関数型言語 Valid における資源管理記述法

小野　諭・高橋　直久・長谷川　隆三・雨宮　真人
（日本電信電話公社　武蔵野電気通信研究所）

"Description of Resource Management
in Functional Programming Language VALID"

Satoshi ONO, Naohisa TAKAHASHI, Ryuzo HASEGAWA and Makoto AMAMIYA
(Musashino Electrical Communication Laboratory, N.T.T.)

Functional programming language VALID is extended to describe abstract data type and resource management. For these applications, simple description of bi-directional communication is essential. The concept of "Color Specification Channel" is proposed for this purpose, and the bi-directional communication protocol is shown which utilizes 4 kinds of uni-directional channels.

Machine-level deterministic programs are presented for each of the channels. These are written in a new data-driven dataflow computation model "Current State Holding Model", and a FIFO (First-In First-Out) channel is implemented without any retrials of operations for mutual exclusion of shared data.

Key word: abstract data type, dataflow model, channel, communication, FIFO queue, history sensitive function, module

1. はじめに

関数型言語は、記述の明瞭さ、高階関数や部分計算による記述力の高さ、並列計算の可能性などの点で、変数への代入に基づいて計算を行う従来型言語より優れているといえる。また、関数型言語に適し、高い並列度を実現できる計算機として、データフローマシン（Data-flow Machine : DFM）が注目されている。

筆者らは、データフローマシン用関数型言語として Valid（Value Identification Language）の開発を進めており[1]、その拡張として、抽象データ型や資源管理の実現に必要な履歴依存型関数（History Sensitive Function）を記述するため、モジュール（Module）およびチャネル（Channel）の概念を導入した[2]。しかし、チャネルのような FIFO（First-In First-Out）の単方向通信路を複数用いて双方向通信を行う場合、並列処理環境のもとでは通信路相互間の同期が必要になる。このため、逐次化を指定する† など記述が複雑になりやすく、また誤記述の検出をコンパイラが行いにくかった。

本稿では、この問題点を解決するため、従来の FIFO チャネルに対し、色指定チャネル（Color Specification Channel : CS Channel）の概念を提案する。そして、Valid のモジュール機構と併用して、抽象データ型オブジェクト、資源管理などを記述した例を示す。次に筆者らが先に提案したデータフロー演算モデルである状態保持モデル（Current State Holding Model : CSH Model）[3]について簡単に説明する。CSH モデルによる待行列は、Catto らの方法[4] のように排他制御のための再試行を必要とせず、顧客の到着順序が正しく保存されるという特徴をもっている。そして、最後に FIFO チャネル、CS チャネルが CSH モデルにより実現できることを示す。

2. モジュールとチャネル

2.1 モジュールの概要

Valid では、履歴依存型関数は、図2－1に示すように、純関数とループとの組合せで実現される。実際には、複数の関数 fi （i ＝1, ・・・, n）が内部状態を共用しうるので、これらの関数群と状態をまとめ、モジュールとして扱う。

Valid のモジュールは、図2－2に示すように、exported function, local function, module body, channel の4要素より構成される[2]。Exported function は、その存在がモジュールの外部にも知られている関数で、モジュールのインスタンスへのアクセスは、すべてこれらの関数を経由して行う。Local function は、exported function により局所的に使用される関数である。Module body は、recurrence 式[1] などにより、過去の状態を保持する。Channel は単方向の通信路で、function と module body 間のデータの授受を行う。

† 文献〔2〕の図4－3のプログラムは、その例である。

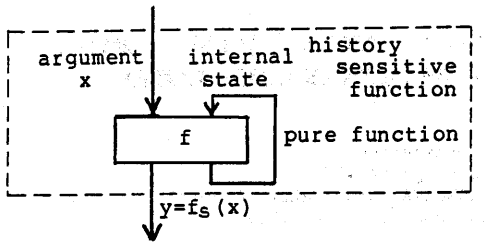


図2－1　履歴依存型関数

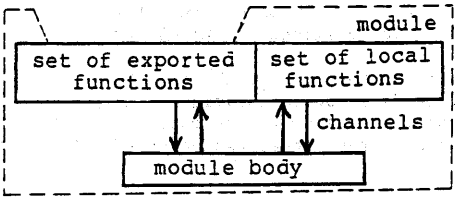


図2－2　モジュールの構成

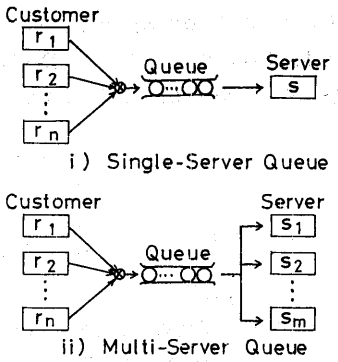


図2－3　単一／複数窓口待行列

2．2　チャネルによる双方向通信

Function と module body 間で、複数のチャネルを用いて双方向通信を行う場合を考える。

Function は並列に呼び出されうるので、複数のプロセスが同時に同一チャネルにデータを送受信することになる。簡単のため、module body 側では同一チャネルへのアクセスは逐次的に行われるとすると、チャネルは図2－3に示す単一窓口待ち行列とみなせる[#]。この場合複数の顧客はチャネルへの並列アクセスに対応し、単一窓口は module body によるチャネルへの逐次アクセスに対応する。キューイングされる対象は、function から module body へのチャネルでは送信データであり、逆方向の場合は function によるチャネルへのデータ受信要求である。

```
(1) Multi-Sender Single-Receiver
    First In First Out Channel
        (MSSR-FIFO Channel)

    From Function:      done=chan!(data)
    To Module Body:     [data,color]=chan?

(2) Multi-Sender Single-Receiver
    Color Specification Channel
        (MSSR-CS Channel)
    From Function:      done=chan!(data)
    To Module Body:     data=chan?(color)

(3) Single-Sender Multi-Receiver
    First In First Out Channel
        (SSMR-FIFO Channel)

    From Module Body:   color=chan!(data)
    To Function:        data=chan?

(4) Single-Sender Multi-Receiver
    Color Specification Channel
        (SSMR-CS Channel)

    From Module Body:   done=chan!(data,color)
    To Function:        data=chan?
```

表2－1　チャネルの種類

双方向通信では、データを定められたプロセスに与えることが必要である。プロセス名を色（Color）と呼び、送信／受信側で相手側の色を指定できるチャネルを色指定チャネル（Color Specification Channel：CS Channel）と呼ぶことにする。すると、チャネルは、通信方向と FIFO／色指定の種別により、表2－1（1）～（4）に示す4種類に分類される。

（1）の MSSR-FIFO チャネルは、function から module body 方向への通信を FIFO で行う。送信側ではデータを引数として与え、相手側がデータを受け取った時点でシグナルを受け取る。また、受信側の module body では、データと送信元の色をリストにしたものを受け取る。（2）の MSSR-CS チャネルも function から module body 方向への通信を行うが、受け取るデータの色は受信側にて引数により指定する。（3）と（4）は Module Body から function 方向への通信を扱うチャネルである。SSMR-FIFO チャネルでは、データを受信したプロセスの色が送信側へ通知される。SSMR-CS チャネルでは、受信側の色は、送信側により指定される。

[#] Module Body 側でのチャネルアクセスが並列の場合、チャネルは図2－3 ii）に示す複数窓口待行列となる。本稿の以下の議論は、文献〔3〕4．3節の手法をもとに、複数窓口待行列の場合に拡張することが可能である。

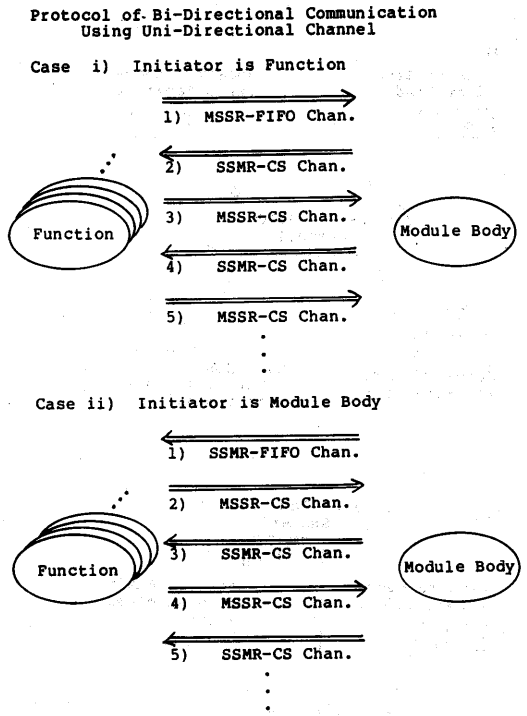


図２－４　双方向通信手順

これらのチャネルを用いて双方向通信を行う場合の手順を図２－４に示す。

i）は function 側から通信を開始する場合であり、FIFO チャネルを通して module body 側にデータと色が送られる。Module body 側ではその色を用いて処理結果を送信し、次のデータを色指定チャネルで受け取る。以下同様にして通信が行われる。ii）は module body 側から通信を開始する場合で、module body は、データが FIFO チャネルを経由して function 側で受け取られると、その受取主の色を結果として受け取る。以下の手順はｉ）の場合と同様である。

２．３　双方向通信の使用例

モジュールと双方向通信を組み合わせることにより抽象データ型オブジェクトや資源管理を実現できる。

図２－５は抽象データ型オブジェクトのプログラム例である。Module arith は、arithobj なる関数を定めている。Fchannel は FIFO チャネルを、また cchannel は色指定チャネルをさす予約語である。通信の方向は！（送信）および ？（受信）で示される。Module body はチャネル command よりコマンドと２オペランドを受取り、演算結果を answer チャネルより送信している。手順は図２－４のｉ）が用いられている。Function 側は、command チャネルへの送信と answer チャネルへの受信を並列に実行し、 arithobj の値として answer チャネルよりの受信結果を返している。

```
arith:module export (arithobj)
={command: fchannel=(list);
  answer : cchannel=(integer);

  arithobj:function (m:list) return (integer)
  ={command!(m); return answer?};

  forever do
   {[[comm,arg1,arg2],color]=command?;
    answer!(
     case
      eq(comm,'add)-> arg1+arg2;
      eq(comm,'sub)-> arg1-arg2;
      eq(comm,'mul)-> arg1*arg2;
      eq(comm,'div)-> arg1/arg2;
     end                          ,color);
   recur}};
```

図２－５　抽象データ型オブジェクト

```
single:module export (get,done)
={request: fchannel=(signal);
  release: cchannel=(signal);

  get:function () return (signal)
  =request?;
  done:function (m:signal) return ()
  =release!(m);

  for (key):('ok) do
   {color=request!(key);
   recur (release?(color))}};
```

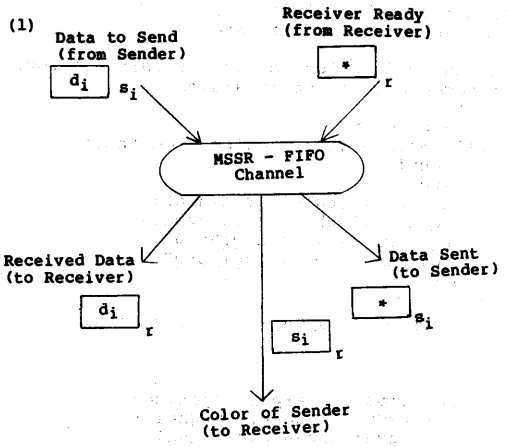
図２－６　単一資源管理

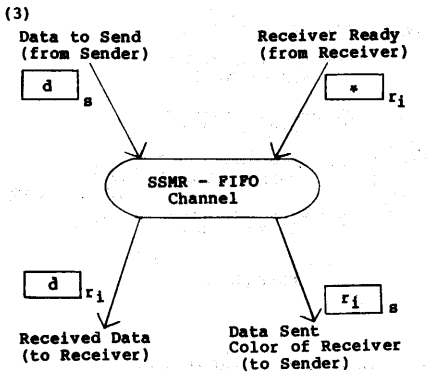
図２－６は、図２－４の手順 ii）を用いた単一資源管理の例である。Module single は資源の要求／開放のため get , done の２関数を定めている。関数 get が呼び出されると、 FIFO チャネル request に受信要求がキューイングされる。資源は module body からの OK シグナルにより与えられる。資源の開放は function done による release チャネルへのシグナル送信により行われ、module body にて再び request チャネルへ出力される。

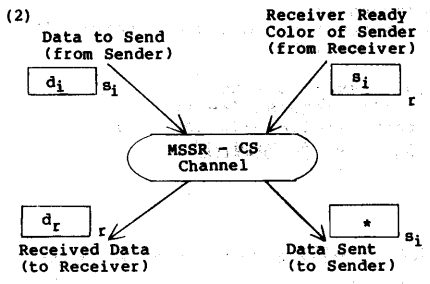
３．　チャネル　の　データフロー　表　現

３．１　チャネルのマクロノード表現

表２－１に示した４種類のチャネルは、データフローモデルのマクロノードとして、図３－１のように表現される。（１）の MSSR-FIFO チャネルでは si （ｉ∈ N）はチャネルの送信側の色を、 di （ｉ∈ N）







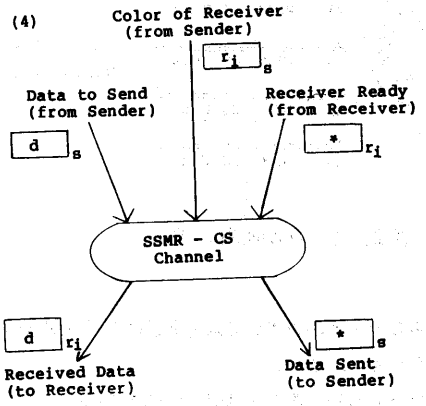


図３－１　チャネルのマクロノード表現

は対応するデータを、また、r は受信側の色である。＊記号は、入力トークン上では don't care を、出力トークン上では unspecified を示している。このマクロノードは、送信データ（ Data to Send ）トークン〔 di 〕si と受信準備完了（ Receiver Ready ）トークン〔 ＊ 〕r が入力アーク上にそろうと発火し、受信側に受信データ（ Received Data ）トークン 〔 di 〕r と送信元色（ Color of Sender ）トークン〔 si 〕r とを、また送信側には送信終了（ Data Sent ）トークン〔 ＊ 〕si を出力する。

（2）の MSSR-CS チャネルでは、送信元色トークンは、受信側によって受信準備完了トークンと兼用して与えられる。（3）の SSMR-FIFO チャネルでは、受信側が複数の色 ri （ i ∈ N ）からなり、色 ri のプロセスがデータを受け取ると、受信先色トークン〔 ri 〕s が送信側に送られる。（4）の SSMR-CS チャネルでは、受信先の色は、送信側のトークンにより指定される。

３．２　ＣＳＨモデル

本節では、図３－１に示したマクロノードをデータフローグラフで実現するための準備として、筆者らが先に提案した状態保持モデル（ CSH モデル ）について概説する。詳細は、文献〔３〕を参照されたい。

本稿で使用する基本演算命令は、図３－２に示すように rcol , excg , wcol , gate 命令の４種類である。rcol 命令は、値として入力トークンの色をとる。Excg 命令は、入力トークンの値と色を交換したものを出力トークンとする。Wcol 命令は、値を第１オペランドの値より、色を第２オペランドの値より定める。Gate 命令は、第２オペランドが到着すると、第１オペランドのトークンを出力する。

色抑止マッチングは、２オペランド命令に付加される属性で、入力アーク間を図３－３に示す矢印で結合する。２本の入力アーク上のトークンは、色にかかわらずマッチングし、ノードが発火する。マッチングは、矢印の先

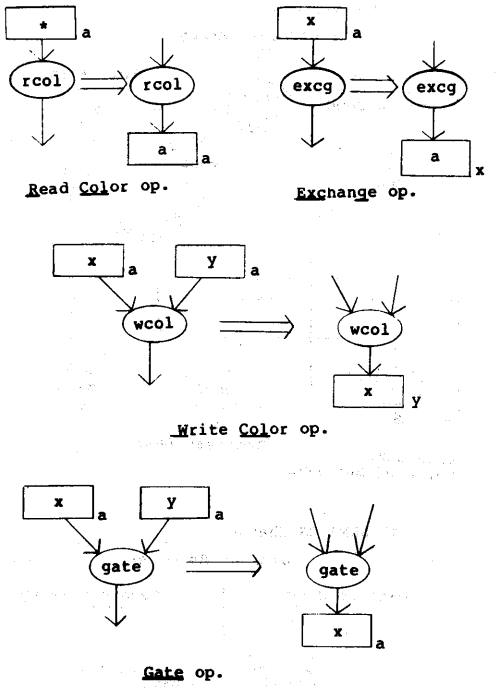


図３－２　基本演算命令

端側の色で起きたとみなす。また、終端側のトークンは、発火しても入力アーク上に残る。

図３－４に示す状態保持アークは、２引数演算ノードの一方向の入力アークに付加可能な属性である。状態保持アークは、初期化後、常にもう一方の入力アークに到着した直前の入力トークンの情報を保持する。

図３－５は、 wcol 演算、色抑止マッチングと状態保持アークを組み合わせた複合ノードの実行例である。このノードは、履歴依存性を持つ１オペランド演算ノードとして機能する。

---

†† Gate 命令など、第一オペランドに重点がある２オペランド命令は、通常、図３－６のように非対称記述する。

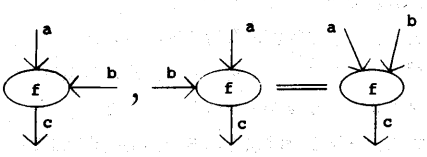


図３－６　非対称記述法

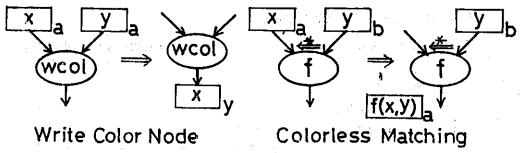


図３－３　色抑止マッチング

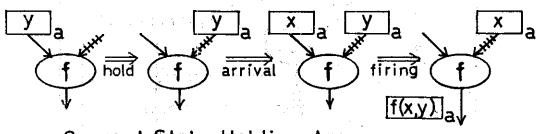


図３－４　状態保持アーク

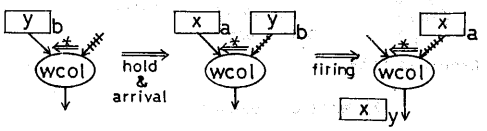


図３－５　複合ノード例

３．３　チャネルのデータフロー表現

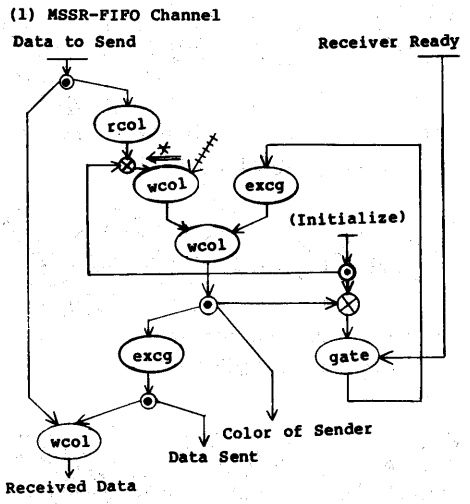
CSH モデルを用いると、図１－１に示したマクロノードを基本演算命令により記述することができる。その結果を図３－７に示す。

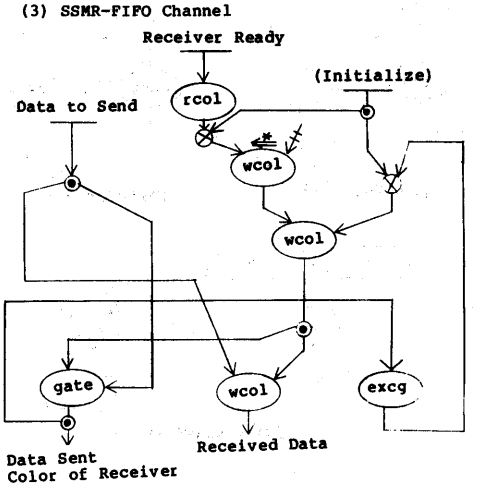
（１）および（３）のFIFOチャネルには、図３－８に示す単一窓口待ち行列が組み込まれている。図中、顧客要求（ Customer Request ）は、待ち行列に入る顧客の入口であり、窓口準備完了（ Server Ready ）は、窓口が処理を終了し、つぎの顧客を受け入れ可能であることを示す。

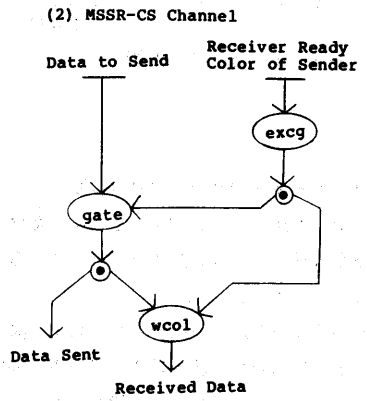
CSH モデルによる待行列は、共有状態の排他制御のための再試行を必要とせず、顧客の到着順序が FIFO に正しく保存されるという特徴を持っている[3]。この性質は FIFO チャネルにおいても、そのまま保存されている。

図３－７の（２）および（４）に示す色指定チャネルでは、指定された色のトークンが、gate 命令により選択される。この選択は、通常ハードウェアにより行われるので、非常に高速である。

図３－７および図３－８の各プログラムの動作の詳細については、それぞれ付録および文献〔３〕４．２節を参照されたい。







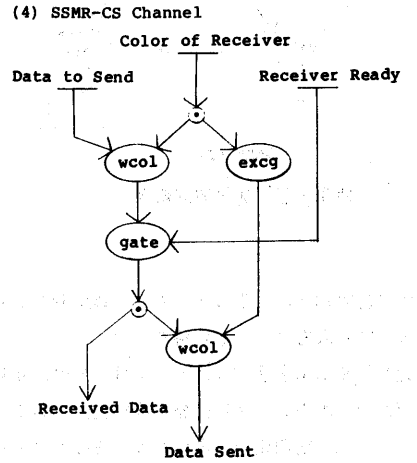


図3－7　チャネルのデータフローグラフ

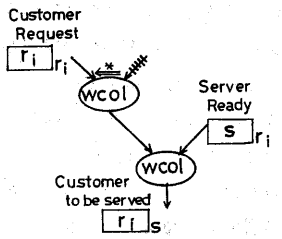


図3－8　単一窓口待行列のデータフローグラフ

4. おわりに

単方向通信路であるチャネルを用いて簡単な記述で双方向通信を行う方法として、色指定チャネルの概念を提案した。そして、Valid のモジュールにおける function と module body 間の双方向通信手順、および、それに必要な4種類のチャネルを示した。さらに、筆者らが先に提案した状態保持モデル（CSH モデル）を用いて、これらのチャネルの実現法をデータフローグラフで示した。

本方式は、共有状態の排他制御のための再試行なしにFIFO チャネルを実現でき、また、色指定チャネルにおけるデータ選択は通常ハードウェアにより行われるので非常に高速である、という特徴をもっている。

今後は、より広範囲の資源管理記述を行うため、Valid の alt 式[1]に対応する非決定性の実現について、データフローの立場から検討していきたい。

最後に，日頃御指導頂く塚本克治基礎研究部第八研究室長、ならびに、御討論頂いたデータフローマシン研究グループの諸氏に感謝します。

〔 参 考 文 献 〕

〔1〕 雨宮, 尾内, "データフローマシン用高級言語 VALID について", 信学技報 電子計算機研究会 EC 82 - 9, pp. 35 - 46 (1982)

〔2〕 小野, 長谷川, 雨宮, "関数型言語 Valid における記号処理機能の拡充", 情処 ソフトウェア基礎論研究会 3 - 9, pp. 57 - 64 (1982)

〔3〕 小野, 高橋, 長谷川, 雨宮, "データフローマシンにおける資源管理方式", 信学技報 電子計算機研究会 EC 82 - 76, PP. 39 - 50 (1983)

〔4〕 A.J.Catto, J.R.gurd, "Nondeterministic Dataflow Graphs", Information Processing 80, S.H.Lavington (ed.), pp. 251 - 256, North-Holland Publishing Company (1980)

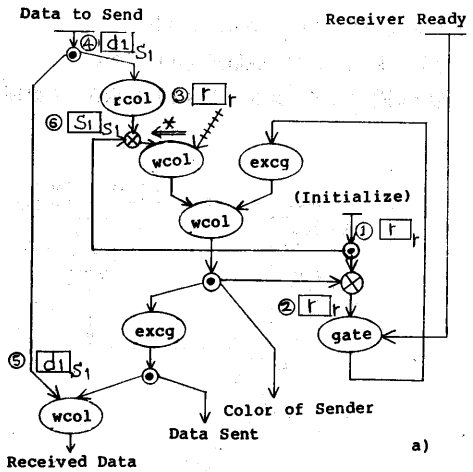
〔 付 録 〕

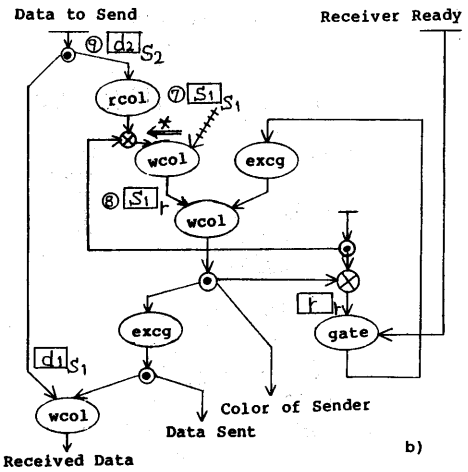
付. 1 MSSR-FIFO チャネル

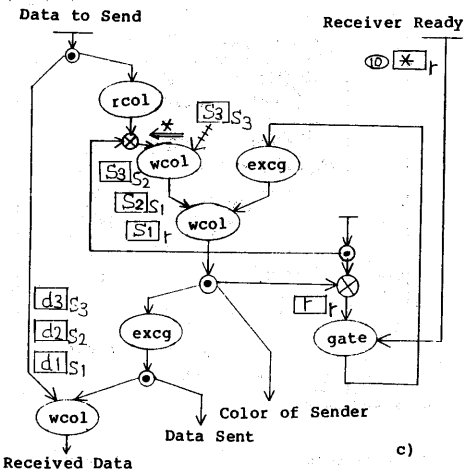
図A－1は、図3－7（1）に示した MSSR-FIFO チャネルの動作説明図である。以下、a）～g）の図に沿って説明する。

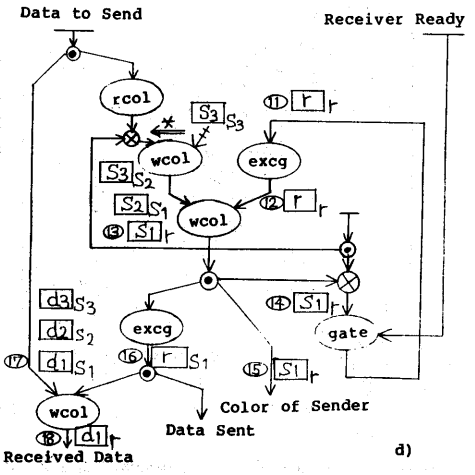
a） 初期トークン〔 r 〕r が Initialize アークに与えられると（1）、 分配ノードにより、一方は gate 命令の第一オペランドになり（2）、他方は状態保持アークの初期値となる（3）（図3－4参照）。ここで、送信データアークにトークン〔 d1 〕s1 が到着すると（4）、 wcol 命令と rcol 命令に送られる（5, 6）。

b） すると（3）と（6）のトークンにより wcol 命令が発火し、状態保持アークにはトークン〔 s1 〕s1 が残り（7）、演算結果トークン〔 s1 〕r が

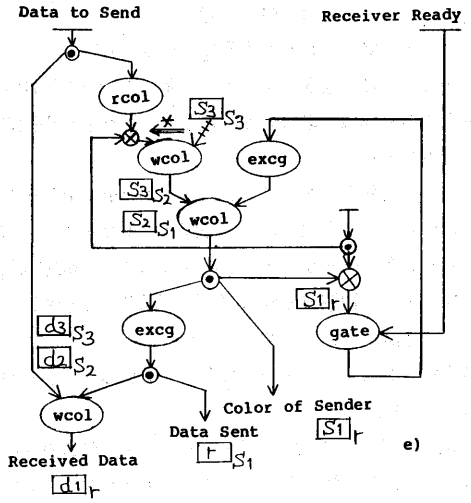


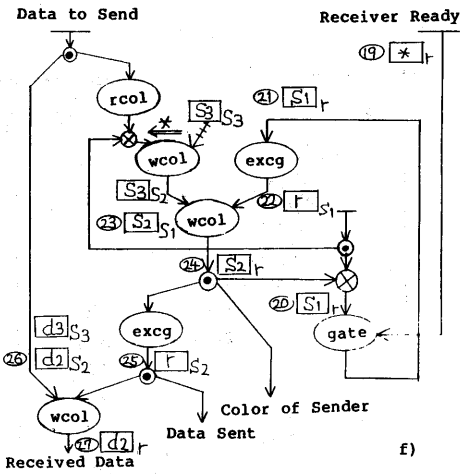


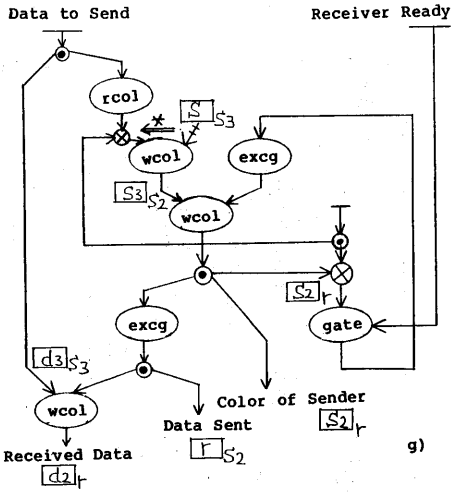




図A－1 MSSR-FIFO チャネルの動作説明図 （次ページにつづく）







図A－1　MSSR-FIFO チャネルの動作説明図 (次ページよりつづく)

出力される（8）。ここで、送信データアークにトークン〔 d2 〕s2 が到着する（9）。

c）　以下同様にして、送信データアークにトークン〔 d3 〕s3 が到着し発火が起こる（説明略）。ここで、受信準備完了アークにトークン〔 ＊ 〕r が到着したとする（10）。

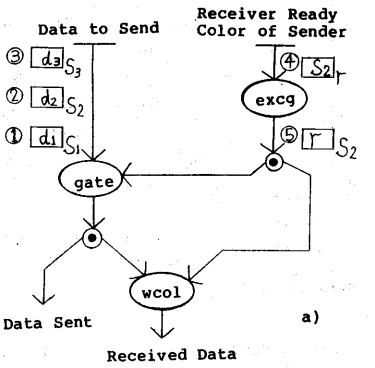
d）　すると、gate 命令が発火し（11）、excg 命令が発火し（12）、（13）のトークンと共に wcol 命令が発火し（14, 15）。続いて excg 命令が発火する（16）。そして、（16）と（17）のトークンにより wcol 命令が発火し、受信データアークにトークン〔 d1 〕r が出力される（18）。

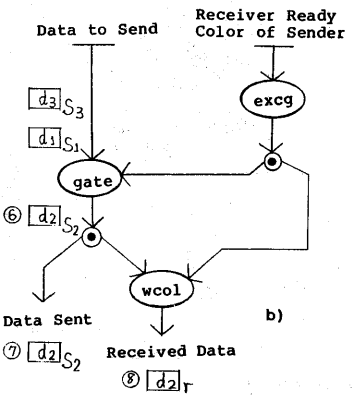
e）　以上で、受信処理が終了する。

f）　更に受信準備完了アークにトークンが到着すると（19）、同様にして（20～26）受信データアークにトークン〔 d2 〕r が出力される。

g）　以上で、次の受信処理が終了する。

以下同様にして、図3－1（1）に示される処理を行う。





図A－2　MSSR-CS チャネルの動作説明図

付. 2　MSSR－CS　チャネル

図A－2 a）～b）に基づいて、図3－7（2）に示した MSSR-CS チャネルの動作を説明する。

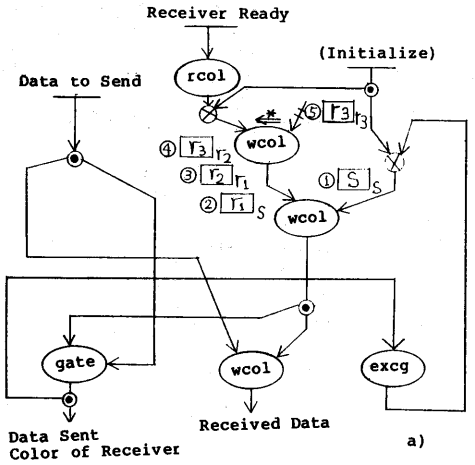
a）　送信データアークにトークン〔 di 〕si （ i = 1, 2, 3 ）が到着している（1～3）。ここで、受信準備完了／送信元色アークにトークン〔 s2 〕r2 が与えられたとする（4）。

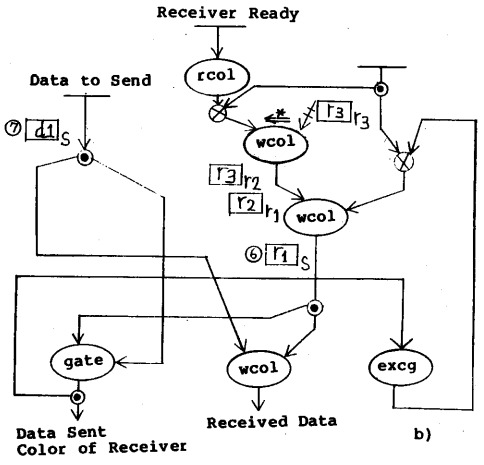
b）　（2）と（5）のトークンにより gate 命令が発火し（6, 7）、続いて（6）と（5）のトークンにより wcol 命令が発火し（8）、受信処理が終了する。

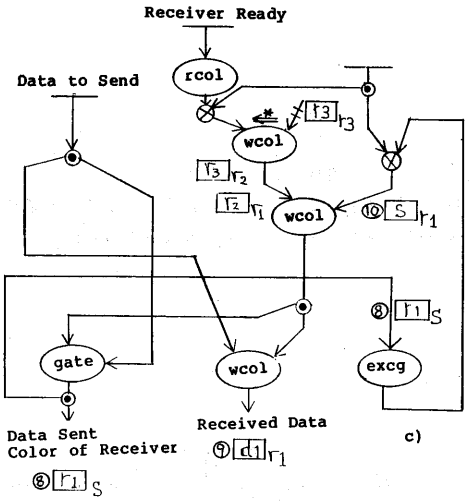
以下同様にして、図3－1（2）の処理を行う。

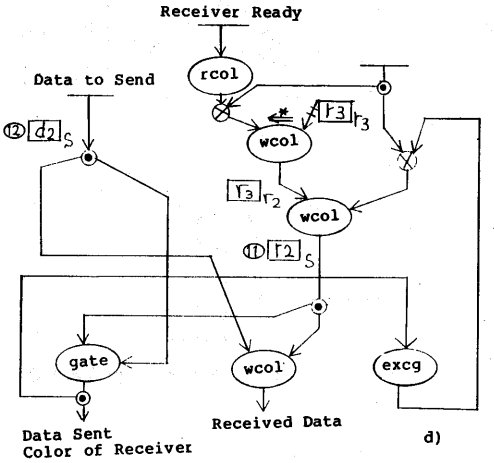
付. 3　SSMR－FIFO　チャネル

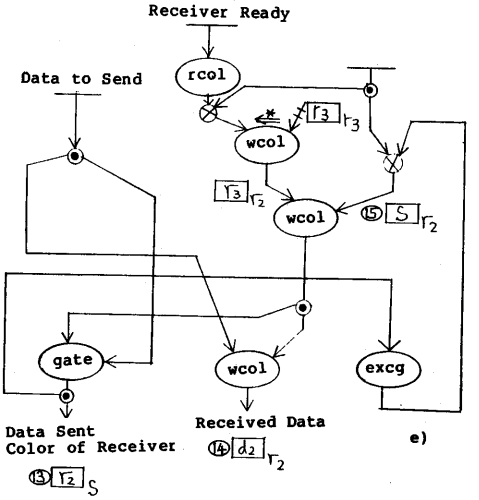
図3－7（3）に示した SSMR-FIFO チャネルの動作については、図A－3 a）～e）を参照されたい。







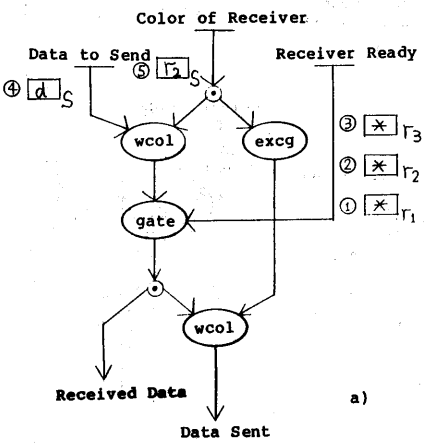


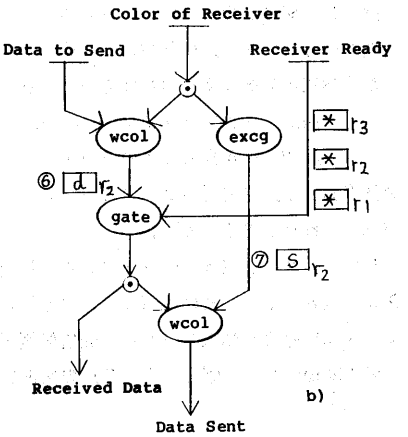


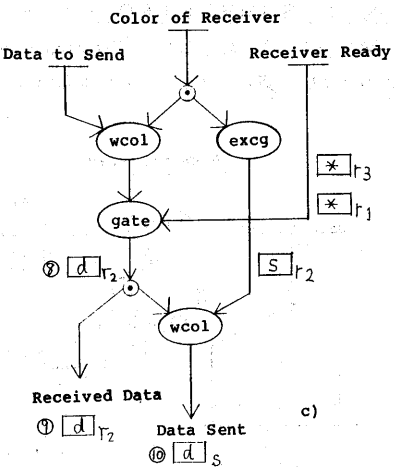
図A－3　SSMR-FIFO チャネルの動作説明図

付. 4　SSMR－CS　チャネル

図3－7（4）に示した　SSMR-CS チャネルの動作については、図A－4 a）～c）を参照されたい。







図A－4　SSMR-CS チャネルの動作説明図