計測自動制御学会東北支部 第199回研究集会 (2001.12.15)
資料番号 199-12

複数電源電圧を用いた低消費電力VLSIプロセッサのハイレベルシンセシス

High-Level Synthesis for Low Power VLSI Processors Using Multiple Supply Voltages

○青山哲也， 張山昌論， 亀山充隆

○Tetsuya Aoyama，Masanori Hariyama，Michitaka Kameyama

東北大学情報科学研究科

Graduate School of Information Sciences
Tohoku University

キーワード： ハイレベルシンセシス (High-Level Synthesis)，複数電源電圧 (Multiple Supply Voltages)，消費エネルギー最小化 (Energy Consumption Minimization)，遺伝的アルゴリズム (Genetic Algorithm)

連絡先： 〒980-8579　仙台市青葉区荒巻字青葉05 東北大学大学院情報科学研究科亀山研究室
青山 哲也，Tel.: (022)217-7155，Fax.: (022)263-9167，E-mail: aoyama@kameyama.ecei.tohoku.ac.jp

1. まえがき

近年，VLSIプロセッサの動作周波数・集積度の向上に伴い，消費電力の増大が深刻な問題となっている[1, 2]．

消費電力の増大が引き起こす問題として，以下の3つが考えられる．まず，熱の問題である．VLSIプロセッサのパッケージには，放熱の限界があるため，発熱がトランジスタの集積度に限界を与えるということである．また，携帯機器などのようなバッテリー駆動デバイスや無線通信システム等におけるバッテリーライフの短縮という問題がある．さらに，VLSIプロセッサの信頼性の問題がある．

しかしながら，現在，消費電力に着目した上位レベルの系統的なVLSIプロセッサの構成理論と，その自動設計法がほとんど報告されていない．そこで，本稿では，VLSIプロセッサの低消費電力化のための最適設計 (ハイレベルシンセシス[3, 4]) を提案している．

最適設計とは，制約条件を処理時間及びチップ面積とし，平均消費電力 (以下では，消費エネルギー) を目的関数としたときの最小化問題とする．

消費エネルギーを削減する最も効果的な方法は，電源電圧の低減である．これは，消費エネルギーが電源電圧の2乗に比例するためである．しかしながら，電源電圧の低減は，同時に遅延時間の増大につながる．従って，遅延時間を増大させずに消費エネルギーを削減することが重要となる．

このような問題を解決する方法として複数電源電圧を用いる方法がある．これは，クリティカルパス上の演算には高い電源電圧を(要求される処理時間制約を満たすため)，非クリティカルパス上

の演算には低い電源電圧を(消費エネルギーを削減するため)割り当てるという方法である．Fig.1 は，演算 O_2，O_3 に低い電源電圧 V'_{dd} を割り当て，その他の演算には V_{dd} を割り当てた例である．例えば，$V'_{dd} = V_{dd}/2$ のとき，消費エネルギーは 70 % に削減される．このように，遅延時間を増大させることなく，消費エネルギーを削減することが可能となる．

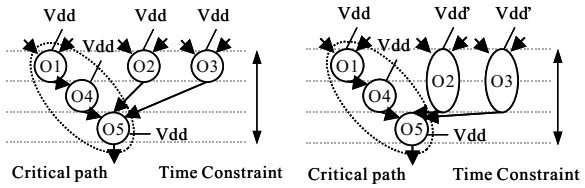

Fig. 1　消費エネルギー削減の概念

本稿では，専用プロセッサの設計を前提としており，アプリケーションのアルゴリズムがデータフローグラフ (DFG) として与えられたとき，並列構造VLSIプロセッサのアーキテクチャモデルを用いて，処理時間及びチップ面積制約下で消費エネルギーを最小とするスケジューリング・アロケーションを提案する．

消費エネルギー最小化のために，消費エネルギー最小化問題を整数計画問題として定式化する．演算ノードの開始ステップの決定 (スケジューリング) と演算ノードを実行する演算器の決定 (アロケーション) の可能な組み合わせに対し，総当り的に消費エネルギーの評価を行えば，一応解は求められる．しかし，探索空間が膨大となり，大規模な問題に応用することは困難である．そこで，本稿では，効率的な探索アルゴリズムである遺伝的アルゴリズム[4, 5]を適用した．探索法について考察した結果を述べる．

2. 消費エネルギー最小化問題の定式化

2.1 問題設定

VLSIプロセッサの動作仕様は，Fig.2 に示すような DFG で与えられるとする．DFG は有効グラフ $G(O,A)$ で，O はノードの集合，A はアークの集合とする．各ノード $O_i \in O$ は演算を示し，各アーク $A_i \in A$ はノードの依存関係を示す

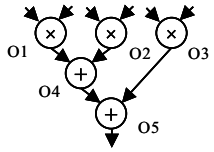

Fig. 2　データフローグラフ (DFG)

Fig.3 にアーキテクチャモデルを示す．相互結合網は多重バス方式に基づいており，必要十分な本数のバスを用いることができる．これにより，任意のモジュール間で任意の並列度のデータ転送を行え，種々のパイプライン・空間的並列構造を実現できる．

Functional Unit (以降では，演算器) は，Table 1 のような演算器ライブラリで与えられ，種々の演算タイプ，電源電圧の演算器を用意する．ここで，DFG の各ノードにどの演算器を割り当てるかを決定すること (アロケーション) が消費エネルギー最小化問題の自由度となる．

また，Gated-Clock を採用することにより，演算器の非動作時における入力信号をカットすることができる．このため，DFG 一連の処理により消費されるエネルギー量を目的関数とすると，演算器の動作時に消費されるエネルギーの総和の最小化を求めることになる．

自由度は，アロケーションとスケジューリングである．スケジューリングとは，DFG の各ノードの開始ステップを決定することである．アロケーションを決定することは，ノードの演算終了時間

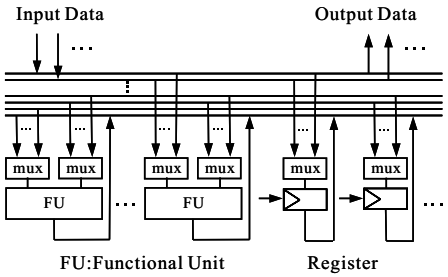

Fig. 3　アーキテクチャモデル

Table 1　演算器ライブラリ

	Functional Unit	Delay	Area (normalized)	Energy (normalized)
F1	ADD(5V)	1 step	1	2
F2	ADD(3V)	2 step	1	1
F3	MUL(5V)	2 step	8	6
F4	MUL(3V)	4 step	8	3

を決定することに対応するため，スケジューリングに対して影響を与える．Fig.4 は，ノード O_1 のアロケーションを $F1$ から $F2$ へ変化させることにより，O_2 のスケジューリングの可能範囲が変化することを示している．同様に，スケジューリングもアロケーションの自由度に影響を与えることから，スケジューリングとアロケーションは，両者を統合して議論することが重要となる．

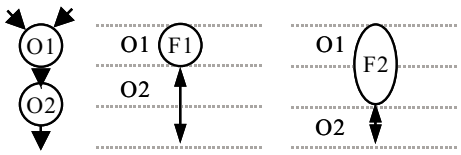

Fig. 4　アロケーションのスケジューリングに対する影響

また，与えられる DFG の処理のステップ数を処理時間制約以内となるように設計すること，並列に用意する演算器の数をチップ面積制約に対応させて設計することを考える．このように，処理時間及びチップ面積制約を考慮することにより，実用的な問題設定となり得る．

消費エネルギー最小化問題を以下のように設定する．VLSI プロセッサの動作仕様として DFG，演算器ライブラリが与えられたとき，処理時間及びチップ面積制約下で消費エネルギーが最小となるスケジューリング・アロケーションを求める．

2.2　整数計画問題へのマッピング

消費エネルギー最小化問題が，スケジューリングとアロケーションを統合した問題に帰着可能であるということに着目し，整数計画問題として定式化する．

はじめに，目的関数の定式化を行う．最小化するエネルギーを各演算ノードで消費されるエネルギーの総和と考える．演算ノード O_i の実行により消費されるエネルギーを E_i，演算ノードの総数を N とすると，以下のように表せる．

$$\sum_{0 \leqq i \leqq N} E_i \tag{1}$$

ここで，E_iは Table 1 の演算器ライブラリから与えられる．

次に，自由度の領域，自由度に対応する変数の定義をする．設定された処理時間制約に対して，DFG の各演算ノードは，時間的自由度を持つ．例えば，Table 1 の演算器ライブラリを用い，Fig.2 の DFG を 5ステップで実行すると仮定した場合，各ノードの開始ステップとして可能なステップの範囲は Fig.5 のように表される．

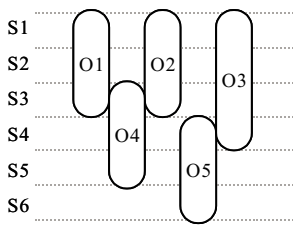

Fig. 5　各ノードの可能なスケジューリングの範囲

このような，ノード O_i の開始ステップとして可能なステップの集合を $mrange(O_i)$ とし，以下

のように定義する．

$$mrange(O_i) = \{S_j | B_i \leqq j \leqq L_i\}$$

ここで，S_j はステップ番号 j を表し，$B_i(L_i)$ はノード O_i 開始ステップとして可能な最も早い(遅い)ステップ番号を表す．これらは，一般に ASAP と ALAP から求められる．例えば，Fig.5 のノード $O3$ では，$B_3 = 1, L_3 = 4$ であるので，$mrange(O_3) = \{S_j | 1 \leqq j \leqq 4\}$ と表せる．

同様に，DFG の各ノードを実行可能な演算器にも自由度が存在する．ノード O_i を実行可能な演算器は，ノードと同一演算タイプの演算器であり，その集合を $fu(O_i)$ とし，以下のように定義する．

$$fu(O_i) = \{F_j \mid (\text{演算器}F_j\text{の演算タイプ}) = (\text{ノード}O_i\text{の演算タイプ})\}$$

ここで，F_j は，演算番号 j を表す．例えば，Fig.2 のノード $O1$ は乗算なので，$fu(O1) = \{F_j | j = 3, 4\}$ と表せる．

各ノードの実行開始ステップを決定するため，$mrange(O_i)$ で与えられた各ステップに対応する変数 x_{ij} を割り当てる．変数 x_{ij}は，スケジューリングに対応し，以下のように定義する．

$$x_{ij} = \begin{cases} 1 & : \text{ノード}O_i\text{の実行開始ステップが} \\ & \quad \text{ステップ}S_j\text{のとき} \\ 0 & : \text{その他} \end{cases}$$

例えば，Fig.5 のノード $O3$ については，$mrange(O3) = \{S_j | 1 \leqq j \leqq 4\}$ であるので，$x_{31}, x_{32}, x_{33}, x_{34}$ が $mrange(O3)$ の各ステップに割り当てられる．

同様に，各ノードを実行する演算器を決定するため，$fu(O_i)$ で与えられた各演算器に対応する変数 y_{ij} を割り当てる．変数 y_{ij} は，アロケーションに対応し，以下のように定義する．

$$y_{ij} = \begin{cases} 1 & : \text{ノード}O_i\text{を実行する演算器が演} \\ & \quad \text{算器}F_j\text{のとき} \\ 0 & : \text{その他} \end{cases}$$

例えば，Fig.2 のノード $O1$ については，$fu(O1) = \{F_j | j = 3, 4\}$ であるので，y_{13}, y_{14} が $fu(O1)$ の各演算器に割り当てられる．

最後に，制約条件を定式化する．ただし，各変数，定数は以下のように与えられる．

A： チップ面積制約

K： 使用可能な演算器の個数

A_{F_i}： 演算器 F_i の面積

D_{F_i}： 演算器 F_i の遅延時間

N_{F_i}： 演算器 F_i の個数

制約条件1： ノードの実行範囲に関する制約

ノード O_i の実行開始ステップは $mrange(O_i)$ の範囲内でなければならない．また，全ての演算の実行開始ステップは可能なステップのうち1個であるため，$mrange(O_i)$ に割り当てられた x_{ij} の合計は 1 である．

$$\sum_{B_i \leqq j \leqq L_i} x_{ij} = 1 \tag{2}$$

制約条件2： ノードの演算器割り当てに関する制約

ノード O_i を実行する演算器は $fu(O_i)$ の範囲内でなければならない．また，全ての演算に対して割り当て可能な演算器は1個であるため，$fu(O_i)$ に割り当てられた y_{ij} の合計は 1 である．

$$\sum_{F_j \in fu(O_i)} y_{ij} = 1 \tag{3}$$

制約条件3： ノード間の依存関係に関する制約

ノード O_i の出力がノード O_j の入力となっている場合，ノード O_i の演算が終わった後にノード O_j の演算が実行されなければならない．

$$\sum_{B_i \leqq k \leqq L_i} (k \times x_{ik}) + \sum_{F_m \in fu(O_i)} (D_{F_m} \times y_{im}) \leqq \sum_{B_j \leqq l \leqq L_j} (l \times x_{jl}) \tag{4}$$

制約条件4： チップ面積に関する制約

演算器の占める面積の合計は，制約条件として与えられるチップ面積 A を越えてはならない．

$$\sum_{1 \leqq i \leqq K} (A_{F_i} \times N_{F_i}) \leqq A \tag{5}$$

式(2)-(5)の制約条件下で式(1)を最小化する x_{ij}, y_{ij} を求めることにより，消費エネルギーを最小と

するスケジューリング，アロケーションが決定される．

整数計画問題は探索空間が膨大であり，大規模な問題を応用することが困難である．そこで次節において，効率的な探索アルゴリズムである遺伝的アルゴリズムに，消費エネルギー最小化問題をマッピングする．

3. 遺伝的アルゴリズムの適用

3.1 遺伝的アルゴリズムの処理フローと遺伝子表現

遺伝的アルゴリズムの処理フローを Fig.6 に示す．その概要は，初期個体群を生成し，以下の step1 から step3 の操作を，終了条件が満足するまで繰り返す処理である．

step1： 各個体の適応度の評価

step2： 適応度に基づく個体群の選択

step3： 選択された個体群の交叉・突然変異による次世代の生成

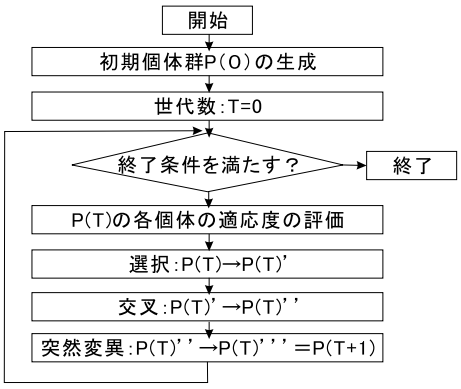

Fig. 6　遺伝的アルゴリズムの処理フロー

遺伝子の表現について，Fig.7 を用いて説明する．Fig.7 の DFG はスケジューリングとアロケーションが決定している．例えば，Parent1 のノード O_1 のスケジューリングは S1 であり，アロケーションは F4 である．これらの DFG を Table 2 のように遺伝子の列で表現する．遺伝子には，ノード番号の順にスケジューリングとアロケーションの情報が組み込まれており，消費エネルギー最小化問題に必要な情報を有している．

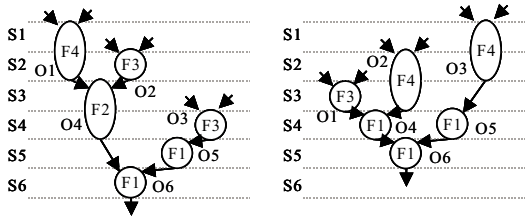

Fig. 7　スケジューリングとアロケーション例 (左：Parent1，右：Parent2)

Table 2　Fig.7 の DFG の遺伝子表現

	O1		O2		O3		O4		O5		O6	
Parent1	S1	F4	S2	F3	S4	F3	S3	F2	S5	F1	S6	F1
Parent2	S3	F3	S2	F4	S1	F4	S4	F1	S4	F1	S5	F1

3.2 致死遺伝子の発生を抑制するグラフ構造に基づく交叉

一般的な交叉である一点交叉について説明する．一点交叉とは，遺伝子の列を任意の交叉点で2つに分割し，それらを交換するという方法である．Table 3 は，Table 2 の遺伝子を用いて，ノード O_3 と O_4 の間を交叉点とし，一点交叉した例である．この方法は，DFG のグラフ構造を考慮することなく，遺伝子レベルの操作により次世代を生成するため，致死遺伝子を発生する確率が非常に高くなる (Fig.8)．このような致死遺伝子の発生は，探索を非効率的にするため，可能な限り発生させないことが重要となる．

Table 3　Table 2 の遺伝子の一点交叉により生成された遺伝子

	O1		O2		O3		O4		O5		O6	
Offspring1	S3	F3	S2	F4	S1	F4	S3	F2	S5	F1	S6	F1
Offspring2	S1	F4	S2	F3	S4	F3	S4	F1	S4	F1	S5	F1

致死遺伝子の発生を抑制するために，グラフ理論のカットセットに着目した交叉について述べる．

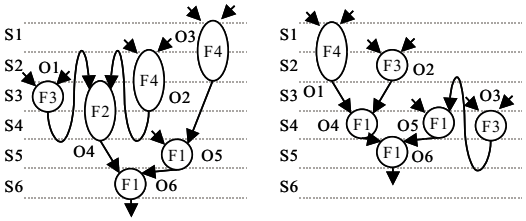

Fig. 8　Table 3 の遺伝子表現の DFG 表現 (左：Offspring1，右：Offspring2)

Fig.9 は連結グラフである．点線で示した辺 wy, xz を除くと，連結グラフが非連結となる．このように，連結グラフを非連結にする辺集合で，その集合に属する辺の幾つか (全部ではない) を除いても連結グラフは非連結にならないという性質を持っているものをカットセットと呼ぶ．Fig.9 のグラフにおいて，wy, xz, xw を除くと非連結となるが，この辺集合は，先に述べた性質を満足していないため，カットセットではない．

ここで，DFG のグラフ構造に着目し，カットセットでグラフを二つのグラフに分割し，それらを交換するという交叉を提案する．Fig.10 は，Fig.7 の二つの DFG の交叉により生成された DFG であり，交差点はノード O_4 とノード O_6 の間の辺 (カットセット) である．カットセットにより分割された二つのグラフ内において，ノードの依存関係が保持されるため，本提案の交叉を用いることにより，致死遺伝子の発生が抑制される．

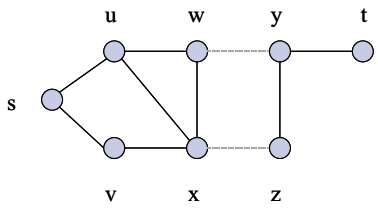

Fig. 9　連結グラフのカットセット

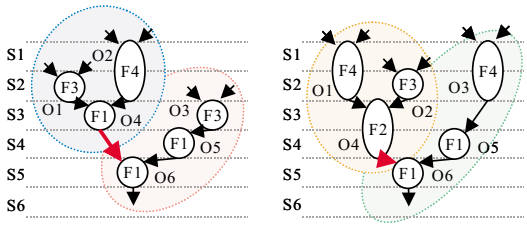

Fig. 10　グラフ構造に着目した交叉により生成した遺伝子の DFG 表現

3.3　遺伝的アルゴリズムと局所探索法のハイブリッド化

遺伝的アルゴリズムは，遺伝的操作 (交叉，突然変異) に基づいた探索法であるため，大局的な探索には非常に効率的である半面，良い解付近での系統的な探索が難しい．一方，局所探索法は，暫定解の近傍を系統的に探索するのに優れている．この二つの探索法を組み合わせることにより，互いの長所を合わせた効率の良いアルゴリズムを実現する．

本アルゴリズムの概要を Fig.11 に示す．遺伝的アルゴリズムの遺伝的操作により生成された個体に対し，局所探索法を適用することで，大局的な探索，系統的な局所探索を行うことができ，探索が効率的となる．

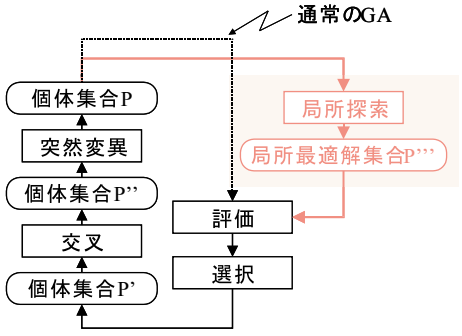

Fig. 11　遺伝的アルゴリズムと局所探索法のハイブリッド化

次に，消費エネルギー最小化問題における局所探索法について Fig.12 を用いて説明する．局所探索は全てのノードに対して行うが，Fig.12 の左図

はノード O_1 を選択したときの例である．前提条件として，全てのノードのスケジューリング・アロケーションが決定している．このとき，ノード O_1 以外のノードに関しては全て固定し，ノード O_1 のみの自由度について総当り的に探索する (局所探索)．ここで，O_1 以外のノードは固定しているため，探索すべき組み合わせ数は数通りしかなく，総当り探索でも十分である．ノード O_1 に関して，局所探索を行った結果が Fig.12 の右図であり，ノード O_1 の電源電圧が削減されており，消費エネルギーが削減している．つまり，局所探索により，解が改善されている．

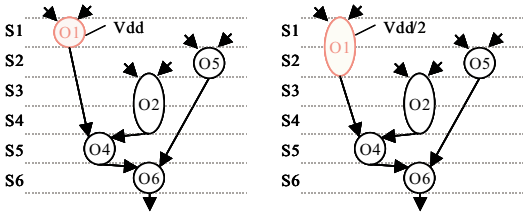

Fig. 12　ノード O_1 の局所探索 (左：初期条件，右：改善例)

3.4　提案手法による解の評価

まず，Fig.13 の左図の DFG，Table 1 の演算器ライブラリを用いて解の評価を行った．結果を Table 4 の上段に示す．処理時間及びチップ面積制約下において，単一電源電圧 (5V) と複数電源電圧 (5V, 3V) により消費されるエネルギーを比較し，その削減率を示した．Exercise の括弧内の数字は，DFG のノード数を示す．処理時間制約を 8, 9, 10step と，変化させて評価を行ったところ，いずれの場合も複数電源電圧を用いたときの方が，消費エネルギーは小さいという結果を得た．また，処理時間制約に自由度が大きいほど，消費エネルギーが削減されるという結果を得た．Fig.13 の右図は，処理時間制約が 8step のときの解である．

次に，総当り探索では実用的な時間で解を求めることが困難な5次楕円フィルタに関して，評価を行った．結果を Table 4 の下段に示す．同様に複数電源電圧を用いることにより，消費エネルギーが削減されるという結果を得た．また，AMD の Athlon 1GHz で計算したところ，約30秒程度の計算時間で解を求めることができた．

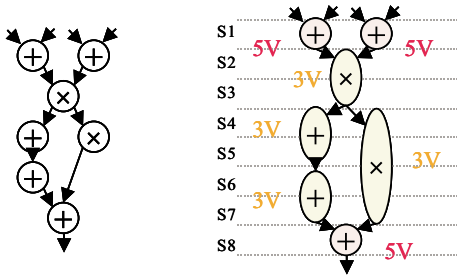

Fig. 13　評価用 DFG (左図) と，処理時間制約が 8step のときの解 (右図)

Table 4　消費エネルギーの削減割合

Exercise	Time Constraint	Area Constraint	Energy using 5V	Energy using 5V, 3V	Reduction Ratio[%]
Test DFG (7)	8step	20	22	17	22.7
	9step	20	22	15	31.8
	10step	20	22	14	36.4
EWF (34)	25step	30	100	69	31.0
	27step	30	100	62	38.0
	30step	30	100	56	44.0

※EWF: 5th order elliptic wave filter

4.　むすび

本稿では，消費電力に着目した VLSI プロセッサの設計法について述べた．まず，処理時間及び面積制約下での消費エネルギー最小化問題を設定し，整数計画問題として定式化した．しかしながら，整数計画問題は探索空間が膨大であり，大規模な問題に対して適用が困難である．そのため，次に，近似的な解を効率的に探索するアルゴリズムである遺伝的アルゴリズムを適用した．その際に，致死遺伝子を抑制する交叉，遺伝的アルゴリズムに局所探索法を組み合わせるアルゴリズムを提案した．最後に，本提案手法を適用し，例題に

よる評価を行ったところ，複数電源電圧を用いることにより消費エネルギーが削減されることを確認した．

今後の展望としては，実用上の VLSI プロセッサを設計する上での総合的な評価を行っていくことが必要である．

参考文献

1) A.P.Chandrakasan，S.Sheng，and R.W.Brodersen，“ Low-power digital CMOS design，” IEEE Journal of Solid State Circuits，pp.473-484，April 1992.

2) S.Raje，M.Sarrafzadeh，“Variable Voltage Scheduling，” In Proceedings of the 1995 International Workshop Low Power Design，1995.

3) 亀山充隆，佐々木正行，“空間的・時間的並列構造融合型VLSIプロセッサの最適設計，” 電子情報通信学会論文誌，Vol. J80-A No.3，pp.499-508，1997.

4) 工藤隆男，張山昌論，亀山充隆，“遺伝的アルゴリズムを用いたロジックインメモリ構造VLSIプロセッサのハイレベルシンセシス，” 計測自動制御学会東北支部，資料番号195-9，2001.

5) 大森健児，“遺伝的アルゴリズムによる高レベル合成，” 電子情報通信学会論文誌，Vol. J81-A No.5，pp.854-862，1998.