論　文

増殖する非同期ビットシリアル回路を表現できる設計記述形式とシミュレータの開発

高野　智明[†]　永本　太一[†]　樋口　準人[†]　柴田裕一郎[††]
小栗　清[††]

Design description form and simulator to express proliferatable asynchronous circuit with bit serial datapath

Tomoaki TAKANO[†], Taichi NAGAMOTO[†], Hayato HIGUCHI[†], Yuichiro SHIBATA[††], and Kiyoshi OGURI[††]

あらまし
C 言語の malloc や C++言語の new などによって，構造体やクラスのインスタンスを動的に作り出すことができるという点がソフトウェアの柔軟性の源となっている．このような自由度の高い資源利用を動的再構成ハードウェアを対象に行うためには，どのような記述形式が必要であるか，またその動作メカニズムはどのようなものであるかを議論し，再帰を許す回路図という記述形式をそのような目的のために使うことができることを明らかとした．さらに回路が動作中に他の回路を構成することができるという PCA（Plastic Cell Architecture）をビットシリアル処理に特化した Bit Serial PCA を前提に，増殖する非同期ビットシリアル回路の動作シミュレータ tanoqs を作成した．

キーワード　PCA 再帰表現 非同期 ビットシリアル シミュレータ

1. まえがき

フォンノイマン型コンピュータの発展の鍵はその汎用性と柔軟性にあり，ハードウェアとソフトウェアからなる構造が汎用性の，メモリ上のデータ表現やオブジェクトインスタンスを malloc や new 操作などによって動的に生成できるヒープ管理が柔軟性の源となっていると考えられる．

PCA [1] [2] [3] はこの汎用性と柔軟性に着目して提案された新しいコンピューティングアーキテクチャである．フォンノイマン型コンピュータがプログラム論理をベースとするのに対し，PCA は布線論理をベースとし，動作中の回路が新たな回路を生成して協調動作する事が出来るという特徴がある (以降このことを動的再構成機能と呼ぶ)．PCA の最初の LSI(PCA-1) は NTT により開発され，特徴的な機能のすべてが正しく動作する事が確認された．同グループは性能向上を目指した二つめの LSI である PCA-2 を開発した [4] [5]．我々は非同期ビットシリアル処理に特化する事により性能の向上を目指した Bit Serial PCA を提案している [6]．

しかし，PCA の最大の特徴である動的再構成機能を積極的に使ったアプリケーションの研究 [7] はあまり活発に行われていない．その理由は，動的再構成機能を使う為の malloc または new 操作の様な動的なインスタンスの生成方法が未だ提供されていない為であると考えられる．

そこで，ハードウェアにおける動的なインスタンスの生成を，再帰表現を用いる事で実現する記述方法を考案し，記述に対する物理メカニズムを決め，シミュレータの作成を行った．

本論文の構成は以下のようである．2. ではこれまで

[†] 長崎大学大学院生産科学研究科，長崎市
Graduate School of Science and Technology, Nagasaki University, 1-14 Bunkyo, Nagasaki-shi, 852-8521 Japan
[††] 長崎大学工学部，長崎市
Department of Computer and Information Sciences, Nagasaki University, 1-14 Bunkyo, Nagasaki-shi, 852-8521 Japan

に存在する動的要素の表現方法について整理し，3. ではハードウェアにおける動的要素の表現として，今回採用した形式とその採用理由について述べ，4. では今回作成したシミュレータ及び再帰表現による回路増殖の実際の回路との対応について説明し，5. では作成したシミュレータを使った増殖する回路の記述実験の結果について述べ，6. で全体をまとめる．

2. 動的要素の表現形式

この章では現在使用されている動的要素の表現法について述べる．

2.1 ソフトウェア

ソフトウェアにおいてリソースを動的に確保して使用しようとする場合，幾つかの方法がある．

2.1.1 malloc や new

c 言語においては malloc，c++言語では new を使用する事によってプログラムの動作中に領域を確保する事が出来る．malloc は確保する領域のサイズを指定する事によりメモリ領域の確保を行い，new ではインスタンスを指定する事によりメモリ領域の確保，インスタンスの配置を行う．new，malloc を使用して作りだした構造体やクラスのインスタンスなどは，ポインタ変数にアドレスを代入する事によって既存部分と接続する．動的要素の使用に関しては，領域確保，配置，接続が基本要素であると思われるが，ポインタ変数に対する値の代入は一般の変数に対する値の代入と同じ扱いであり，動的要素の接続という形式が特別設けられているわけではない．

2.1.2 再 帰 表 現

再帰表現を用いる事でも再帰呼び出しする関数のインスタンス (ローカル変数など) を動的に生成する事が出来る．再帰表現はスタックを用いる事で領域を確保し処理を行う．再帰表現では，再帰呼び出しを行う関数を単位として，その入出力関係，すなわち接続関係が記述されており，malloc，new に比べると動的要素の領域確保，配置，接続のすべてが形式として用意されていると考える事が出来る．ただしソフトウェアの再帰では処理中の関数は 1 つのみで，関数の処理が完了すればその関数のインスタンスは消されてしまうので再帰により作られた構造そのものを繰り返し使用するという事はない．

2.1.3 fork や thread 実行

fork によるプロセスの複製や thread の実行によっても動的に実行インスタンスを増やす事が出来る．

2.2 ハードウェア

上述の通り，ソフトウェアでの動的要素の表現は各種存在するが，ハードウェア記述言語でハードウェアを対象として動的要素の表現形式を用意したものは現在のところ存在しない．

3. 再帰構造の使用

動的に回路を生成して使用するという事を表現できる設計記述方式の検討に当たり，当初，空き領域を管理する回路が要求に応じて領域を払い出す malloc や new と同様の仕組みが使えるものと考えていた．しかし，払い出し要求がこの回路に集中してしまい布線論理の並列性を活かす事が出来ない，またハードウェアにはポインタ変数など存在しないので新しく配置したハードウェアと既存部分との接続関係を表現する為には新たな表現形式を考案しなければならないといった問題があった．

そこで，再帰表現が形式として領域の確保，インスタンスの配置，接続のすべてを持っている事に着目し，再帰表現をハードウェアの動的要素の表現として使えるのではないかと考える様になった．ただし，ソフトウェアの様な関数を単位とする再帰表現で，関数の呼び出しでインスタンスが生成され，関数の終了でインスタンスが削除されるというのではなく，回路の部品種を単位とする再帰表現で，特別な信号 new によってインスタンスが生成され，特別な信号 delete でインスタンスが削除されるという，再帰表現をベースとして new，delete の機能を加える事とした．例えば回路種 A の定義の中に部品として回路種 A が使われることを許し，回路種 A は特別な入力端子 new と delete を持つという様にする．この再帰表現ではポインタを使用した接続関係の表現とは違い，新たに追加した回路をどのようにして既存部分に接続するかが明示されており，これにより必要に応じて回路を作り出して使用するという設計を直接的に表現できる様になった．以下，この再帰表現による回路の動的割り付けを回路増殖と言う．

4. 回路増殖を表現するシミュレータ

非同期回路では回路を動的に構成する場合のタイミング制御が簡単となり，ビットシリアル回路では接続が広がらないため，動的構成のために接続部に設けられたスイッチによる性能低下が相対的に緩和されるとともに，演算回路を小さくできるため，データをメモ

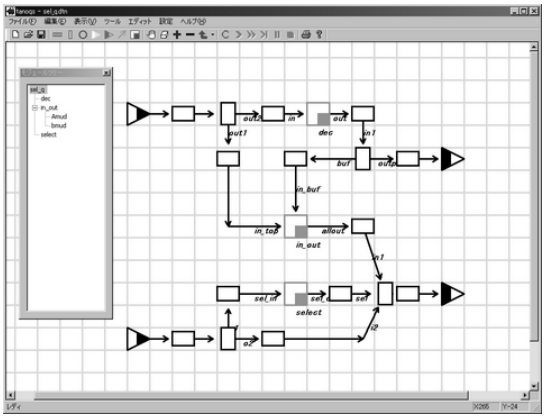
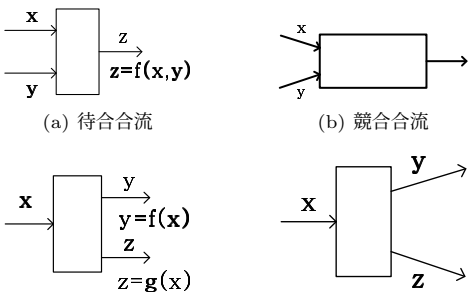


図 1　tanoqs の GUI
Fig. 1　GUI of tanoqs



(a) 待合合流　(b) 競合合流
(c) 複数分岐　(d) 選択分岐

図 2　処理の合流，分岐
Fig. 2　interfluent and branching process

リと演算回路の間を何度も往復させるのではなく，処理のままに配置されたたくさんの演算回路の中を流す事によって性能向上を達成できるので，我々は非同期ビットシリアル回路が将来の動的再構成アーキテクチャの主流になると考えている．このため再帰表現による回路増殖をどのように行うかも，この非同期ビットシリアル回路を前提に行う事とした．

これまでに非同期ビットシリアル回路の設計，動作検証を目的に非同期ビットシリアル回路をペトリネットでモデル化したシミュレータ QROQS を開発してきた [8]．しかし，設計する非同期ビットシリアル回路が大規模化するにつれ，QROQS での設計，シミュレーションは時間が掛かるようになってきている．そこで，より抽象度を上げることと再帰表現を可能にする事を目指した新しいシミュレータ tanoqs を開発する事とした．これは再帰表現を具体的にどうするかは設計記述表現の全体の中でしか考える事が出来ないため，閉じた世界の全体を tanoqs という形で実現してみるためである．従って以下，再帰表現を含む設計記述表現は tanoqs の機能という形で説明する．tanoqs は視覚的に回路の構築，検証を行える様マウスによるドロー系 GUI を採用した (図 1)．

まず，ペトリネットレベルから抽象度を上げた新たな構成プリミティブについて述べ，次に再帰表現をどのように取り込むのか，また物理的メカニズムをどのようにするのかについて述べる．

4.1　抽象度の向上

これまでに QROQS により非同期ビットシリアル回路で設計された FIR フィルタ，FFT，DCT 回路，雑音を排除する音声の圧縮方式である IMBE 用のデコード回路などを調査したところ典型的な回路パターンが数多く出現する事が分かった [9] [10]．典型的なパターンとは，ステートマシンとシフトレジスタ，カウンタである．ビットシリアル処理での四則演算はフルアダー＋フィードバックレジスタ程度で実現できるため汎用の制御構造であるステートマシンに含めてしまう事が出来る．FIR フィルタや FFT などの少し複雑な処理は四則演算と遅れ素子で実現できるため，ステートマシンに加えてシフトレジスタがあれば良い．シフトレジスタは記憶にも用いる事が出来るので，ビットシリアル処理に必要なプリミティブはステートマシンとシフトレジスタ，そしてビット列の区切りを管理するためのカウンタとなる．そこで，これらを tanoqs のプリミティブとする事により記述の抽象度を上げる事とした．カウンタをステートマシンに含めてしまう事も可能であったが，別にカウンタを用意する方が望ましいとここでは判断した．

4.2　処理の合流，分岐

IMBE の様な複雑なアルゴリズムは処理の合流，分岐によって実現される．非同期回路における処理の合流，分岐は以下の 4 通りとなる．

（1）　待合合流 (join)

図 2(a) が待合合流である．入力が複数必要な処理を行う場合，すべての入力が揃ってから処理が行われる．幾つの入力を待ち合わせるかは状況によって変化する．

（2）　競合合流 (conflict)

図 2(b) が競合合流である．一つの資源に対して複数の処理要求が出された場合に調停が必要となる．

（3）　複数分岐 (fork)

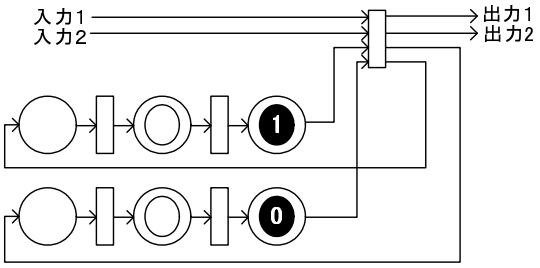


図 3　ステートマシンのペトリネット表現
Fig. 3　State machine in petri net expression

図 2(c) が複数分岐である．複数の結果を出力する場合である．幾つ出力を行うかは状況による．

（4） 選択分岐 (select)

図 2(d) が選択分岐である．いずれかに要求を出力する場合である．

4.3 プリミティブの定義

抽象度の向上によって抽出されたステートマシン，シフトレジスタ，カウンタと，合流，分岐機能を結合する事により，tanoqs のプリミティブを以下の様に定義した．

4.3.1 ステートマシン

ステートマシンは複数の状態を保持する事が出来，状態ごとに処理式，状態遷移式を記述し，どの状態が初期状態であるかを指定する事によりステートマシン全体の機能を表現する．ステートマシンは現状態の入力がすべて揃うまで待ち合わせを行う．また複数の入出力に対し，状態ごとに異なった部分を処理式，状態遷移式の中で指定することにより図 2(a) 待合合流のどれだけの入力を待ち合わせるか，また図 2(c) の複数分岐あるいは図 2(d) の選択分岐を表現できる．ステートマシンは QROQS で記述形式として採用したペトリネットで表現すると，待ち合わせと複数分岐を表現するために図 3 のようになり，入出力ともペトリネットのトランジションからの信号となる．従って QROQS と同様の方法で非同期回路へマッピングする事を前提とすると，ペトリネットにおいてトランジションとトランジションは接続できないのでステートマシン同士の接続は禁止されることとなる．

4.3.2 シフトレジスタ

シフトレジスタはステートマシンの入出力，バッファなどとしてデータを保持するためあるいは時間遅れを実現するために使用する．シフトレジスタの保持ビット数を設定する事により記憶ビット数だけでなく

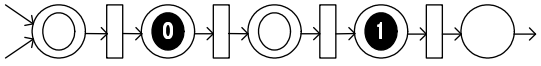


図 4　シフトレジスタのペトリネット表現
Fig. 4　Shift register in petri net expression

遅れ時間数を定義する．また，シフトレジスタは通常 1 入力 1 出力であるが，複数の入力を繋ぐ事により競合合流を表現することとした．シフトレジスタはペトリネットでは競合合流を表現するために図 4 のようになり，入出力はペトリネットのプレースからのものとなる．従ってシフトレジスタ同士は接続できず，シフトレジスタはステートマシン，カウンタ，ソーストランジション，シンクトランジション，モジュールに接続される．また，シフトレジスタはあらかじめ値を入力しておく事も出来る様にした．

4.3.3 カ ウ ン タ

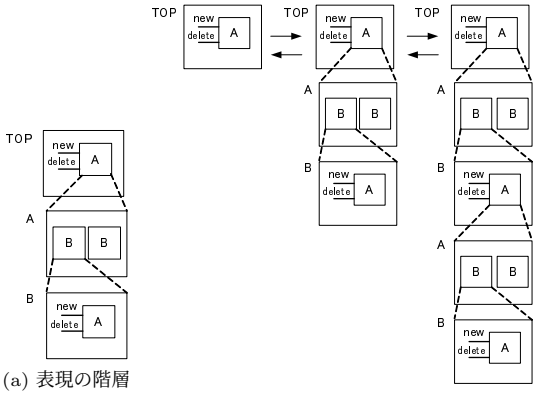
カウンタはカウント数を設定して使用する入力 1 つ出力 1 つのプリミティブである．カウント数は初期状態ではカウンタ変数にコピーされている．カウンタは入力が 0 であると 0 を出力し，入力が 1 であるとカウンタ変数を −1 し 0 を出力する．ただし，−1 した結果，カウンタ変数が 0 となる時は 1 を出力すると共にカウント数をカウンタ変数にコピーする．カウンタはペトリネットでは入力 1，出力 1 のステートマシンと同様になる．従ってカウンタの入出力はシフトレジスタと接続される．

4.3.4 入　出　力

シミュレーションを行う際のデータの入出力プリミティブとしてソーストランジション及びシンクトランジションを設けた．ソーストランジションからデータを入力し，シンクトランジションへデータを出力する．これらはシフトレジスタに接続される．また，これらは後述するモジュールの入出力としても使用される．

4.3.5 モジュールと階層構造

回路の規模の増大につれ，ステートマシンやシフトレジスタなどの構成プリミティブ数が膨大になり，接続関係が複雑化してしまう．また，各種演算などの構成が様々な場所で複数使用される事が多々ある．そのため，管理と構築の容易化，また回路の再使用を考え，同じ構成を持ち，複数存在する部分をモジュールとして階層化して設計を行える様にした．tanoqs 上ではモジュールプリミティブを追加し，上位モジュールから接続する事によってモジュール化と階層構造を表現できるようにした．また，この階層構造は後述する再



(a) 表現の階層関係　　(b) 実現される回路

図 5　再帰表現の例
Fig. 5　Example of recursive description

帰構造にも使用される．下位モジュールでは，上位モジュールからの入力はソーストランジションからの入力に見え，上位モジュールへの出力はシンクトランジションへの出力に見える．従ってモジュールの入出力はシフトレジスタに接続されなければならない．再帰を行わない場合，モジュールとの接続を行うソーストランジション，シンクトランジション共に無視され，ソーストランジション，シンクトランジションに接続された各々のシフトレジスタは連結して扱われる．

4.4 再帰構造

tanoqs では再帰表現により実行時に回路を任意個追加することができる．設計時に作成したモジュールに new という端子が存在する事により動作時にモジュールの作成を行うモジュール，delete という端子が存在する事により動作中にモジュールの削除を行うモジュールと判断される．このような new あるいは delete 端子を持つモジュールの配下に同一種のモジュールを含ませる事により任意個のモジュールを作成，削除できる．new や delete という端子の性質を設けただけでは任意個のモジュールを作り出し削除する事は出来ない事に注意されたい．再帰表現と組み合わせる事によってはじめて任意個のモジュールの作成，削除が可能となる．

図 5(a) に再帰表現を持つ回路の例を示す．図 5(a) 上段は TOP モジュール，中央は A モジュール．下段は B モジュールであり，A モジュールは new 端子と delete 端子を持っている．この回路が動作した時の回路構成の変化を図 5(b) に示す．初期状態では TOP モジュールのみの構成となる．TOP モジュール内の A モジュールの new 端子に要求が到着した時に回路の作成が行われ，回路構成は図 5(b) 中央の様になる．B モジュールは new 端子を持たないためそのまま展開される．図 5(b) 中央の B モジュール内の A モジュールは new 端子を持つため展開されない．図 5(b) 中央の B モジュール内の A モジュールの new 端子に要求が到着した場合さらに回路が作成され，図 5(b) の右の様になる．図 5(b) の右に示す回路構成で，上から 3 番目の B モジュールに含まれる A モジュールの delete 端子に要求が到着するとこの A モジュールが削除され図 5(b) の中央の様になる．さらに TOP に含まれる A モジュールの delete 端子に要求が到着するとこの A モジュールが削除され図 5(b) 左のトップモジュールだけの回路構成となる．もし図 5(b) の右の回路構成で TOP 直下の A モジュールの delete 端子に要求が到着すると，この A モジュールは削除されるがその中に含まれていた A モジュールはどこにも接続されない状態で残ってしまう．new と delete は本来独立であるが delete は new を持った場合に限って持つ事が出来る事とした．

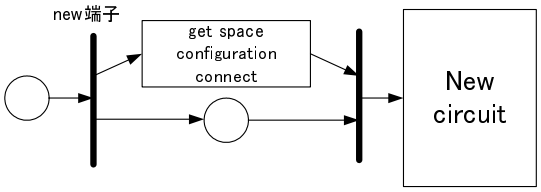


図 6　new 端子
Fig. 6　new signal

4.4.1 再帰構造の回路構造

new 端子を持つモジュールを直下に含むモジュールは new 端子を持つモジュールの回路情報を持ち，その回路情報を元に組み込み部に対して必要な空きスペースを作るためのコマンド，回路を構成するためのコマンド，構成した回路の端子を自モジュールに接続するためのコマンドを送出する回路を持つ．この回路は new 端子に要求が来た時に動作し，要求そのものは図 6 の様に new 端子を持つモジュールが完成し無事に接続されるのを待って，このモジュール内に入っていく．

delete 端子を持つモジュール，従って new 端子も持つモジュールを直下に含むモジュールは new の動作を行う回路に加えて delete を行う回路を持っており，new の時の情報を元に回路の削除と接続の解放を行う．

4.4.2 アーキテクチャに要求される機能

任意の回路を構成できる可変部が，コマンドを送りつける事によって通信や構成などのあらかじめ決まったより上位の機能を持つ組み込み部を使う事が出来るという点が PCA の本質である．従って再帰構造と new 端子や delete 端子による回路増殖や縮小を可能とするには組み込み部の機能として

- 空き領域の確保
- 回路の構成
- 回路の接続
- 回路の削除
- 接続の解放

があれば良いといえる．各々の詳細については別途述べる予定である．

4.5 シミュレーション

ここではプリミティブが構築され，実際にシミュレーションを行う際の流れを述べる．

4.5.1 チェ　ック

シミュレーションを行う前に，正しくプリミティブが構築されているかのチェックを行う．チェックの内容はステートマシンの各種式，各プリミティブの属性値，階層構造の接続関係である．また，階層構造を持つ場合，下位モジュールの内容を展開し接続する．new 端子を持ち回路増殖を行うモジュールは開始時点では展開しない．

4.5.2 実　　行

チェックによりエラーが発見されなければシミュレーションを実行する．シミュレーションは

（1） ソーストランジションにからのデータの取り出し

（2） シフトレジスタの処理

（3） ステートマシン，カウンタの入出力チェック

（4） ステートマシン，カウンタの処理，ステートマシンの状態遷移

（5） シフトレジスタの処理

（6） シンクトランジションへのデータの書き込みを順番に繰り返す．すべての場所で処理を行う事が出来なくなった時，シミュレーションを停止する．結果はシンクトランジションで確認できる．

4.5.3 new 端子と delete 端子

new 端子を持つモジュールは new 端子にデータが到着した時点でモジュールの内容を展開し，delete 端子を持つモジュールは delete 端子にデータが到着した時点でモジュールの内容を削除する．シミュレータは再帰構造であるかどうかをチェックしておらず，あくまで new と delete 端子だけの処理を行う．しかしながら本質が再帰表現との組み合わせにある事は先に述べたとおりである．

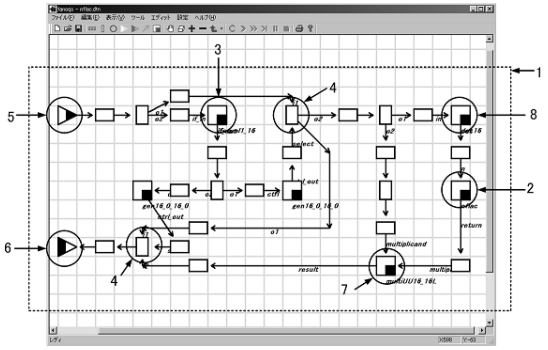


図 7　tanoqs で作成した階乗回路
Fig. 7　Factorial circuit in tanoqs

増殖を行うモジュールが複数の入力端子を持つ場合に new 端子にデータが到着する前に他の入力端子にデータが到着した場合はエラーとなりシミュレーションは停止する．new 端子にデータが来る前に delete 端子にデータが到着した場合もエラーとする．すでに存在しているモジュールの new 端子に到着したデータはそのままモジュールに渡される．なお，値に意味のある要求や信号をデータとしているが，受け取る側で値を使わない事も可能であるのでデータという用語でこれらすべてを表している．

5. 記 述 実 験

今回，再帰により処理を行う回路として，階乗計算を行う回路の記述を行った．

5.1 作成した回路

図 7 が tanoqs 上で作成した再帰により階乗計算を行う回路である．

図 7 の，1 は階乗器の定義，2 は再帰呼び出しを行うモジュール，3 は再帰を行うかを判断するモジュール，4 は再帰により新たに作られたモジュールに接続を切り換えるスイッチ，5 は入力，6 は出力，7 は乗算器，8 は減算器である．

5.2 処理の流れ

処理の流れを以下に示す．

（1） 図 7 の 5 からデータが入力される．データは LSB から入力される．

（2） 入力されたデータは図 7 の 3 において再帰を

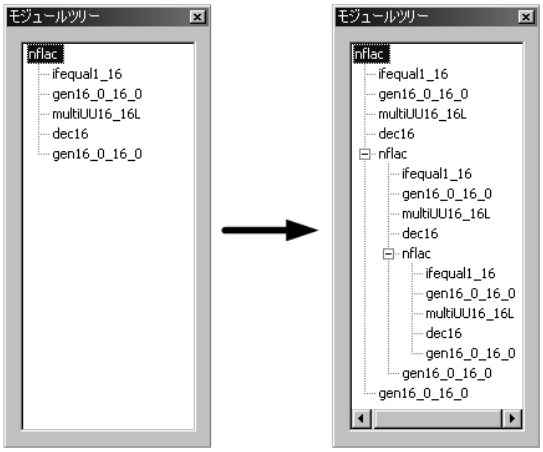


図 8　モジュール状態の変化
Fig. 8　Change of module's state

行うかどうか (入力値が 1 であるかどうか) 判定される．判定結果は図 7 の 4 のスイッチに送られ，再帰を行う場合，行わない場合のそれぞれが切り替えられる．

（3） 再帰を行う場合，データは分岐し，図 7 の 7 と 8 に送られる．8 ではデータを 1 減算する．減算されたデータは図 7 の 2 に送られる．図 7 の 2 のモジュールは new 端子にデータが到着した時点で再帰呼び出しが行われると判断し，自モジュールの内容 (階乗器) を展開し，到着したデータを入力として (1) からの処理を行う．

（4） 再帰を行わない場合は図 7 の 6 の出力からデータが出力される．再帰呼び出しが行われ，下位モジュールの入力データが 1 となった場合 (これ以上再帰呼び出しを行わない場合) は，出力されるデータは図 7 の 2 の出力となり，図 7 の 7 の乗算器により上位モジュールのデータ (1 引かれていないデータ) と掛けられる．その結果は再び図 7 の 6 から出力され，次々にデータは上位へと伝搬され，階乗計算が行われる．

5.3 実 行 結 果

図 8 に入力データを 3 とした時の回路状態の変化を示す．2 回再帰呼び出しが行われ，回路が生成された事が分かる．

6. む す び

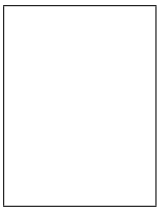
今回，増殖する非同期ビットシリアル回路の表現法として再帰表現を取り上げ，再帰表現により回路の増殖を設計記述できるシミュレータ tanoqs の機能として整理した．さらに PCA の組み込み部に必要とされる機能と物理的メカニズムとの対応を明らかにした．小さな回路について記述実験を行い，シミュレータが正しく動作する事を確認した．この回路はソフトウェアの再帰表現をそのまま回路にしているためハードウェアの並列性を引き出せていないが，再帰構造を持つ回路図の可能性を指摘した事の意義は大きいと考える．

今後は作成したシミュレータ tanoqs を元に記述実験を進め，ハードウェアの並列性と柔軟性をどのように活かすかの検討を行う予定である．

文　　献

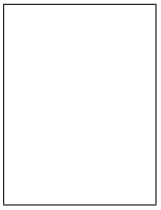
[1] 小栗清，“布線論理による新しい汎用情報処理アーキテクチャPCA(1)，”bit，vol.32，no.1，pp.27-35, 2000.
[2] 小栗清，“布線論理による新しい汎用情報処理アーキテクチャPCA(2)，”bit，vol.32，no.3，pp.54-62, 2000.
[3] 小栗清，“布線論理による新しい汎用情報処理アーキテクチャPCA(完)，”bit, vol.32，no.7，pp.51-59, 2000.
[4] H. Ito, R. Konishi, H. Nakada, H.Tsuboi, Y.Okuyama and A. Nagoya, “Dynamically Reconfigurable Logic LSI:PCA-2,”IEICE Transactions on Information and Systems, vol.E87-D, no.8, pp.2011-2020, Aug. 2004.
[5] H. Ito, R. Konishi, H. Nakada, H.Tsuboi and A.Nagoya, “Dynamically Reconfigurable Logic LSI designed as Fully Asynchronous System - PCA-2,” Proc. of COOL Chips VI, pp.84, Apr. 2003.
[6] Kiyoshi OGURI, Yuichiro SHIBATA, Nobuyoshi MURAKAMI, Akira KAWANO and Akira NAGOYA, “Asynchronous Bit-Serial Datapath for Objectoriented Reconfigurable Architecture PCA,” IEICE TRANS.INF.&SYST., vol.E82, No.1, JANUARY 2005.
[7] 神山知己，倉田圭吾，池畑陽介，北道淳司，黒田研一，“動的再構成デバイス PCA 上での自己複製アプリケーション設計容易化手法の提案と実装，” 情報処理学会研究報告 IPSJ SIG Technical Reports, vol.2005, No.8, pp.141-146, 2005.
[8] 坂本博和，永本太一，柴田裕一郎，小栗清，“非同期ビットシリアル回路シミュレータ QROQS の開発，” 信学論 (D-I)，vol.J88-D-I，no.2, pp155-162, Feb.2005.
[9] 永本太一，坂本博和，柴田裕一郎，小栗清，“低ビットレートボコーダ IMBE の固定小数点 DSP による実装，” 信学論 (D-I)，vol.J88-D-I，no.2, pp344-352, Feb.2005.
[10] 永本太一，坂本博和，小佐々武志，柴田裕一郎，小栗清，“アプリケーションからの PCA 基本要素の抽出，” 第 4 回リコンフィギャラブルシステム研究会，pp.87-94, Sep.2004.

（平成 xx 年 xx 月 xx 日受付）



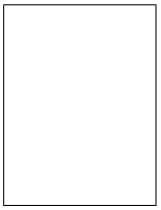
髙野　智明

平成 16 長崎大・工・情報システム卒．同年同大大学院生産科学研究科電気情報工学専攻修士課程に入学，現在に至る．計算機アーキテクチャ，ビットシリアル処理回路設計環境の研究に従事．



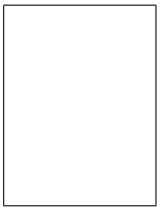
樋口　準人

平成 17 長崎大・工・情報システム卒．同年同大大学院生産科学研究科電気情報工学専攻修士課程に入学，現在に至る．計算機アーキテクチャの研究に従事．



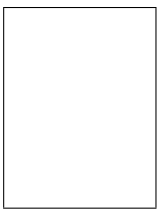
永本　太一

平成 12 年米子高専・電気工学科卒．平成 14 年長崎大学・情報システム工学科卒．平成 16 年長崎大学大学院・生産科学研究科博士課程前期了．同年同大学大学院生産科学研究科システム科学専攻博士課程後期入学，現在に至る．プラスティックセルアーキテクチャの研究に従事．



柴田裕一郎　（正員）

平 8 慶大・理工・電気卒．平 13 同大大学院理工学研究科計算機科学専攻後期博士課程了．博士（工学）．同年長崎大学工学部情報システム工学科助手．現在，同講師．平 10〜平 13 日本学術振興会特別研究員．リコンフィギャラブルアーキテクチャの研究に従事．平 15 年度本会論文賞．



小栗　　清　（正員）

昭和 49 年九州大学理学部物理学科卒業．昭和 51 年九州大学大学院理学研究科修士課程修了．昭和 51 年日本電信電話公社（現 NTT) 入社．平成 12 年同社退職．昭和 12 年長崎大学工学部教授，現在に至る．計算機アーキテクチャ，設計自動化，ハードウェア記述言語，論理合成システム，再構成可能ハードウェア，非同期回路，ビットシリアル処理の研究に従事．工学博士．電子情報通信学会，情報処理学会各会員．昭和 62 年元岡賞受賞．平成 2 年情報処理学会論文賞受賞．平成 4 年大河内記念技術賞受賞．平成 12 年電子情報通信学会業績賞受賞．

Abstract Flexibility of software realized that 'malloc' on C language or 'new' on C++ can make structure or instance of class dynamically. We disputed about what engineering form is needed and what is its mechanism to make dynamic reconfigurable hardware execute very flexible operation directly, and showed clearly to be able to use the description form of the circuit diagram which allows recursion for such purpose. Furthermore, we created the simulator tanoqs for the proliferatable asynchronous bit serial circuit on assumption that Bit Serial PCA specialized PCA which other circuits can be constituted while the circuit operated in bit serial processing.

Key words PCA, asynchronous, bit serial, simulator, recursive expression