

シフトレジスタ構造に基づく ロジックインメモリVLSIプロセッサとその応用

Shift Register Based Logic-in-Memory VLSI processor and Its Application

○工藤隆男*, 亀山充隆**

○Takao Kudoh*, Michitaka Kameyama**

*八戸高専, **東北大学情報科学研究科

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

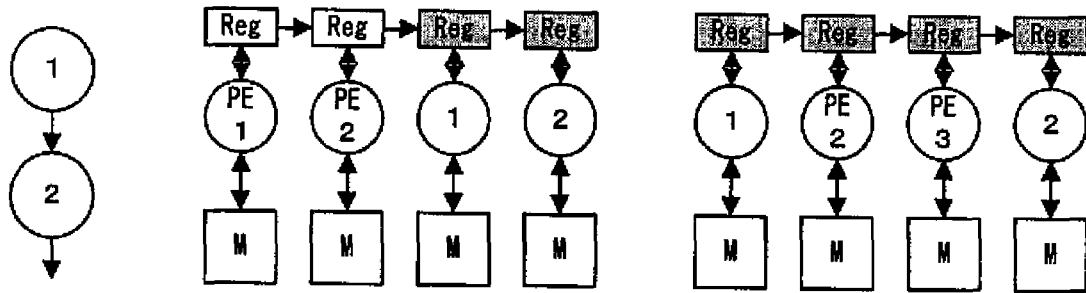
キーワード: 並列構造VLSIプロセッサ(parallel VLSI processor), 転送ボトルネックフリーアーキテクチャ(Bottleneck free architecture), ハイレベルシンセシス (High level synthesis), ロジックインメモリ構造 (Logic-in-memory architecture), 分枝限定法 (branch and bound method)

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

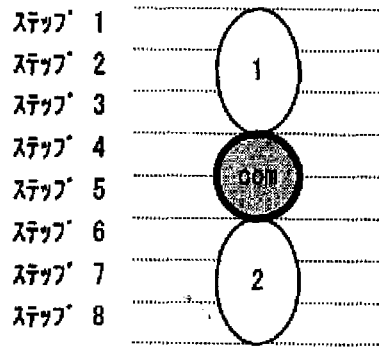
1. はじめに

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通信ボトルネックのないデータ転送が重要である. 通信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)}, この一般的设计法の開発が望まれる. 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演算を最小演算遅れ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

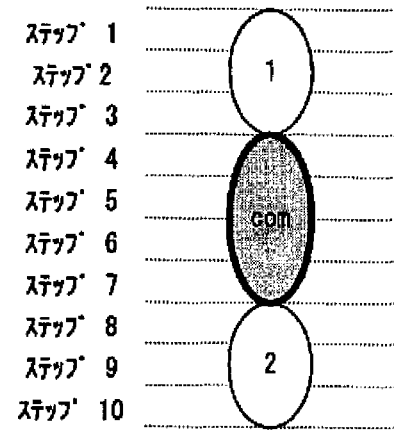
たデータ依存グラフに対して, 演算器のアロケーションとスケジューリングは互いに影響し合うため, スケジューリングとアロケーションを統合した設計が必要になる. そこで, スケジューリングとアロケーションの膨大な組合せに対して, 分枝限定法に基づき最適解を探索する方法を提案している. 最後に, データ依存グラフで与えられた演算に対し, 最適なスケジューリングとアロケーションを行った場合の転送時間を含む全体の処理時間が, 大幅に短縮できることを示す.



(a) データ依存グラフ



(b) アロケーション1



(c) アロケーション2

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

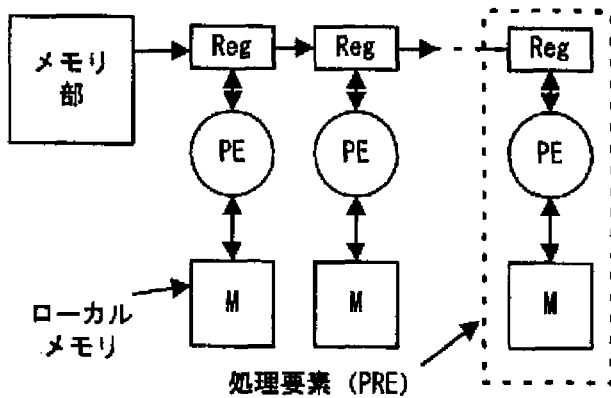


Fig. 1 ハードウェアモデル

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間の通信は、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、通信は、1ステップの間に隣接したPREへ転送できるものとする。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つのノード間の通信時間を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演算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。

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*にFig. 3のようなデータ依存関係があり、ノード*i*が*S_i*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*への通信時間を*T_{com}*とすると、*j*のスケジューリング可能なステップ*S_j*は、式(1)で表すことができる。

$$S_i + Top + T_{com} - 1 \leq S_j \quad (1)$$

次に、ノード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*a_i*であり、スケジューリング可能なステップの個数が*b_i*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*C_i*とおくと、*C_i*は式(2)で表すことができる。

$$C_i = a_i \times b_i \quad (2)$$

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。

$$N = C_1 \times C_2 \times \dots \times C_m \quad (3)$$

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

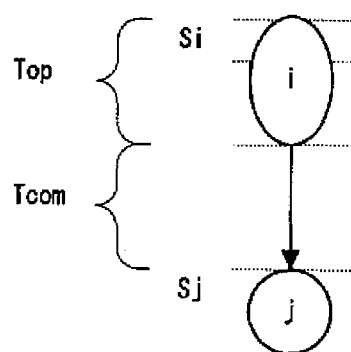


Fig. 3 ノード*j*のスケジューリングにおける制約条件

*i*につき*C_i*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査して、等価的にすべての枝を探査したことになる探索方法が望まれる。

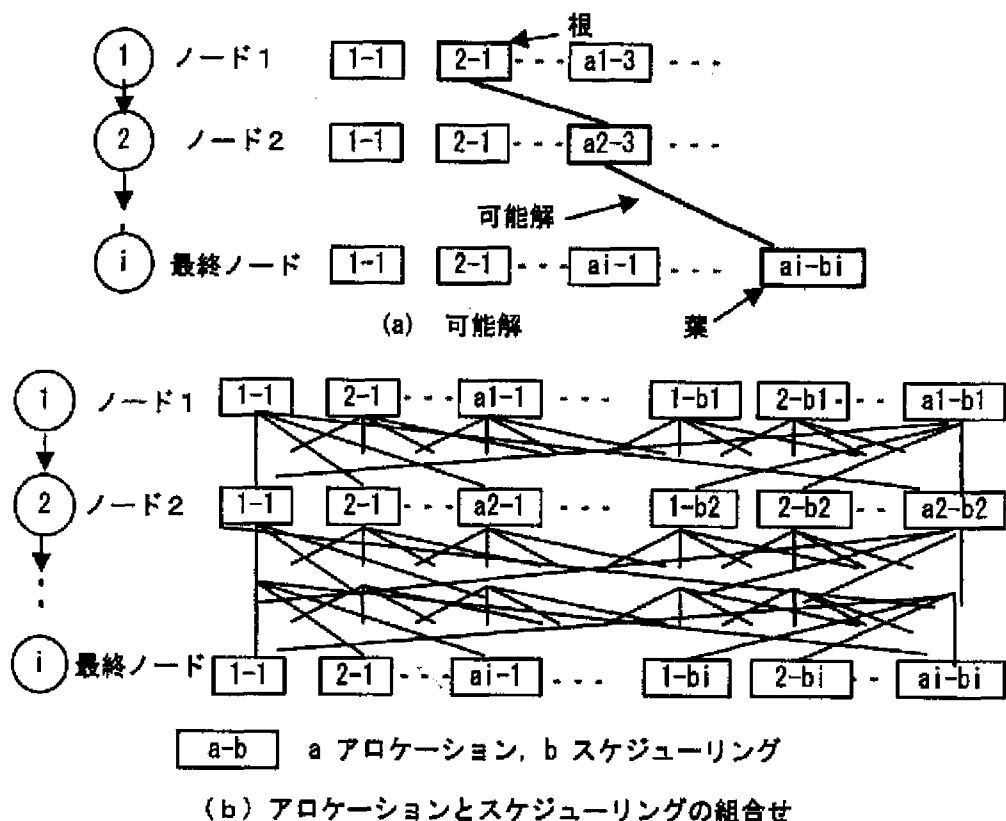


Fig. 4 アロケーションとスケジューリングの組合せと可能解

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³⁾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。

$$D = S_i + Dimin \quad (4)$$

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 S_i に等しいことから、Dを求めるためにはノードi

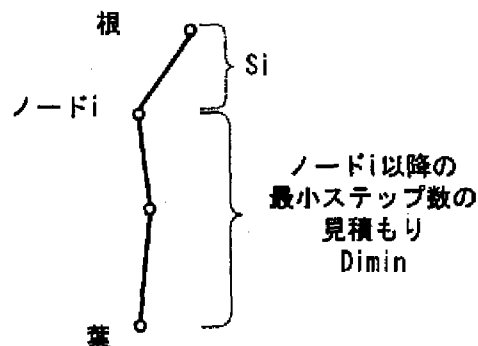
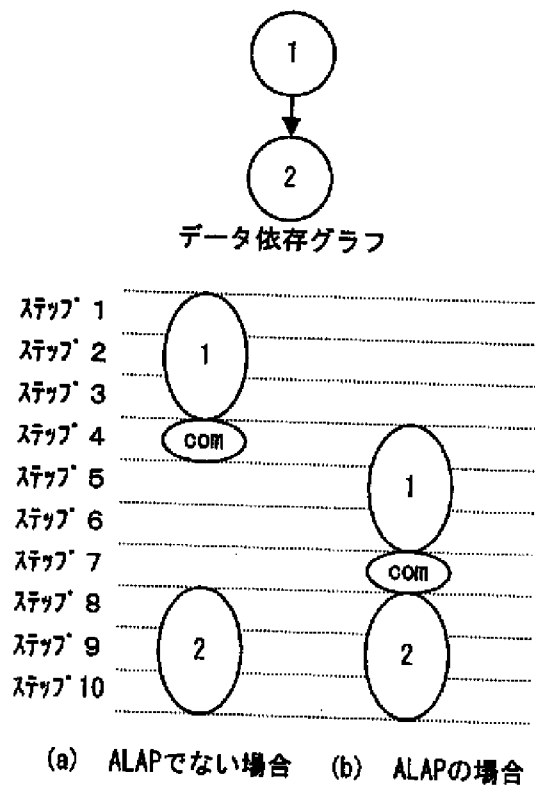


Fig. 5 最小演ステップ数の見積もり

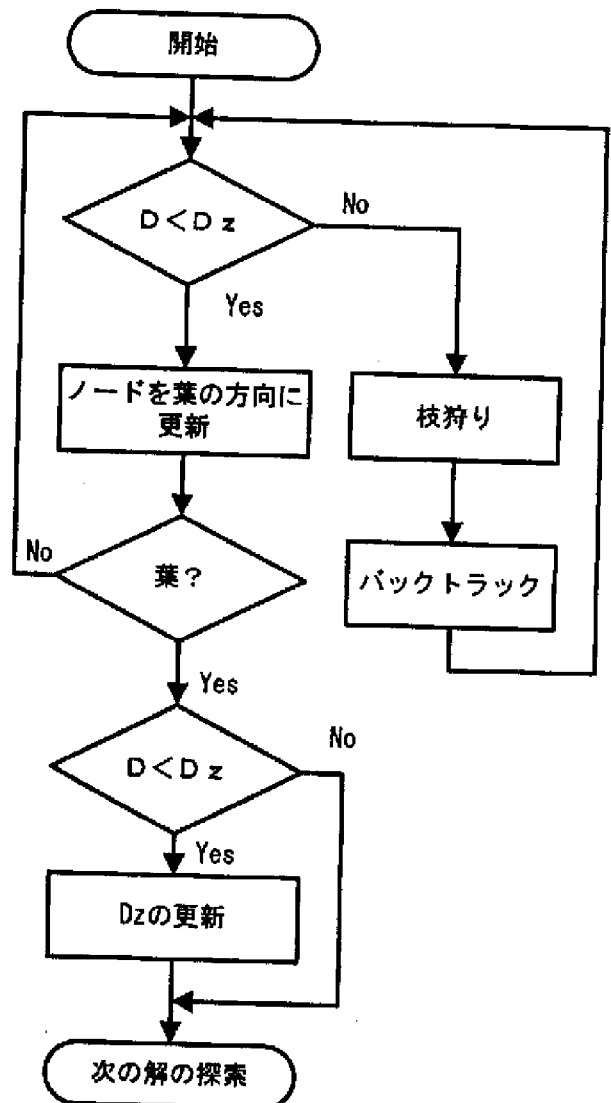
以降の最小ステップ数Diminの見積もりができればよい。そこで、あらかじめデータ依存グラフのすべてのノードについて最終ノードまでの最小ステップ数の見積もりDiminをALAP⁴⁾ (as last as possible) に基づき求めておくことにする。ALAPに基づくDiminの決定は次のようにする。たとえば、Fig.6のデータ依存グラフのスケジューリングは、各ノードの演算に3ステップ、ノード間の通信に1ステップ必要であるとき、(a)や(b)の可能性があり、(b)は可能な限り各ノードを遅いステップにスケジューリングしたALAPの場合である。(b)においてはノード1がスケジューリングされてか



(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

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 D_{1min} は7ステップである。 D_{imin} 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 S_i と D_{imin} 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 D_z を持つ解が発見されているとき、 $D < D_z$ の場合は、新しい最適解を発見できる可能性があるため、探索中の枝について探索を続行する。逆に、 $D \geq D_z$ 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暫定的な最小ステップ数より必ず小さいか等しい



D_z : 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

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プロセッサを設計する。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、ウィンドー処理のデータ依存関係は入力データのみ依存する。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。9個のノードにFig.9のハードウェアモデルの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。すなわち、目的関数はステップ数である。木の探索途中における最小ステップの見積もりは、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で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。現在探索しているノード集合から次のノード集合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。つまり、ウィンドーの移動には、少なくとも1ステップ必要であることから、次のノード集合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の見積もりは、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。つまり、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。最小ステップ数の通信と演算を実現できるアロケーションとスケジューリングを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、通信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必要としない。

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ことを保証できる。

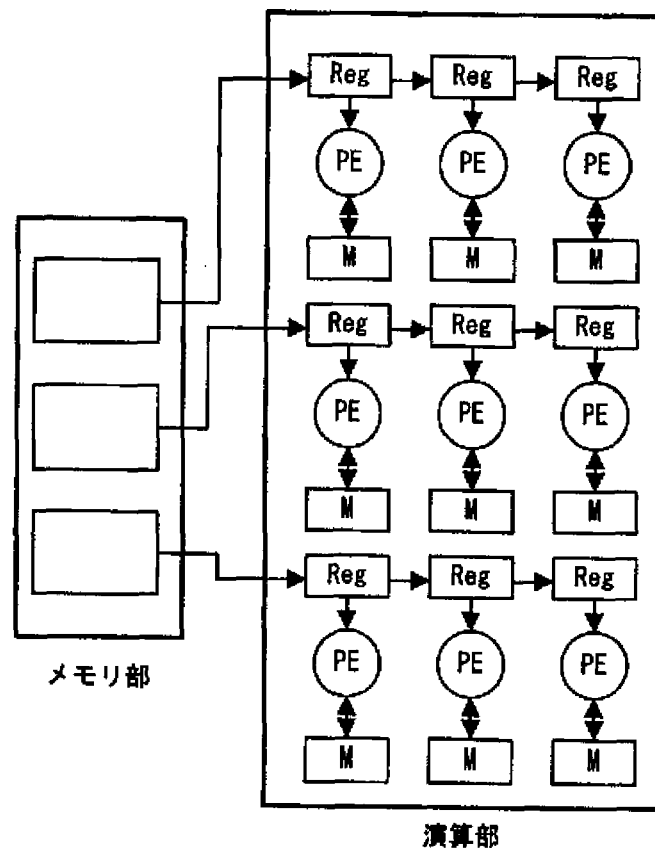
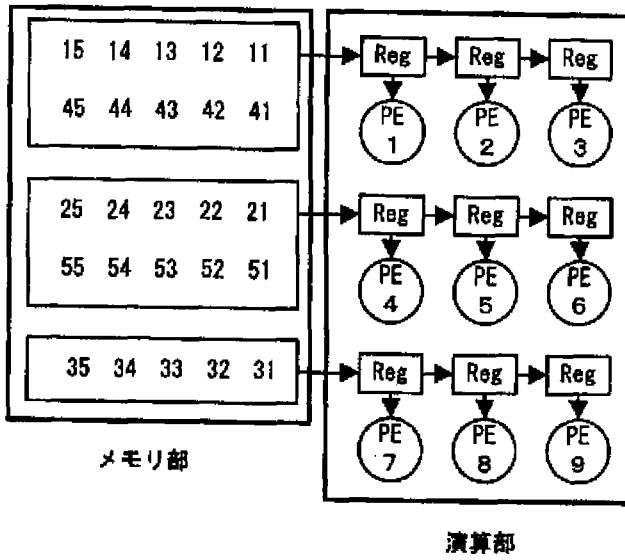


Fig. 9 評価に用いたハードウェアモデル

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、通信時間も含め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演算を最小の処理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⁵⁾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



(a) ハードウェア構成

	PE 1	PE 2	PE 3	PE 4	PE 5	PE 6	PE 7	PE 8	PE 9
ステップ 1	11			21			31		
ステップ 2	12	11		22	21		32	31	
ステップ 3	13	12	11	23	22	21	33	32	31
ステップ 4	14	13	12	24	23	22	34	33	32
ステップ 5	15	14	13	25	24	23	35	34	33
ステップ 6	41			21			31		
ステップ 7	42	41		22	21		32	31	
ステップ 8	43	42	41	23	22	21	33	32	31
ステップ 9	44	43	42	24	23	22	34	33	32
ステップ 10	45	44	43	25	24	23	35	34	33
ステップ 11	41			51			31		
ステップ 12	42	41		52	51		32	31	
ステップ 13	43	42	41	53	52	51	33	32	31
ステップ 14	44	43	42	54	53	52	34	33	32
ステップ 15	45	44	43	55	54	53	35	34	33

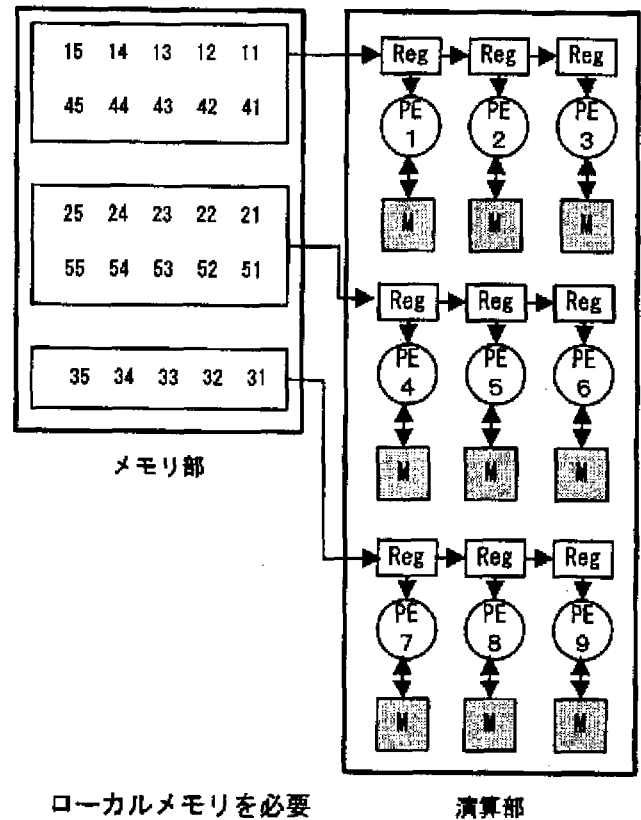
(b) 最小ステップの通信と演算

Fig. 10 最適なアロケーションとスケジューリング

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ミュレーション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



ローカルメモリを必要

演算部

(a) ハードウェア構成

	PE 1	PE 2	PE 3	PE 4	PE 5	PE 6	PE 7	PE 8	PE 9
ステップ 1	11			21			31		
ステップ 2	12	11		22	21		32	31	
ステップ 3	13	12	11	23	22	21	33	32	31
ステップ 4	13	12		24	23	22	34	33	32
ステップ 5	13	12		24	23	22	34	33	32
ステップ 6	13	12	14	24	23	22	34	33	32
ステップ 7	13		14	25	24	23	35	34	33
ステップ 8	13	15	14	25	24	23	35	34	33
ステップ 9	41			21			31		
ステップ 10	41			22	21		32	31	
ステップ 11	41	42		23	22	21	33	32	31
ステップ 12	41	42		23	22	21	33	32	31
ステップ 13	41	42	43	23	22	21	33	32	31
ステップ 14	44								
ステップ 15	43	44							
ステップ 16	42	43	44	24	23	22	34	33	32
ステップ 17	45			25	24	23	34	33	32
ステップ 18	44	45		25	24	23	34	33	32
ステップ 19	43	44	45	25	24	23	35	34	33
ステップ 20	41			51			31		
ステップ 21	42	41		52	51		32	31	
ステップ 22	43	42	41	53	52	51	33	32	31
ステップ 23	44	43	42	54	53	52	34	33	32
ステップ 24	45	44	43	55	54	53	35	34	33

[n] : ローカルメモリに記憶
多くの通信のステップを必要

(a) 通信と演算

Fig. 11 最適ではないアロケーションとスケジューリング

- 3) たとえば石畑清, "アルゴリズムとデータ構造", 岩波書店
- 4) Daniel Gajski, Nikil Dutt, Allen Wu, Steve Lin, "HIGH-LEVEL SYNTHESIS Introduction to Chip and System Design", Kluwer Academic Publishers
- 5) J.N.Warfield, "Binary Matrices in System Modeling", IEEE Trans.Syst., Man, Cybern., vol.SMC-3, No.5, Sept. 1973.

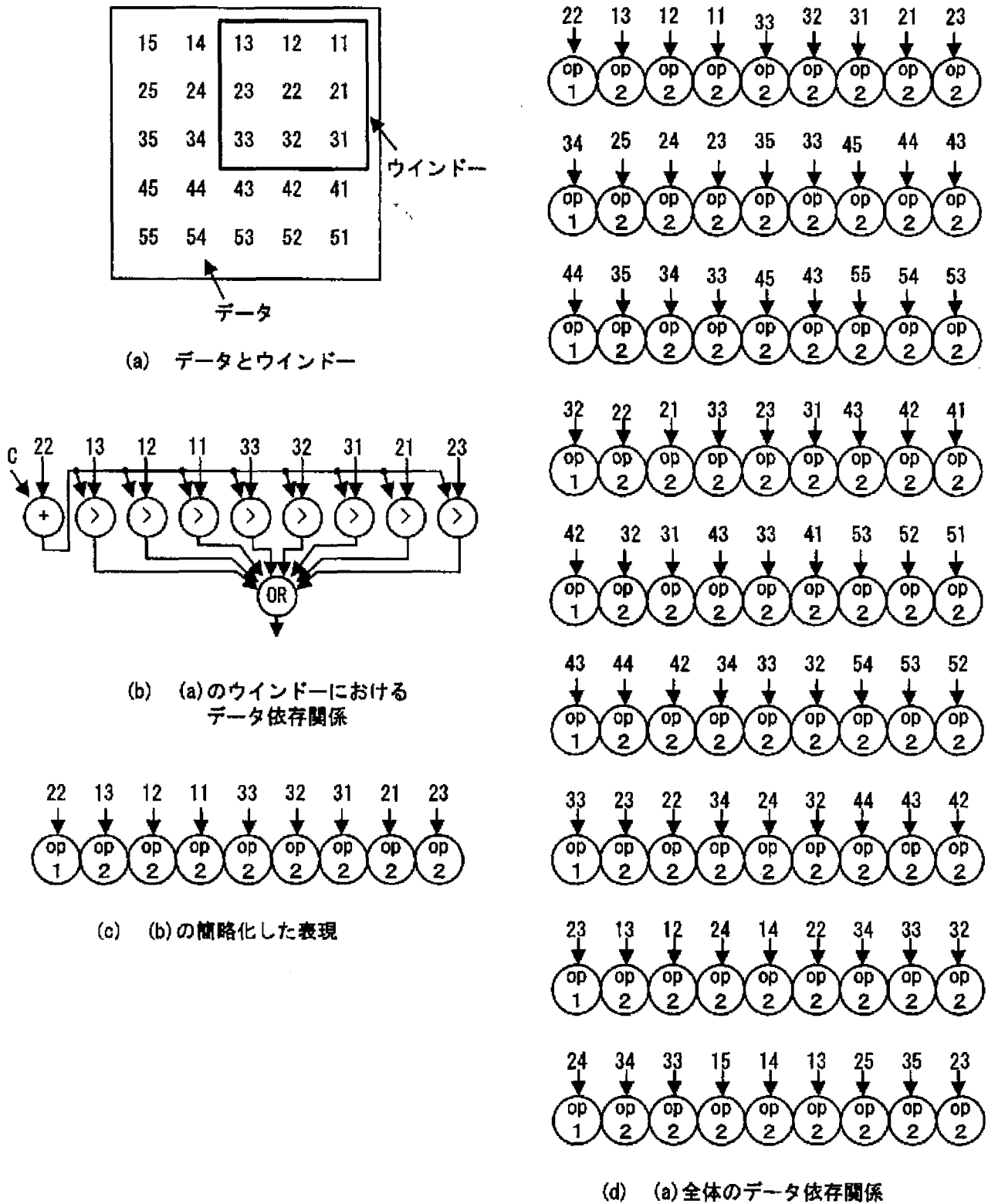


Fig. 8 評価