지연 제약 조건을 고려한 새로운 ILP 스케줄링 알고리즘

김 기 복, 인 치 호
세명대학교 컴퓨터학과
Email: kgb1587@daum.net, ich410@semyung.ac.kr

Ki-Bog Kim, Chi-Ho Lin
Dept. of Computer Science
Semyung University

A New ILP Scheduling Algorithm that Consider Delay Constraint

Abstract

In this paper, we suggested the integer linear programming (ILP) models that went through constraint scheduling to simple cycle operation during the delay time.

The delayed scheduling can determine a schedule with a near-optimal number of control steps for given fixed hardware constraints. In this paper, the resource-constrained problem is addressed, for the DFG optimization for multiprocessor design problem, formulating ILP solution available to provide optimal solution.

The results show that the scheduling method is able to find good quality schedules in reasonable time.

Ⅰ. 서론

컴퓨터를 이용하여 디지털 시스템을 설계하고자 하는 자동설계 기술(design automation technique)은 대단히 복잡하고도 방대한 설계 과정을 거쳐야만 하는 것이기 때문에, 전 설계과정을 다음과 같은 3단계의 합성과정으로 구분지어, 보다 체계적으로 설계과정의 효율을 극대화 시킨다. 즉, 전 설계과정은 동작영역(behavioral & structural domain) 합성과정과 논리영역(logical domain) 합성과정 및 하위영역(physical domain)합성과정으로 구분되어, 톱다운(top-to-down) 설계방식에서는, 각 합성과증의 입력정보로서 한 단계 shb은 상위영역의 합성과정으로부터 얻어지는 출력정보(net list)를 취하게 된다.

동작영역의 자동합성기술에 대한 연구는 70년대 중반 이후부터 대학의 연구소에서 진행되었으며 [1], 이후 미국 Carnegie-Mellon 대학의 CMU-DA 시스템, System Atchtect's Workbench, Facet/Emerald 시스템, Stanford 대학의 Flamel/Olympus 시스템, USC의 ADAM 시스템, 서독의 Kiel 대학의 MIMOLA 시스템, 스웨덴 Linköping 대학의 CAMAD 시스템, 카나다 Carleton 대학의 ELF 시스템 및 HAL 시스템, 화란 IMEC 연구소의 CATHEDRAL-II/CATHEDRAL-III 시스템, 대만 Tsing-Hua 대학의 ALPS/LYRA/ARSY 시스템등과 같은 동작영역 합성용 툴이 개발되었다.

이러한 동작영역의 합성과정은 다시 동작기술(behavioral description), 변환(transformation), 데이터패스합성(data path synthesis) 및 제어패스합성(control path synthesis)으로 세분화 되어, 이 범주에 속하는 합성이론을 다우는 연구 분야를 상위수준합성(high-level synthesis)이라 부른다.

이러한 상위수준합성의 각 단계 중 가장 중요하고도 핵심적인 것은, 시스템의 동작알고리즘으로부터 비교적 구체적인 데이터패스의 구조적인 형태를 추출해 내는 데이터패스 합성단계이다.

따라서, 상위수준합성에 관한 대부분의 연구는 데이터

패스 합성단계의 스케줄링과 하드웨어 할당에 초점을 두고, 이의 효율성과 성능을 개선하고자 노력하였다.

기본적으로, 스케줄링 단계에서는 동작알고리즘의 모든 연산을 적절한 제어스텝(control step)에 할당시키고, 하드웨어할당 단계에서는 연산자(operator) 일명 기능단위(functional unit)라고도 불리운다

가장 기본적인 스케줄링 기법은 하드웨어 차원(hardware resource)과 동작속도의 제약이 가해지지 않은 상태에서 행할 수 있는 ASAP(As Soon As Possible)스케줄링 기법을 들 수 있다.

기존의 정수계획법 스케줄링 방식은 비선형적인 형태로 기술한 후 이를 선형적인 형태로 변환하기 때문에 수식이 난해함은 물론 수식이 복잡하고 비구조적이기 때문에 수식을 생성하고 분석하는데 상당한 시간이 요구되며 비용제약 스케줄링의 경우 전 스케줄링 단계에 걸쳐 자원의 제한 조건이 불필요하게 중복되므로 스케줄링의 효율이 좋지 않다는 단점을 갖고 있다 .

따라서 본 논문에서는 이러한 단점의 보완과 아울러 보다 개선된 방식의 스케줄링 기법을 제안하였다.

Ⅱ. ILP 기본 스케줄링

데이터 흐름도(DFG)상의 연산을 제어스텝에 할당 시키는 과정은 연산을 제어스텝에 사상 시키는 과정을 의미한다. 이를 위해서 연산과 제어스텝은 일정한 관계를 유지하여야만 한다. 연산간의 선후관계와 연산이 할당될 수 있는 제어스텝의 범위로 표현되는 연산의 속성은 일정한 관계식으로서 임의의 시간간격인 제어스텝의 속성에 결합될 수 있다.

지연 시간동안 ILP 모형들이 단순 주기 연산에 제약 스케줄링을 시키었던 것들이 개발되어진 것이다. 목적함수는 고정된 시간 제약을 주었던 자원의 수를 최소화하는 것이다.

제약들은 노드 제약 방정식에 의해서 기술되어지며 이 모형은 핸들과 다중 주기 연산으로 확장되어진다. 본 논문에서는 고정시간 조절 제약을 주었던 자원에서 피크 전원 영역과 수를 최소화 한 다중 주기 작동에 제약된 스케줄링을 시키었던 지연시간 동안 ILP 모형을 기술하였으며, ILP 의한 스케줄링 기법을 데이터 패스 합성에 실제적으로 적용시키기 위해서는 연산의 지연시간, 사용하고자하는 연산자의 연산 유형, 구성하고자 하는 데이터패스의 형태에 따라 각기 적용시킬 수 있는 ILP가 요구된다. 스케줄링 단계에서 필히 고려하여야만 하는 중요한 요소로서는 제어스텝의 시간간격을 들 수 있다. 왜냐하면 사용가능한 자원의 수가 충분한 경우, 제어스텝이 시간간격이 크면 클수록 많은 연산이 1-싸이클 타임 이내에서 수행될 수 있기 때문이다. [3]

Ⅲ 지연 조건을 고려한 ILP 스케줄링 알고리즘

자원의 제한조건 하에서, 연산이 할당되는 제어스텝의 최하한 값 T_{max}을 수하기 위한 지연스케줄링 기법은 ASAP 스케줄링이나, 또는, 기타의 성능제약 스케줄링 결과를 이용하여, 자원의 제약이 가해진 상태에서의 스케줄링을 행하는 방법이다.

ASAP 스케줄링, 또는, 성능제약 스케줄링에 의해 각 제어스텝에 할당된 연산은 뒤쪽제어스텝에 할당되어야만 한다. 즉 제어서텝 T_j에 할당되어야만 하는 경우 연산 Oi 는 (j-i)스텝 만큼 연산의 할당이 미루어져야만 한다. 이와같이 자원 제약이 가해지지 않은 상태하에서 임의의 연산이 할당될 수 있는 제어스텝과 자원제약이 가해진 상태하에서 동일 연산이 할당될 수 있는 제어스텝과의 차를 지연상수(Delay constant)라 정의할 때, 지연스케줄링은 이러한 지연상수를 최소화 시키기 위한 스케줄링이라고 볼수 있다.

정의된 연산의 지연상수와 가용자원 간에는 다음과 같은 관계식이 성립한다. Oi의 지연상수를 Ai, 연산 Oi의 최근접 후행연산으로서 연산 Oi가 할당된 제어스텝의 바로 뒤 제어스텝에 할당된 연산 Oj의 지연상수를 Aj라 할때 Aj는 최소한 Ai보다는 크거나 같은 값을 가져야 만 한다. 다시 말해서 연산 Oi의 할당이 유예된 만큼 또는 그 이상의 시간연산 O의 할당이 유예되어야만 한다. 따라서 Ai와 Aj사이에는 다음과 같은 관계식이 성립한다.

$$A_i \leqq A_j \qquad \text{(식1)}$$

제어스텝 k 내에 살당된 연산 유형 t인 연산의 개수를 nt, 연산유형 t인 연산을 구현할 수 있는 연산자의 개수를 Mt(Mt<Nk), 제어스텝-k에 할당된 연산의 지연상수를 Ai(i=1...m),지연상수가 Ai인 연산의 선행 연산의 지연상수를 ai라 할때 제어스텝 -(k+1)로 이동되어야만 제어스텝 -k 에서의 자원제한 조건이 만족된다. 따라서 이들 연산 사이에서는 다음과 같은 조건식이 만족된다.

$$\sum_{i=1}^{n_t} Ai \geqq (n_t - M_t), \quad (t = 1, \cdots, m) \qquad \text{(식2)}$$

또한 제어시텝-(k-1)에서 연산의 이동이 존재할 경우는 이동만큼의 연산이 추가로 후행 제어스텝으로 이동되어야만 한다. 그림1과 같이 제어스텝-(k-1)에 존재하는 연산의 지연상수와 부가상수를 각각 $a1, a2, a3$ 및 ρ1, ρ2,ρ3라 할때 제어스텝-(k-1)로 부터의 + 연산이동에

따른 식 (2)의 변화는 다음과 같다.

$$(1-a1)*\sum_{i=1}^{p}A1_i+(1-a2)*\sum_{i=1}^{q}A2_i \geqq (n_t-M_t)+\sum_{i=1}^{3}\rho i*ai \quad (식3)$$

식을 정리하면

$$\sum_{i=1}^{p}A1_i+\sum_{i=1}^{q}A2_i \geq (n_t-M_t)+(\sum_{i=1}^{p}A1_i+\rho 1)*a1 + (\sum_{i=1}^{q}A2_i+\rho 2)*a2+\rho 3*a3 \quad (식4)$$

또한 $\rho 1=-p,\ \rho 2=(1-q),\ \rho 3=1$이므로 식 (4)는 다음과 같이 표현된다.

$$\sum_{i=1}^{p}Ai_1+\sum_{i=1}^{q}A2_i \geqq (n_t-M_t)+(a2+a3) \quad (식5)$$

* 연산의 이동에 따른 수식 역시 (5)와 동일하다 그림 1의 *연산을 +연산으로 +연산을 * 연산으로 대치시키게 되면 식 (5)와 동일한 결과를 얻을 수 있다.

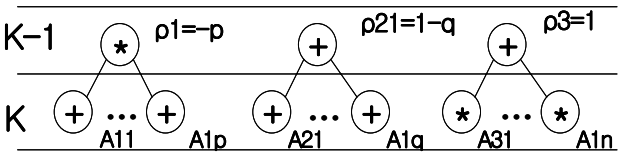

그림 1. 연산의 지연상수와 부가상수와의 관계
(Relations between delay constants and additive constants)

식 (1)과 식 (5)의 관계식이 만족되는 범위 하에서 제어스텝의 하한값을 최소로 하기 위한 스케줄링을 행할 경우 주어진 자원의 제한조건을 만족한은 비용제약 스케줄링의 결과를 얻을 수 있다, 또한 가용자원의 개수를 나타내는 M_t의 값을 최소로 하기 위한 스케줄링을 행하는 경우는, 즉 주어진 제어스텝의 범위 안에서 모든 연산을 수행시킬 수 있는 최소비용을 구할 수 있다.

Ⅳ 스케줄링 구현 및 결과

그림 2는 DFG 모델에 대해 ASAP 스케줄링을 행한 결과이다. 각 연산의 좌측 상단에 대한 표시된 기호는 연산의 지연상수를 나타낸 것이다.

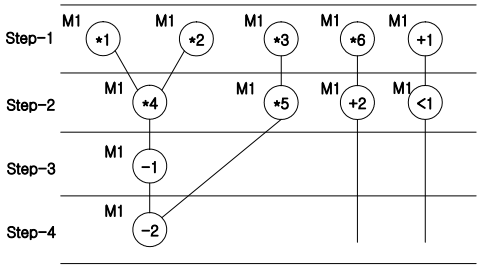

그림2. DFG 모델

식1은 각 연산의 지연상수가 만족하여야만 하는 조건식이므로 이에 따라 지연상수 간의 관계식을 기술하면 다음과 같다.

$M1 \leqq M4, \quad M2 \leqq M4$

$M4 \leqq S1 \leqq S2$

$M3 \leqq M5 \leqq S2$

$M6 \leqq A2$

$A1 \leqq C1$

식 5는 자원의 제한조건을 초과하는 제어스텝에서의 지연상수 간의 관계를 기술한 식이므로 이에 따른 수식의 전개는 다음과 같다.

제어스텝-1 M1+ M2+ M3+ M6 ≥(4-2)

제어스텝-2 M4+ M5≥(2-2)+ M1+ M2+ M3+ M6
A2+ C1≥(2-1)+ A1

제어스텝-3 0≥(0-2)+ M4+ M5
S1 ≥(1-2)=A2+ C1

제어스텝-4 S2≥(1-2) + S1

이러한 관계식을 만족하면서 제어스텝의 개수 데이터패스의 지연시간을 최소로 하는 지연상수 값은 M1 = M4 =M5= M6 =A2= S1= S2=1 이며, 데이터 패스의 지연시간은 5-싸이클타임이다. 따라서 2개의 승산기와 하나의 가산기로는 5-싸이클타임 이내에서 모든 연산의 수행이 가능하다.

이러한 관계식과 지연상수 간의 관계식을 이용하여, 지연 스케줄링을 행하면 표1과 같은 지연 상수값이 구해진다. 구해진 지연상수의 값에 해당되는 만큼 연산을 뒤쪽 제어스텝으로 이동시키게 되면 전 제어스텝 내에서 주어진 자원의 제한 조건이 만족되고 있는 스케줄링 결과를 얻을 수 있다 데이터패스의 지연시간이 성능제약 스케줄링 결과에 비해 2-스텝 만큼 확장되었다.

표1 지연상수의 값

A1	0	A6	0	A11	1	A16	2	A21	0	A26	2	M5	0
A2	0	A7	0	A12	0	A17	2	A22	2	M1	0	M6	2
A3	0	A8	0	A13	2	A18	1	A23	0	M2	1	M7	1

A4	0	A9	1	A14	0	A19	1	A24	1	M3	0	M8	2
A5	0	A10	1	A15	1	A20	2	A25	2	M4	1		

Ⅵ. 결론

모든 스케줄링 방식이 정수계획법에 의해 이루어 질 수 있도록 ASAP 및 ALAP 스케줄링을 휘한 정수계획법을 제안하여, 동일 환경하에서 일률적으로 스케줄링이 행해지도록 하였으며, 연산의 체이닝과 멀티싸이클링에 의한 데이터패스의 지연시간을 비교분석하였으며 도한 비용제약 스케줄링 기법으로서, 성능제약 스케줄링의 결과를 이용하여 자원의 제약이 가해진 상태에서의 스케줄링을 행할 수 있는 지연스케줄링 기법을 제시함으로서 성능제약 스케줄링과 비용제약 스케줄링이 동일 과정에서 행해짐은 물론 부분적인 설계공간의 탐색이 가능하였다.

제안된 정수계획법 수식의 변수는 $O(s \times n)$인 형태로 증가하고 생성되는 방정식은 $O(s \times m + n + r + e_f))$로 증가하기 때문에, 복잡도는 기존의 정수계획법 수식의 복잡도와 동일하다. 그러나, 자원의 제약이 가해진 상태에서의 스케줄링 기법인 지연스케줄링의 경우는 수식의 변수가 $O(s \times m)$의 형태로 증가하기 때문에 복잡도가 매우 낮다는 장점을 갖고 있다.

참고 문헌

[1] D.Gajaski, N.Dutt, A. wu an Li허n,g "고무High-Level synthesis : Introduction to chip and system design

[2] R.S.Martin and J.P.Knight, "*Optimizing Power ASIC Behavioral Synthesis*", IEEE Design and test computer.

[3] E.Musoll and J.Cortadella, " *High - Level Synthesis Techniques for Reducing the Activity of Functional Units*" International symposium for Low Power Design.

[4] A.P.Chandrakasan and R.W.Brodersen, "*Low Power Digital CMOS Design*", Kluwer Academic Publishers.

[5] G. Goossens, J. Rabaey, J. Vandewalle, and H. De Man, *"An efficient microcode compiler for application specific DSP processors,"* IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 9, no. 9, pp. 925-937, September 1990.

[6] T.F. Lee, A.C.H. Wu, Y.L. Lin, and D.D. Gajski, *"A transformation-based method for loop folding,"* IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 13, no. 4, pp. 439-450, April 1994.

[7] V.K. Madisetti, *VLSI Digital Signal Processors, An Introduction to Rapid Prototyping and Design Synthesis,* IEEE Press and Butterworth Heinemann, Boston, 1995.

[8] J. Sanchez and H. Barral, "*Multiprocessor implementation models for adaptive algorithms,*" IEEE Transactions on Signal Processing, vol. 44, no. 9, pp. 2319-2331, September 1996.