計算機アーキテクチャ　115－2
設　計　自　動　化　78－2
（1995．12．14）

レジスターレジスタ間データ転送を考慮したパイプライン方式データパスのスケジューリング法

原嶋 勝美† 小味 弘典‡ 福永 邦雄†
harasima@cs.osakafu-u.ac.jp

† 大阪府立大学工学部
〒593 堺市学園町1-1

‡ 日立製作所
〒244 横浜市戸塚区吉田町292

ハイレベルシンセシスにおいてスケジューリングは重要な要素技術のひとつである．本稿では，ハードウェアコストの最小化を考慮して，パイプライン方式のデータパスをスケジューリングする手法を提案する．従来の手法では，パイプライン方式によって生じるレジスターレジスタ間のデータ転送のための接続コストは考慮されていなかったのに対し，本手法では，バスを介してレジスターレジスタ間データ転送を行うデータパスモデルを導入し，整数線形計画法により他のハードウェアコストと同時に最小化を行っている．これにより，スケジューリング時にさらに厳密にハードウェアコストを見積もることが可能になった．

A Scheduling Method for Pipelined Datapaths Considering Register-to-Register Data Transfers

Katsumi HARASHIMA† , Hironori KOMI‡ and Kunio FUKUNAGA†

† Faculty of Engineering, University of Osaka Prefecture
1-1, Gakuen-cyo, Sakai, Osaka, 593 Japan

‡ Hitachi ltd.
292, Yoshida-cyo, Totsukaku, Yokohama, Kanagawa, 244 Japan

In high level synthesis, scheduling is an important stage which assigns each operation appeared in a data flow graph to a specific control step, of which results influence the design quality directly. This paper describes a scheduling approach for pipelined datapaths. Since no previous approach estimates the interconnection cost between registers (register-to-register cost), our approach introduces a datapath model with the interconnection between registers across buses, and minimizes the total hardware cost including the register-to-register cost with Integer Linear Programming approaches. Consequently the proposed approach can estimate the hardware cost exactly in the scheduling phase.

1 まえがき

ハイレベルシンセシスにおけるスケジューリングは回路の状態遷移を決定する重要な要素技術であり，近年では，データパスのパイプライン化を考慮したスケジューリング法が数多く提案されている [1]–[8]．データパスをパイプライン化すると単位時間当たりの処理効率(スループット)を向上させることができるが，ノンパイプラインの場合より同時に処理する演算数が増えハードウェアコストが増加する．本論文では，パイプラインのスループットを制約条件として与え，データパスのハードウェアコスト最小化を考慮したスケジューリングを行う手法を提案する．

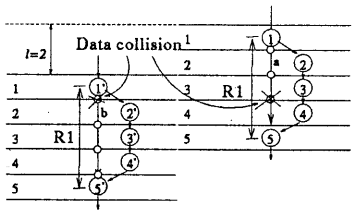
ハードウェアコストの最小化を考える場合，チップ内の接続コストは最も重要な要因であるといえる．従来のスケジューリング法の多くは，演算器とレジスタ間のデータ転送に必要なバスの本数を見積もって，これを接続コストとして扱い，スケジューリング時に最小化を行っている．しかし，データパスをパイプライン化した場合，レジスタにおけるデータ衝突を避けるため，レジスターレジスタ間でデータ転送を行う必要が生じる場合があり，これに必要となる接続コストは，従来のスケジューリング法では考慮されていなかった．

そこで，本手法ではレジスターレジスタ間におけるデータ転送と演算器ーレジスタ間データ転送が同じバスを共有するシステムをデータパスモデルとして考え，スケジューリング時に，そのバスの本数を他のハードウェアユニットと同時に最小化する．これにより，パイプライン化されたデータパスにおけるレジスターレジスタ間データ転送のための接続コストをより厳密に評価し，最小化することを可能としている．また，最適化手法として，整数線形計画法を用いており，ハードウェアコストを厳密に最小化し，スケジューリングすることを可能にしている．

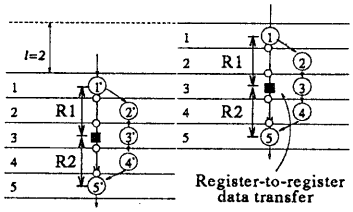
2 スケジューリング問題のモデル化

2.1 レジスターレジスタ間データ転送

各演算の出力データがレジスタで保持される期間をライフタイムと呼ぶが，パイプライン方式データパスでは，ライフタイムが単位処理の開始間隔(レイテンシ)よりも長い場合，レジスターレジスタ間でデータを転送する必要が生じる．例えば，与えられたデータフローグラフ(DFG)を図1(a)のようにレイテンシ l=2 としてスケジューリングを行った場合，演算1の出力データは演算5の入力データとして用いられるため，4ステップの間レジスタにおいて保持される必要がある．一方，演算1′は演算1の出力データのライフタイムの間に $R1$ へ演算結果を出力するため，演算5には次段の演算1′からの出力データが入力され，誤った計算が生じる．そこで，図1(b)のように次段の実行結果が出力される前に，レジスタ $R1$ からレジスタ $R2$ へデータ転送を行う．このレジスターレジスタ間転送によって，演算1と演算1′の出力結果はそれぞれ $R2$, $R1$ において同時に保持することができ，不当なデータ衝突を避けることが可能となる．



(a) レジスタにおけるデータ衝突



(b) レジスターレジスタ間データ転送

図 1: レジスターレジスタ間データ転送

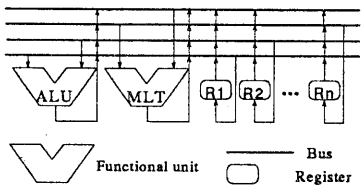


図 2: データパスモデル

2.2 データパスモデル

本手法では，レジスタからレジスタへのデータ転送と，演算器とレジスタ間のデータ転送が同じバスを共有するデータパスモデルを考え，そのバスのコストをスケジューリング時に考慮することで，レジスターレ

ジスタ間データ転送のための接続コストを厳密に見積もっている.

本手法で用いるデータパスモデルは, 図2のように各演算器, レジスタはそれぞれ固有の入力バスを持ち, 任意のバスにデータを出力することができるものとする [9].

3 整数線形計画法による定式化

本手法では, スケジューリングを行う際, 整数線形計画法 (ILP: Integer Linear Programming) により組合せ最適化問題を解くことにする. 以下,Gebotys, Hwang らによって提案されている制約式 [9][10], および本手法で提案しているレジスターレジスタ間データ転送のための制約式について述べ, 最小化すべき目的関数を説明する.

3.1 データ依存関係に関する制約式

スケジューリング時にDFG $D(V,E)$ (Vは演算の集合, Eはデータ依存関係) 中の各演算を割り当てることができる制御ステップは, ASAP(As Soon As Possible) およびALAP(As Late As Possible) スケジューリングによって求めることができる. ここで, 演算 i を $o_i(o_i \in V)$, o_iがASAP, ALAP によって割り当てられた制御ステップを各々$S_i(asap)$, $S_i(alap)$, o_iの割り当て可能な制御ステップを $R(i) = \{s|S_i(asap) \leq s \leq S_i(alap)\}$ で表す.

初めに, 各演算の制御ステップへの割当を表すために, 次の 0-1 変数 $x_{i,j}$を定義する.

$$x_{i,j} = \begin{cases} 1 & \left(\begin{array}{l}\text{演算 } i\text{の実行開始が制御}\\ \text{ステップ } j\text{に割り当て}\\ \text{られた時}\end{array}\right) \\ 0 & (\text{それ以外のとき}). \end{cases} \quad (1)$$

この $x_{i,j}$を用いて, データ依存を保証するための制約式を以下に示す.

$$\sum_{j \in R(i)} x_{i,j} = 1 \qquad \forall i, \quad (2)$$

$$\sum_{\substack{j2 \leq j \\ j2 \in R(i2)}} x_{i2,j2} + \sum_{\substack{j1 \geq j - d_{i1} + 1 \\ j1 \in R(i1)}} x_{i1,j1} \leq 1 \quad (3)$$

$$\forall (o_{i1} \rightarrow o_{i2}),\ j \in (R(i1) + d_{i1} - 1) \bigcap R(i2).$$

ただし $o_{i1} \rightarrow o_{i2}$は演算 $i1$ が演算 $i2$ の出力データを入力データとして直接用いるような, o_{i1}, o_{i2}の組を表し, d_iはo_iがマルチサイクリングを用いて実行されるのに必要な制御ステップ数である [10].

3.2 ハードウェアコストに関する制約式

3.2.1 演算器

タイプ fの演算器数を $n_{fu(f)}$とすると, その演算器で処理される演算数は各パイプステージにおいて $n_{fu(f)}$以下である. このため, 演算器に関する制約式は次式で与えられる.

$$n_{fu(f)} \geq \sum_{o_i \in FU(f)} \sum_{j \in S(p)} x_{i,j} \qquad \forall p,\ f. \quad (4)$$

ただし, $FU(f)$ はタイプ fの演算器によって処理される演算の集合, $S(p)$ はパイプステージ p $(1 \leq p \leq l)$ において処理される制御ステップの集合である.

3.2.2 レジスタ

同時刻にライフタイムを持つデータは, 同じレジスタを共有できないため, 必要とされるレジスタ数は, 各パイプステージの境界においてその境界と交差するライフタイムを持つデータ数を数え, その最大値として見積もることができる. 従って, レジスタに関する制約式は以下のようになる [9].

$$\sum_{(o_{i1} \rightarrow o_{i2}) \in SL(p)} \sum_{j \in S(p)} \Bigg(\sum_{\substack{j1 \leq j - (d_{i1}-1) \\ j1 \in R(i1)}} x_{i1,j1} + \sum_{\substack{j2 > j \\ j2 \in R(i2)}} x_{i2,j2} - \sum_{\substack{j3 \leq j \\ j3 \in R(i2)}} x_{i2,j3} - \sum_{\substack{j4 > j - (d_{i1}-1) \\ j4 \in R(i1)}} x_{i1,j4} \Bigg) \leq 2n_{reg}$$
$$\forall SL(p) \subseteq L(p),\ p. \quad (5)$$

ここで, n_{reg}はレジスタの必要数, $L(p)$ は $(o_{i1} \rightarrow o_{i2})$ の全集合のうちパイプステージ p を横切り, データのライフタイムとなることができる演算組 (データの先端と終端) の集合である. また, $SL(p)$ は, $L(p)$ のうち各データの終端を唯一として選んだ $L(p)$ の部分集合である.

3.2.3 演算器ーレジスタ間データ転送のためのバス

各パイプステージおいて, 演算器ーレジスタ間のためのデータ転送数は各演算の入力データ数と出力データ数の合計で見積もることができ, 演算器ーレジスタ間のデータ転送のために必要となるバス数 n_{bus}は, その最大値として与えられ, 以下の制約式で表される [9].

$$\sum_{j \in S(p)} \left(\sum_i In(Ty(i)) x_{i,j} + \sum_i x_{i,j-d_i+1} \right) \leq n_{bus} \qquad \forall p. \quad (6)$$

ただし, $In(f)$ はタイプ f の演算器が必要とする入力バス数, $Ty(i)$ は o_i を処理する演算器のタイプを表す. また, 第1項, 第2項が各々入力データ数, 出力データ数の合計を表す.

3.3 レジスターレジスタ間データ転送の考慮

3.3.1 データ転送のスケジューリング

3.2.2 で述べたデータのライフタイムを決定する可能性のある演算組のうち, ライフタイムがレイテンシ l より大きくなり得る組を $(o_{i1} \stackrel{\geq l}{\rightarrow} o_{i2})$ で表する. ある演算組で定義されるライフタイムは, 最も大きい場合を $LT(i1,i2)$ とすれば $LT(i1,i2) = S_{i2}(alap) - (S_{i1}(asap) + d_{i1} - 1)$ となり, $(o_{i1} \stackrel{\geq l}{\rightarrow} o_{i2})$ の場合 $LT(i1,i2) \geq l+1$ であるといえる. 本手法では, ライフタイムが l を越えるデータは, その間にレジスタからレジスタへデータ転送が行われるとし, 1つのレジスタに保持される時間がレイテンシ以下になるように各データ転送のスケジューリングを行う. 図3のように, 接続コストの抑制を考慮して各データのレジスターレジスタ間データ転送は1つのパイプステージに属する制御ステップで行われるようにする.

ここで, データ転送の有無を表すために, 0–1 変数 $v_{i,p}$ を以下のように定義する.

$$v_{i,p} = \begin{cases} 1 & \left(\begin{array}{l}\text{演算 } i \text{の出力データが}\\ \text{パイプステージ } p \text{で}\\ \text{レジスターレジスタ間}\\ \text{転送される時}\end{array}\right) \\ 0 & (\text{それ以外のとき}). \end{cases}, \qquad (7)$$

データのライフタイムがレイテンシより大きい場合, $v_{i,p}$ はいずれかのパイプステージにおいて1にならなければならない. 従って, 以下の制約式を満たす必要がある.

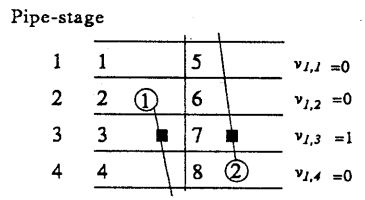


図 3: レジスターレジスタ間データ転送のスケジューリング

$$(LT(i1,i2) - l)\sum_p v_{i1,p} \geq (lt(i1,i2) - l) \quad \forall(o_{i1} \stackrel{\geq l}{\rightarrow} o_{i2}), \qquad (8)$$

$$\sum_p v_{i1,p} \leq 1 \qquad \forall(o_{i1} \stackrel{\geq l}{\rightarrow} o_{i2}). \qquad (9)$$

ただし, $lt(i1,i2) = \sum_{j\in R(i2)} j \cdot x_{i2,j} - (\sum_{j\in R(i1)} j \cdot x_{i1,j} + d_{i1} - 1)$ とする.

3.3.2 レジスターレジスタ間データ転送のためのバス

各パイプステージ p において o_i の出力データがレジスターレジスタ間データ転送されるために必要なバス数 $b_{i,p}$ は, そのデータが保持される期間内においてパイプステージ p で処理される制御ステップ数で見積もることができる. 従って, $b_{i,p}$ は以下の制約式で表される.

$$\sum_{j\in R(i2)} G(j,p)\cdot x_{i2,j} - \sum_{j\in R(i1)} G(j+d_{i1},p)\cdot x_{i1,j} \leq b_{i1,p} + K_1(1 - v_{i1,p}) \quad \forall p,\ (o_{i1} \stackrel{\geq l}{\rightarrow} o_{i2}), \qquad (10)$$

$$b_{i,p} \leq K_2 \cdot v_{i,p} \qquad \forall p, o_i \in SV. \qquad (11)$$

ただし, $G(j,p) = \lfloor (j-1+l-p)/l \rfloor$, SV はレイテンシ l よりも長いライフタイムを持つ可能性のあるデータを出力する演算の集合である. また, K_1, K_2 は十分大きな正の整数とする.

式(10)は, 右辺第二項により $v_{i1,p} = 1$ の時のみ制約条件として意味を持ち, その時, 各 $x_{i,j}$ に係数 $G(j,p)$ を掛けることで $b_{i1,p}$ を与えている. この $b_{i,p}$ を用いて, レジスターレジスタ間データ転送を考慮したバスの本数は, 制約式(6)を変形して以下のように表される.

$$\sum_{j\in S(p)} \left(\sum_i In(Ty(i))x_{i,j} + \sum_i x_{i,j-d_i+1}\right) + \sum_{o_i\in SV} b_{i,p} \leq n_{bus} \qquad \forall p. \qquad (12)$$

この制約式により, 演算器ーレジスタ間とレジスターレジスタ間データ転送の両方を同時に考慮して, バスの本数を見積り最小化することができる.

3.4 目的関数

以上に述べた制約式のもとで, チップ全体のハードウェアコストを最適化する場合, ILP により最小化すべき目的関数は以下の式で与えられる [9].

表 1: スケジューリングに用いた DFG

EWF	Elliptical wave filter[12]
SEC	Second-order direct form filter[13]
APL	Fourth-order all-pole lattice filter[6]

表 2: スケジューリング結果

Data	T	l	#bus	#mut	#add	#reg	cpu
EWF	18	16	9	2	3	18	5.8
	18	14	11	2	3	18	19.1
	18	11	12	3	3	18	115.6
	17	16	12	3	4	16	1.9
	17	10	15	4	5	19	8.9
	17	7	17	3	5	28	10.1
SEC	6	2	11	2	2	12	0.6
	6	1	22	4	4	27	1.0
	5	2	11	2	2	10	0.6
	5	1	20	4	4	27	0.6
APL	13	11	6	1	2	10	4.8
	13	9	6	2	2	10	2.1
	13	8	8	1	2	10	22.2
	12	9	9	2	3	10	0.6
	12	8	9	1	3	10	0.7
	12	7	8	1	2	10	0.5
	12	6	9	2	3	14	0.9

Minimize:

$$\sum_f c_{fu(f)} \cdot n_{fu(f)} + c_{reg} \cdot n_{reg} + c_{bus} \cdot n_{bus}. \quad (13)$$

ただし, $c_{fu(f)}$, c_{reg}, c_{bus} はそれぞれタイプ f の演算器, レジスタ, バスの単位ユニット当たりのコストである. 制約式を満たし, この目的関数を最小にする $x_{i,j} = 1$, $v_{i,p} = 1$ となる組合せを ILP により導くことで, 与えられた DFG に対し, ハードウェアコストを最小にする大域的最適解を求めることができる.

4 実験

4.1 実験条件および結果

スケジューリングを行う DFG として, 表 1に示すデータを用いた. 演算の処理時間は EWF の乗算を除きすべて 1 ステップとし, EWF の乗算は 2 ステップで処理されるものとした [9][10].

目的関数 (13) において, 加算器をタイプ 1, 乗算器をタイプ 2 とし, 式 (13) 中の各係数 $c_{fu(1)}$, $c_{fu(2)}$, c_{reg}, c_{bus} はそれぞれ 50, 250, 15, 100 とした [9].

表 1のデータをスケジューリングした結果を表 2に示す. また, 実際のスケジューリング例として, l=7,

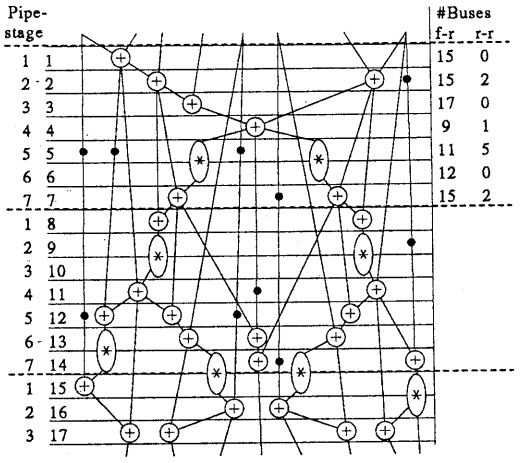


図 4: スケジューリング例 (EWF)

$T = 17$ として求めたときの EWF の結果を図 4に示す. 図中, 枝上の黒丸はレジスターレジスタ間のデータ転送を表し, #Buses で示した列中の (f-r), (r-r) はそれぞれ各パイプステージにおいて演算器ーレジスタ間データ転送, レジスターレジスタ間データ転送のために必要とされたバス数を表しており, 一部のパイプステージに集中することなく, 2 種類のデータ転送が効率よくデータバスを共有するようスケジューリングされていることがわかる.

4.2 考察

一般に ILP を解くための計算時間は, 最悪の場合で整数変数の数に対して指数関数的に増加する. 従って, ILP を用いた解法では, 実際の問題規模と計算時間の関係を考慮する必要がある.

本手法では, 計算時間に関して以下のような傾向がある.

1. レイテンシが小さいほど計算時間は増加する.
2. 遅延時間が長いほど計算時間が増加する.

しかし, 以下に示すような実際的な制約から, 多くの問題に本手法を適用し, 有効な時間内にスケジューリング結果が得られると考えられる.

1. 各繰り返し単位間にデータ依存関係がある回路の場合, 無制限にレイテンシを小さくすることができない [3][6].
2. レイテンシを極端に小さくすると処理並列度の著しい増加をまねき, 回路の実現が困難になる.

3. 遅延時間を長くすることは回路のパフォーマンスを低下させる.

従って, 本手法のようにILPに基づく厳密手法も, より短い計算時間で問題を解くことができ, さらに詳細な制約条件を付け加えてハードウェアコストを見積もり最小化できる可能性があると考えられる.

5 むすび

本論文では, パイプライン方式データパスをハードウェアコストの最小化を考慮し, スケジューリングを行う手法を提案した. パイプライン時に生じるレジスターレジスタ間データ転送をバスを介して行うデータパスモデルを導入し, 従来考慮されていなかったレジスターレジスタ間データ転送のための接続コストを見積もることを可能にした.

今後の課題として, 条件分岐, 繰り返し処理される演算列間のデータ依存関係といった実際的な制約を考慮できるようにし, より大規模な問題にも適用できるようにすること, また, スケジューリング時にアロケーションを同時に考慮するよう本手法を拡張することが挙げられる.

参考文献

[1] Park N. and Parker A.C.: "Sehwa: A Software Package for Synthesis of Pipelines from Behavioral Specifications", IEEE Trans. Comput.-Aided Des. Integrated Circuits & Syst. , **7**, 3, pp336–370 (Mar. 1988).

[2] Jain R., Parker A.C. and Park N.:"Predicting System-Level Area and Delay for Pipelined and Nonpipelined Designs", IEEE Trans. Compt.-Aided Des. Integrated Circuits & Syst., **11**, 8, pp.955–965 (Aug. 1992).

[3] Maciel F.B., Miyanaga Y. and Tochinai K.:"Optimizing and Scheduling DSP Programs for High Performance VLSI Designs", IEICE Trans., **E75-A**, 10, pp.1191-1201 (Oct. 1992).

[4] Komi H., Harashima K. and Fukunaga K., "Throughput Optimization in Pipelined Data-Path Scheduling", Proc. of JTC-CSCC'93, pp.675-680 (July 1993).

[5] Hwang C.T., Hsu Y.C. and Lin Y.L.: "Scheduling for Functional Pipelining and Loop Winding", Proc. 28th ACM/IEEE Design Automation Conference, pp.764–769 (1991).

[6] Heemstra de Groot S.M., Geres H. and Herrmann O.E.: "Range-Chart-Guided Iterative Data-flow Graph Scheduling", IEEE Trans. Circuits & Syst., **39**, 5, pp.351–364 (May 1992).

[7] Hwang C.T., Hsu Y.C. and Lin Y.L.: "PLS:A Scheduler for Pipeline Synthesis", IEEE Trans. Circuits & Syst., **12**, 9, pp.1279–1286 (Sep. 1993).

[8] Hwang K.S., Casavant E., Chang C.T. and d'Aabreu M.A.:"Scheduling and Hardware Sharing in Pipelined Data Paths", Proc. ICCAD-89, pp.24–27 (1989).

[9] Gebotys C.H. and Elmasry M.I.:"Simultaneous Scheduling and Allocation for Cost Constrained Optimal Architectural Synthesis", Proc. 28th ACM/IEEE Design Automation Conference, pp.2–7 (1991).

[10] Hwang C.T., Lee J.H. and Hsu Y.C.: "A Formal Approach to the Scheduling Problem in High Level Synthesis", IEEE Trans. Comput.-Aided Des. Integrated Circuits & Syst., **10**, 4, pp.464–475 (April 1991).

[11] Gebotys G.H. and Elmasry M.I.:"Global Optimization Approach for Architectural Synthesis", Trans. Comput.-Aided Des. Integrated Circuits & Syst., **12**, 9, pp.1266–1278 (Sep. 1993).

[12] Kung S.Y., Whitehouse H.J. and Kailath T.:"VLSI and Modern Signal-Processing", Englewood Cliffs, NJ: Pretice Hall (1985).

[13] Schwarts D.A. and Barnwell III T.P.:" Cyclostatic Multiprocessor Scheduling for the Optimal Realization of Shift-invariant Flow Graphs ", Proc. ICASSP-85,pp1384–1387(Mar. 1985).