設 計 自 動 化 60－5
（1991．12．12）

動作記述に条件分岐を含むデータパスの高位レベル合成アルゴリズム

大橋 尚子　若林 真一　吉田 典可
広島大学 工学部

本稿では，動作記述に条件分岐を含むデータパスの高位レベル合成に対し，データパスの回路方式がパイプラインの場合と非パイプラインの場合のそれぞれについて合成アルゴリズムを提案する．前者に対しては，ステージ数制約の下での相互排他な演算間のモジュール共有問題において，著者等が既に発表しているモジュール共有グラフを用いた解法をデータパス中に演算連鎖を許した場合に拡張し，シミュレーション実験により評価を行った結果を示す．また，後者に対しては，データパスの実行時間の期待値の最小化を目的とし，データフローを考慮しながら条件分岐による実行径路単位でスケジューリングを行う．

Algorithms for Data Path Synthesis with Conditional Branches

Naoko Ohashi　Shin'ichi Wakabayashi　Noriyoshi Yoshida
Faculty of Engineering, Hiroshima University
4-1, Kagamiyama 1-chome, Higashi-Hiroshima 724, JAPAN

We present synthesis algorithms for pipelined and non-pipelined data paths with conditional branches. For the former, we consider the problem of minimizing data path area under stage constraint by sharing modules among mutually exclusive parts. We've already proposed a method using the module sharing graph under a constraint that each stage has one or less data dependent operation. We extend this algorithm to allow operation chainings, and show experimental results. For the latter, we consider the problem of minimizing the expected excution time. We schedule each path of conditional branches considering data dependencies and relations among paths.

1．まえがき

高位レベル合成におけるデータパス合成では，従来，条件分岐を含まない動作記述中における演算の並列性に注目した最適化手法が多く提案されている[3]．これに対し，動作記述中に条件分岐が含まれている場合，分岐先の演算間の相互排他性を陽に考慮することにより，データパス中の演算器数を少なくし，より効率的な制御シーケンスを生成できる[1][2]．例えば，Wakabayashi等は非パイプライン方式のデータパス合成に対し，与えられたデータフローグラフの中の相互排他な実行経路(パス)を条件ベクタを用いて表し，リストスケジューリングする手法を提案している[4]．また，Camposano等は同じく非パイプライン方式のデータパス合成に対し，コントロールステート数の最小化を目的としたパス単位でスケジューリングを行う手法を提案している[1]．一方，パイプライン方式のデータパス合成に対しては，Hwang等が演算の相互排他性を考慮したスケジューリング手法を提案している[2]．本稿では，動作記述に条件分岐を含むデータパスの高位レベル合成問題に対し，合成の対象となるデータパスの回路方式をパイプライン，及び非パイプラインとした場合のそれぞれについて合成アルゴリズムを提案する．

パイプラインの場合について，与えられたステージ数の制約の下でのモジュール面積の最小化を目的として既に著者等はモジュール共有グラフを用いた解法を提案している[5][6]．従来のHwang等の方法が演算器を共有可能な演算をグリーディに探索するものであるのに対し，著者等の手法は元の問題をモジュール共有グラフ上での経路探索問題に帰着して解いており，精度の良い解が得られるという特長を持つ．著者等の以前のアルゴリズムでは，データ依存関係のある演算は1ステージに1つ以下という制約を与えていたが，本稿ではこれを演算連鎖を許した場合に拡張し，更にシミュレーション実験による結果を示す．

非パイプラインデータパスの場合には，パイプラインの場合と異なり，最も長い実行時間を持つパスにデータパス全体の実行時間は制約されないため，分岐によって各パスはそれぞれ異なる時間で終了可能である．そこで，データパスの実行時間の期待値の最小化を目的とする合成手法について考察する．分岐条件が決定した後は，各実行経路は独立に取り扱えるため，パスを単位としたスケジューリング手法が提案されているが，Camposano等の従来手法[1]では，各パスが直列的である必要があるため，一般の入力に対しては前処理が必要である．本稿では，一般のデータフローグラフを入力としたパス単位のスケジューリング手法について考察する．

2．パイプラインデータパス合成問題

2.1 モジュール共有問題

著者等は文献[5]，[6]において，条件分岐を含む動作記述に対するパイプラインデータパス合成問題に対し，モジュール共有グラフを用いた合成手法を提案している．以下では，文献[5]で定義した合成問題を演算連鎖を許した問題に拡張する．演算連鎖を許した場合のステージ制約付面積最小化問題を以下のように定式化する．

[入力] (1)制御データフローグラフ　CDFG＝(V，E)，
(2)モジュールレパートリ　MR，
(3)最大ステージ遅延　msd，
(4)最大許容ステージ数　maxs．

[出力] パイプラインモジュール割付け．

[制約条件] (1)各ステージの演算実行遅延 ≦ msd，
(2)総ステージ数 ≦ maxs．
(3)パイプラインレーテンシィ ＝ 1．

[目的関数] モジュールの総面積の最小化．

[定義2.1] (制御データフローグラフ)

制御データフローグラフCDFG＝(V，E)はデータパスの動作を表す有向グラフである．節点集合Vは演算節点(○)，条件分岐と合流を表すフォーク(▲)，ジョイン(▼)からなる．演算節点 $v \in V$ は動作記述中の算術論理演算の集合$\{vo_0, vo_1, \cdots, vo_m\}$であり，各演算はスケジューリングにおいて同じステージに割り当てられなければならない．各演算vo_iはそれぞれ算術論理演算子$op(vo_i)$，及び，相互排他性を表す条件ベクタ[4]$vec(vo_i)$を持つ．但し，入力の制御データフローグラフでは1節点に1演算が割り当てられている．2つの演算が1つの演算器を共有することが決定されると，CDFG上の対応する2つの演算節点がマージされる．各枝 $e \in E$ は，その始点から終点までの演算開始

可能時間の差を枝重みベクタwei(e)として持ち，各節点はその出力枝のベクタの中で要素ごとに最大の値を節点重みベクタwei(v)として持つ．フォークとジョインの出力枝のベクタは<0>とする．ここで，<0>は要素がすべて0のベクタを表す．これらの各ベクタはすべて条件ベクタのビット数と同じ数の要素数を持つ．　□

[例2.1] 制御データフローグラフの例を図1に示す．□

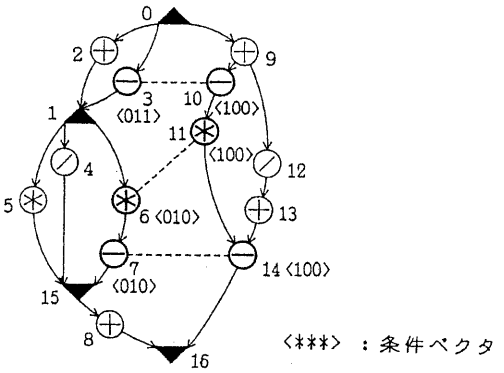

図1 制御データフローグラフ

2.2 ヒューリスティックアルゴリズム

2.1で定義したモジュール共有問題に対するアルゴリズムの概要を以下に示す．

[アルゴリズムの概要]

S1：上のすべての節点間の半順序関係を最短径路アルゴリズムを用いて求める．

S2：CDFG上の各節点の割り当てられる最小ステージをベクタを用いたASAPにより決定する．

S3：モジュール共有グラフを構成し，グラフ上の適当な径路の選択により，モジュール共有演算子対を決定する．

S4：ステージ制約を満たしているかどうかチェックする．制約を満たしていなければ満たすまで適当なモジュール共有演算子対を除去する．

S5：CDFGの縮約を行う．

S6：S1～S5をモジュール共有グラフが構成されなくなるまで繰り返す．

2.2.1 モジュール共有グラフ（S3）

ここでは，演算連鎖を取り扱うために，モジュール共有グラフ[5]の定義を少し拡張する．まず，使用する記号を説明する．uを始点，vを終点とするCDFG上の部分グラフを$G[u,v]=(V',E')$と表す．また，2節点u，vの実行可能開始時間の差を距離ベクタdis(u,v)とする．

[定義2.2] (モジュール共有グラフ)

モジュール共有グラフMSG=(P,Q,w,d)は制御データフローグラフCDFG=(V,E)とモジュールレパートリMRより構成される閉路を持たない重み付き有向グラフである．P，Q，w，dは以下の様に定義される．

(1) 節点集合$P=\{p_0(=p_s), p_1, \cdots, p_\alpha, p_{\alpha+1}(=p_t)\}$の各節点は制御データフローグラフ上のモジュール共有可能演算子対を示す．節点pとなるCDFG上の演算子対$uo_i \in u$と$vo_j \in v$は以下の①，②の条件を必ず満たし，③または④を満たす．u，v∈Vである．

①$op(uo_i)=op(vo_j)$．但し，u＝vも可．

②$vec(uo_i) \wedge vec(v_j)=<0>$．

③u↛v，かつ，v↛u．

④u→v(またはv→u)の時は，$G[u,v]=(V',E')$上でMAX_E(dis(u,v)+wei(v))≦msd．ここで，MAX_Eはベクタの要素の最大値とする．

但し，→はCDFG上の2節点間に有向道が存在することを，↛は存在しないことを示す．

(2) $Q=\{q_1,q_2,\cdots,q_\beta\} \subseteq P \times P$は有向枝の集合であり，各枝$q_k=(p_k,p_l)\in Q$は，次の①，②の条件を満たすときに存在する．ここで，$p_k=(uo_{k,i},vo_{k,j})$，$p_l=(uo_{l,i},vo_{l,j})$，$uo_{k,i}\in u_k$，$vo_{k,j}\in v_k$，$uo_{l,i}\in u_l$，$vo_{l,j}\in v_l$である．

①$G[u_k,v_k] \cap G[u_l,v_l]=\phi$．

②の各節点u_k，u_l，v_k，$v_l \in V$は，ui→uj，かつ，vi→vj，または，ui→vj，かつ，vi→ujのどちらかが成立する．

(3) wはP→N(自然数の集合)の関数である．w(p)は節点に付けられた重みであり，pの示す演算子対がモジュールを共有するときの減少面積を示す．

(4) dはQ→Nの関数である．$d((p_i,p_j))$は枝の重みであり，p_i，p_jに対応する演算子対がモジュール

を共有するときのステージ数の増加分を表す．

モジュール共有グラフ上で，枝の重みの総和をステージ制約以下とし，節点の重みの総和を最大とするパスを選ぶことにより，モジュールを共有する演算を決定することができるが，枝の重みに関して，モジュール共有グラフの推移性が崩されている場合があるので，その場合は，ステップ4でパスのバックトラックを行う．

2.2.2 制御データフローグラフの縮約（S5）

ステップ3で演算子対$uo_i \in u$，$vo_j \in v$によって演算の共有が決まったとき，部分グラフG[u,v]=(V',E')を1節点に縮約し，これを新しい節点newとする．newは次のような節点となる．

①new$=\cup_{x \in V'} x$．このとき，uo_iとvo_jは1要素$neo_k \in$ newとなり，$vec(neo_k)=vec(uo_i) \vee vec(vo_j)$である．

②節点newの入力枝，出力枝は，それぞれe_{in}=[u',v'], u'∈V', v'∈V', e_{out}=[u',v'], u'∈V', v'∈V'となる枝をすべて含む．$\forall e_{in}$=[u',v']を縮約した新たな枝e_{in}=[u',new]の重みwei(e_{in})は，縮約以前の重みを保持する．$\forall e_{out}$=[u',v']を縮約した新たな枝e_{out}= [new,v']の重みwei(e_{out})は(dis(u,u')＊vec([u',v'])＋wei([u',v']))とする．ここで，vec(e)は枝e=[u,v]の重みwei(e)のベクタの各要素が0以上なら1，0ならば0となる0－1のベクタであるとする．

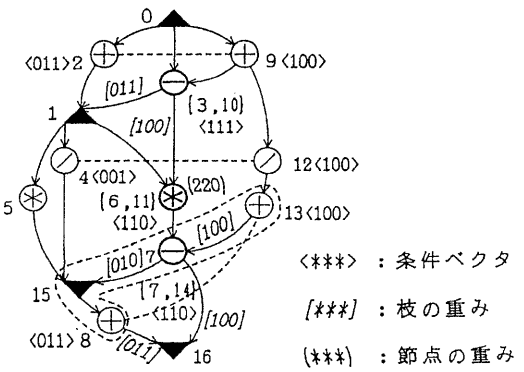

図2 縮約したデータフローグラフ

[例2.2] 図1を縮約したCDFGを図2に示す．アルゴリズムの次のステップでは点線で囲まれた部分グラフが1点に縮約される． □

2.3 実験結果

本手法をSPARCstation2上でC言語を用いて実現した．実験データを表1に示す．制御データフローグラフは人手により作成した動作記述をトランスレータにより変換して用いている．msdを3，及び4とした場合の実験結果を表2に示す．表中のアスタリスク(＊)は計算途中で打ち切ったことを表している．実験結果より，最適解と比較して多くの場合2％増程度の解を1秒程度の計算時間で求めていることがわかる．但し，まれに非常に精度の悪い解を求めることがあるため，その制御のためのパラメータを新たに導入する予定である．

表1 制御ﾃﾞｰﾀﾌﾛｰｸﾞﾗﾌ

ﾃﾞｰﾀ No.	節点数	最大面積	ﾃﾞｰﾀ No.	節点数	最大面積
0	22	26	5	32	139
1	24	78	6	36	120
2	29	100	7	40	120
3	30	90	8	41	147
4	32	112	9	52	232

表2 実験結果

msd	ﾃﾞｰﾀ No.	最小ｽﾃｰｼﾞ	本手法 面積	本手法 時間(s)	分枝限定法 面積	分枝限定法 時間 (s)
3	0	4	66	0.1	63	1.7
	1	4	52	0.1	48	0.3
	2	6	61	0.2	61	0.4
	3	4	49	0.2	44	0.3
	4	6	60	0.2	60	1.1
	5	7	84	0.3	84	9874.4
	6	3	75	0.5	75	289.4
	7	6	63	0.6	63	1.0
	8	7	79	0.7	79	4784.7
	9	9	161	1.4	158＊	10.3(h)
4	0	3	63	0.1	59	2.6
	1	2	48	0.1	43	0.3
	2	4	61	0.2	61	1.0
	3	3	49	0.2	44	0.3
	4	3	58	0.2	58	0.5
	5	5	84	0.3	78	8118.6
	6	3	75	0.4	75	4027.1
	7	5	63	0.7	63	2.4
	8	6	98	0.7	76	40329.0
	9	8	161	1.5	158＊	4.2(h)

3．非パイプラインデータパス合成問題

3.1 スケジューリング問題の定式化

非パイプラインデータパスの実行時間の期待値を最小化を目的とするスケジューリング問題を以下のように定式化する．

[入力] (1)制御フローグラフ　CFG＝(V_C, E_C),
(2)モジュールレパートリ　MR,
(3)条件分岐における実行確率(optional).

[出力] スケジューリングS．

[制約条件] $m\_nu(S, mo_i) \leqq max\_a(mo_i)$　$1 \leqq i \leqq m$,

[目的関数] データパスの実行時間の期待値の最小化．

但し，モジュールレパートリMRには，データパス中に使用可能なモジュール$\{mo_1, mo_2, \cdots, mo_m\}$とそれぞれの個数$max_a(mo_i)$ ($1 \leqq i \leqq m$)が与えられているものとする．また，$m_nu(S, mo_i)$はスケジューリングSにおけるモジュールmo_iの個数であるとする．簡単のため，演算連鎖は考えない．

[定義3.1] (制御フローグラフ)

制御フローグラフCFG＝(V_C, E_C)は入力動作記述より，条件分岐による処理の並列性を示した有向グラフであるとする．節点集合は$V_C = V_{op} \cup F \cup J \cup \{start, end\}$と表される．$V_{op}$とFの各節点は算術論理演算，代入といった動作記述中の1つの処理に対応し，特にFは条件分岐を表す．それぞれ演算節点，条件節点と呼ぶ．Jは分岐の合流点を，また，start，endはそれぞれグラフの始点，及び，終点を表し，データパス中の処理は含まないダミーの節点である．Jは合流節点と呼ぶ．有向枝は入力記述の順序に従って付けられており，V_{op}は入力枝，及び出力枝を1本づつ，Fは2本以上の出力枝を，また，Jは2本以上の入力枝を持つ．　□

[例3.1] 制御フローグラフの例を図3に示す．　□

制御フローグラフ上で始点startから，終点endまでのすべてのパスの集合を$P = \{p_1, p_2, \cdots, p_s\}$とする．各パスはある条件分岐によって実行される演算の系列を示している．パスp_iの実行される確率を$prob(p_i)$とする．$\Sigma prob(pi) = 1$である．スケジューリングSにおける各パスの実行遅延時間を$p_delay(S, p_i)$と表すとき，実行遅延時間の期待値は$e_delay(S) = \Sigma prob(p_i) * p_delay(S, p_i)$である．

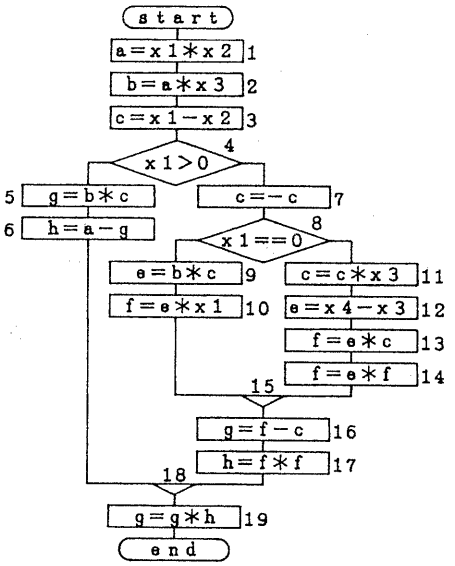

図3 制御フローグラフ

3.2 ヒューリスティックアルゴリズム

以下に，提案するヒューリスティックアルゴリズムの概要を示す．

S1：制御フローグラフより，始点startから終点endまでのすべてのパスを求める．また，各パスに1ビットを割り当てたパスベクタを各節点に付ける．

S2：各パスごとにデータフローグラフを構成する．

S3：パスごとにデータ枝のみを用いてスケジューリングする．実行確率の高いものから，必要なコントロールステップ数の多いものの辞書式順序でパスをソーティングする．

S4：ソーティングの順序に従って，パスをコントロールステートにマッピングする．

ステップ1については3.1の制御フローグラフの定義により簡単に導かれるので，ステップ2以降について具体的に説明する．

3.2.1 データフローの導出（S2）

パスを終点endよりトラバースして，各パスごとにデータフローグラフを作成する．データフローグラフは

パス上の節点をすべて持つ．枝はデータ依存関係を表すデータ枝，及び，制御の流れを表す制御枝の２種類が存在する．条件節点，及び，合流節点から出る枝は制御枝のみである．図３の制御フローグラフに対する各パスのデータフローグラフを図４に示す．この図では，合流節点は省略してある．

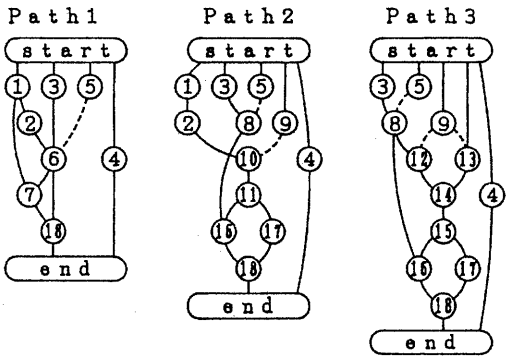

図４ データフローグラフ

3.2.2 パスごとのスケジューリング（Ｓ３）

各パスごとに面積制約を満たし，必要なコントロールステップ数ができるだけ少なくなるようにリストスケジューリングする．その後，ステップ数を増加させることなくパスベクタの同じ節点同士を同じステップに割り当てられるようにする．

１）各ステップごとに節点のパスベクタの排他的論理和をとり，ビット値が１の個数をsim(i)とする．ｉはステップの番号．

２）Σsim(i)の値を小さくするように節点の割り当てを変更する．

各ステップに含まれる節点のパスベクタは同じ値にしておく．実行確率の高いパスから，ステップ数の多いパスの順番にソーティングする．

3.2.3 コントロールステートへのマッピング（Ｓ４）

ステップ４でソートされた順序に従い，演算使用ベクタ[4]を用いて各パスをステートにマッピングする．ステート数を抑えるため，同じ節点番号の分岐節点は必ず同じステートに割り当て，同じ節点番号の演算節点はできるかぎり同じステートに割り当てるとする．また，既に構成されているステートでは数が不足する場合は，新しいステートを加える．

１）既にステートの決まっている分岐節点があれば，そのステップはステートを決定する．但し，演算節点でモジュール制約を満たさないものがあれば，その演算を別のステージへ移動する．別のステージへの移動不可能ならステージ数を増やす．

２）１つのステージに含まれているすべての演算を１ステートに割り当てするような最大のマッチングを求める．

３）割り当てできなかったステージに対し，新しいステートを割り当てる．

3.2.4 演算の繰り上げ実行

Ｓ３のパス単位のスケジューリングにおいて，CFGにおいて条件分岐の後続節点であった演算を条件分岐に先行させて実行させることによりパスの実行時間が短縮できる場合がある．このような演算のスケジューリングを演算の繰り上げ実行という．あるパスで演算を繰り上げて実行すると，その後にスケジューリングされるパスの実行時間を増大させる可能性がある．そこで，演算の繰り上げ実行により実行時間が短縮できるパスに対し，繰り上げ実行するスケジューリングとしないスケジューリングの２つのスケジューリングを行い，以降それぞれについてＳ３を繰り返し，求まったスケジューリングの中で実行時間の期待値の最小のものをアルゴリズムの出力とする．但し，すべての組み合わせを行うと計算時間が膨大になる場合があるので，あらかじめユーザによって指定された数のパスだけ２種類のスケジューリングを行うものとする．ユーザの指定したパス数以上のパスに対しては，ユーザのあらかじめ与えた指定により常に繰り上げ実行を伴うスケジューリングか，または伴わないスケジューリングを行うものとする．

４．あとがき

条件分岐を含むデータパス合成において，回路方式がパイプラインの場合，モジュール共有グラフに基づくモジュール共有アルゴリズムをデータパスが演算連鎖を許す場合に拡張した．シミュレーション実験の結果より，多くの場合，高速に良い近似解を求めることがわかった．しかし，入力データによって，解の精度

にばらつきが見られるため，ばらつきを抑えるためのヒューリスティックの導入が必要であると思われる．また，非パイプラインの場合，データパスの実行時間の期待値を最小化する問題に対し，データフローを考慮したパス単位のスケジューリングを行うヒューリスティックアルゴリズムを提案した．この手法では，実行確率の高いパスから優先的にスケジュールするため，実行時間の期待値の最小化が容易であると予測できる．但し，データパスの制御のためのステート数が文献[4]のような手法と比較して増大する可能性があり，現在，計算機実験による評価を行っている．

謝辞

日頃，ご指導を賜ります広島大学総合科学部助教授宮尾淳一先生に深謝致します．

文献

[1] R.Camposano: "Path-based scheduling for synthesis," IEEE Trans. Computer-Aided Design, Vol.10, pp.85-93, January (1991).

[2] K.S.Hwang et al.: "Constrained conditional resource sharing in pipeline synthesis," Proc. ICCAD-88, pp.52-55 (1988).

[3] M.C.McFaland et al.: "The high-level synthesis of digital systems," Proc. IEEE, vol. 78, pp.301-318, February (1990).

[4] K.Wakabayashi and T.Yoshimura: "A resource sharing and control synthesis method for conditional branches," Proc. ICCAD-89, pp.62-65 (1989).

[5] 若林，他：" 動作記述に条件分岐を含むパイプラインデータパスの高位レベル合成手法"，信学技報，VLD91-10 (1991).

[6] K.Yokoyama et al.: "Optimum resource sharing in pipeline synthesis," Proc. Int. Workshop on Discrete Algorithms and Complexity, pp.47-54 (1989).