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

2004－ＳＬＤＭ－115 (10)
2004／5／28

SystemCを用いた動的再構成可能アーキテクチャPCAのためのシミュレーション法の提案

浅野 賢司† 北道 淳司†† 黒田 研一†

† 会津大学大学院コンピュータ理工学研究科
†† 会津大学コンピュータ理工学部
〒965-8580 福島県会津若松市一箕町鶴賀
E-mail: †{m5081207,kitamiti,kuroken}@u-aizu.ac.jp

あらまし 本稿では,動的再構成可能アーキテクチャPCAのためのSystemCを用いたシミュレーション法を提案する.提案手法を用いることによって,抽象度の高い要求仕様を記述するレベルから具体的なアーキテクチャ上における設計を行なうレベルまで,様々なレベルの設計に対するシミュレーションが可能である.本稿では,直交変換の一つであるアダマール変換を行なう回路について,一つのモジュールで変換全体を行なう要求仕様レベルから,実際にPCA上で動作するレベルまで段階的に設計を行ない,各レベルでシミュレーションが可能であることを確認した.

キーワード 動的再構成可能アーキテクチャ,SystemC,段階的設計,シミュレーション

Proposal of a Simulation Method for Dynamical Reconfigurable Architecture PCA using SystemC

Kenji ASANO†, Junji KITAMICHI††, and Kenichi KURODA†

† Graduate School of Computer Science and Engineering, The University of Aizu
†† School of Computer Science and Engineering, The University of Aizu ,
Tsuruga, Ikki-machi ,Aizu-Wakamatsu City,Fukushima 965–8580 Japan
E-mail: †{m5081207,kitamiti,kuroken}@u-aizu.ac.jp

Abstract In this paper, we propose a simulation method for dynamical reconfigurable architecture PCA using SystemC. Using proposed method, we are able to verify the specifications from the level of system requirement to the concrete level at which the system performs on the FPGA devices. In this paper, we design Hadamard transform circuit, one of orthogonal transforms, with step-wise refinement from system requirement level to concrete level at which the system runs on PCA, simulate each specification in SystemC and evaluate proposed method.

Key words Dynamical Reconfigurable Architecture, SystemC, Step-wise Refinement, Simulation

1. はじめに

近年,LSI製造技術の進歩によりシステムLSI,SoC(System on a Chip)のように大規模システムを1チップ上に実現することが可能となっている.一方,FPGA技術に基づき,システムに対する動的な処理の要求の変化や,内部リソースの使用状況の変化に応じて内部構成を変化させる動的再構成可能デバイスあるいはそれらとASICとの組み合わせによる実現に関する研究が行われている[1][2][3].動的再構成可能デバイスは必要に応じて回路動作を変更できるので,小規模のデバイスでもリソースを有効に利用できる可能性がある.

動的再構成可能デバイスへのシステム実装に関しては,いくつかのデバイスの設計事例およびそのデバイスにおけるアプリケーションの実装事例など報告されているが,それらの多くは,専用のシステムあるいは回路記述言語とそのデバイス専用のCADを有しており,ある設計を他のデバイスにリターゲットすることは困難である.

我々研究グループでは,SystemC[4]を用いた動的再構成可能アーキテクチャのためのシステム設計レベルからの段階的設計法を提案[5]している.本研究では,NTTが開発したPCA(Plastic Cell Architechture)を対象に提案手法を適用した.

PCAに対して,デバイスレベルでの記述およびPCAの実行単位であるPCAcellを複数使用するモジュールに対して内部構造を抽象化した記述をSystemCを用いて行ない,PCAの具体

レベルでのシミュレーション性能および抽象レベルでのシミュレーション性能を調べた. また, 信号処理における直交変換の一つであるアダマール変換について, 入力に対して変換を行ない出力することを一つのモジュールにおいて行なうという抽象度の高いレベルから具体的に PCA で動作するレベルまで段階的に設計を行ない, 各レベルにおいてシミュレーション性能を調べた.

以下,2 章では動的再構成可能デバイスのための段階的設計法,PCA の概略,SystemC を用いたシミュレーション法について,3 章では, 設計例題として信号処理における直交変換アルゴリズムの一つであるアダマール変換アルゴリズムの段階的設計例について述べ,4 章でまとめとする.

2. 動的再構成可能アーキテクチャPCA のための段階的設計法とシミュレーション法

2.1 段階的設計法

動的再構成可能アーキテクチャ上で動作するシステムのための回路モデルおよびそこからの段階的設計法について述べる. 設計初期における抽象レベルでは, システムを以下のように捉える. システムは, いくつかのモジュールにより構成され, それらは互いにインターフェースを通じてイベントを通信しあう. 各モジュールは, 受け取ったイベントおよびシステムの内部状態に応じて指定された演算を行ない, その結果に従って内部状態を更新し, 他のモジュールにイベントを送出する. 各モジュールは並列に動作を行なう. 各動作および通信には遅延が与えられる場合 (SystemC における Timed Functional Model に相当する), および遅延がない場合 (SystemC における Untimed Functional Model に相当する) の両方およびそれらが混在する場合が考えられる.

動的再構成可能アーキテクチャ上では, さらに, 各モジュールおよびインターフェースの動的な生成, 変更, および抹消が, 基本動作として定義される.

このような回路モデル上では, 設計 (詳細化) は, 一つのモジュールをより具体的な複数のモジュールを組み合わせて実現する, あるいは, インターフェースを具体化することとみなすことができる. 機能モジュールは, より具体的な遅延あり機能モジュールあるいは遅延なし機能モジュールを複数組み合わせて実現される.

最も具体的なレベルでは, 全モジュールおよびインターフェースが, 実際に用いる動的再構成可能 デバイス上での動作によって定義される. モジュールおよびインターフェースは, 用いるデバイスの遅延を反映した遅延のある機能モデルを用いて実現され, モジュールおよびインターフェースが静的に存在するように定義される.

抽象レベルで用いられていたモジュールとインターフェースの生成, 変更および抹消は, 例えば, マルチコンテキスト型 FPGA では, 各モジュールおよびインターフェースの配置および配線が, 対応するロジックブロックおよび配線用のスイッチおよびコネクションブロックの設定記憶に保持されるべきデータとして実現される. システム規模が大きな場合は, システム分割 (コンテキスト分割) が行なわれ, 動的なモジュールおよびインターフェースの生成, 変更および抹消が, マルチコンテキストの切替え (コンテキスト選択レジスタの値を変更すること), 使用されていないコンテキストへのデータの書き込みを行なう外部システム (FPGA 内部で実現することもできる) によって実現される.

本研究では, 動的再構成可能デバイスとして NTT が提唱, 実現を行なった PCA に対して, 提案手法を適用した. 各レベルのモジュールを SystemC を用いてモデル化する. **2.2** では, ターゲットデバイスである PCA について,**2.3** では, PCA を SystemC を用いてモデル化した例について述べる.

2.2 動的再構成可能アーキテクチャPCA

PCA(Plastic Cell Architecture) [3] は NTT 未来ネット研究所によって提案された動的再構成可能アーキテクチャであり,PCA-1 及び 2 がチップ化されている.PCA は, 非同期回路構成, 基本モジュールの 2 次元アレイ構成, 自律再構成可能などの特徴を有する.PCA は PCAcell と呼ばれるモジュールを 2 次元アレイ状に敷き詰めることによって実現されている (図 1(a)) .PCAcell は, 回路機能を実現する Plastic Part (PP) と複数の PCAcell で構成されるモジュール同士の通信および PP の再構成を制御する Built-in Part (BP) によって構成されている (図 1(b)) . PP は 8x8 個の BasicCell から構成される (図 1(c)). 各 BasicCell は 4 個の LUT(Look up Table) からなり, 組み合わせ回路あるいは記憶回路として動作する (図 1(d)).BP は入力データを命令とみなして実行し, 隣接する BP との通信経路の制御および PP の回路設定を行なう.

PCAcell 間は 4 相ハンドシェイク束データ方式の非同期通信 (図 2) が用いられる. 通信には data 信号,Request 信号,Acknowledge 信号が用いられる.PCA-1 では 1 つの data 信号は 4bit 幅のデータを用いて通信が行われる.

現在システムを PCA 上で動作させるために, 回路設計支援ツールとしてスケマティックエディタ PCA-SE およびシミュレータ PCASimII を用いて設計することが可能である.PCA-SE

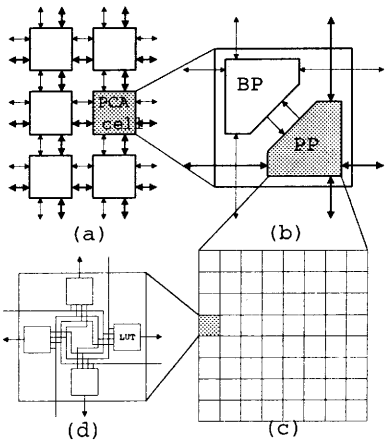

図 1 PCA の階層的構成の概要

を用いることによって, 各 LUT の設定内容を設計することができる.Scheme 言語を拡張した KL-Scheme を用いて, モジュールの配置, モジュール間の通信経路の設定などを設計することがができる. 以降これらの情報をシミュレーション実行情報と呼ぶ. シミュレータ PCASimII を用いることによってシミュレーション実行情報に対して, シミュレーションを行ない, 動作検証を行なうことができる.

2.3 PCA に対する SystemC モデル

PCA のデバイスレベルでのシミュレーションを行なうために,SystemC を用いて **2.2** で述べた PCA アーキテクチャを, 階層構造に基づいて設計した. 各モジュールの各動作は遅延をもち, それらの遅延は PCASimII の遅延モデルを参考にした.

SystemC の module として,PCAcell,BP,PP を構成する回路を対応させ, 階層 module 構成をもつ.PP の内部には,LUT の実行を process に対応させており,1PP モジュールは,5x8x8 個の process をもつ. 設計初期には PP の設計は,BasicCell および LUT のレベルまでモジュールとして実装し階層設計を行なっていたが, シミュレーション実行に必要となるメモリが多くなった. 以下で述べるシミュレーション実行結果では,PP のレベルまでモジュール階層を除去した設計を用いた結果について述べる.

BuildPart.{cpp,h}:BP に対応するレベルの記述. 入力されたコマンドに従い各命令を実行する動作が定義されている.BP の構造を図 3 に示す. 外部からのデータの受け渡しを行う入出力ポートは 5 個ずつあり,PP の制御用と PP の LUT へのアクセス用の計 12 個のモジュールから構成されている. 各モジュール間は PCAcell 間の通信と同様に, 束データ方式による非同期通信が行われる. BP では 4 方向の外部入力と PP からの入力の 5 方向からの入力を同時に処理する必要がある. そのため各ポートを独立して動作させるために SystemC のプロセス文により並列動作を記述した. 束データ方式は図 4 のように記述した.

PlasticPart_flat.{cpp,h}:PP に対応するレベルの記述.PP 内部にある 5x8x8 個の LUT を動作させ, 内部情報を入出力させる動作が定義されている.PP 内での BasicCell は 4 入力 4 出力の LUT により構成されている. この BasicCell への入力はそれぞれ独立して変化するため, 実装においては,LUT を, 入力とその遅延を処理するプロセスと,LUT の内容を読み出し出力するプロセスに分割した. 構成及びそれぞれの SystemC での記述を図 5 に示す.

提案する SystemC モデルに対して, シミュレータ PCASimII のためのシミュレーション実行情報を利用して動作することができるようにするため, シミュレーション実行情報を入力とし

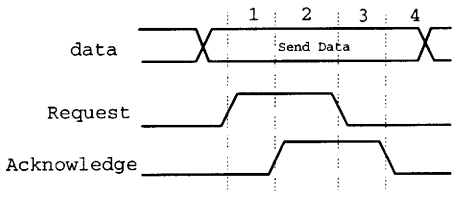

図 2　4 相ハンドシェイク束データ方式の非同期通信の例

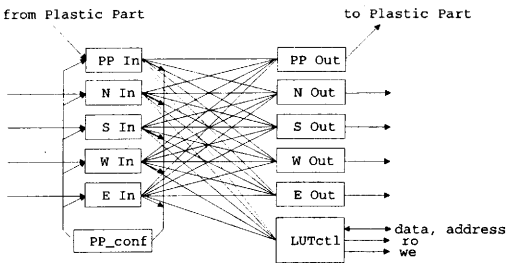

図 3　BP 内のモジュール間の通信

```
while(true) {
     while(in_req.read() != true) wait();
     data = in_data.read();
     in_ack = sc_logic(true);
     while(in_req.read() != false) wait();
     in_ack = sc_logic(false);
}
```

図 4　束データ通信の SystemC による記述

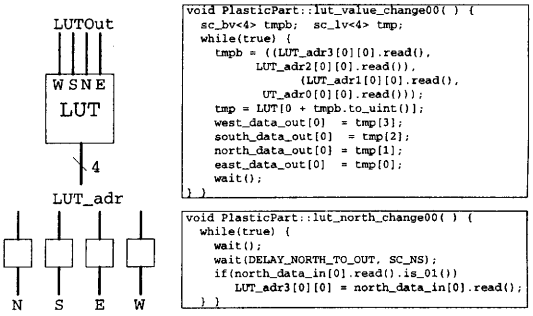

図 5　BasicCell の構成と SystemC による記述

FPGA 設定データ系列に変換し,SystemC モデルにダウンロードすべき PCA の端子にデータを転送する特別なモジュールを設計した.

設計した SystemC モデルによる実行時間と, シミュレーションツール PCASimII の実行時間とを比較した. 回路のサイズは, PCAcell6x6 である. 各回路には PCAcell6x6 の初期化および各データを実行モジュールまで転送させるために必要な回路動作に必要とする時間が含まれている. 初期化のみに必要な時間は,0.99sec. である. 各実行時間として, 5 回の試行の平均実行時間を示す.

PCA デバイスの異なる 4 種類の基本動作毎にサンプル回路を設計し, シミュレーション時間を計測した.Test_inc4 は,PP を使った組み合わせ回路（インクリメンタ）に対して 4096 個のデータを与え連続実行させる. Test_mm は,PP を記憶素子として 1024 個の連続データの書き込み読み出しを 16 回行なわせる回路である.Test_coci は,BP から PP に対して 1PCAcell の回路設定情報の書き込みと読み出しを 16 回繰り返す回路である. Test_bp_route は, 39PCAcell を通過する BP 間の通信経路を設定し,4096 個のデータを転送し, 通信経路を解除する動作を行なう回路である.

処理時間の結果を表 1 に示す.SystemC モデルに対する動

作環境は,Pentium4 1.8GHz 1Gmemory, RedHat9, gcc3.2.2 である. 比較とした PCASimII の動作環境は,Pentium3 933MHz 512Mmemory, WindowsXP である. SC1, SC2, SC3 は, それぞれ波形出力なし,PCAcell 間の通信のみ波形出力,PCAcell 間, BP-PP 間,BasicCell 間のすべての接続端子を出力した場合の実行時間である.SC の数字が多いほうが, 多くのシミュレーション情報を取得することが可能である. 出力された波形情報は, 別の波形ビューアを用いて動作を確認することができる.

	PCASimII	SC1	SC2	SC3
Test_inc4	6	3.73	12.8	1051
Test_mm	7	1.09	1.05	1.89
Test_cico	7	2.05	4.89	338
Test_bp_route	1.5	4.69	7.27	48.7

表 1 基本回路を SystemC で実行したシミュレーション時間 (sec.)

結果として,SC1 では Test_bp_route 以外は,PCASimII よりも高速にシミュレーションすることができた. しかしながら SC2,SC3 など記録する情報が増加するにつれて遅くなる.

PCASimII は,PCA の動作をグラフィカルに表示させ, いくつかのブレークポイントにより回路動作を一時停止させるなどの機能はあるが, 過去の波形を見ることができない. 提案する SystemC モデルにより波形出力を行なわせることで, 実行履歴を過去に遡り確認することができる. また SystemC に対応した汎用ツールを利用でき, 設計および検証が容易になる.

3. アダマール変換アルゴリズムの設計例

本章では, デジタル信号の直交変換処理の一つであるアダマール変換の高位から動的再構成可能デバイス PCA-1 で動作するレベルまでの設計に対して提案手法を適用した結果を述べる. アダマール変換はアダマール行列を基底関数とした直交変換である. アダマール行列の各要素は,+1 または-1 であり加算と減算のみを用いて計算が可能であるため, 直交変換としては小規模な回路にて実装できる. アダマール変換の概要と, 高速アダマール変換アルゴリズムを PCA に実装するための段階的設計について述べる.

我々の研究グループでは, アダマール変換アルゴリズムに対して, 同じ仕様から, アダマール行列の各要素を逐次生成し加減算を繰返し実行する実装 [6] と高速アダマール変換アルゴリズムを動的再構成が容易になるように実装 [7] を行なった. 今回は, 後者の高速アダマール変換アルゴリズムを段階的に設計した例を報告する.

3.1 アダマール変換と高速アダマール変換アルゴリズム

アダマール変換は式 1 ,2 に示す漸化式によって定義されるアダマール行列との行列演算により直交変換が行われる. このときアダマール行列の要素は 1 または-1 である.

$$H(2) = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad (1)$$

$$H(2^k) = \begin{bmatrix} H(2^{k-1}) & H(2^{k-1}) \\ H(2^{k-1}) & -H(2^{k-1}) \end{bmatrix} \quad (2)$$

$$F = \frac{1}{\sqrt{n}} Hf \quad (3)$$

アダマール行列の行における符号の変化の回数を交番数と呼ぶ. この交番数は三角関数による直交変換のときの周波数に相当する. 直交変換したあとの圧縮などの処理では交番順に並ぶことにより, 処理が簡単になる可能性がある. しかし, 式 1,2 のアダマール行列では各行が交番順には並んでいない. そのため交番順に並べられたアダマール行列は式 4 によって定義することができる.

$$F(u) = \sum_{x=0}^{N-1} f(x) \cdot b(x,u) \; where \; u = 0, ..., N-1 \quad (4)$$

$$b(x,u) = \sum_{i=0}^{b-1} x_i \cdot (u_{b-i} + u_{b-i-1}) \quad (5)$$

アダマール変換には, フーリエ変換における高速フーリエ変換に相当する高速な計算アルゴリズムが提案されている. 例えば, 交番順の 8 点アダマール変換は, 図 6 のように 2 点アダマール変換回路 (2pHT. 2 つの入力に対し, その和と差を出力する) とその間の規則的な接続によって実現することができ, 計算量は $O(n^2)$ から $O(n\ logn)$ に削減することができる. ただし, この変換には式 3 での $\sqrt{n}$ による除算は含まれていない.2pHT を図 6 のように構成し, 8 点高速アダマール変換が実現できる. 図 6 では, 偶数行目 2 番目以降の 2pHT の出力を入れ替えている. この交差によって, 全体を規則的に構成することができ, さらに出力が交番順で得られる.

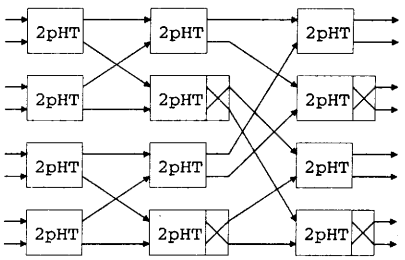

図 6 8 点高速アダマール変換アルゴリズムの概略図

3.2 設計手順

以下では, 一次元 8 点の 4bit 入力データに対するアダマール変換に対して, 段階的設計を行なった概要について述べる.

仕様レベルでは, アダマール変換を行なうモジュールが一つあるレベルである. そこから, 3 段階に設計を行ない PCA デバイスレベルまで設計を行なった. 設計の概要を図 7 に示す.

PCA アーキテクチャのチップ実現である PCA-1 では,PCA-cell へのデータ入力が 4bit であるため, アダマール変換への入力も符号無し 4 bit 整数とした. 出力に関しては, アダマール変換を行なって得られる出力値の範囲が-128～120 までになる為, 各モジュールのデータは符号付 8bit 整数として処理する. 外部からの入力データも符号拡張し符号付 8bit 整数として表現される.

Level1 での設計はアダマール変換の処理に対する設計の中

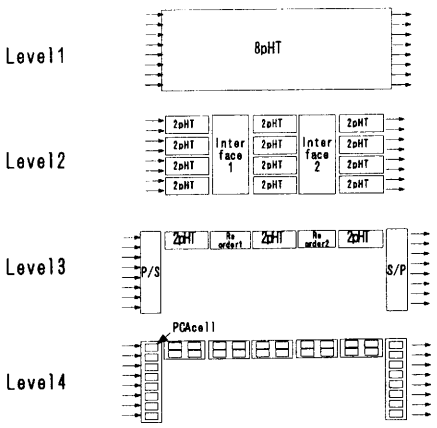

図 7 アダマール変換アルゴリズムの段階的設計の概略

で一番抽象度が高い設計である. 処理全体を行なうモジュール 8pHT が一つあり, 内部における処理のアルゴリズムは式 4, 5 を用いる. 入出力は, データ信号が 4bit × 8, 処理開始を表す start 信号, 終了を伝える finish 信号からなる. この設計に対して,SystemC を用いてテストベンチを作成した. 適切なタイミングの制御信号とテストデータをモジュール本体に与え, アダマール変換の処理結果を確認した. これ以降のレベルでの設計でも, 入出力は同じものを利用するため, 各レベルにおける動作確認にこのテストベンチを用いた.

アルゴリズムレベル (Level2) では高速アダマール変換アルゴリズムを採用し,12 個の 2pHT およびその間のインターフェース (Interface1,2) により実現している.2pHT は 2 入力 2 出力の回路であり, 式 1 より 2 点アダマール変換の計算は加減算により実現される. インターフェース 1, 2 はそれぞれ, 4 および 8 個の入力データに対して必要な交差を行なった結果を出力する.Level1 以降の設計では, 入出力の仕様は同じであり, 内部構成のみ詳細化される.

次のレベル (Level3) では, ハードウエア量を減らすために, 並列度を 1/4 に減らす実現を行なっている. 入力側において, 8 入力のパラレルデータを 2 入力のシリアルデータに変換する (モジュール P/S). その後,2pHT を実行する. 次に, インターフェース 1 が行なっていた交差を, シリアルデータに対して実施するモジュール Reorder1 を用いる. さらに 2pHT,Reorder2 を行なった後, シリアルデータをパラレルデータに変換 (モジュール S/P) して全体の出力とする. 並列度を削減した際の各モジュール P/S,S/P,Reorder1 および Reorder2 の回路の実現方法の詳細は, 文献 [7] に述べられており, ここでは省略する.

デバイスレベル (Level4) は, 各モジュールおよびインターフェースを PCA デバイス上で動作するレベルである. すでに PCA-SE を用いて設計された高速アダマール変換回路があり, そのシミュレーション実行情報ファイルを SystemC モデルに入力させることによりデバイスレベルのシミュレーションが可能である. このレベルで必要とする PCAcell は 118 個 (うち,BP のみ使用 81 個,PP のみ使用 6 個, 両方使用 31 個) であった.

3.3 シミュレーション結果

アダマール変換アルゴリズムに対して段階的設計して得られた各レベルにおいてシミュレーションを行なった結果について述べる.SystemC のシミュレーションは,**2.3** で述べた SC1 のレベルで行なった. シミュレーション結果を表 2 に示す. 各レベルにおいて入力データ数を 10000 および 100000 とした場合の実行時間を計測した.

抽象度が高くなるにつれてイベントの発生数が減り, 同じ処理内容であるが高速なシミュレーションが実行可能であることがわかる.

いずれのレベルにおいても初期化に必要とする時間が含まれているので, 実行時間が入力数に比例していない.

設計レベル／ 入力データ数	10000	100000
Level1	0.31	1.59
Level2	0.52	5.19
Level3	0.92	8.92
Level4	N/A	N/A
PCASimII	482	4555

表 2 アダマール変換アルゴリズムの各レベルを SystemC で実行したシミュレーション時間 (sec.)

この結果および **2.3** で述べた実験結果により, 従来 PCA に対して行なわれていた Level4 あるいは PCASimII を用いた設計レベルで直接設計するよりも, 抽象度の高いレベルから設計を行なうことにより, 高速なシステムシミュレーションを行ない, 設計初期における動作確認, あるいはモジュール間の通信量の見積りなどが可能である. これらの高位レベルでのシミュレーション結果を利用することにより, 下位レベルでの最適な設計を容易にする可能性がある.

Level4 におけるシミュレーションは, 実行途中において正しいシミュレーション結果が得られていない. この原因は PCASimII のタイミングモデルが一部正しく実現されてないためと思われる. 現在原因を調べているところである.

今回は十分な時間がなく実施できなかったが, 提案手法では, 複数のモジュール構成のシステムに対し, 異なるレベルが混在する状況でシミュレーションも可能である. これはあるレベルからの設計において全体を具体化するのではなく, 一部のみ具体化して全体シミュレーションが可能であり, これによってあるレベルの仕様から個々のモジュールを分散設計する場合, 有用と考えられる.

4. あとがき

本稿では, 動的再構成可能アーキテクチャのための段階的設計に対して,SystemC を用いたシミュレーション法を提案した. 仕様を記述するレベルから動的再構成可能アーキテクチャ PCA 上で動作するレベルまでの様々なレベルでシミュレーションを行ない提案手法を評価した.

現在の時点で,SystemC にモジュール単位での動的再構成の機能が実現されていないが, これらの機能が実現されれば抽象レベルにおいてモジュールを動的再構成することが可能である.

高速アダマール変換アルゴリズムの構成を, 任意の $(1/2)^n$ 倍のハードウェアリソースで構成することができる. 文献 [7] では, 並列度, 処理次数, 処理次元の各場合における動的再構成の方法についてまとめており, 本提案手法に基づき, 抽象レベルにおける動的再構成システムのシミュレーションを行ない, 提案手法を評価したい.

今後, 本手法の PCA 以外のデバイス, 例えばマルチコンテキスト型の動的再構成可能アーキテクチャへの適用などを考えたい. 本手法は,SystemC を用いて新たな動的再構成可能アーキテクチャのプロトタイピングおよび性能評価などにも有用と考えられる.

文　献

[1] 末吉, 飯田: "リコンフィギャラブル・コンピューティング," 情報処理,Vol.40, No.8,pp.777–782 (1999).

[2] T. Fujii, K.-i. Furuta, M. Motomura, M. Nomura, M. Mizuno, K.-i. Anjo, K. Wakabayashi, Y. Hirota, Y.-e. Nakazawa, H. Ito, and M. Yamashina: "A Dynamically Reconfigurable Logic Engine with a Multi-Context/Multi-Mode Unified-Cell Architecture ." ISSCC'99, pp.364–365(1999).

[3] K.Nagami, K.Oguri, T.Shiozawa, H. Ito and R. Konishi: " Plastic cell architecture: A scalable device architecture for general-purpose reconfigurable computing," IEICE Trans. Electron., Vol.E81-C, No.9, pp.1431–1437 (1998).

[4] T.Grötker, S, Liao, G.Martin and S, Swan (柿本他訳) : "SystemC によるシステム設計," 丸善 (2003).

[5] 北道, 黒田, "可変構造システムの段階的設計のための抽象モデルの提案,"FIT2003, 第一分冊,pp.313–314 (2003).

[6] 高橋, 北道, 西村, 黒田, "ブロックサイズおよび並列度を動的変更可能なアダマール変換回路の設計," 第 1 回リコンフィギャラブルシステム研究会論文集, pp.157–164(2003).

[7] 高橋, 北道, 黒田, "N 次元高速アダマール変換アルゴリズムの提案と動的再構成可能デバイスへの実装," 情処研報 SLDM113,pp.53–58(2004).