5K−3

ハードウェアリソース最小化をめざしたスケジューリング法の検討

池田充郎　宮崎敏明
NTT LSI研究所

1　はじめに

LSI設計の短TAT化のために、上位合成技術研究の実用化が強く望まれている。本報告では、上位合成を構成する一分野であるスケジューリング問題を取り上げ、入力仕様動作の実行時間制約下で演算器数の最小化をめざした新たなスケジューリングアルゴリズムを提案する。同種のスケジューリング法としてForce-Directedスケジューリング法(FDS法)[2]が知られているが、本アルゴリズムは、FDS法と同等かあるいはより少ない演算器数でスケジューリングすることができる。

2　スケジューリングアルゴリズム

スケジューリングとは入力動作仕様内の各演算をどの時刻(制御ステップ)で実行するかを決定することである。スケジューリングの結果、動作仕様を処理するのに必要な演算器数や制御ステップ数が求まる。ここで、最終的に必要な演算器数は各制御ステップ内の演算数の最大値であるから、演算器数を最小化するためには各ステップの演算数の最大値を小さく抑える必要がある。

2.1　割り当て法

提案手法の概要を述べる。ここでは、与えられた制御ステップ数の範囲内で、各ステップに、可能な限りひとつずつ演算を割り当てていくという方針をとる。各ステップひとつずつの割り当てですべての演算の割り当てが終わらなかった場合、再度、同様の割り当てを行ない、すべての演算を割り当てるまで繰り返す。もし、その繰り返しが1回で終われば、対応する演算器は1つでよく、繰り返しが2回で終われば、演算器は2つ必要となる。この方法では、可能な限り各制御ステップにひとつずつ演算を割り当てていくため、同じ制御ステップに多数の演算が集中するということが起こりにくく、最終的な演算器数を小さく抑えることができる。

割り当ての際、同じ制御ステップに割り当てることが可能な演算が複数個存在する場合は、ある基準に従って演算に優先順位をつけ、優先度の高い演算から先に割り当てていく。本手法では、その優先基準として、「演算の自由度の変化量」を定義する。

2.2　優先基準: 自由度の変化量

図1は“Diffeq”と呼ばれる動作記述[1]に対するデータフローグラフおよび各演算の自由度(Mobility)を示したものである。

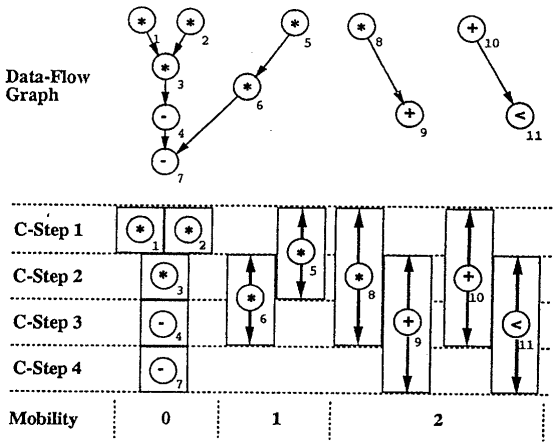

図1: データフローグラフと演算の自由度

データフローグラフ上のデータ依存関係から、演算を割り当てることが可能な制御ステップ(C-Step)の上限と下限がわかり、各演算の自由度が得られる。ここで、演算の自由度とは、「割り当てることが可能な制御ステップ数−1」とする。例えば、図1において乗算1は制御ステップ1にしか割り当てることができないので自由度は0である。一方、乗算5は制御ステップ1もしくは制御ステップ2に割り当てることができるので自由度は1である。なお、この状態における各演算の自由度の総和は10である。

この自由度の総和がスケジューリングに伴い、どのように変化するかを図2に示す。これは、“Elliptic Wave Filter(以下EWF)”[2]に対する実験結果である。与えた制御ステップ数は、最短ステップ数より2ステップ多い19ステップである。提案手法以外にFDS法, ALAPスケジューリング法およびListスケジューリング法[1]を使用したときの実験結果も示した。図中の“Result”は各スケジューリングの結果、必要となった演算器(加算および乗算)の個数を表す。どのスケジューリングも割り当て(Assignment)が進むにしたがって自由度(Mobility)は減少するが、減少の仕方は手法によって異なることがわかる。ALAPスケジューリングのように、割り当ての早い段階で自由度が急激に減少すると、残った演算が同じ制御ステップに集中して割り当てられてしまう可能性が高い。以上の考察から、演算の割り当てに際しては、できるだけ自由度の総和を保ったまま、割り当てることが望ましいことがわかる。

そこで、割り当ての際の優先基準として、演算を割り当てることによって自由度の総和の減少ができるだけ少ないものを先に選ぶことにする。

A Scheduling Algorithm to Minimize Hardware Resources
Mitsuo Ikeda, Toshiaki Miyazaki
NTT LSI Laboratories

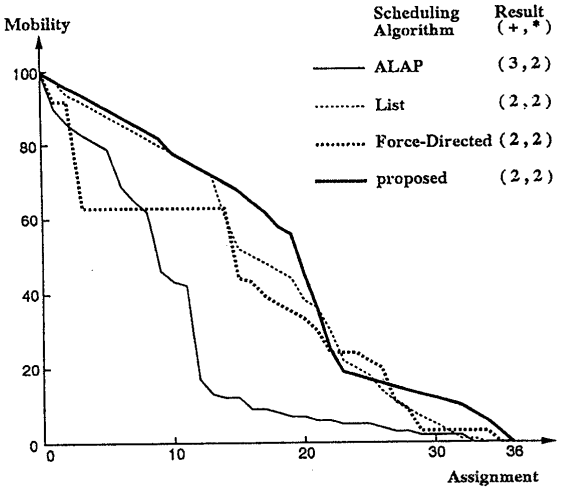

図 2: 割り当てにともなう自由度の変化

2.3 アルゴリズム

i,j を制御ステップの番号、k,l を演算の種類の番号とする。また、制御ステップの最大値を S_{max}、演算の種類の総数を F とする。さらに、制御ステップ i に種類 k の演算を割り当てたかどうかのフラグを FLG_{ik} とおく。

Step1 フラグ $FLG_{ik}(1 \leq i \leq S_{max}, 1 \leq k \leq F)$ を 0 に初期化する。

Step2 $FLG_{ik} = 0$ である制御ステップ i, 演算の種類 k に該当する割り当てが可能なすべての演算について、

- 割り当てることが可能な各制御ステップに仮割り当てを行なう。
- 仮割り当てをしたとき自由度が変化する演算について変化量を計算し、その総和を求める。

Step3 自由度の変化の総和が最も少ない演算をひとつ割り当てる。ただし、自由度の変化の総和が等しい場合には、割り当てる演算自身の自由度の変化が少ない方を選ぶ。

その制御ステップを j, 演算の種類を l とすると、その割り当てに対応するフラグ FLG_{jl} を 1 にする。

Step4 $FLG_{ik} = 0$ である制御ステップ i, 演算の種類 k が存在し、かつ、それに該当する割り当てが可能な演算が存在するか。存在するなら、Step2 へ。

Step5 すべての演算の割り当てが終わったか。まだならば、Step1 へ。

実際に実現したプログラムでは、計算量を減少させるために以下の改良を行なっている。スケジューリングの際、割り当てられる制御ステップが他の演算の割り当て等によって自動的に決定される演算が存在するが、先のアルゴリズムではそのような演算に対しても、割り当ての際の自由度を調べることになる。そこで、スケジューリングを始める前、あるいは、他の演算が制御ステップに割り当てられたときに自動的に制御ステップが決定される演算について、仮割り当ての計算を省略するようにアルゴリズムを改良した。プログラムは C 言語を用いて UNIX マシン (Sparcstation1+) 上に実現した。

3 実験結果

EWF に対し、提案手法と FDS 法,List スケジューリング法を適用した結果を表 1に示す。これは EWF の演算のうち乗算に制御ステップが 2 ステップ必要であるとして実験したものである。表 1で、"+","*" の右の数字はスケジューリングの結果必要となった加算器, 乗算器数をそれぞれ表している。

表 1: EWF のスケジューリング結果

Steps	17			18			19,20		
手法	FD	List	本手法	FD	List	本手法	FD	List	本手法
+	3	3	3	3	3	2	2	2	2
*	3	3	3	2	2	2	2	2	2

処理時間は、CPU 時間で数秒から十数秒であった。実験結果をみると、最短の制御ステップ数 (17 ステップ) では、必要とする演算器数に差は出なかったが、与える制御ステップ数の制約を 1 ステップ緩めた時に、FDS 法より少ない演算器数でスケジューリングを行なった。また、乗算に必要な制御ステップを 1 ステップとした場合も同様の結果を得た。

4 まとめ

本報告では、必要とする演算器数をできるだけ少なくするために、(1) 可能な限り、各制御ステップにひとつずつ演算を割り当てて行くこと、(2) 割り当ての優先順位を決めるために、演算の自由度の変化量に着目し、できるだけ自由度が減らない割り当てを行なうこと を特徴とするスケジューリング法を提案した。

また、同じ制御ステップ数を与えたときに、提案手法は、必要とする演算器数の面で FDS 法と同等かそれを上回る良好な結果が得られることを実験により確かめた。

今回は条件分岐やループ処理を含まない単純な動作記述にスケジューリングを限定したが、条件分岐などを含む動作記述からのスケジューリングが今後の課題である。

謝辞

日頃から有益な助言を頂く NTT LSI 研究所 設計システム研究部 安達徹主幹研究員および星野民夫主幹研究員に感謝します。

参考文献

[1] M.C.McFarland,A.C.Parker and R.Camposano, *"Tutorial on High-Level Synthesis,"* Proc. of the 25th Design Automat. Conf.,pp.330-336,June 1988.

[2] P.G.Paulin and J.P.Knight, *"Force-Directed Scheduling for the Behavioral Synthesis of ASIC's,"* IEEE Trans. on CAD, Vol.8 No.6,pp.661-679,June 1989.