;
  while (C>B) do
    begin
      Q:=Q*2; C:=C%2;
      if (R>=C) then
        begin
          Q:=Q+1; R:=R-C;
        end;
    end;
end_slow-div
```

그림 2. BSDL의 동작 기술부의 예

2.2 CDTF의 생성 - ICSC의 Parser -

ICSC는 [3-6]과는 달리 데이타전송부와 제어부 모두를 고려 하여 상태합성을 행하므로 이를 위해 반복문은 if-then-else 문으로,산술문과 데이타 전송분은 그대로, reserved word는 모두제거 하는 대응 관계에 따라 BSDL로부터 CDTF를 생성한다. 그림 2의 BSDL에 대한 CDTF는 그림 3과 같다.

```
1. C:=B
2. (C<=A)/(3,5)
3. C:=C*2
4. ->2
5. R:=A
6. Q:=0
7. (C>B)/(8,14)
8. Q:=Q*2
9. C:=C%2
10. (R>=C)/(11,13)
11. Q:=Q+1
12. R:=R-C
13. ->7
14. END
```

그림 3. 그림2에 대한 CDTF

그림3에서 각 구문의 선두에 있는 숫자는 각 구문에 대한 Label이며 4번문과 같은 조건/판단문은 괄호안의 조건식이 참이면 3번문으로, 거짓이면

5번분으로 천이함을 의미하는 조건적 분기문이며 4번분은 2번분으로 무조건 천이함을 의미한다.

2.3 State Synthesizer

종래의 [3-6]과 같은 Silicon Compiler의 State Synthesizer는 모두 data-path generater의 일부로서 데이타 전송부만을 고려하였다. 따라서, 제어부의 state 결정과 데이타 전송부와 제어부 사이의 연결은 각각 설계를 완료한후 행하여야 하였다. 최근 [9]에서는 데이타 전송부와 제어부의 설계 요구 조건을 precedence graph 개념을 이용하여 state synthesis를 구성하였으나 하드웨어 기술언어의 reserved word와 연산식 모두를 vertex로 설정하고 이들 사이에 연산 수행 순서에 따라 edge를 설정하였으나 system이 커지는 경우 vertex 수와 edge수가 증가하여 vertex search 시간이 증가한다. 따라서, ICSC에서는 CDTF로부터 다음과 같이 CCG와 VCG를 구성한후 이들을 이용하여 상태합성을 행하는 방법을 제안한다.

2.3.1 CCG 의 구성

CDTF의 임의의 2개 구문에 대하여 구문의 수행 순서에 따라 CCG를 구성한다.

정의) CCG = (V,E)

(1) CCG의 vertex 집합인 V는 CDTF의 각 구문 label이다.

(2) CDTF의 i-구문이 수행된 직후 j-구문이 수행되면 CCG에 방향성 edge (Vi,Vj) (단, (Vi,Vj)∈V)를 설정하는데 이와같이 결정된 edge의 집합을 CCG의 E로 한다.

이와같이 설정된 CCG에서 (Vi,Vi+1)∈E이고 (Vj,Vi)∈E (단,i<j)이면 Vi는 CCG에 속하는 feed back path의 시작 구문이므로 임의 State의 시작 구문이어야 한다. 그림3의 CDTF에 대한 CCG는 그림4.(a)와 같으며 구문 2와 7은 위의성질을 만족하므로 임의의 State 시작 구문이어야 한다.

2.3.2 VCG 의 구성

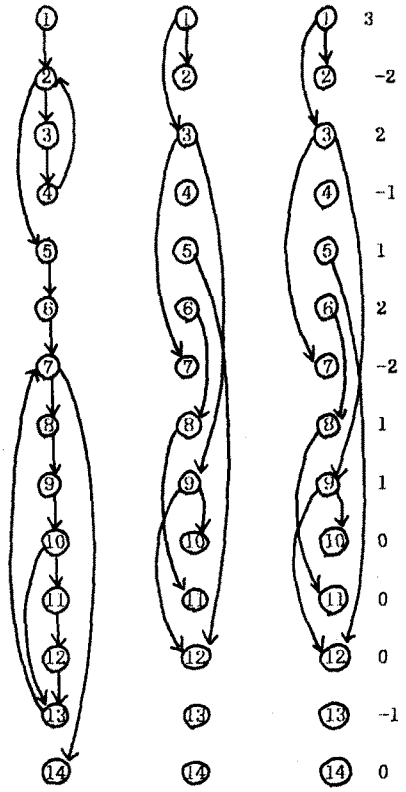
CDTF의 임의의 2개 구문에 대하여 변수의 발생과 사용 여부에 따라 VCG를 한다.

정의) VCG = (V,E)

(1) VCG의 Vertex 집합인 V는 CDTF의 각 구문 Label이다.

(2) CDTF의 i-구문과 j-구문 사이에 CCG에서 경로가 존재하고 i-구문에서 생성된 변수가 j-구문에서 생성되거나 사용되면 VCG에 방향성 edge (Vi,Vj) (단,Vi,Vj ∈V)를 설정하는데 이와같이 설정된 edge의 집합을 E로 한다.

이와같은 결정된 VCG에서 방향성 edge (Vi,Vj)가 존재하면 CDFTF에서 i-구문의 연산자는 j-구문의 연산자보다는 먼저 수행되어야 하므로 i-구문이 j-구문보다는 선행 State에 할당되어야 한다. 그림3의 CDTF에 대한 VCG는 그림4.(b)이다.



(a) CCG (b) VCG (c) (b)에 label 값 부여

그림 4. 그림3에 대한 CCG와 VCG 및 label값 부여

2.3.3 상태합성 알고리즘

종래의 데이타 전송부 설계를 위한 상태합성 알고리즘으로는 ASAP(As Soon As Possible)과 ALAP(As Late As Possible) 및 mobility[15]의 개념등을 이용하여 최소의 상태로, 또한 가능한한 적은 hardware 면적으로 실현되도록 상태를 합성하는 방법이 제안되었다. 본 논문에서는 데이타 전송부와 제어부를 함께 고려하면서 위의 조건을 만족하는 상태합성 알고리즘을 제안하는데 다음과 같다.

단계 1. VCG의 각 vertex에 label 값을 다음에 따라 부여한다.

경우1) CCG의 E에 (Vi,Vi+1)와 (Vj,Vi) (i<j)를 만족하는 Vi가 존재하면 Vi의 label값 =-2 이다.

경우2) CCG의 Vi가 무조건 천이문이면 Vi의 label값=-1이다.

경우3) 경우1),경우2)가 아니면 Vi에서 임의의 vertex까지의 경로 길이중 최대 경로 길이를 Vi의 label 값으로 한다.

위와같이 부여된 label값 중 최대값을 M이라 하면 상태합성의 최소 상태수는 (M+1)이다. 상태변수 n을 초기적으로 1로 한다.

단계 2. VCG의 탐색 시작 vertex를 i=1, 탐색 종료 vertex의 집합 VE의 임의의 원소 j의 설정은

다음과 같다.

경우1) Vk의 label값=-2 인 vertex k가 존재하면 k는 모두 j로 설정 한다.

경우2) CCG의 Outdegree=0 인 vertext를 j로 설정한다.

구성된 VE의 원소들을 증가 순으로 Sorting 한다.

단계 3. VE의 원소들 중 i보다는 크고 VE중 최소인 원소를 j라 한다.

경우1) i부터 (j-1) 까지의 VCG Vertex 중 indegree=0 이고 label값=(M-n+1) 인 Vertex중 구문 번호가 최대인 vertex를 k라 할때 i 부터 k까지의 구문의 연산자를 n번째 상태에 속하는 연산자로한다.

경우2) (k+1)번째 구문의 vertex label값=-1 이면 (k+1)구문도 n 번째 상태에 포함 시킨다.

n번째 상태에 할당된 구문에서 다른 구문으로 향하는 모든 edge를 VCG에서 제거한다. 또한, VE에서 j도 제거한다.

단계 4. 단계3에서 경우2)를 만족하였으면 i를 (k+2)로 하며 그렇지 않으면 i를 (k+1) 로 한다. VE의 원소들중 i보다 큰 원소가 있으면 n를 1증가하여 단계3으로 가고 없으면 모든 구문의 상태합성이 완료되었으므로 알고리즘을 종료한다.

3. 실 험 예

그림4 (a)의 VCG의 각 vertex에 대해 제안한 알고리즘의 단계1에서 label 값을 부여하면 그림 4(c)와 같으며 M=3이다. 단계2에서 VE를 구하면 Vl={2, 7,14}이며 n=1일때 단계3에서 초기적으로 i=1,j=2이고 조건을 만족하는 구문은 1번구문이다. n=2일때 단계3에서 i=2,j=7이므로 조건을 만족하는구문은 2-6이다. n=3일때 단계3에서 i=7, j=14이므로 조건을 만족하는 구문은 7-9이다. n=4일때 단계3에서 i=12,j=14이므로 조건을 만족하는 구문은10-14이다. 따라서 본 논문의 출력은 [14]의 8상태보다 효율적이다.

4. 결 론

본 논문에서는 digital system의 동작특성을 기술한 하드웨어 기술언어에 대하여 데이타 전송부및 제어부를 함께 고려하여 상태합성을 행하는 새로운 방법을 제안했다.

하드웨어 기술언어 BSDL를 입력하여 if - then - else문, 산술문및 데이타 전송문만으로 구성되는 CDTF을 생성하였다. CDTF에서 각 문의 수행 순서에 따라 CCG를, 변수의 생성 및 사용 여부에 따라 VCG를 구성한후 VCG의 각 vertex에 label값을 부여하였다. CCG와 VCG를 이용하여 데이타 전송부와 제어부에 대해 상태합성을 함께 행하는 새로운 상태합성 알고리즘을 제안하고 실제의 예에 대해 실험을 행하였다.

앞으로의 과제로는 임의의 상태수에 따라 하드웨어의 면적을 최소로 하는 상태합성 알고리즘에 대한 연구가 요구될 것이다.

참 고 문 헌

[1] D.D.Gajaski et al,"Silicon Compilation,"NATO Study Institute on Logic Synthesis and Silicon compilation for VLSI, 1986.

[2] A.V.Goldberg et al,"Approachs Toward Silicon Compilation,"IEEE Circuits and Devices Magazine, Vol. 1, No. 3, May 1985, pp.29-39.

[3] P.G.Paulin et al,"HAL: A Multi-Paradigm Approach to Automatic Data Path Synthesis,"Proc. of the 23rd DAC, 1986, pp.263-270.

[4] D.E.Thomas et al,"Automatic Data Path Synthesis,"IEEE Computer, December 1983, pp.59-70.

[5] A.C.Parker et al,"MAHA: A Program for Data path Synthesis,"Proc. of the 23rd DAC, 1986, pp.461-466.

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

[7] C.S.Jhon et al,"Silicon Compilation Based on a Data-Flow Paradigm," IEEE Circuits and Devices Magazine, Vol. 1, No. 3, May 1985, pp.21-28.

[8] Z.Peng,"Synthesis of VLSI Systems with the CAMAD Design Aid,"Proc. of the 23rd DAC, 1986, pp.278-284.

[9] R.Camposano et al,"Combined Synthesis of Control Logic and Data Path," Proc. of the ICCAD'87, 1987, pp.327-329.

[10] 조중휘,"VLSI의 논리 설계 자동화를 위한 Symbolic Description Language에 관한 연구," 한양대학교 박사학위 논문,1986.

[11] J.Allen,"Computer Architecture for Digital Signal Processing,"Proc. of the IEEE, May 1985, pp.852-873.

[12] G.A.Frank et al,"An Architecture Design and Assessment System," VLSI Design, Augst 1985, pp.30-50.

[13] E.F.Girczyc et al,"An ADA to Standard Cell Hardware Compiler Based on Graph Grammars and Scheduling,"Proc. of the ICCD'84, 1984, pp.726-731.

[14] R.Jamier et al,"APPLON, A Data Path Silicon Compiler,"IEEE Circuits and Devices Magazine, Vol. 1, No. 3, May 1985, pp.6-14.