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

2005－ＳＬＤＭ－119（６）
2005／3／17

再構成可能加算を考慮したLSI高位設計手法

渡辺 貴宏　　伊藤 和人

埼玉大学工学部電気電子システム工学科

〒338-8570 埼玉県さいたま市桜区下大久保255

TEL/FAX 048-858-3731

E-mail: {watanabe,kazuhito}@elc.ees.saitama-u.ac.jp

あらまし　特定用途専用LSIの設計では、要求性能を満たしつつシステムコストを最小化することが求められている。処理内部の演算精度が不均一な場合に各精度の演算に専用の演算器・レジスタを備えることは、資源利用効率の面で不利でありシステムコスト増加を招く。精度を切り替えて演算実行可能な再構成可能加算器とレジスタ再構成を考慮することでシステムコストを最小化したLSIを設計するための高位合成手法を提案する。与えた演算器構成についてレジスタコストを最小化する演算スケジュールを求める手法を用い、演算器構成を徐々に増大させることで最小コストのシステムを探索する。

キーワード　専用LSI設計、再構成可能演算器、レジスタ再構成、高位合成

A High-Level Synthesis for LSI Design by Considering Reconfigurable Adders

Takahiro Watanabe　　Kazuhito Ito

Department of Electrical and Electronic Systems, Saitama University
225 Shimookubo, Sakura-ku, Saitama 338-8570, Japan

Tel/FAX +81-48-858-3731

E-mail: {watanabe,kazuhito}@elc.ees.saitama-u.ac.jp

Abstract　In application specific LSI design, it is requested to minimize the system cost of the LSI while satisfying performance requirements. If the operation precisions are not uniform in the given processing algorithm, one of the solutions is to build the system with many types of functional units each of which is dedicated to a particular operation precision. However, the usage of functional units may become low and result in an expensive system. In this paper, a high-level synthesis method is proposed to minimize the system cost by considering reconfigurable adders and reconfiguration of registers. By using a technique to obtain the operation schedule which minimizes the register cost for a given functional unit configuration, the system with the minimum cost is searched by gradually increasing the configuration of functional units.

Keyword　Application Specific LSI Design, Reconfigurable Functional Unit, Register reconfiguration, High-level Synthesis

1. はじめに

与えられた用途に専用のLSIの設計では、要求性能を満たしつつシステムのコストを最小化することが求められている。ディジタル信号処理などでは、内部の演算精度が不均一なことが一般的である。各精度の演算にそれぞれ専用の演算器を備えることはコストの面で好ましくない。精度を切り替えて演算実行可能な再構成可能演算器を用いることで演算器資源の利用効率を向上し、システムコストを最小化することができる。

再構成可能加算は短ビット長の加算器を組み合わせて長ビット長の加算を実現する手法である[1]。この手法を用いて同一の加算器資源を短ビット長(低精度)と長ビット長(高精度)の演算で共用することで、資源利用効率を改善することができる。本研究では、2つの単精度加算または1つの倍精度加算を実行可能な再構成可能加算器を用いる。また、単精度データ用の短ビット長レジスタを2個組み合わせて、倍精度データ用の長ビット長レジスタとして利用するレジスタ再構成も考慮する。

システムコストを総演算器コストと総レジスタコ

ストの総和で見積もり、最小コストのシステムを求める。与えられた処理を実行可能な最小コストの演算器構成から演算器数を徐々に増やし、それぞれの構成における必要レジスタ数を求めていくことで最小システムコストのシステムを探索する。

レジスタ数は、スケジューリングによって決定された演算の実行時刻に依存して定まる。最小コストのシステムの探索においては、再構成可能演算器を考慮してレジスタコストを最小化する演算スケジューリングが必要となる。演算のスケジューリング優先順序をSimulated Annealingによって探索することで、レジスタ数を最少化した最適なスケジュールを得る手法を提案する。

2. 関連する研究

演算精度を考慮して処理に最適化された専用LSIの設計に関して、いくつかの研究がなされている。変数のビット長を指定することが可能な拡張C言語を用いて異なる演算精度に対応する方法[2][3]では、データパス幅を自由に指定できるソフトコアプロセッサを対象に、各変数のビット長を指定可能なリターゲッタブルコンパイラを用いて設計最適化を図る。コアプロセッサのデータパス幅は指定できるが設計時に固定され、実行中にデータパス幅を切り替えて実行するものではない。実行するアプリケーションに特化したデータパスでシステムを構成する手法[4]では、ビット長の異なる複数種類の演算器・レジスタを用いることでコスト最小化を図るが、例えば長ビット長演算器を短ビット長演算に流用するといった演算器利用効率向上が図られていない。

3. 不均一精度演算

3.1. 再構成可能加算

一般的にディジタル信号処理などでは内部の演算精度は不均一である。図1で与えられるデータフローグラフ(DFG)を考える。DFGのノードは演算を表し、枝は演算間のデータ依存関係を表している。図1のDFGでは、演算aは16bit加算、演算b, cは16bit乗算である。乗算は16ビット精度のデータに16ビット精度の係数を乗じ、その結果(積)は16ビットよりも高い精度となる。精度の高い積どうしを正確に加算するため、加算の前に精度を16ビットに落とすのではなく、高精度の加算を行った後に16ビット精度に落とす。そのため、演算dでは32bit加算を行う。

図1のDFGをコスト1単位、演算時間1クロックの16bit加算器を1個、コスト2単位、演算時間1クロックの32bit加算器を1個、コスト3単位、演算時間2クロックの16bit乗算器を1個の演算器構成でスケジュールした結果を図2(a)に示す。16bit加算器、32bit加算器、16bit乗算器をそれぞれ1個ずつ必要とし、総演算器コストは6となる。16bitと32bitの2個の加算器はそれぞれ1回ずつしか使用されず、資源利用効率が悪い。

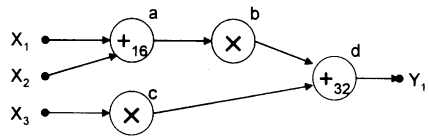

図1 データフローグラフ

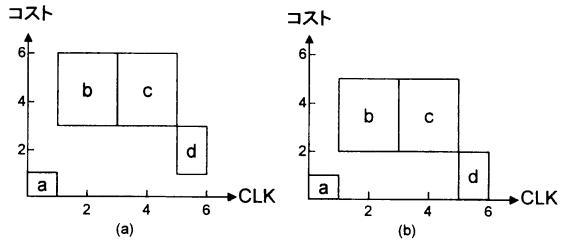

図2 演算スケジュール (a)演算精度ごとに専用の加算器を使用, (b)再構成可能加算器を使用

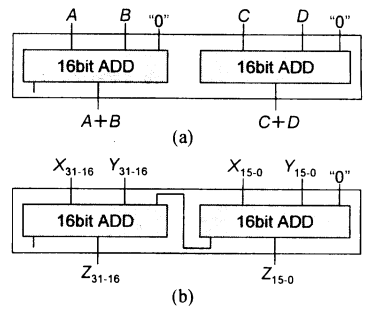

図3 再構成可能加算器 (a)2個の16bit加算器として使用, (b)1個の32bit加算器として使用.

精度を切り替えて演算実行可能な再構成可能演算器[1]を用いることで演算器コストを削減することができる。再構成可能演算器は、同一の演算器資源を低精度の演算と高精度の演算に切り替えて利用可能な演算器である。

図3に示す再構成可能加算器では、2個の単精度加算器として使用する場合(a)と1個の倍精度加算器として使用する場合(b)を切り替えて使用することができる。同一の加算器資源を低精度と高精度の演算で使用することで、資源利用効率を改善することができる。さらに図3の再構成可能加算器の場合、単精度加算と倍精度加算の間の再構成は極めて容易であり、回路規模は単精度加算のほぼ2倍である。また、もともと倍

精度演算が図 3(b)のように下位桁から上位桁への桁上げを行う回路構成を採用している場合、再構成可能加算器を用いたとしても動作速度の低下はほとんどない。以上のように、再構成可能加算器を用いることによる回路規模と動作速度のオーバーヘッドはごくわずかである。2 つの単精度加算器を用いて倍精度加算を行う再構成可能加算器を図 1 の DFG に適用すると、図 2(b)のようなスケジュールが得られる。同一の再構成可能加算器が、クロック 0 では 16bit 加算、クロック 5 では 32bit 加算を実行している。このスケジュールでは、再構成可能加算器、16bit 乗算器をそれぞれ 1 個ずつ必要とし、再構成可能加算器のコストは 16bit 加算器の 2 倍の 2 単位であるので、総演算器コストは 5 へと削減されている。

3.2. レジスタ再構成

単精度データ用の短ビット長レジスタを 2 個組み合わせて倍精度データ用の長ビット長レジスタとして利用するレジスタ再構成によってレジスタの利用効率を改善し、システムの最適化を図ることができる。例えば、図 4 のように 2 個の 16bit レジスタ(REG16)を用いて 32bit データを保持することができる。

4. 低コストシステムの探索

4.1. コスト関数

LSI 高位設計は，専用 LSI として実現したい処理内容、演算器仕様(演算実行時間およびコスト)、処理実行速度要求(最大クロック数)が与えられて、要求された処理実行速度を満たし、システムコストを最小化したスケジュールを得ることを目的とする。

構成 X のシステムコスト $C(X)$ を以下の式で見積もる。

$$C(X) = C_{fu}(X) + C_{reg} \cdot N_{reg}(X)$$

ここで、構成 X は演算器構成(演算器種類および各種類の演算器の個数)であり、

$C_{fu}(X)$:構成 X における総演算器コスト
C_{reg} :単精度レジスタ 1 個のコスト
$N_{reg}(X)$:単精度レジスタ数

とする。総演算器コスト $C_{fu}(X)$ は以下の式で与える。

$$C_{fu}(X) = C_{radd} \cdot N_{radd}(X) + C_{add} \cdot N_{add}(X) + C_{mul} \cdot N_{mul}(X)$$

ここで、各パラメータを以下のように定義する。

C_{radd} :再構成可能加算器 1 個のコスト
C_{add} :単精度加算器 1 個のコスト
C_{mul} :乗算器 1 個のコスト
$N_{radd}(X)$:再構成可能加算器数
$N_{add}(X)$:単精度加算器数
$N_{mul}(X)$:乗算器数

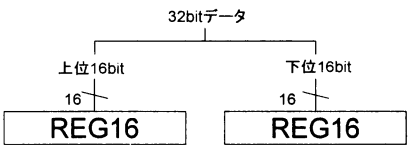

図 4　レジスタ再構成

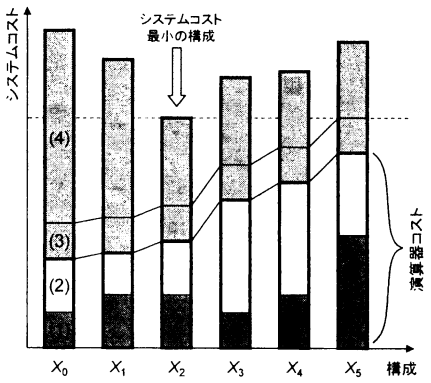

図 5　コスト最小システムの探索

4.2. コスト最小システムの探索

システムコストは、演算器コストとレジスタコストの和で与えられる。システムコストが最小のシステムの探索の様子を図 5 に示す。入力として与えられた処理を実行可能な最小コストの演算器構成 X_0 について、要求された実行速度を満たして処理実行に必要最小なレジスタコストを求める。図 5 において、(1)～(4)はそれぞれ加算器コスト、乗算器コスト、レジスタコスト下限、レジスタコスト(下限分を除く)を表す。(1)と(2)の和が演算器コスト、(3)と(4)の和がレジスタコスト、(1)～(4)の総和がシステムコストを意味する。レジスタコストの下限とは、処理実行に必ず使用されるレジスタのコストであり、例えば処理入出力データの保持に必要なレジスタなどによって決まる。

演算器コストが徐々に増える構成を X_1、X_2、…のように生成し、それぞれの構成についてシステムコストを最小化するようレジスタコストを最小化する。構成を X_4 まで生成したとき、最小のシステムコスト C_{min} を与える構成は X_2 となっている。X_4 の次に演算器コストの小さい構成 X_5 では、演算器コストとレジスタコスト下限の和((1)～(3)の和)が C_{min} 以上(この例では一致)となっている。レジスタコスト(4)は常に 0 以上であることから、構成 X_5 のシステムコストは C_{min} 以上であることが分かり、ここでシステムコスト C_{min} が与えられた処理について最小のシステムコスト(最適な構成は構成 X_2)であることが確定するので構成の探索を終了する。

以上述べたコスト最小システム探索手順を図6に示す。図6において N_{regLB} はレジスタコスト下限を単精度レジスタのコストに換算した場合のレジスタ数であり、$C_{reg} \cdot N_{regLB}$ がレジスタコスト下限を表す。初期演算器構成は次のように求める。すなわち、再構成可能加算器と乗算器が1つずつからなる構成についてレンジチャートガイドスケジューリング[5]を行い、要求された処理実行速度を満たし実行可能なスケジュールが存在するか調べる。スケジュールが存在しなければ、次に演算器コストが小さい構成を生成し、スケジューリングを行う。これを繰り返し、実行可能なスケジュールが存在する最小演算器コストの構成を初期演算器構成とする。

4.3. スケジューリング

演算の実行時刻が定まると、各演算結果データについて保持が必要な時間区間が定まる。処理実行の開始から終了までのすべての時刻に対して、同時に保持が必要なデータの最大数を求めると、これが必要最小なレジスタコストを決定する。

本手法で用いたスケジューリングアルゴリズムを以下に示す。資源制約されたレンジチャートガイド手法[5]に基づいており、入力として演算のスケジューリング優先順位と演算器構成を与え、レジスタコストを最小化するスケジュールを出力する。演算を順次選び、レジスタコストを最小化する時刻を割り当てており、局所最適解に陥る可能性があるが、計算複雑度が比較的低い利点がある。

演算 n のスケジュールレンジとは、その演算が実行可能な時刻の下限 LBn と上限 UBn に囲まれた時間区間である。この範囲外の時刻ではいずれかの演算間に先行制約が違反するので、この範囲内の時刻に演算実行をスケジュールしなければならない。演算器数制約や演算の先行制約からスケジュールレンジ内に演算が実行できない場合、スケジュール失敗とみなす。

1. まだ実行時刻の決定していない演算のうち、最も優先順位の高い演算 n を選ぶ。
2. 演算 n のスケジューリングレンジ[LBn, UBn]を求める。
3. 実行時刻 $t_n \leftarrow$ LBn、レジスタコスト $CR_{min} \leftarrow \infty$
4. 演算 n を時刻 t_n で実行すると仮定する。演算器数が超過する場合、6.へ進む。
5. 保持必要な時間区間が確定したデータについて、保持に必要なレジスタコスト CR を求める。
   $CR_{min} > CR$ ならば実行時刻候補 $\tau \leftarrow t_n$、$CR_{min} \leftarrow CR$
6. $t_n \leftarrow t_n + 1$、$t_n \leqq$ UBn ならば4.へ戻る。

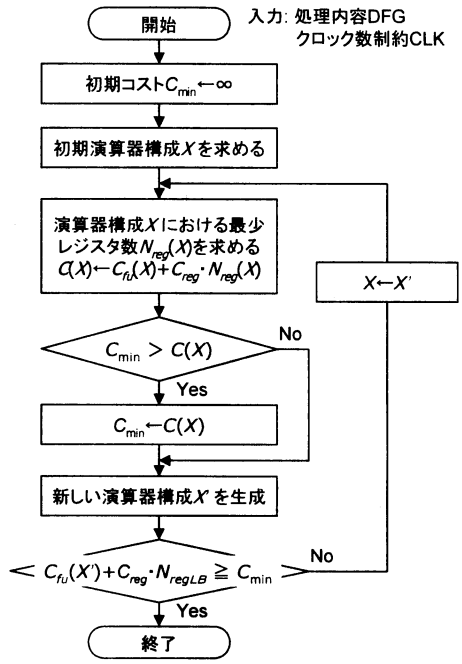

図6 最小コストシステム探索アルゴリズム

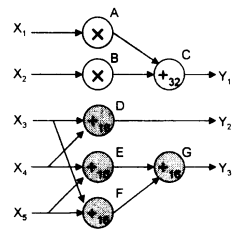

図7 DFG例

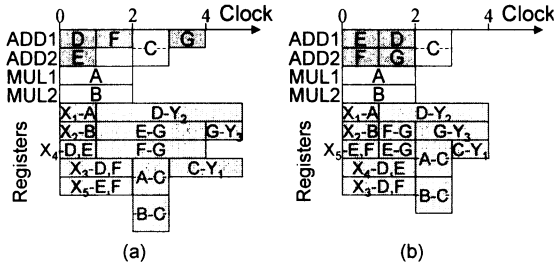

図8 スケジューリング結果
(a) 演算優先順位 a, (b)演算優先順位 b

7. $CR_{min} = \infty$ ならば、演算 n は実行できないので終了(実行可能なスケジュールなし)。さもなければ、演算 n の実行時刻を τ とする。
8. まだ実行時刻の決定していない演算が存在すれば1.へ戻る。

9. 演算スケジュールに基づいてレジスタコストを算出する。

4.4. レジスタコスト最小スケジュールの探索

4.3 節に述べたスケジューリング手法において、実行時刻を決定する演算の順序、すなわち演算の優先順位がレジスタコストに影響する。例として図 7 に示す DFG に対して実行時間制約を 4 クロックとし、演算の優先順位を高い順に A, B, C, D, E, F, G とした場合(a)と、A, B, C, E, F, D, G とした場合(b)のスケジューリング結果を図 8 に示す。図 8 から分かるように、レジスタコストは演算優先順位に依存する。そこで、Simulated Annealing（焼きなまし法, SA）を用いて最適な演算優先順位を探索する。

SA による最適優先順位探索を図 9 に示す。初期温度 T=20 とし、初期演算優先順位について 3.3 節で述べたスケジューリングを行ってレジスタ数を得る。新しい演算優先順位の生成は、いずれか 1 つの演算を選び、その優先順位を変更することで行う。例えば図 8 の例では、演算 D を選び、その優先順位を演算 E の次へと変更している。得られた演算優先順位についてスケジューリングを行ってレジスタ数を評価し、(1)前の演算優先順位よりレジスタ数が減少していた場合は新しい演算優先順位を採用、(2)前の演算優先順位と比べてレジスタ数が等しい場合は 50%の確率で新しい演算優先順位を採用、(3) 前の演算優先順位よりレジスタ数が増加した場合は温度 T に依存した受理確率 ((exp(T)/exp(20)×0.5)×100(%))で新しい演算優先順位を採用する。これを 1000 回行い、その後 T を 0.695% 下げる。T が 10 以下になれば終了する。T=20→10 としたとき、受理確率は 50%→0.002%へと変化する。

5. 実験

提案手法を C 言語によって計算機上に実装し実験を行った。実験に使用した計算機は、クロック周波数 2GHz、主メモリ 512M バイトである。設計対象処理として 8 ポイント 1 次元離散コサイン変換(DCT, 図 10)および 5 次ウェーブ楕円フィルタ(WEF, 図 11)を用いた。乗算後の加算を倍精度で行うと仮定し、乗算結果はレジスタ再構成を用いて 16bit レジスタ 2 個を用いて格納する。また、倍精度加算結果は上位 16bit をレジスタに格納すると仮定した。用いた演算器コストは、16bit 加算器を 1、32bit 加算器を 2、再構成可能加算器を 2、16bit 乗算器を 3、16bit レジスタを 1、32bit レジスタを 2 とした。

得られた結果を表 1 に示す。表 1 の項目は左から順に DFG、繰り返し周期(クロックステップ数)要求(I)、設計手法(method)、最小システムコストの場合の 16bit

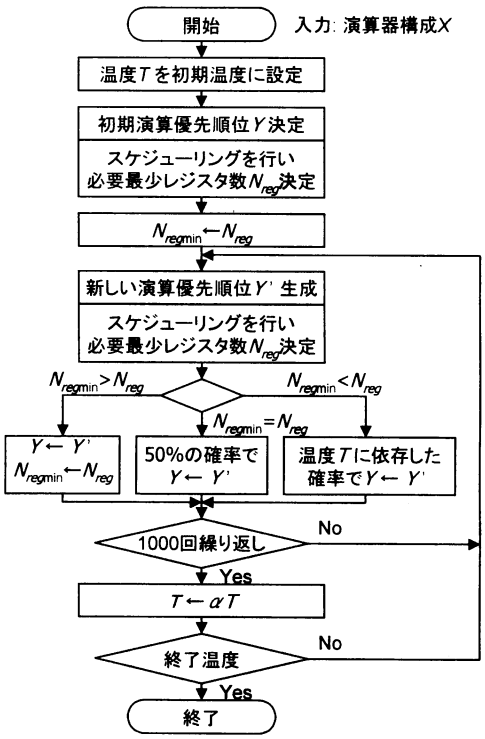

図 9　Simulated Annealing アルゴリズム

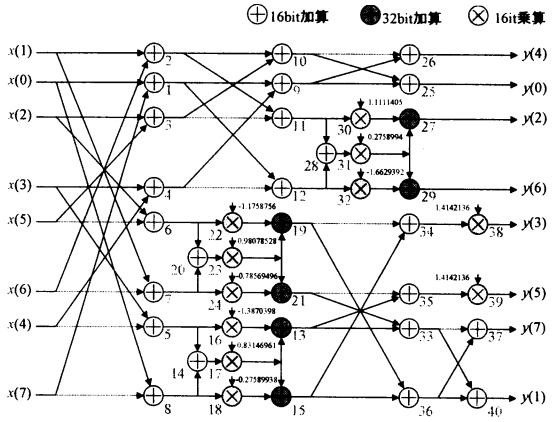

図 10　8 ポイント 1 次元 DCT

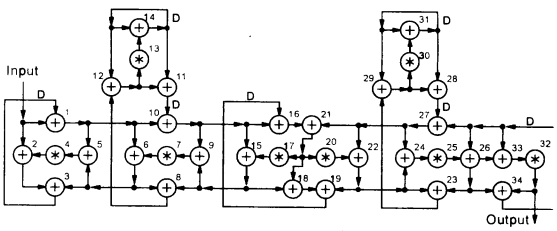

図 11　5 次ウェーブ楕円フィルタ

加算器数(A16)、32bit 加算器数(A32)、再構成可能加算器数(AR)、16bit 乗算器数(M16)、16bit レジスタ数(R16)、32bit レジスタ数(R32)、システムコスト(SC)、求解に要した CPU 時間を示している。設計手法は'w/o recon'は再構成可能加算器を用いない場合，'w recon'が再構成可能加算器を用いる提案手法を意味する。CPU 時間の単位は分(m)または時間(h)であり、例えば'>12h'は 12 時間経過しても設計が完了せず、途中で打ち切ったことを意味する。この場合、表 1 には打ち切りまでに得られた最良の設計結果を示している。図 12 は、処理 DCT、繰り返し周期 23 について、最適演算器構成が得られたときの SA によるレジスタコスト(16bit レジスタに換算したレジスタ数)の最小化の様子を示している。横軸は温度、縦軸は各温度における 1000 回の繰り返しで得られるレジスタ数の平均値を示す。この結果より、最終温度ではレジスタ数を最小にしたスケジュールが得られていることが確認できる。また、このときのスケジュール図 13 に示す。加算部分で縦に同じ数字(演算番号)が並ぶ部分で 32bit 加算を行っており、それ以外の部分では 16bit 加算を行っている。

表 1 から、再構成可能加算器を考慮することで提案手法によりシステムコストの削減された設計が得られることが確認できる。

処理が DCT、繰り返し周期要求が 8 のとき、再構成可能加算器を考慮する設計では 113 通りの演算器構成を調べている。多数の構成候補を調べるため、求解に比較的長い時間を要している。繰り返し周期要求が 23 の場合は、調べるべき構成候補は減少するが、各演算のスケジューリングレンジが大きくなり、スケジューリングに時間を費やしている。

再構成加算を考慮しない場合に求解時間が長いのは、同じ演算器コストとなる 16bit 加算器と 32bit 加算器の組み合わせが多く、結果として調べる構成候補が多く存在するためである。

6. まとめ

本研究では、精度の異なる加算に再構成可能加算を用いた低コスト LSI システムの高位設計手法を提案した。計算機実験により、要求される処理速度を満たす最小コストシステムを得ることができ、本手法の有効性が確認された。今後の課題として、最適演算器構成探索の効率化、演算器およびレジスタ間データ通信のためのマルチプレクサコストの考慮などが挙げられる。

表 1　実験結果

DFG	I	method	A16	A32	AR	M16	R16	R32	SC	CPU
DCT	8	w/o recon	4	4	-	6	12	6	54	>12h
		w recon	0	0	4	6	14	0	40	172m
	23	w/o recon	2	1	-	1	8	4	19	>24h
		w recon	0	0	1	1	11	0	16	401m
WEF	17	w/o recon	3	2	-	3	10	2	30	197m
		w recon	1	0	2	3	12	0	26	102m
	21	w/o recon	2	1	-	1	9	1	18	198m
		w recon	0	0	1	1	11	0	16	104m

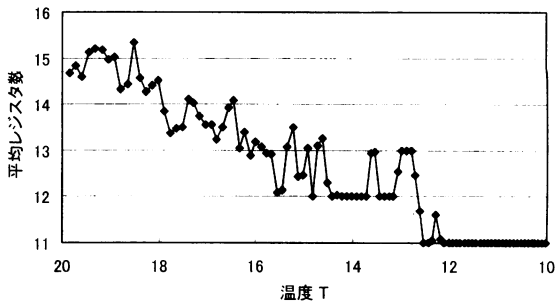

図 12　SA によるレジスタ数最少化の様子

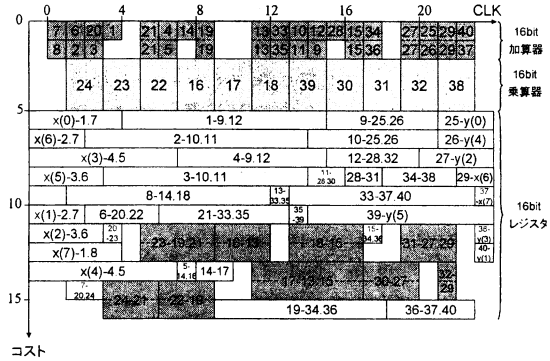

図 13　処理 DCT、繰り返し周期 23 の最小システムコストのスケジュール

文　献

[1] S. Perri, P. Corsonnello, and G. Cocorullo, "A High-Speed Energy-Efficient 64-Bit Reconfigurable Binary Adder," IEEE Trans. on VLSI Systems, pp. 939-943, 2003.

[2] U. M. Mesbah, C. Yun, H. Yasuura, "An Efficient Exploration Scheme for Datapath Width Optimization of Embedded Processor Systems," IEICE Tech. Report, VLD2001-150, pp. 10-16, Mar. 2002.

[3] B. Shackleford, M. Yasuda, E. Okushi, H. Koizumi, H. Tomiyama, H. Yasuura, "Memory-CPU Size Optimization for Embedded System Designs," Proc. DAC 1997.

[4] Y. Miyaoka, N. Togawa, M. Yanagisawa, T. Ohtsuki, "A Cosynthesis Algorithm for Application Specific Processors with Heterogeneous Datapaths," Proc. ASP-DAC2004.

[5] S. M. Heemstra de Groot, S. H. Grez, and O. E. Herrmann, "Range-Chart-Guided Iterative Data-Flow Graph Scheduling", IEEE Trans. Circuit Syst.- I: Fund. Theory & Appl., vol. CAS-39, pp. 351-364, 1992.