計測自動制御学会東北支部 第188回研究集会 (2000.6.23)
資料番号 188-6

転送ボトルネックフリーVLSIシステムのハイレベルシンセシス

High-Level Synthesis of Bottleneck Free VLSI Systems

○工藤隆男*，亀山充隆**

○Takao Kudoh*, Michitaka Kameyama**

*八戸高専, **東北大学　大学院　情報科学研究科

*Hachinohe National college of Technology,
*Graduate School of Information Sciences, Tohoku University

キーワード： 並列構造VLSIプロセッサ(parallel VLSI processor)，転送ボトルネックフリーアーキテクチャ(bottleneck free architecture)，ハイレベルシンセシス (high level synthesis)，ロジックインメモリ構造 (logic-in-memory architecture)，分枝限定法 (branch and bound method)

連絡先： 〒039-1192　八戸市田面木字上野平16-1 八戸工業高等専門学校　電気工学科 工藤隆男
Tel.: (0178)27-7279，Fax.: (0178)27-7279，E-mail: tkudoh-e@hachinohe-ct.ac.jp

1. はじめに

並列構造VLSIプロセッサを設計する場合，メモリ部から演算部への転送ボトルネックのないデータ転送が重要である．転送ボトルネックを解消する手法として，筆者等は演算機能と記憶機能を融合させたロジックインメモリVLSIアーキテクチャに基づくプロセッサを提案しており[1, 2, 3]，この一般的設計法の開発が望まれる．この有効な一方法として，ハイレベルシンセシスが挙げられる．ハイレベルシンセシスは，与えられたディジタルシステムの動作仕様に対して，論理最適化や下位レベルの仕様を生成する設計技術である．

本稿では１個の演算器と数ワードの記憶機能からなるロジックインメモリ構造のモジュールを処理要素（PRE）とし，隣接したPREを接続したハードウェアモデルに対して，データ依存グラフで与えられた動作仕様を，転送時間と演算時間を含めた最小の処理時間で実行可能なスケジューリングとアロケーションを決定する方法を提案する．与えられたデータ依存グラフに対して，演算器のアロケーションとスケジューリングは互いに影響し合うため、スケジューリングとアロケーションを統合した設計が必要になる。そこで，スケジューリングとアロケーションの膨大な組合せに対して、分枝限定法に基づき最適解を探索する方法を提案している．最後に、データ依存グラフで与えられた動作仕様に対し，最適なスケジューリングとアロケーションを行った場合の処理時間が，大幅に短縮できることを示す．

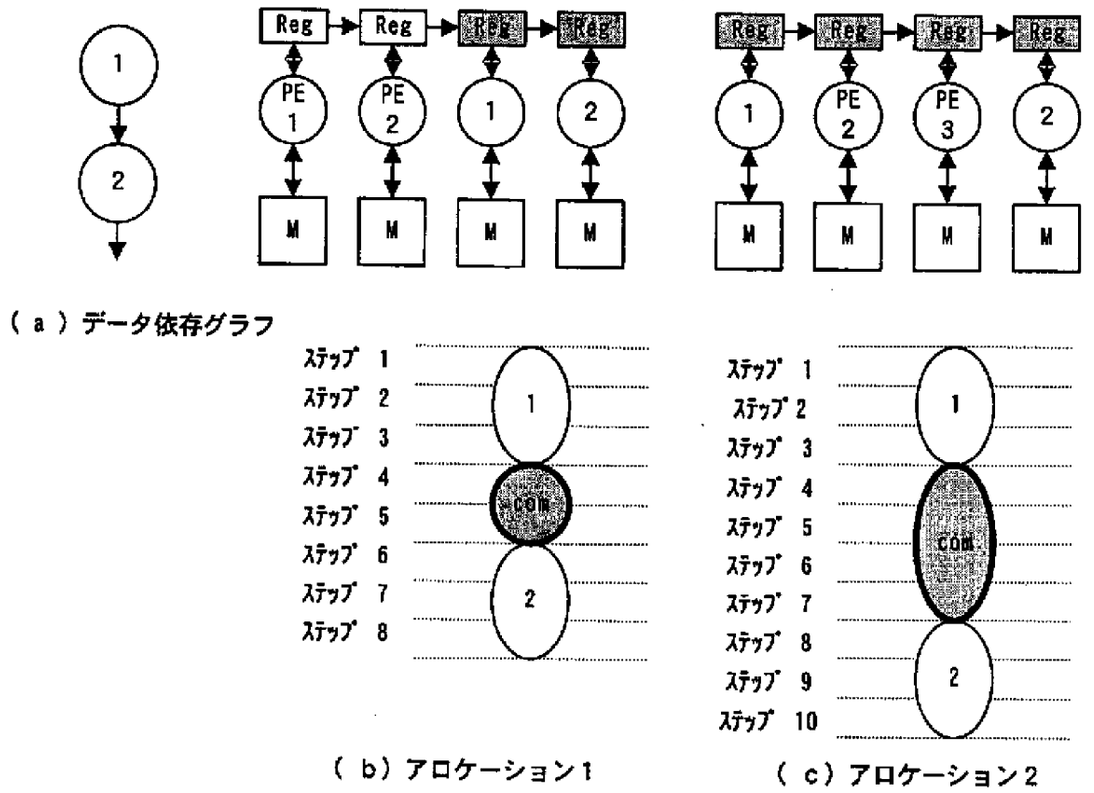


Fig. 2　アロケーションの違いによるスケジューリングへの影響

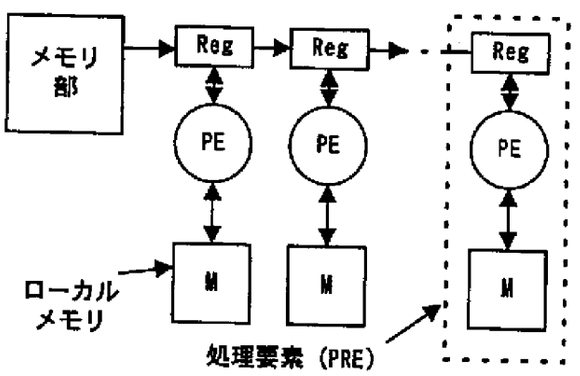


Fig. 1　ハードウェアモデル

2.　アロケーションとスケジューリングの統合の必要性

Fig. 1のように、1個の演算器（PE）と1ワードのレジスタと小規模のメモリモジュール（M）からなる処理要素（PRE）を用意し，隣接したPREのレジスタどうしを直接接続したハードウェアモデルにおいて、演算器のアロケーションとスケジューリングは互いに影響し合うことについて考える．各PEにおける演算の種類は同じで、PE間の転送は、隣接するPREに対して1方向のみ可能とする。PEの演算に必要なステップ数はTop（＞1）ステップとし，転送は1ステップの間に隣接したPREへ実行できるものとする。まず，アロケーションがスケジューリングへ影響することについて考える．たとえばFig. 2 (a)のデータ依存グラフに対して（b）のように隣接する2つのPEをアロケーションする場合と、（c）のように離れた2つのPEをアロケーションする場合とでは，異なるスケジューリングが得られる．次に，スケジューリングがアロケーションへ影響することについて考える．たとえば，Fig.2(a)のデータ依存グラフにおいて2つのノード間の転送時間を1ステップだけに限定すると，隣接する2つのPE以外にはアロケーション不可能である．このように，アロケーションとスケジューリングは互いに影響し合うことから、データ依存グラフで与えられた処理を最小ステップ数で実行できる並列構造VLSIプロセッサを設計するためには，アロケーションとスケジューリングを統合した設計が必要になる．

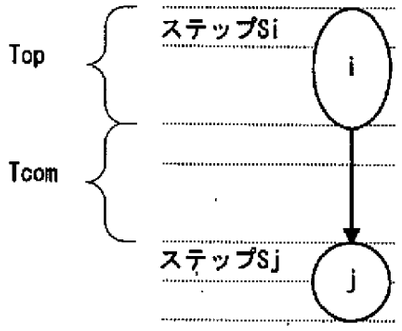


Fig. 3　ノードjのスケジューリングにおける制約条件

3. アロケーションとスケジューリングを統合した設計法

アロケーションとスケジューリングの可能な組合せを統合した設計問題を木探索問題に帰着させることについて考える．まず，スケジューリング可能な場合の数について考える．ノードi,jにFig. 3のようなデータ依存関係があり、ノードiがSiにスケジューリングされているとき、iの演算時間をTopステップ，iからjへの転送時間をTcomとすると、jのスケジューリング可能なステップSｊは、式（1）で表すことができる。

$$Si+Top+Tcom-1 \leqq Sj \qquad (1)$$

次に，ノードiへのアロケーションの可能なPEの数について考える．ノードiがアロケーションを必要としているときm個のPEがアロケーション可能であれば，ノードiへのアロケーション可能なPE数はm個である．　よって，ノードiに対してアロケーション可能なPEの個数がaiであり，スケジューリング可能なステップの個数がbiであるとき，ノードiのアロケーションとスケジューリングの可能な組み合わせ数をCiとおくと，Ciは式（2）で表すことができる．

$$Ci = ai \times bi \qquad (2)$$

以上のことから，データ依存グラフ全体のノード数がmのとき，データ依存グラフのアロケーションとスケジューリングの組合せ数Nは，式（3）で表すことができる．

$$N = C1 \times C2 \times \cdots\cdots Cm \qquad (3)$$

式（3）で表されるN個の可能な組合せを木で表現する．データ依存グラフのあるノードiに着目する．ノードiの演算器アロケーションを行い，さらにあるステップを割り当てる，すなわちノードiにつきCi個ある節の中から１個の節を選ぶことを，ノード１から最終ノードまでについて行ったとき，１個の可能解はFig.4(a)のようにノード１から最終ノードまでの枝の連なりで表すことができる．ここでノードiに相当する節に１－２と記されている場合は，ノードiに１番のPEがアロケーションされ，そのスケジューリングは第２ステップであることを示す．可能解の探索問題はFig. 4（b）のようにN本の枝を持つ木の探索問題に帰着できる．この木に対し，深さ優先探索で，根から葉まで到達すると、根と葉およびその間に挟まれた節の組み合わせが可能解となる．

3.1　最適解の探索問題

最適解とは，データ依存グラフで与えられた演算を最小ステップ数で実行できるときのアロケーションとスケジューリングの組み合わせをいうことにする．最適解を決定するためには，Fig. 4（b）のN個のすべての可能な組合せ（これを可能解と呼ぶことにする）の中から最適解を見つけられればよい．しかし，可能解の個数は演算ノード数が多くなると爆発的に多くなることから，すべての可能解をしらみつぶし的に調べることは実用的ではない．そこで，可能な限り最適解を持つ枝だけを探索して，等価的にすべての枝を探索したことになる探索方法が望まれる．

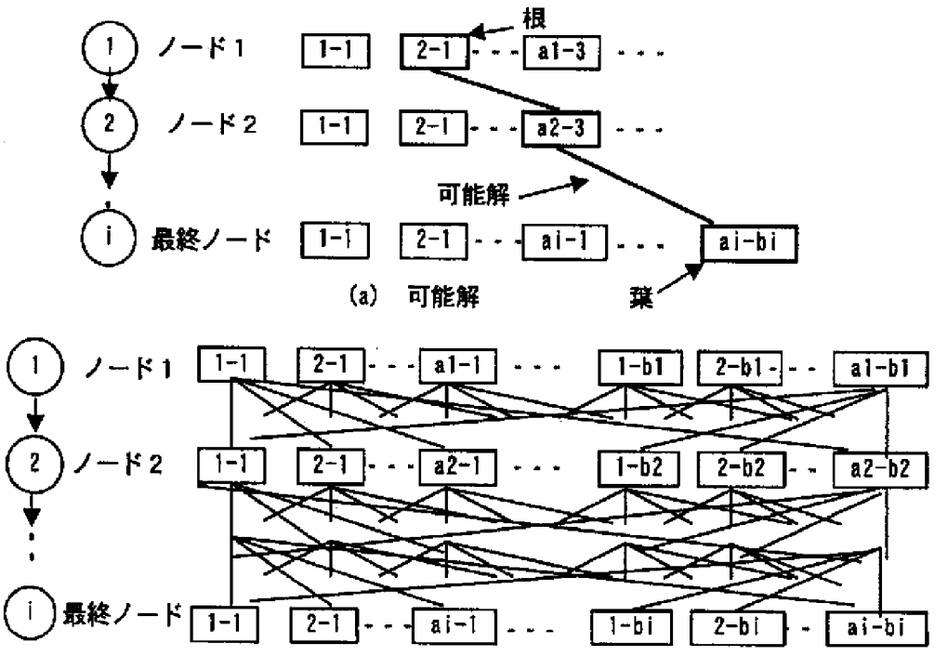


Fig. 4 アロケーションとスケジューリングの組合せと可能解

3.2 分枝限定法に基づく最適解の探索

探索途中で最適解が含まれないことが明らかになった枝を刈り取ることができれば，刈り取った枝も等価的に探索したことになることから，最適解を持つ可能性のある枝を探索するだけで，すべての可能解を探索したと見なすことができ,効率的な解の探索が可能になる．

枝に最適解がないことを評価するための目的関数を用い，最適解のない枝を刈り取ることにより，等価的にすべての枝を探索する方法に分枝限定法[4]がある．今、目的関数として探索途中の枝の最小ステップ数の見積もりを導入する。最小ステップ数の見積もりをDで表す。Dは，探索の途中でノードiへのアロケーションとスケジューリングが決定されるとき，式（4）で表すことができる．

$$D = Si + Dimin \qquad (4)$$

式（4）の関係をFig.5に示す．Dを求めることについて考える．根からノードiまでのステップ数はノードiがスケジューリングされたステップSiに等しいことから，Dを求めるためにはノードi以降の最小ステップ数Diminの見積もりができればよい．そこで，あらかじめデータ依存グラフのすべてのノードについて最終ノードまでの最小ステップ数の見積もりDiminをALAP[5]（as last as possible）に基づき求めておくことにする．ALAPに基づくDiminの決定は次のようにする．たとえば，Fig.6のデータ依存グラフのスケジューリングは，各ノードの演算に3ステップ，ノード間の転送に1ステップ必要であるとき，(a)や(b)の可能性があり，（b）は可能な限り各ノードを遅いステップにスケジューリングしたALAPの場合である．(b)

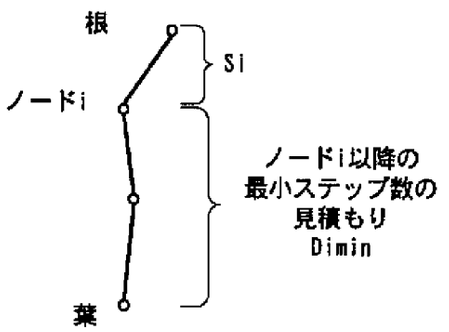
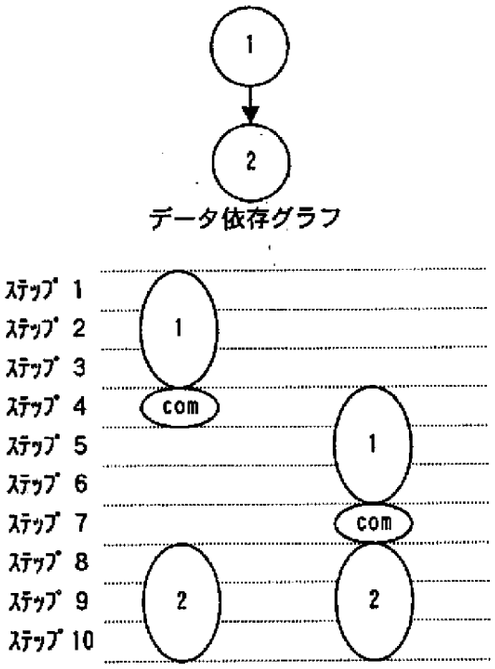


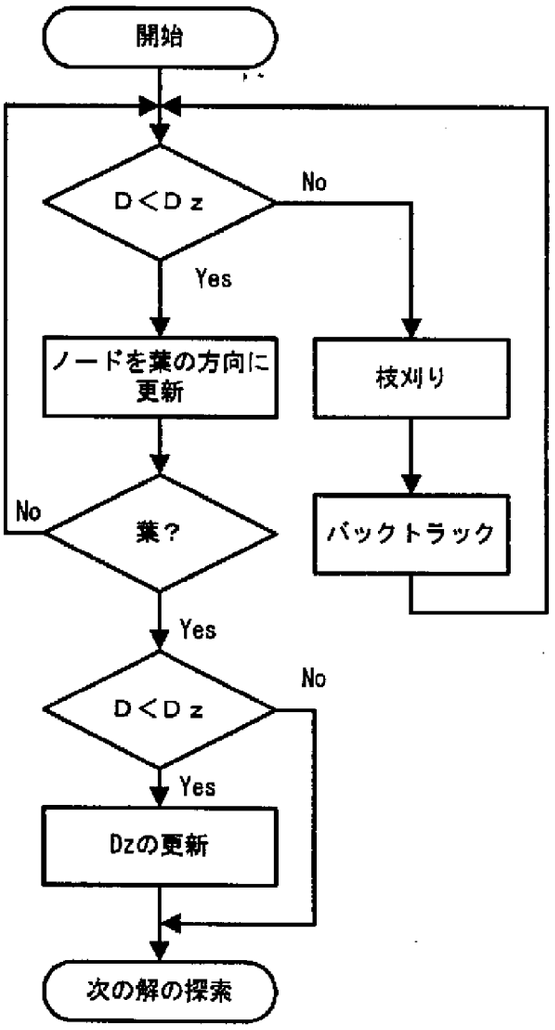
Fig. 5 最小演ステップ数の見積もり



(b)から，ノード1のクリティカルパスは
7ステップと見積もることができる

Fig. 6 ALAPに基づく最小ステップ数の見積もり

においてはノード1がスケジューリングされてからノード2の演算が終了するまで7ステップ必要である．このとき，ノード1のD1minは7ステップである．Diminはノードiに相当する節から葉までの最小ステップ数の見積もりであり、どのようにアロケーションしてもこれ以上小さくすることはできない．よって，可能解の最小ステップ数Dの見積もりは，式（4）のようにすでに決定されたステップSiとDiminの和で表すことができることから，実際に得られる可能解のステップ数はD以上になる．よって，木全体の探索の途中において，暫定的な最小ステップ数Dzを持つ解が発見されているとき，D＜Dzの場合は，新しい最適解を発見できる可能性があるので，探索中の枝について探索を続行する．逆に，D≧Dzの場合は，新しい最適解を発見できる可能性は無いので，その先の枝を刈り取り，バックトラックする． 探索の手順をFig.7に示す．この手順によると，刈り取られないで残る枝に含まれる可能解のステップ数は



Dz：暫定的な最小ステップ数
D：可能解の最小ステップ数の見積もり

Fig. 7 解の探索手順

暫定的な最小ステップ数より必ず小さいか等しいことから，少なくとも最後に発見された解が最適解になる。また、可能な限り最適解が含まれる枝だけを探索し，等価的にすべての枝を探索できたことになる．

4. 評価

提案する設計法の有用性を評価するために，Fig.8(a)のような5×5のデータに3×3のウィンドー処理をする並列VLSIプロセッサの設計を例題とする．ウィンドー処理は(b)のようにウィンドーの中心にあるデータと定数の加算をし，これと周辺のデータと比較した結果のOR演算をとるとき，ウィンドー処

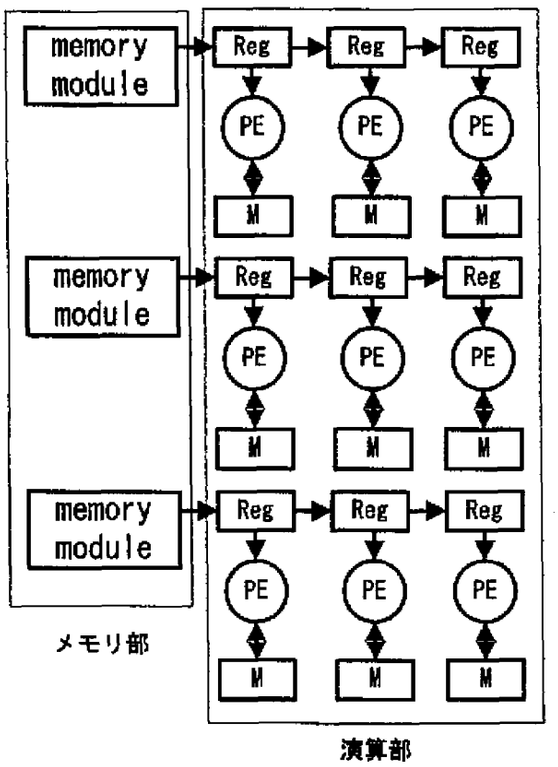
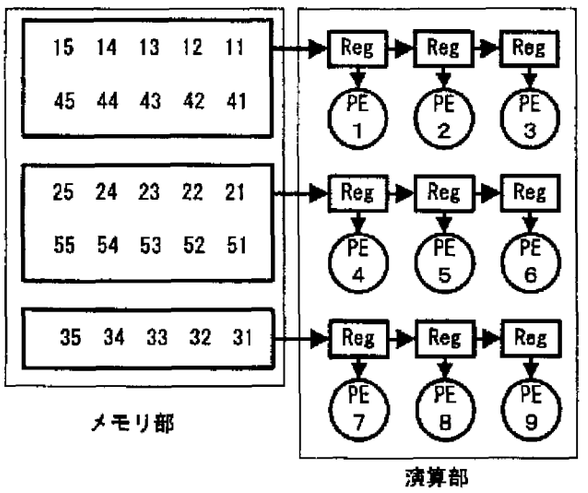


Fig. 9　評価に用いたハードウェアモデル

理のデータ依存関係は入力データのみに依存する．よって，データ依存関係を(c)のように略記できることから，(a)の全データに対するウィンドー処理のデータ依存関係は，(d)のようにウィンドー処理毎の９個の演算ノードの集合の集まりとして表すことができる．ハードウェアモデルは，ウィンドーの形状からFig.9にする．

メモリモジュールへのデータの記憶について考える，並列アクセスを可能とするために，Fig.8(a)の１行目のデータ(11,12,13,14,15)と，２行目のデータ(21,22,23,24,25)と３行目のデータ(31,32,33,34,35)は，異なる３つのメモリモジュールへアロケーションする．４行目のデータは2行目と3行目のデータと同時にアクセスできることが必要であるが，１行目のデータと同時にアクセスされる必要はないので，１行目のデータがアロケーションされたメモリモジュールにアロケーションする．このように，常に連続する３行のデータへの並列アクセスを可能とするためには，データはFig.9のようにアロケーションすればよい．



(a)　ハードウェア構成

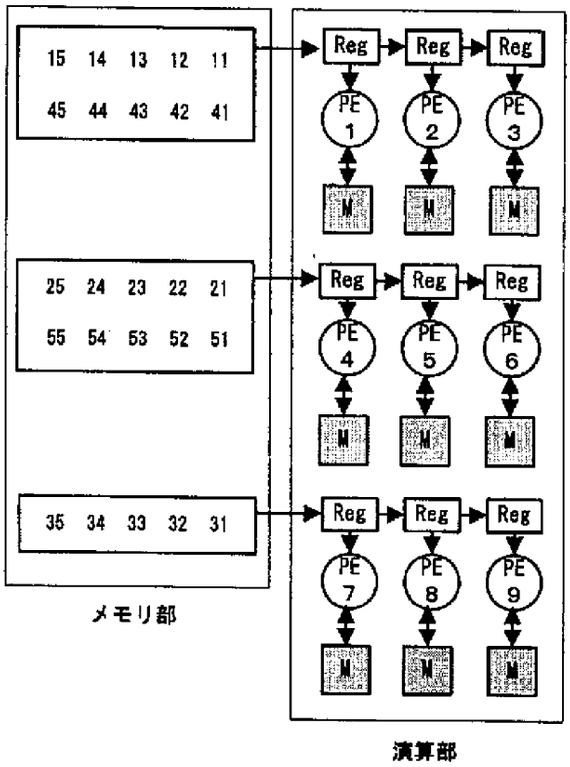
	PE1	PE2	PE3	PE4	PE5	PE6	PE7	PE8	PE9
ｽﾃｯﾌﾟ1	11			21			31		
ｽﾃｯﾌﾟ2	12	11		22	21		32	31	
ｽﾃｯﾌﾟ3	13	12	11	23	22	21	33	32	31
ｽﾃｯﾌﾟ4	14	13	12	24	23	22	34	33	32
ｽﾃｯﾌﾟ5	15	14	13	25	24	23	35	34	33
ｽﾃｯﾌﾟ6	41			21			31		
ｽﾃｯﾌﾟ7	42	41		22	21		32	31	
ｽﾃｯﾌﾟ8	43	42	41	23	22	21	33	32	31
ｽﾃｯﾌﾟ9	44	43	42	24	23	22	34	33	32
ｽﾃｯﾌﾟ10	45	44	43	25	24	23	35	34	33
ｽﾃｯﾌﾟ11	41			51			31		
ｽﾃｯﾌﾟ12	42	41		52	51		32	31	
ｽﾃｯﾌﾟ13	43	42	41	53	52	51	33	32	31
ｽﾃｯﾌﾟ14	44	43	42	54	53	52	34	33	32
ｽﾃｯﾌﾟ15	45	44	43	55	54	53	35	34	33

(b)　最小ステップの転送と演算

Fig. 10　最適なアロケーションとスケジューリング

９個のノードにFig.9のPEをアロケーションする時，ウィンドー処理を最小ステップで実行できるアロケーションとスケジューリングを決定する．

木の探索途中における最小ステップの見積もりについて考える．探索途中までに必要としたステップ数に，これから探索するノード集合において必要と見込まれる最小ステップの見積もりを加えればよい．すなわち，現在探索しているノード集合iから次のノード集合jに演算器をアロケーションすることは，ウィンドーを移動することに等しい．ウィンドーの移動には，少なくとも１ステップ必



ローカルメモリを必要

(a) ハードウェア構成

	PE1	PE2	PE3	PE4	PE5	PE6	PE7	PE8	PE9
ｽﾃｯﾌﾟ1	11			21			31		
ｽﾃｯﾌﾟ2	12	11		22	21		32	31	
ｽﾃｯﾌﾟ3	13	12	11	23	22	21	33	32	31
ｽﾃｯﾌﾟ4	[13]	[12]		24	23	22	34	33	32
ｽﾃｯﾌﾟ5	[13]	[12]		[24]	[23]	[22]	[34]	[33]	[32]
ｽﾃｯﾌﾟ6	13	12	14	24	23	22	34	33	32
ｽﾃｯﾌﾟ7	[13]		[14]	[25]	[24]	[23]	[35]	[34]	[33]
ｽﾃｯﾌﾟ8	13	15	14	25	24	23	35	34	33
ｽﾃｯﾌﾟ9	41			21			31		
ｽﾃｯﾌﾟ10	[41]			22	21		32	31	
ｽﾃｯﾌﾟ11	[41]	[42]		23	22	21	33	32	31
ｽﾃｯﾌﾟ12	[41]	[42]		[23]	[22]	[21]	[33]	[32]	[31]
ｽﾃｯﾌﾟ13	41	42	43	23	22	21	33	32	31
ｽﾃｯﾌﾟ14	44								
ｽﾃｯﾌﾟ15	43	44							
ｽﾃｯﾌﾟ16	42	43	44	24	23	22	34	33	32
ｽﾃｯﾌﾟ17	45			25	24	23	34	33	32
ｽﾃｯﾌﾟ18	44	45		[25]	[24]	[23]	[34]	[33]	[32]
ｽﾃｯﾌﾟ19	43	44	45	25	24	23	35	34	33
ｽﾃｯﾌﾟ20	41			51			31		
ｽﾃｯﾌﾟ21	42	41		52	51		32	31	
ｽﾃｯﾌﾟ22	43	42	41	53	52	51	33	32	31
ｽﾃｯﾌﾟ23	44	43	42	54	53	52	34	33	32
ｽﾃｯﾌﾟ24	45	44	43	55	54	53	35	34	33

[n] ：ローカルメモリに記憶

多くの転送ののステップを必要

(b) 転送と演算

Fig. 11 最適ではないアロケーションとスケジューリング

要であることから，ノード集合jに演算器をアロケーションするためには少なくとも1ステップ必要であると見込むことができる．よって，これから探索するノード集合に対し必要と見込まれる最小ステップ数は，未だ探索していないノード集合の総数に等しいと見込むことができる．このことから，現在探索途中の可能解の最小ステップ数の見積もりは，探索途中までに必要としたステップ数に，未だ探索していないノード集合の総数を加えることで見積もることができる．

最小ステップ数で転送と演算を実現できるアロケーションとスケジューリンをFig.10に，最小ステップ数でないアロケーションとスケジューリングをFig.11に示す．最適なアロケーションとスケジューリングをすることにより，転送と演算を合わせた総ステップ数つまり処理時間を大幅に減少でき，さらに各PREのローカルメモリを省略できる．

Fig.10はすべての可能解を探索して得られた解に等価であることから，これ以下のステップ数で実行できるアロケーションとスケジューリングの組合せは存在しない，すなわちFig.10は最適解であることを保証できる．

5. むすび

ハードウェアモデルとデータ依存グラフが与えられるとき，演算時間と転送時間を合わせた処理時間全体が最小となるアロケーションとスケジューリングの決定問題を最適解の探索問題に帰着させた．この設計法を用いると，ハードウェアモデルが与えられているとき，データ依存グラフで定義される処理を最小の時間で実行可能なアロケーションとスケジューリングを決定できる．但し，データ依存グラフのノード数が多い場合は，分枝限定法を用いても実用的な時間内での探索が不可能になるという問題が残されている．これに対してはISM法[6]により，あらかじめノード数が多いデー

タ依存グラフをデータ依存関係の弱い部分から切り分けたデータ依存グラフについて，このハイレベルシンセシスを適用することが考えられるが，これについてはあらためて検討したい．

このアロケーションとスケジューリングとを統合したハイレベルシンセシスの考え方は，ロジックインメモリVLSIプロセッサに限らず、演算器のアロケーションとスケジューリングが相互に影響する一般のプロセッサのハイレベルシンセシスにも適用可能である．

参考文献

1) 工藤隆男，羽生貴弘，亀山充隆，"ロジックインメモリアーキテクチャに基づく道路抽出VLSIプロセッサとその応用"，電気関係学会東北支部連合大会講演予稿集，2H12，1998.

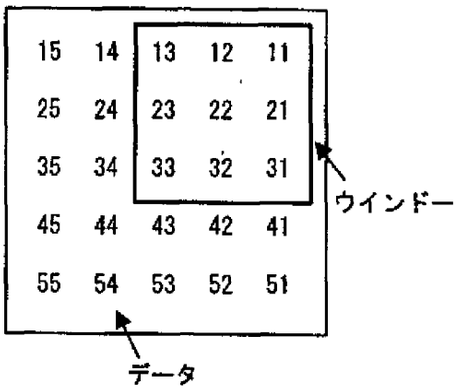
2) 堀井崇史，羽生貴弘，亀山充隆，"共通バス本数最小化に着目したロジックインメモリVLSIシステムの設計"，電気関係学会東北支部連合大会講演予稿集，2H17，1998.

3) 張山昌論，季昇桓，亀山充隆，"転送ボトルネックのないセンサ・メモリアーキテクチャに基づくモーションステレオVLSIプロセッサの構成"，電学論E，120巻5号，p.237-244，2000.

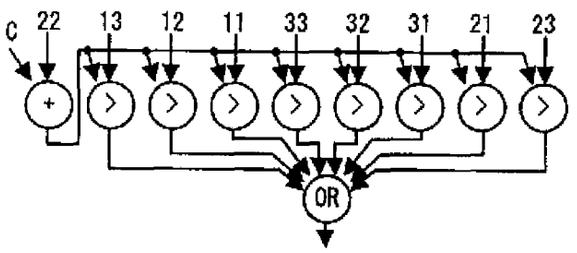
4) たとえば石畑清,"アルゴリズムとデータ構造",岩波書店

5) Daniel Gajski,Nikil Dutt,Allen Wu,Steve Lin,"HIGH-LEVEL SYNTHESIS Introdeution to Chip and System Design",Kluwer Academic Publishers

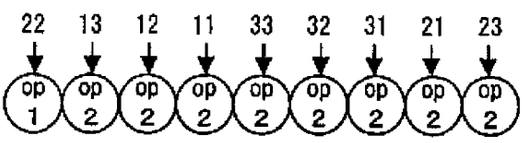
6) J.N.Warfield,"Binary Matrices in System Modeling", IEEE Trans.Syst.,Man, Cybern.,vol.SMC-3,No.5,Sept. 1973.



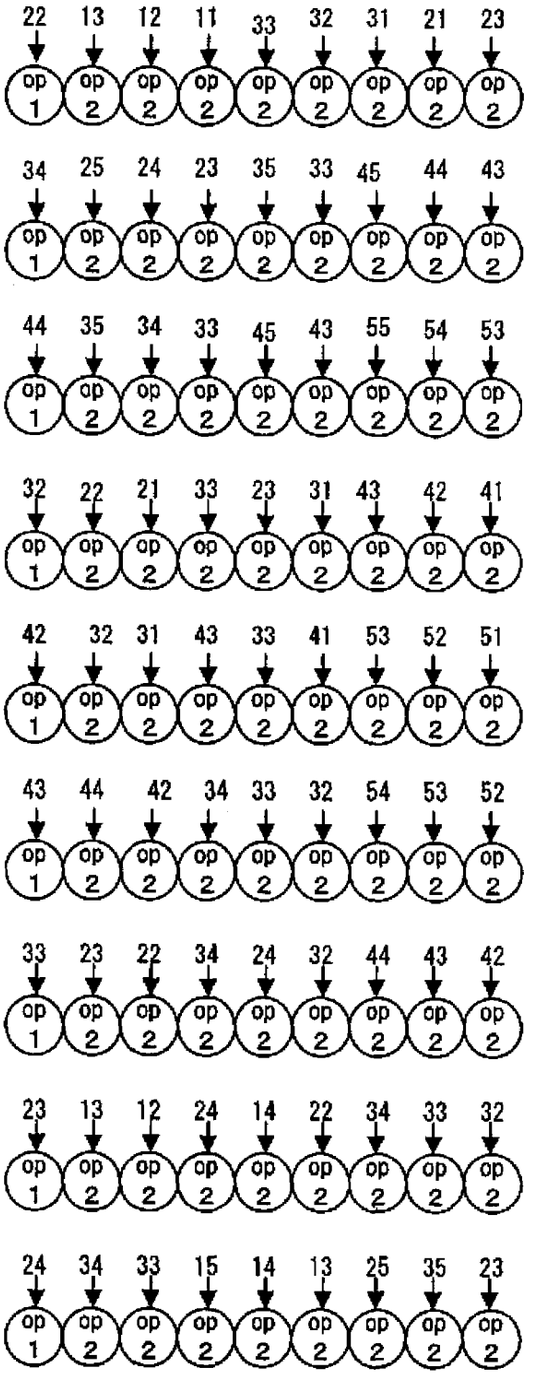
(a) データとウインドー



(b) (a)のウインドーにおける
データ依存関係



(c) (b)の簡略化した表現



(d) (a)全体のデータ依存関係

Fig. 8 評価