社団法人　情報処理学会　研究報告
IPSJ SIG Technical Reports

2004－SLDM－117（33）
2004／12／2

非同期式パイプライン制御回路の論理合成法

大西　陽三†　　籠谷　裕人††　　杉山　裕二††　　岡本　卓爾†††

† 岡山大学大学院 自然科学研究科 〒700-8530 岡山県岡山市津島中 1-1-1
†† 岡山大学工学部通信ネットワーク工学科 〒700-8530 岡山県岡山市津島中 1-1-1
††† 岡山理科大学工学部電子工学科 〒700-0005 岡山県岡山市理大町 1-1
E-mail: †yozo@infsys.cne.okayama-u.ac.jp, ††{kagotani,sugiyama}@cne.okayama-u.ac.jp,
†††okamoto@ee.ous.ac.jp

あらまし　本論文では，依存性グラフと依存関係の組で与えた仕様から，非同期式プロセッサのパイプライン制御回路を合成する手法を提案している．この手法により，依存性グラフと依存関係にしたがって各ノードに対応する制御モジュールを適宜ハンドシェーク接続するだけで制御回路を実現させることができる．特に，制御の流れを稼働相と休止相に分け，それらを並列に動作させることにより，パイプラインの高速化を図っている．また，この手法により，従来の手法よりもハードウェア量が小さくなると考えられる．
キーワード　非同期式回路，論理合成，パイプライン化，依存性グラフ

A Synthesis Method of Control Circuits for Pipelined Asynchronous Prosessors

Yozo ONISHI†, Hiroto KAGOTANI††, Yuji SUGIYAMA††, and Takuji OKAMOTO†††

† Graduate School of Natural Science and Technology, Okayama University
†† Dept. of Communication Eng., Okayama University
††† Dept. of Electronics Eng., Okayama University of Science, Japan
E-mail: †yozo@infsys.cne.okayama-u.ac.jp, ††{kagotani,sugiyama}@cne.okayama-u.ac.jp,
†††okamoto@ee.ous.ac.jp

Abstract　We propose a synthesis method of control circuits for pipelined asynchronous processors from specifications given as a dependency graph and dependencies. This method gives control circuits by connecting handshaking control modules corresponding to graph nodes according to the edges of the graph and dependencies. These control modules are designed to achieve efficient pipelines by parallelizing working and idle phases, which are composing a micro-operation. This method can synthesize smaller hardware than a conventional one.
Key words　Asynchronous circuit, Logic synthesis, Pipeline synthesis, Dependency graph

1. はじめに

非同期式プロセッサのパイプライン (以下，単にパイプラインという) は，同期式プロセッサの場合と同様に，処理速度，特にスループットを向上させる上で極めて重要であり，これまでに多くの報告がなされている [1]～[4]．しかし，パイプラインを組織的に実現しようとする方法は，筆者らの知る限り Teifel らの方法 [5] と籠谷らの方法 [6] の 2 つしか提案されていない．

Teifel らの方法は，データフローグラフを利用して組織的にパイプラインを実現しようという方法であり，そのための制御回路はデータパスとともに自動的に設計されることになる．しかし，この方法では，同一の処理が重複して多数回出現する場合，別々の処理として扱われるので，回路合成したときのハードウェア量，特にレジスタ数が多くなり過ぎる傾向にある．これに対して籠谷らの方法は，依存性グラフ [7], [8] を利用して，パイプラインの処理手順を合理的に決定・表現しようとする方法であり，同一処理が別の処理として表現されるようなことは原理上起こり得ない．しかし，条件分岐を含むパイプラインの依存性グラフ作成法についてはまだ開発途上であり，また，依存性グラフからの制御回路の合成法については全く手がつけられていない．

本研究の目的は，籠谷らの方法において，依存性グラフが

作成済みであるとしたときのパイプラインのための制御回路(以下，パイプライン制御回路という) 合成法を与えることにある．ここでは，依存性グラフにおけるノードの種類毎に制御モジュールを構成している．これらのモジュールでは特に相互間の信号授受や制御モジュールとデータパスとの信号授受を4相ハンドシェークにより行っている．また，データパスの動作を実質的な演算処理を行う稼働相と回路の状態を初期化する休止相とに分けて，ノード間に依存関係がない限り，休止相実行中でも後続の処理を開始することで，パイプラインの高速化を図っている．全体の制御回路は，依存性グラフのノードと有向辺にしたがってこれらの制御モジュールを単に接続するだけで実現される．

以下，2. ではパイプライン化のための仕様の表現方法と制御回路の合成方針を述べる．続いて 3. では各制御モジュールの回路構成法について述べる．最後に 4. では依存関係が存在するときの回路補正方法と制御回路の合成例について述べる．

2. パイプラインの仕様と制御回路の合成方針

2.1 パイプラインの仕様の表現法

パイプラインの仕様は依存性グラフにより与えられる．このグラフは5種類のノード (基本操作ノード，フォークノード，ジョインノード，セレクトノード，マージノード)，有向辺，および初期トークンから構成される．基本操作ノードは変数間の代入や代入時の演算などの基本操作 (micro-operation) を表し，フォークノードおよびジョインノードはそれぞれ並列処理の開始およびその終了を表し，セレクトノードおよびマージノードはそれぞれ条件分岐の開始およびその終了を表す．処理の流れは有向辺に沿ったトークンの移動により示されるが，初期トークンはその開始位置を表す．

仕様の例を図1に示す．これは，あるRISCプロセッサからJMP(ジャンプ) 命令とLOAD (読み出し) 命令の実行部のみ抽出した仮想的な命令セットプロセッサの仕様であり，次のような処理を繰り返し実行することを表している．基本操作 A および B ではそれぞれ，命令のフェッチおよびプログラムカウンタのインクリメントを行い，それに続く左のループでは，JMP命令であればプログラムカウンタをジャンプ先アドレスに更新したのち，LOAD命令であればそのまま，次の回の A, B の処理に進む．また，右のループでは，基本操作 C および D においてそれぞれソースレジスタのアドレスの指定およびデスティネーションフィールドの値を読み込みを行い，さらに S_2 で分岐したのち次の回の A の処理に進むとともに，LOAD命令であればデスティネーションレジスタにソースレジスタで指定されたアドレスの値を読み込んだのちに，JMP命令であればそのまま，次の回の C, D 処理に移る．特に，基本操作 F と次回の基本操作 A や，基本操作 C, D, F およびセレクト S_2 と次回の基本操作 B がそれぞれパイプライン化されている．

点線および破線は，依存性グラフの構成要素ではないが，ここではそれぞれ始点の基本操作の実行結果を格納したレジスタの値が終点の基本操作の実行結果に更新されるような関係である Write-after-Write 関係 (WAW 関係) および始点の基本操作

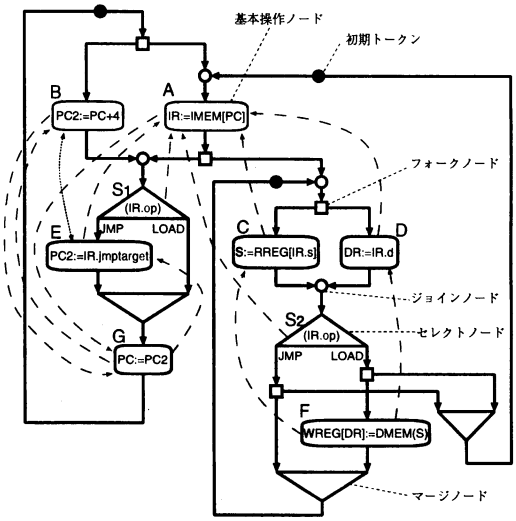

図1 パイプライン化依存性グラフの例

で参照されたレジスタの値が終点の基本操作の処理結果に更新されるような関係である Write-after-Read 関係 (WAR 関係) を示す．以下，これらの関係のことを依存関係という．これらは後章で述べるように，処理効率のよいのよい制御回路を合成するために仕様として必要である．

2.2 パイプライン制御のモデルとその構成方針

非同期式プロセッサのハードウェアは，同期式の場合と同様にデータパス部と制御部に分けることができるが，処理のパイプライン化はこのうちの制御部の構成により決まる．そこで本論文では，制御部をデータパス部から分離し，一括してパイプライン制御回路として取り扱うものとする．また，基本操作におけるデータパス部と制御部との間の通信は，処理の開始を指令する要求信号 (Req) とその終了を伝える応答信号 (Ack) とのハンドシェークにより行なわれるものとする．

次に，素子遅延について述べる．本論文で対象とするパイプライン制御回路は，非同期式回路であるから，設計時においては素子遅延に対する観点が必要である．ここではそのための仮定として，ゲートも配線も有限であるが上界の未知な遅延があるする Delay-Insensitive モデルに等時分岐 (Isochronic-Fork) の仮定を加えた Quasi-Delay-Insensitive(QDI) モデル [9] を採用する．

一つの基本操作実行のための通信の例を図2に示す．$Req = 1$ となると演算を開始し，これが終了すると $Ack = 1$ となるが，次の処理の演算を行えるようにするためには Req, Ack ともに0にリセットすることが必要である．ここでは，Req が1となってから Ack が1となるまでの期間を稼働相 (working phase)，Req が0となってから Ack が0となるまでの期間を休止相 (idle phase) と呼ぶ．

データパス部とパイプライン制御回路全体の入出力関係を図3に示す．データパス内で実行される各基本操作と制御回路との間は一対の $Req - Ack$ 線により結ばれている．汎用性のある

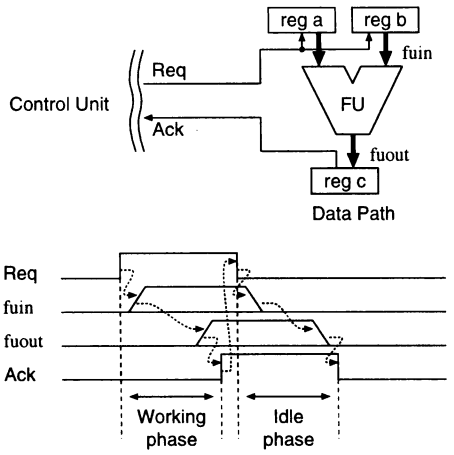

図 2　基本操作の実行

制御回路を得るための一つの方法は，すでに示唆したように依存性グラフのノードの種類ごとに上位からの処理開始指令をハンドシェークで受信し，さらに処理終了後の下位への処理開始指令もハンドシェークにより送出するようにすることである．ここではこのような方針のもとでさらにデータパス部での処理開始を極力早めるために，依存関係がない限り，稼働相が終了し次第下位のノードの処理を開始することにする．

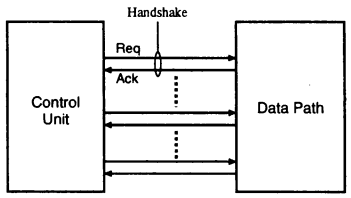

図 3　制御回路とデータパス回路間のハンドシェーク

3. 制御モジュールの設計

本論文では，すでに述べたように，依存性グラフの構成要素を対応する制御モジュールに直接写像することによりパイプライン制御回路を合成する．ここではそのための手段として，依存性グラフ上の同種のノードごとに同一の制御モジュール (初期トークンに対しても制御モジュールを用意する) を用意し，グラフのノードに対応して配置し，かつ有向辺にしたがって相互に接続することにより制御回路を構成する．特に本節の設計では，簡単化のために 2. で述べた依存関係が存在しないものとして制御モジュールを設計し，これが存在するときの扱いは 4.1 で述べる．

3.1　基本操作ノードに対応する制御モジュール

基本操作ノードに対応する制御モジュール (基本操作モジュール) に要求されるの入出力構造を図 4(a) に示す．(xi, xo), (yi, yo) および (ai, ao) は，それぞれ上流のモジュール，下流のモジュールおよびデータパス部に接続される入出力対である．基本操作モジュールに要求される機能を信号遷移図 (STG) で示すと，同図 (b) のようになる．信号に付した + はその信号の 0 → 1 変化を，− はその信号の 1 → 0 変化を表す．矢印はその終点の信号遷移が起こるために始点の信号遷移が完了しなければならないことを表す．黒丸はトークンの初期位置を表す．すべてのトークンが初期位置から移動して，稼働相・休止相のすべてが完了すると基本操作の処理が終了し，再び初期位置に戻る．

基本操作モジュールの回路構成例を図 5 に示す．C_1, C_2 は Muller の C 素子である．これは直感に基づいて構成したので以下のようにして正当性を確かめた．図 6 は QDI モデルのもとで，General Single Winner(GSW) 法 [10] により求めた図 5 の回路の状態図である．状態は信号線 (xi, xo, ai, ao, yi, yo) の値の組で表されている．0,1 に付したアンダーバーは，その状態において変化し得る信号を表し，矢印に付加した数字は遷移後の状態が遷移前の状態の左から何番目の信号変化により生起したかを表す．図中の最上部の点線で囲んだ状態が初期状態である．この状態図は図 4(b) の STG を満たしている．

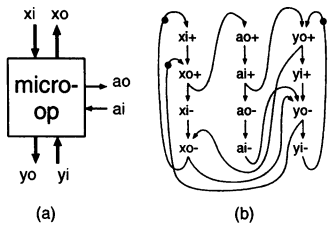

図 4　基本操作モジュールの入出力構造とその信号遷移図

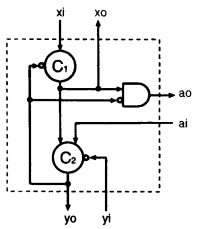

図 5　基本操作モジュールの回路構成例

3.2　フォークノードとジョインノードの制御モジュール

フォークノードおよびジョインノードに対応する制御モジュール (フォークモジュールおよびジョインモジュール) は，データパスにおいて演算処理を制御する基本操作モジュールとは異なり，基本操作モジュール間の接続を行うだけである．このために必要なフォークモジュールの入出力構造を図 7(a) に示す．(xi, xo) は上流のモジュールに接続される入出力対，(yi, yo),(zi, zo) は下流のモジュールに接続される 2 組の入出力対である．これに要求される機能を信号遷移図で示すと，図 7(b) のようになる．また，この信号遷移図を満たす回路構成例を図 8 に示す．

ジョインモジュールも図 9(b) の信号遷移図から容易に図 9(c) のように構成される．

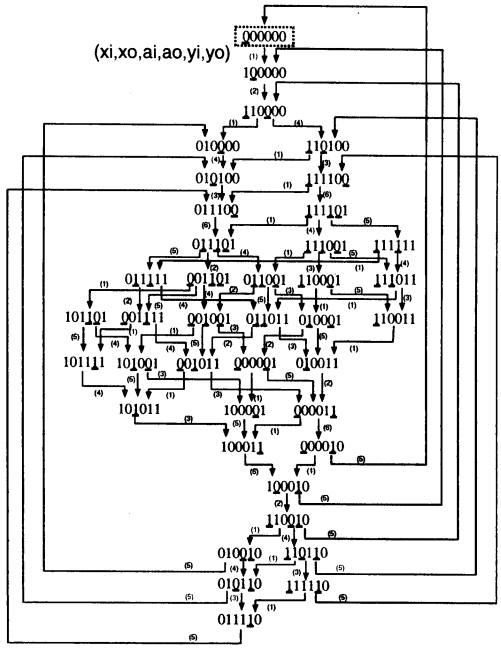

図 6　基本操作モジュールの状態遷移図

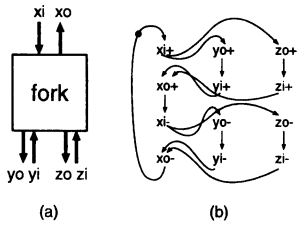

図 7　フォークモジュールの入出力構造とその信号遷移図

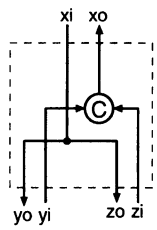

図 8　フォークモジュールの回路構成例

3.3　セレクトノードとマージノードの制御モジュール

セレクトノードの機能は，処理の選択実行のための制御の分岐である．したがって，これに対応するモジュール (セレクトモジュール) は，分岐条件の演算処理部 (基本操作と同じ) との間で信号のやりとりを行うことが必要である．これに対して，マージノードの機能は，フォークモジュールやジョインモジュールと同様に基本操作モジュール間を接続するだけである．

セレクトモジュールの入出力構造を図 10(a) に示す．(xi, xo) は上流のモジュールに接続される入出力対，(yi, yo),(zi, zo) は

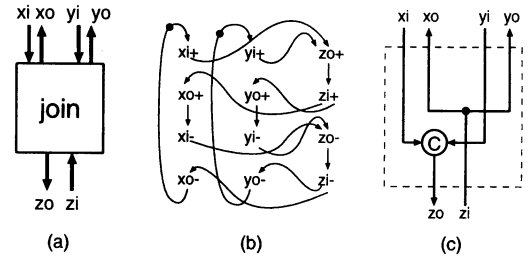

図 9　ジョインモジュールの入出力構造，信号遷移図および回路構成例

下流のモジュールに接続される 2 組の入出力対，(ai, ao) はデータパス部に接続される入出力対である．また，これとは別にデータパス部から送られる条件判定結果を示す入力 b をもつ．この制御モジュールに要求される機能を信号遷移図で表すと，図 10(b) のようになる．図中の白丸はペトリネットのプレースに対応する．その信号遷移図を満たす制御モジュールの回路構成例を図 11 に示す．この制御モジュールには，上述した分岐条件発生のためのデータパスを制御するために基本操作モジュールと同一構成の制御モジュールが内蔵させてある．この制御のもとで決定された分岐条件により上流の制御モジュールが下流の右または左の制御モジュールに接続される．

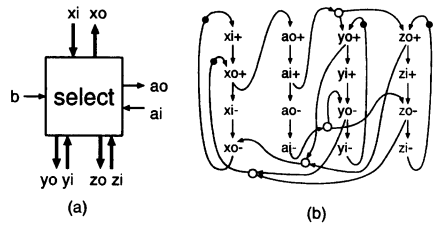

図 10　セレクトモジュールの入出力構造とその信号遷移図

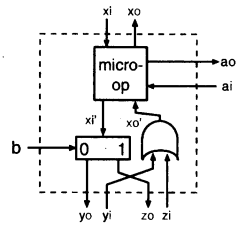

図 11　セレクトモジュールの回路構成例

マージノードに対応する制御モジュール (マージモジュール) の入出力構造および状態遷移図をそれぞれ図 12(a) および 12(b) に示す．(xi, xo), (yi, yo) は上流のモジュールに接続される 2 組の入出力対，(zi, zo) は下流のモジュールに接続される入出力対である．この状態遷移図を満たす回路構成例を図 13 に示す．特に，上流の制御モジュールからの 2 つの入力 xi, yi がラッチを介して下位に伝達されるようにしたのは，一方の上流モジュールと下流のモジュールとがマージノードを介してハンドシェー

ク信号のやりとりを行っている間に上流の他方のモジュールから処理開始指令が到来したときの競合により発生する誤動作を回避するためである．この回路が状態遷移図の機能を満たしていることは明らかである．

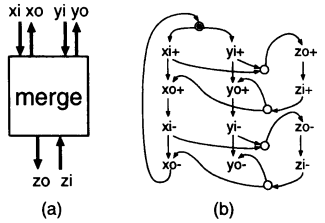

図 12　マージモジュールの入出力構造とその信号遷移図

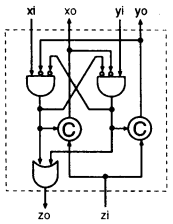

図 13　マージモジュールの回路構成例

3.4　初期トークンに対応する制御モジュール

初期トークンを生成するためのモジュール (トークンモジュール) の入出力構造，状態遷移図および回路構成例をそれぞれ図 14(a)，(b) および (c) に示す．

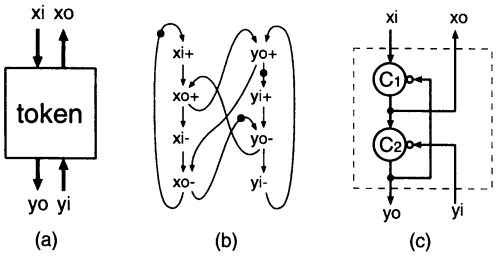

図 14　トークンモジュールの入出力構造，信号遷移図および回路構成例

4.　パイプライン制御回路の合成法

4.1　依存関係のある基本操作間での制約

3. で述べた基本操作モジュールでは，処理速度向上のため，そのモジュールの基本操作の休止相と後続の基本操作モジュールでの稼働相が並列に実行される．このため，基本操作間に 2. で示した依存関係が存在する場合にそのまま動作したとすると，レジスタの値の整合性が保証されず誤動作が起こりうる．そこで以下本節では若干の回路を付加してこの問題を解決する方法について述べる．

(a)　WAW 関係の場合

図 15(a) に示すように基本操作 A, B 間に WAW 関係がある場合，演算回路の遅延によっては基本操作 A によるレジスタ b への入力データが初期状態に戻る前に，基本操作 B による入力データと衝突する可能性がある．この誤作動を回避するためには，基本操作 A の休止相の終了を確認するまで基本操作 B の稼働相を実行しないように並列性を制限すればよい．ここではこの制御を図 15(b) のようにして実現する．すなわち，基本操作 A の休止相が完了すると，データパスからの応答信号 (ai) が 0 になるので，基本操作 B の AND ゲートの出力が 1 となり，基本操作 B の稼働相が実行可能となる．また，基本操作 B で更新したレジスタ b も基本操作 A の次回の実行によって更新されるため，同様に AND ゲートで制御する必要がある．このように AND ゲートを付加したとしても，一方の信号線 ao の信号変化が他方の制御モジュールの信号線 ai が 0 となるまで遅れる可能性があるだけで，図 6 の状態遷移に変化はない．

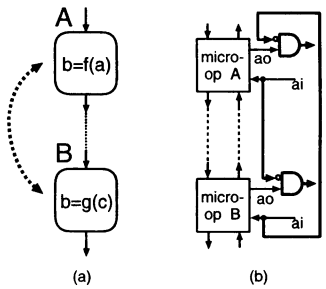

図 15　WAW 関係が存在する場合の回路補正

(b)　WAR 関係の場合

図 16(a) に示すように基本操作間に WAR 関係がある場合，演算回路の遅延によっては基本操作 A によるレジスタ a への入力データが，初期状態に戻る前に基本操作 C による入力データと衝突する可能性がある．この誤作動を回避するために基本操作 A の休止相の開始を確認するまで基本操作 C の稼働相を実行しないように並列性を制限すればよい．ただし，更新されるレジスタが初期状態になるまで下流の基本操作の稼働相を開始することができない WAW 関係の場合とは異なり，基本操作 A で参照するレジスタ a が初期状態になった時点で基本操作 C の稼働相を開始することができる．また，基本操作 C が稼働相にあるときに，下流の処理がつぎつぎと進んでいき，再び基本操作 A の稼働相が開始されると，基本操作 C が稼働相の途中でも休止相が開始されてしまい，誤動作が起こりうる．ここでは，これらの制御を図 16(b) のようにして実現する．すなわち，基本操作 A の休止相が開始されると，データパスからの応答信号 (ao) が 0 になるので，基本操作 C の AND ゲートの出力が 1 となり，基本操作 C の稼働相が実行可能となる．また，基本操作 C が稼働相であるときに基本操作 A の稼働相が開始しないように，同様に AND ゲートで制御する必要がある．このように AND ゲートを付加したとしても，一方の信号線 ao の信号変化が他方の制御モジュールのデータパスへの入力が 0 とな

るまで遅れる可能性があるだけで，基本操作は図 6 の状態遷移にしたがって実行される．

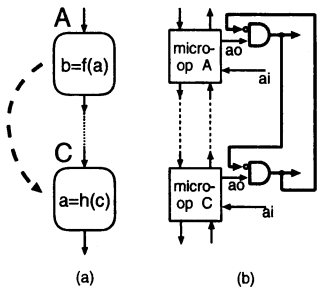

図 16　WAR 関係が存在する場合の回路補正

4.2　制御回路の合成法

これまでに述べた合成手法を，図 1 の依存性グラフに適用した．図 17 にその合成後のパイプライン制御回路を示す．四角は基本操作モジュールであり，その内部に書かれた A, B などの記号は依存性グラフのどの処理に対応するかを表す．ただし基本操作モジュールと接続されるデータパスの構造は省略している．セレクトノードの制御モジュール内のデマルチプレクサへの入力は，分岐先を決定する分岐信号の値が書き込まれたレジスタからの入力となる．特に，依存関係がある場合の付加回路および信号線は太線で表している．

この回路の正当性は，当研究室で開発した verilogHDL シミュレータにより確認した．このシミュレータは回路の構成要素の遅延時間を固定しているが，現実的なモデルとしては十分実用的であると考えられる．

5.　む　す　び

本論文では，依存性グラフと依存関係との組で与えた仕様から，ノードの種類毎に制御モジュールを設計し，依存性グラフと依存関係にしたがってこれらを配置接続することにより，任意のパイプライン制御回路を合成することができることを示した．

ここで示したモジュールは単なる一構成例であり，対応する信号遷移図を満たす限り，どのように構成しても制御回路の動作は保証される．また，条件分岐とその合流に対応するセレクトモジュールとマージモジュールについては，2 分岐とその合流のみを対象としているが，必要なら 3 分岐以上に接続することは容易であろう．

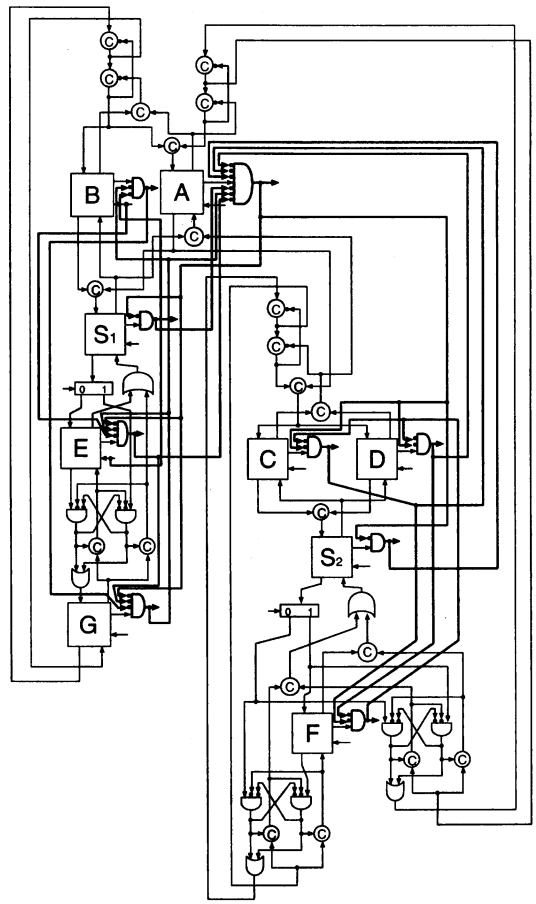

図 17　パイプライン制御回路の合成例

文　献

[1] Ivan E. Sutherland: “Micropipelines.” Communications of the ACM, Vol.32, No.6, pp.720-738, (1989)

[2] K.Y.Yun, P.A.Beerel, J.Arceo: “High-performance asynchronous pipeline circuits.” In Proc. International Symposium on Advanced Reserch in Asynchronous Circuits and Systems. IEEE Computer Society Press, (1996)

[3] D.A.Gilbert and J.D.Garside: “A result forwarding mechanism for asynchronous pipelined systems.” In Proc. International Symposium on Advanced Reserch in Asynchronous Circuits and Systems. pp.2-11. IEEE Computer Society Press, (1997)

[4] Sam S.Appleton, Shannon V.Morton, and Michael J.Liebelt. “Two-phase asynchronous pipeline control. In Proc. International Symposium on Advanced Reserch in Asynchronous Circuits and Systems. pp.12-21. IEEE Computer Society Press, (1997)

[5] J. Teifel and R. Manohar: “Static tokens: Using dataflow to automate concurrent pipeline synthesis”, Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems, IEEE Computer Society Press, pp.17-27, (2004).

[6] 籠谷裕人，杉山裕二，岡本卓爾: “非同期式プロセッサのパイプラインアルゴリズム -条件分岐のない場合-”, VLD2004-33, (2004)

[7] T.Nanya, Y.Ueno, H.Kagotani, M.Kuwako and A.Takamura: “TITAC:Design of a quasi-delay-insensitive microprosessor”, IEEE Design & Test of Computers, 11, 2, pp.50-63, (1994).

[8] H. Kagotani and T. Nanya: “A synthesis method of quasi-delay-insensitive prosessors based on dependency graph”, Asia-Pacific Conference on Hardware Description Languages (APCHDL), pp.177-184, (1994)

[9] Scott Hauck: “Asynchronous design methdologies” An overview. Processing of IEEE, Vol.83, No1, pp.69-93, (1995).

[10] Brzozowsk, J.A. and Yoeli,M.: “Gigital Networks”, Prentice-Hall, (1786)