システムＬＳＩ設計技術　103－19
（2001. 11. 30）

# 通信する非同期オブジェクトによるシリアル乗算器の実現

村上　宣義†

小佐々　武志‡　　柴田 裕一郎‡　　小栗 清‡

†長崎大学大学院 生産科学研究科 電気情報工学専攻

‡長崎大学 工学部 情報システム工学科

〒852-8521 長崎県長崎市文教町 1-14

E-mail: †nobu@cis.nagasaki-u.ac.jp,
‡{b698420, shibata, oguri}@cis.nagasaki-u.ac.jp

**あらまし** 本稿ではシリアルなデータ通信に容易に接続することができるシリアル乗算器を提案する。回路の動作が要求されない場合には消費電力を節約するために、乗算器は非同期回路として実現する。さらに、この乗算器は、一種類の乗算オブジェクトから成り、オブジェクトの動的な生成および配置によって可変長乗算を形成することができる。また、このPCA(Plastic Cell Architecture)での実装についても述べる。

**キーワード**　PCA、非同期回路、乗算器、シリアル回路

# Implementation of a serial multiplier by communicating asynchronous objects

Nobuyoshi Muakami †

Takeshi Kozasa ‡,　Yuichiro Shibata ‡,　Kiyoshi Oguri ‡

† Department of Electrical Engineering and Computer Science,
Graduate School of Engineering Science, Nagasaki University

‡ Department of Computer and Information Sciences,
Faculty of Engineering, Nagasaki University
1-14 Bunkyo-machi, Nagasaki, 852-8521, Japan

E-mail: †nobu@cis.nagasaki-u.ac.jp,
‡{b698420,shibata,oguri}@cis.nagasaki-u.ac.jp

**Abstract** This paper proposes a serial multiplier that can be easily connected to serial data communication interface. To save the power consumption when the operation is not required, the multiplier is implemented on an asynchronous circuit. In addition, the multiplier consists of a single sort of one-bye-one multiplication objects and any width multiplier can be configured by dynamic generation and placement of the objects. Implementation of the multiplier on PCA(Plastic Cell Archtecture) is also shown.

**Key words**　PCA, Asynchronous circuit, multiplier, serial circuit

## 1. はじめに

一般に、データ伝送はシリアル、つまり直列な転送で行われる。ここで、並列な演算器を回路上に作成した場合、計算するデータは並列として扱う必要がある。だが、入力されるデータはもともと直列で与えられるので、結局は、直列（入力）→並列（変換して計算）→直列（出力）という流れで計算しなければならない。つまり、一度入力されたデータを並列に変換して計算し、計算後は結果を直列に直して出力しなくてはならない。この方法だと、データを変換して計算する必要があるから、それだけ全体のスピードが低下すると考えられる。それならば、並列に変換して計算するような回路よりは、直列すなわちシリアルなまま計算できる回路の方が速度向上を望めるはずである。以上のことからシリアルなまま計算する回路を作成し、その処理速度向上を目指す。すなわち、1 ビットずつ入力、そのまま計算して 1 ビットずつ出力するような演算器である。

また、演算処理が必要でないときの消費電力を抑えることが出来る非同期回路への設計を目標とし、1 種類の“1 ビット×1 ビットの乗算オブジェクト”での可変長乗算の実現を目指し、またそれをＰＣＡに実装することを目標とした。

本稿の構成は以下の通りである。2 章はＰＣＡについての基本的な説明を行う。3 章では乗算器の基本的な形を示し、1 ビット分のシリアルな乗算回路の例を挙げる。4 章では非同期回路に用いられるペトリネットの説明を行う。5 章では非同期で動作するステートマシンについて解説し、また回路の例を挙げた。6 章では非同期で動作する乗算オブジェクトの設計例を示し、その動作について述べる。7 章では乗算オブジェクトの結果について述べ、8 章はまとめである。

## 2. ＰＣＡ

PCA(Plastic Cell Architecture)は、FPGA よりもさらに汎用性を向上させ、CPU とメモリによるコンピュータの汎用性に近づけたアーキテクチャである。PCA は構成可能な回路である可変部と、これを制御し通信する組み込み部からなるプラスティックセルを 2 次元メッシュ状に並べ、接続したものである。プラスティックセルの可変部は隣接するプラスティックセルの可変部と回路を作るために接続し、プラスティックセルの組み込み部は、隣接するプラスティックセルの組み込み部と通信するために接続し、組み込み部と可変部は、組み込み部から可変部を制御するため、組み込み部が可変部に通信要求を送るため、可変部が組み込み部に通信要求を送るため接続する。

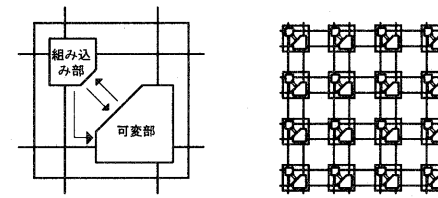


図１：ＰＣＡの構造

### 2．1　ＰＣＡ命令セット

組み込み部には独立に動作する入力ポートが 5 つあり、これは 2 次元メッシュ状に構成されたプラスティックセル上でルーティングスイッチを形成するとともに可変部を制御する。入力ポートは 5 つとも同一構造であり、それぞれ東西南北あるいは可変部からの要求を受け付け、入力ポートの状態に応じた処理を行う。要求は 10 種類のコマンドで行われ、5 ビットでコーディングされている。以下に命令コードを示す。

表１：コマンド

| | | |
|---|---|---|
| ＣＬＥＡＲ | ０ＸＸＸＸ | 経路開放 |
| ＯＰＥＮ | １０００Ｘ | 接続 |
| ＣＬＯＳＥ | １００１Ｘ | 切断 |
| ＣＯ | １０１０Ｘ | 書き込み |
| ＣＩ | １０１１Ｘ | 読み出し |
| ＷＥＳＴ | １１０００ | 経路設定（西） |
| ＮＯＲＴＨ | １１００１ | 経路設定（北） |
| ＥＡＳＴ | １１０１０ | 経路設定（東） |
| ＳＯＵＴＨ | １１０１１ | 経路設定（南） |
| ＰＰ | １１１ＸＸ | 経路設定（可変部） |

これらのコマンドのうち、経路設定に関するものが 5 つある。入力ポートはこれらのコマンドを初期状態で受け取ると、これらのコマンドを消費するとともに後続のコマンドを指定された向きへ転送し続けるように、すなわちシフトレジスタとなるように設定される。

### 2．2　オブジェクト

可変部に構成された回路のうち動作中には変更さ

れず、組み込み部としか接続をもたないものをオブジェクトと呼ぶ。オブジェクトは回路構成の単位とし、これはオブジェクト指向プログラミングのオブジェクトと同様、処理とデータが一体になったものと考えることができる。

複数のオブジェクトが存在する場合、オブジェクト間の通信は組み込み部を介してのみ可能となる。

## 3. 乗算器

まずは今回作成した乗算器の構成を述べる前に、単純なシリアル乗算器について述べることにする。

最も基本的な乗算は、2 個の正の 2 進数で積の生成を行うことである。これを筆算のやり方で見てみる（図 2）と、部分積と部分和とシフトから成立しているとわかる。ここで 2 進の 1 ビット×1 ビット乗算は論理 AND 演算と等価である。

```
              X2X1X0 被乗数
×             Y2Y1Y0  乗数
                X0Y0
              X1Y0
            X2Y0
              X0Y1
            X1Y1
          X2Y1
            X0Y2
          X1Y2
＋    X2Y2
  P6P5P4P3P2P1P0   積
```

図 2：筆算による乗算

図 3 にシリアルな乗算回路の 1 ビット分の例を示す。この構造では、シフトされた部分積の列を順次加算することによって、乗算が実行される。上部から入力されるのが被乗数であり、左方から入力されるのが乗数である。

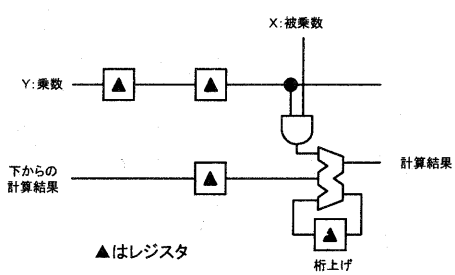


図 3：1 ビット分の乗算回路

図 4 にシリアルな乗算回路の全体例を示す。被乗数はあらかじめ入力されており、左から下位のビット、すなわち LSB から入力され、右になるにつれて上位のビットとなる。一番右が MSB である。被乗数は常に AND 演算を通じて加算回路で待ち合わせをしている。

入力される乗数は下位のビットすなわち LSB からであり、順次入力され回路をシフトしていく。計算結果は下位のビット、すなわち LSB から出力される。

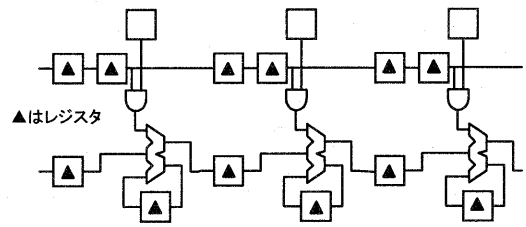


図 4：並列接続した乗算回路

同期回路で作成する場合には、図 4 にあるようなレジスタを組み込んでいればタイミングに関しては問題なく動作し、正しく結果が出力される。ここで、計算の途中結果が乗数の流れるスピードの倍の速さで流れるということが同期でのタイミングをとる点において重要である。しかし、今回目標としたのは非同期回路による乗算回路である。従って、回路間のハンドシェイクなどが重要な要素となる。これを非同期で設計するとなると、ハンドシェイクやハザードの問題がからみ非常に考えにくい。よって次に非同期設計において見通しのよいペトリネットを用いて考えることにする。

## 4. 非同期設計

### 4．1 ペトリネットでの表現

ペトリネットとは並行システムのモデル化言語である。ペトリネットは、プレースとトランジションという二種類のノードをもつ。プレースは円で表されるノードであり、条件を表す。トランジションは棒または箱で表されるノードであり、事象を表す。そして、これらを結ぶアークは条件、事象の間の関係を表す。プレース、トランジション、アークがシステムの構造を表現する。プレースの上には、トークンが置かれる。トークンは点で表され、条件の成立を表す。プレース上のトークンの配置をマーキングといい、システムの状態を表す。図 5 にペトリネ

ットの例を示す。

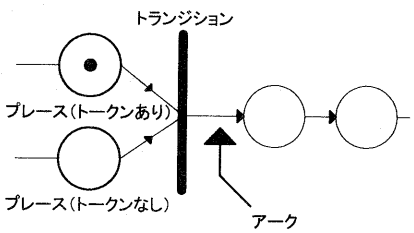


図5：ペトリネットの例

この例では上のプレースにトークンが入っており、トランジションの前で下のプレースにトークンが入るのを待っている状態である。下のプレースにトークンが入ると発火、トランジションを越えて一つ先のプレースへトークンが遷移する。発火したあとのプレースはトークンが空になり、次のトークンを待つ状態になる。

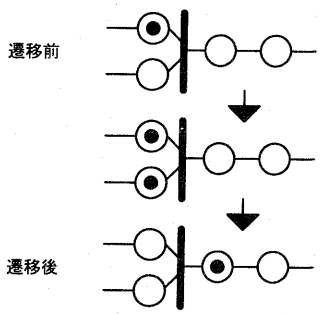


図6：状態の遷移

トランジションが発火するためにはトランジションの前にトークンが揃う必要がある。また移動先のプレースが全て空であるという条件も必要であるとする。

このペトリネットを使用した考え方は非同期回路設計において非常に役に立つ。トランジションは待ち合わせとして、プレースは入力線として、入力線の値をトークンとして考えることができる。

### 4．2　C素子

非同期設計において、基本回路素子としてのMullerのC素子はなくてはならないものである。回路としては図7のようなものを例として挙げる。

C素子は入力がともに1になったときに出力が1になり、入力がともに0になったときに0になるトランジスタ回路で、内部に記憶を持っており入力が異なっている間は直前の値を保持し出力は変化しない。出力が反転することをC素子が反応するということにするなら、反応させることのできる入力値というものを考えることができる。C素子の出力が1であるなら反応させることのできる値は0であり、両方の入力が0になるとC素子が反応して出力が0になる。このとき入力0はもはやC素子を反応させることのできる入力値ではないので、これはまさしくペトリネットのトランジションの発火とプレースのトークンの関係である。従って、C素子をトランジション、C素子の反応を発火、入力線をプレース、入力線の反応をさせることのできる入力値をトークンと対応させることができる。図8にC素子回路のペトリネットでの表現を示す。

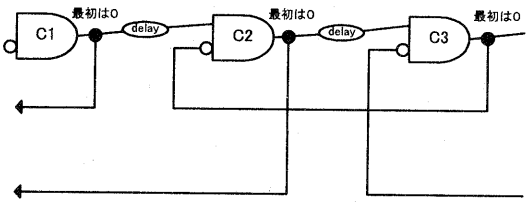


図7：C素子回路

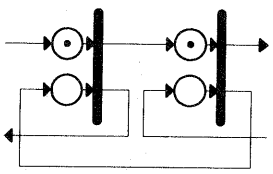

図8：ペトリネットでの表現

### 5. ステートマシン

乗算器の制御にはステートマシンが必要になる。データ保持にはレジスタを用いずにラッチとC素子を用いる。そのステートマシンをペトリネットで表現したのが図9である。(図9の上部はラッチを含んだ回路であり、下部はペトリネットでの表現である)

ループしているプレースをトークンが遷移し、1と0のトークンが常に存在しているものとする。1は黒丸（●）で、0は白丸（○）で表現した。回路は閉じているのでトークンが発火により消滅することはない。下のプレースにトークンが入力され、その上部のプレースにトークンが遷移した時に発火する。これは回路のラッチが動作してデータを記憶、データはそのまま論理回路へと流れ出力されることに対応する。ラッチは3つ存在しここでデータの保持を行う。発火したトークンは2つ出力される。片方は先のプレースへ、もう一つはラッチでのデータ保持のため回路を戻り後ろのプレースへ入力される。

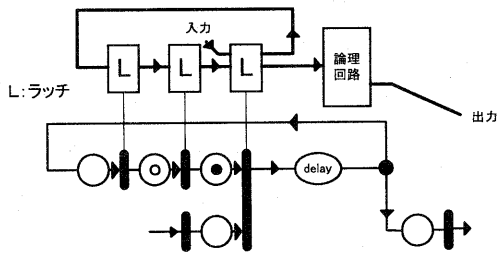


図 9：ステートマシンのペトリネット

トークンの遷移を図 10 に示す。太い黒線で表現されたところがトランジションの発火、すなわち C 素子が 1 または 0 になったということを表す。閉路である部分に常にトークンがあり、その閉路で反応した C 素子に対応したラッチにおいてデータの取り込みが行われる。データは論理回路に流れると同時にデータの保持のため閉路を巡回しつづける。次の入力がないかぎりはそのデータを保持し続ける。

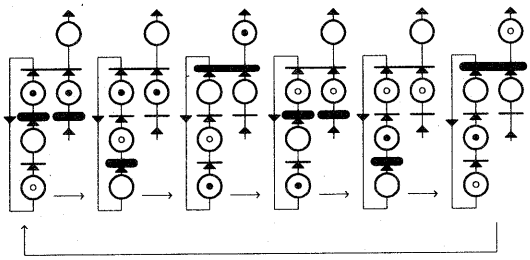

図 10：トークンの遷移状態

## 6. 乗算器の設計

ここでは非同期かつシリアルで動作するような乗算オブジェクトの PCA 上での設計について述べる。

### 6．1　オブジェクト

PCA セルを用いた「1 ビット×1 ビット乗算オブジェクト」（以下乗算オブジェクト）の動作を説明する。今回はこの同じ乗算オブジェクトを用いて n ビット×m ビットの乗算を行うことを目的とした。以下、簡単な例として 4 ビット×4 ビットの乗算を用いて説明する。

この乗算オブジェクトは 3 つの状態を内部に持っており、最初に行われる初期化直後は初期状態である。状態、および乗算オブジェクトが持つ機能を表 2 にまとめた。

以下このコマンドについて詳細に説明する。

表 2：乗算オブジェクトが持つ機能

| 場合 | 状態 | データ | 説明 |
|---|---|---|---|
| ① | X | １１ＸＸＸ | Through |
| ② | 準備 | １００ＸＢ | 被乗数取り込み |
| ③ | 演算 | １００ＸＸ | Through |
| ④ | X | １０１ＡＳ | 演算 |
| ⑤ | X | ０ＸＸＸＸ | 初期化（未設定） |

① PCA における経路設定のためのコマンドである。乗算回路には最初にこの経路設定データが必要であり、これにより回路の経路をどこにとるか、また組み込み部と可変部を接続するかを決定する。これは PCA での経路設定であるので、乗算オブジェクトの回路によらない。よって回路にデータは流れるが回路で処理する必要はないのでそのままスルーするものとした。

② このデータにより乗算における被乗数を設定する。被乗数は固定であり、回路を流れるのは乗数とする。これは筆算においても被乗数は常に固定であり、乗数が変動するのと同じイメージである。入力データの LSB を回路に取り込み、データ自体は出力せずに次のデータを待つものとする。

③ 被乗数設定と同じデータであるが、すでに回路に被乗数が入力されているので、これはデータをそのまま次のオブジェクトにスルーするものとする。つまりオブジェクトの被乗数設定は最初に LSB から入力し、決定されるものとする。

④ 演算を行うデータである。A の部分に乗数部分のデータ、S は下のビットからの計算結果である。入力されるデータとしての乗数も LSB から入力されるものとする。
演算した結果の出力、すなわち解答は出力の LSB が相当する。
ここで演算において、必要なデータだけ回路に流せばよいわけではなく、回路のトークンを動作させるために乗数データのあとにデータ 0 をいくつか送る必要がある。

⑤ 回路のクリアを行うデータである。しかし、今回作成した回路ではこれについては考慮していない。

### 6．2　乗算の全体例

図 12 に 4 ビット×4 ビットの全体例を示す。乗算は図のように 4 個の乗算オブジェクトでできる。

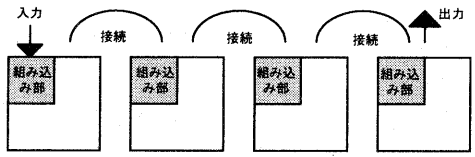


図 12：乗算器全体のイメージ

ここで、同期での乗算と同じく計算データの部分は乗数よりも 2 倍の速度で流れる必要がある。タイミングを合わせるために回路の加算器で被乗数、乗数、桁上げを全て C 素子で待ち合わせして加算を行う。

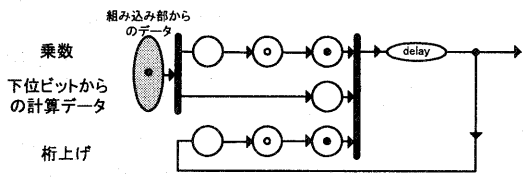


図 13：ペトリネットでの加算タイミング表現

トークンが発火するためには発火先であるプレースが全て空でないといけないので、初期状態では図 13 のようにトークンを設定し、これにあわせラッチを用意し順次トークンを動作させ値を保持していけば上述のようなタイミングをとることができる。

## 6．3　実装

回路が最初に初期化された時点で回路上にトークンが設置され、乗算オブジェクトは動作可能になる。図 14 に実際の 1 ビット×1 ビットの乗算オブジェクトの回路を示す。データ判別にデコーダを用い、スルー、被乗数とりこみ、演算を決定している。

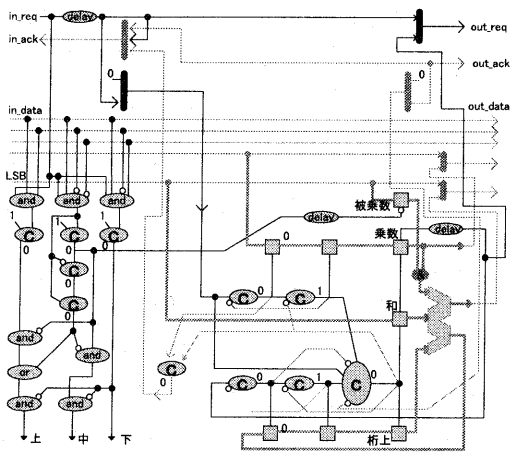


図 14：乗算オブジェクトの回路図

データ保持が必要なのは加算器における桁上げなので、そこで前述のステートマシンを利用している。図のように C 素子の値を設定することによってトークンを配置したことと同じ意味を持ちハンドシェイクの問題も解決することができる。

なお、出力に関しては LSB から出力されると述べたが、回路に設定したトークンの都合上出力の最初は空のトークンの値すなわち 0 が出力することになった。今回はこの回路を PCA 用の回路図エディタである PCASE を用いて設計した。使用したセルは 4×5 である。

## 6．4　結果と今後の課題

その作成した回路を PCA のシミュレータである PCASIM を用いて検証を行った。出力結果の例としては以下のようになった。

| | |
|---|---|
| 計算例 | 1101×1011=10001111 |
| 出力結果 | 20 21 21 21 23 22 20 22 21 |

結果は 10 進数で表現されているので、これを 2 進数に変換して出力結果である LSB に着目すると 20=0,21=1,22=0,23=1 であるから結果が 10001111 となっており正しく計算がなされている。ここで回路のトークンを動作させるために乗数の後に 0 を 5 つ入力した。前述したように、出力の最初に 0 があるのは回路上の空のトークンが出力されたためである。また、シミュレーション時間は、以下のようになった。

| | |
|---|---|
| 回路構成時間 | 0m,821u,160n,200p SEC |
| 計算時間 | 0m,67u,132n,300p SEC |

## 7.　まとめと今後の課題

本稿で作成したのは 1 ビット×1 ビットの乗算オブジェクトである。これは非同期かつシリアルで演算を行うものであり、可変長乗算が可能である。

今後の課題としては、回路の縮小化、速度向上、また負の数への対応が考えられる。

## 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 PCA①，” bit,vol.32,no.1,pp.27-35,Jan.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 PCA②，” bit,vol.32,no.3,pp.54-62,March.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.51-69,July.2000

[4] 村野忠夫,ペトリネットの解析と応用,近代科学社,1992

[5] Neil H.E. Weste & Kamran Eshraghian : CMOS VLSI 設計の原理,pp299-306,丸善,1988.