「OR条件」と「外因子」の導入によるビットレベル並行計算モデルAPECの拡張：APEC+

Two Extensions of Bit-Level Concurrent Computation Model APEC by Introducing “OR-Condition” and “External-Factor” : APEC+

網代　孝*
Takashi AJIRO

1．はじめに

CPU のマルチコア化などによってコンピューターの性能が著しく向上した近年では、従来は専用ハードウェアで行っていた高負荷の処理をもソフトウェアで行えるようになりつつあり、特にサーバーの仮想化をはじめとするコンピューター・ハードウェアの挙動をソフトウェア処理で再現するハードウェア・エミュレーションは既に広く普及している。また、GPU などの特定用途向け超並列プロセッサを、マルチメディア処理・ニューラルネットワーク型 AI・仮想通貨マイニングなどの膨大な演算が要求される用途に転用する動きもあり、並行処理性能の向上に依存したコンピューターの性能向上が近年の大きな流れとなっている。しかしながら、それらは一般に高い並行性を内包したプログラム記述が要求されるため、現在主流の逐次処理を基礎とした計算モデル・プログラミング言語上で実装することは決して容易ではなく、プロセッサの並行処理性能が向上し続けたとしても、その潜在性を十分に引き出せるソフトウェアを開発することは次第に困難になっていくことが予想される。

一方で、チップ上に大量に敷き詰められた機能ブロックや論理素子などの基本演算素子を自由に繋ぎかえて超並列処理を実現する再構成可能コンピューティングの研究［1］［2］が以前から行われており、それらは専用ハードウェアの高速性とソフトウェアの柔軟性を兼ね備えているのが特徴である。しかしながら、あくまでも柔軟なハードウェアという位置付けのため、人間が直接的に結線（プログラミング）して目的の処理を実装することは想定されておらず、一般にはハードウェア記述言語（HDL）などを用いたハードウェア設計に近いものとなっている。もし、人間が直接的に基本演算素子を結線して容易に処理を実装できるような計算モデルと言語があれば、高い並行性を内包したプログラムの記述が容易になるだけでなく、その基本演算素子を並列ハードウェア上で効率的にあるいは直接的に実行されることで、並行処理性能を際限なく引き出すことも可能になると考えられる。

そこで筆者はまず、2003 年に、ビットストリームによって結合された多数の演算素子を組み合わせてビットレベルの並行処理を記述できるビジュアルプログラミング言語（VPL）A-BITS を開発した［3］。次に、その言語システムの問題点を洗い出した後、厳密な計算モデルで基本演算素子を定義することの重要性から、「プリミティブ」と呼ばれる演算素子間を「キャリア」と呼ばれる媒体がキャッチボールのように行き来することで「ネットワーク」の計算過程が進行する APC（Ajiro Program Circuit）

＊　埼玉工業大学人間社会学部非常勤講師

モデルを考案した[４]。その後、若干の修正を行って形式的定義を洗練したものを APEC（Asynchronous Program Elements Connection）モデルと命名した。ビットストリームやデータフロー型の計算モデルでは、演算素子間の同期に由来する結線の複雑化が大きな問題の一つであったが、APEC モデルにおけるプリミティブ間を往復するキャリアは同期と通信の両方を担うことから、同期機構による結線の複雑化を抑制できる点で優位性がある。2008 年の論文［５］では、APEC モデルの形式的定義に加えて、２次元セルオートマトン（Fredkin's replicator）や簡単な４ビットコンピュータを APEC ネットワークとして構成した例も示した。その後、APEC モデルを基礎とした VPL APECbits を開発して前述の構成例をビジュアル・プログラムとして記述し、それが実際に正しく動作することを確認した［６］。

本論文では、プリミティブの動作を定義するルール部に「OR 条件」（OR-Condition）、それから外的要因によるキャリア状態の変更を表現する「外因子」（External-Factor）を導入して APEC モデルを拡張した「APEC+ モデル」について説明する。また、本モデルによって、ビットレベル並行計算で多用される同期機構を APEC モデルで構成するよりもさらに単純に実現できることを示す。

２．APEC モデルの概要

APEC モデルの基本演算素子をプリミティブと呼び、３個の「端子」と高々４個の「ルール」を持ち、端子上に静止するキャリアと呼ばれる媒体が保持する値・状態をルールに従って変更する動作を行う。各プリミティブは互いに非同期的に動作し、キャリアが値を変化させながらプリミティブ間を行き来することで演算ネットワーク全体の計算過程が進行する。キャリアは端子間を移動中である「出発」か、端子上に静止する「滞在」かのどちらかの「行動状態」を取る。１つの端子は１つのキャリアによってのみ他の端子と接続でき、ループバック接続も許可されるが、一方で複数のキャリアが１つの端子を共有することはできない。図１は、APEC モデルにおけるキャリアの接続イメージとそれに対応する図記法を示したもので、また、「A.x」はプリミティブ A の端子 x という意味である。例えば、（a）の A.x には１を保持する滞在キャリアがあり、（d）の B.y には０を保持する出発キャリアがある。

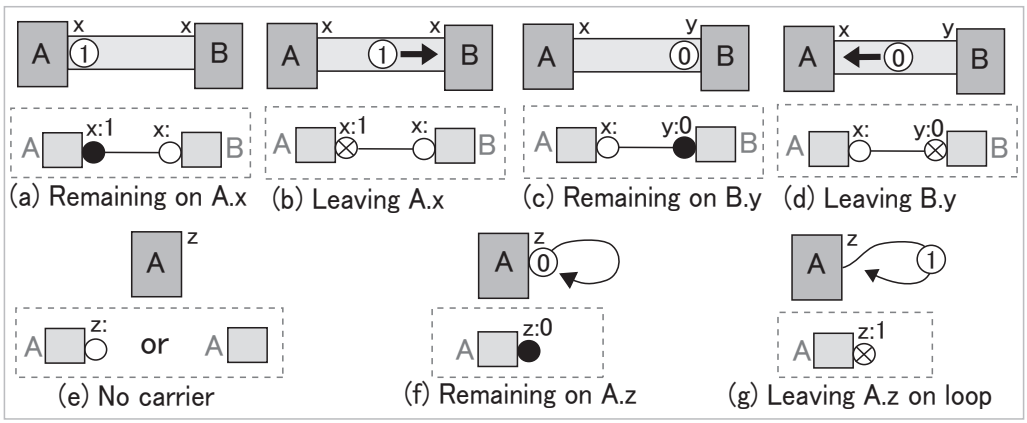

図１．キャリアの接続イメージ（上部）と対応する図記法（下部）

各プリミティブの振る舞いは「ルール表」によって定義され、図２の（a）は経路結合を担うJOINプリミティブの動作を定義するルール表を例示したものである。ルール表の上部には「ルール名」が表記されているが、一般には、そのルールが実現する動作を容易に連想できる短い文字列を充てることから、これを「ニーモニック」とも呼ぶ。１つのルール表が持つルールは最大４個で上位ほど優先順位が高く、各端子上の全ての滞在キャリアが表の左列にある「条件値」を満たしたときに、表の右列にある「行動値」に変更されることを意味している。「＊」はその端子を無視するという意味で、また、「`0」はキャリアの値を０にしてかつ出発状態にするという意味である。キャリアを滞在状態のまま値だけを変更する場合は、「`」を付加せずに単に「0」や「1」と表記する。図２の（b）はJOINプリミティブに上から２番目のルールが適用されたときのキャリア状態の変化を示したものである。

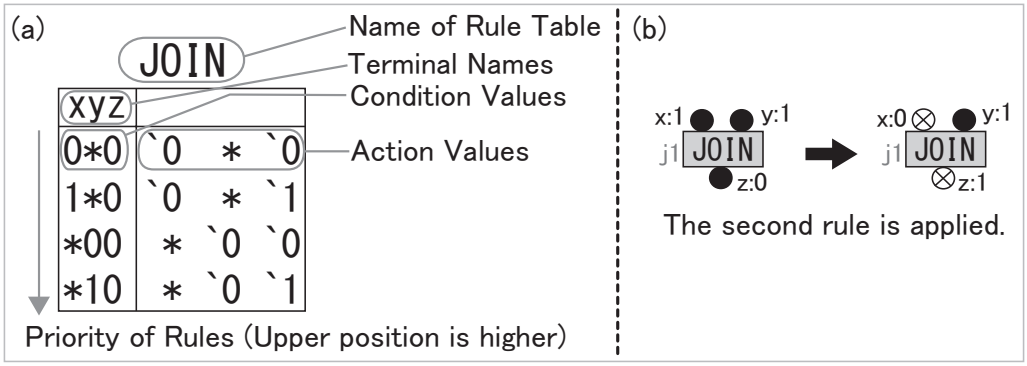

図２．プリミティブの動作を定義するルール表

ネットワークの状態が変更されることを「遷移」と呼び、「プリミティブ遷移」と「キャリア遷移」の２種類の遷移がある。前者は、全ての端子上の滞在キャリアがルール表の少なくとも１つのルールに適合する場合に非同期的に起こり、一番優先順位の高い適合ルールに従ってその端子上のキャリア群が操作される。後者は、あるキャリアが出発状態のときに非同期的に起こり、キャリアが送信先に到着して滞在状態になる。ここでS → pは、あるネットワーク状態Sにおいて、プリミティブpに遷移が起こった後のネットワーク状態を示すものとする。同様にS → p.xは、p.x上にあるキャリアに遷移が起こった後の状態を示すものとする。このとき、S → p → p.xは（S → p）→ p.xと解釈されて、状態（S → p）においてキャリア遷移p.xが起こった後の状態を意味している。このようなネットワーク状態の連続的な変化を記述したものを、「遷移系列」と呼ぶ。この表記法では一度に１つの遷移しか記述できないが、それは単に因果的な順序を意味するだけのため、現実の時間経過とは無関係である。また、どのような遷移も起こることができないネットワークの状態を「完了状態」と呼ぶ。

図３はDUPプリミティブの動作をテストするためのネットワーク例で、（A）はDUPの働きを簡易的に表現したもの、（a1）～（a3）はネットワークの状態変化を示したものである。DUPはxから受け取ったキャリアの値をyとzから出力することで値複製器として働く。DMYプリミティブはyの値をxから出力するか、あるいはxで値を受け取ることでテスト入出力として働く。このように、固定サイズのルール表によって各端子上のキャリアの振る舞いを入力・出力・記憶になるように制御することで、

多種多様な演算素子を定義できるのがAPECモデルの特徴である。このネットワークの完了状態までの遷移系列の１つは（a1）→d1→d1.x =（a2）→u1→u1.x→u1.y→u1.z =（a3）であるが、遷移の生起タイミングは非同期であることから、例えばu1.x、u1.y、u1.zの順序が入れかわった系列も考えられる。各プリミティブはキャリアによって分離され、かつ各キャリアはプリミティブによって分離されているので、同種の連続する遷移の順序を入れ替えても同じ結果状態に収斂する。

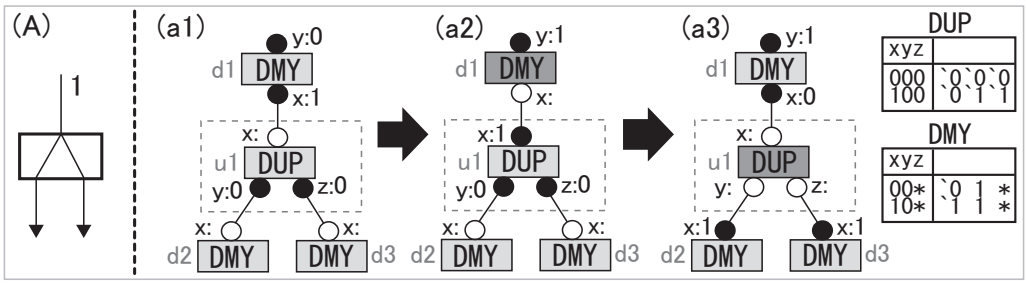

図３．APECネットワークの遷移（DUPプリミティブ）

３．APEC モデルの拡張

多種多様なプリミティブを組み合わせてビットレベル並行計算のための主要な機能部品が構成できることを論文［５］で例示しているように、APECモデルは既に十分な計算記述能力を有しているが、後に若干の拡張によって重要な機能部品を単一のプリミティブで構成できることが明らかとなった。また、本モデルは外界の影響を受けない「閉じた」ネットワークを構成することのみを指向したため、多くのプログラミング言語処理系が備えている外部入出力やAPIアクセスなどの外界との通信を担う特殊機構を直接表現できなかったが、後に若干の拡張によってそれらも表現可能になることが明らかとなった。そこで、これらに関する２つの機能を導入してAPECモデルを拡張したものをAPEC+モデルと命名し、次節より、導入した機能の詳細および本モデルの形式的定義について説明する。

（１）OR 条件の導入

プリミティブの動作を定義する各ルールの条件値には「＊」「0」「1」があるが、まずはこれを「＊⇒00、0⇒01、1⇒ 10」として２ビット値で符号化する。次に、行動値には「0」「1」「`0」「`1」があるので、同様に「0⇒00、1⇒01、`0⇒10、`1⇒11」として符号化する。厳密には「＊」もあるが、これは条件値が「＊」のときには常に行動値も無視されることを示す値なので、符号化の必要はない。このとき、ルール１個の条件値は２ビット×３端子＝６ビット、行動値は２ビット × ３端子＝６ビットの計12ビット必要なので、ルール表は12ビット × ４個 =48ビット（６バイト）で構成できることになる。理論的に扱う限りでは、ルールの構成に必要なデータ量にあまり意味はないが、ソフトウェア処理やハードウェアとして実装する場合には消費メモリ領域や回路規模に影響するので重要である。ところで、条件値の符号「11」だけは未定義のため、これに新たな条件値を割り当てることでデータ量を増やさずにAPECモデルを機能拡張できる。そこで、滞在キャリアがあるときに適合する条件値「c」を「11」に割り当てることで、消費ルール数を減らせる可能性があることから、プリミティブの能力を

若干高められる余地が生まれる。例えば､AND プリミティブは条件値の「x y」部に「0 0」「0 1」「1 0」「1 1」の４ルールが必要だが、条件値「c」を用いると「1 1」（出力が１の場合）と「c c」（出力が０の場合）の２ルールで定義できる。また、EXOR の場合は「0 0」「1 1」（出力が０の場合）「c c」（出力が１の場合）の３ルールで定義できる。そして、余ったルール部は後述する演算プリミティブの「責任化」（Responsibilization）のために利用できる。この新しく導入した条件値「c」は、滞在キャリアの値が「0」または「1」の場合に適合することから、OR 条件と呼ぶことにする。

（2）外因子の導入

プリミティブ端子上の滞在状態のキャリアはプリミティブ遷移によってのみ状態が変更され、また、出発状態のキャリアはキャリア遷移によってのみ状態が変更される。どちらの遷移が起こるにせよ、いかなる外界の影響も受けることがないため、ネットワークは完全に閉じているといえる。閉じたネットワークは並行計算過程を完全に表現することはできるが、外界との通信過程については、DMY プリミティブなどのダミー接続の端子を接点に見立てて擬似的に表現するしかない｡そこで､外界の影響によって滞在キャリアの状態がひとりでに変化するようなプリミティブ端子を導入すると、その端子上ではどちらの遷移でもない第３の遷移が起こることになる。そのような端子を外因子、また、そこで起こる遷移を「外因子遷移」と呼ぶことにする。外因子が存在するネットワークは外界の影響を受けることから「開いて」おり、例えば接続された周辺機器とのデータ通信、別のネットワークとの接続部分、API 呼び出しなどの機構をネットワーク内に組み込んで直接的に表現することができる。

あるネットワークの状態 S において、プリミティブ p_e 上の端子 t_e を $p_e.t_e$ として、かつ $p_e.t_e$ が外因子であるとき、$p_e.t_e$ 上に滞在するキャリアの状態が外因子遷移によって変更された状態を $S \rightarrow p_e.t_e<v>$ および $S \rightarrow p_e.t_e<\grave{}v>$ と表記する。ここで、前者は $p_e.t_e$ 上のキャリアが滞在かつ値 v を保持した状態、後者はキャリアが出発かつ値 v を保持した状態に変更されることを意味している。また、外因子のあるネットワークでは、３種類のどのような遷移も起こることができない完了状態に加えて、外因子遷移のみが起こりうる状態を「準完了状態」と呼ぶことにする。プリミティブ遷移とキャリア遷移では、同種の遷移の順序を入れ替えても結果状態が変わることはないが、外因子遷移については、同一の外因子で連続的に起こる場合にのみそれは成り立たない。例えば、$p_e.t_e<0> \rightarrow p_e.t_e<\grave{}1>$ の順序を入れ替えた $p_e.t_e<\grave{}1> \rightarrow p_e.t_e<0>$ では２番目の遷移が起こることはできず、また、$p_e.t_e<0> \rightarrow p_e.t_e<1>$ とその順序を入れ替えた $p_e.t_e<1> \rightarrow p_e.t_e<0>$ の結果状態は異なってしまう。

図４は外因子を含む APEC+ ネットワークの例で、(a) は外因子 n1.x からの値「1」によって d1.x からのデータ「1」を通過させるように動作し、(b) は d3.x からの値「1」を外因子 n2.x に出力する動作を行う。なお、外因子となるプリミティブ端子は円形ではなく菱形の図記号で表記している。GT プリミティブは y が１のときは x の値（入力値）を z から出力し、y が０のときは入力値を破棄する。NOP プリミティブは全く何もしないが、外因子のための見かけ上のプリミティブとして使用している。(a) と (b) は実際には外因子を介して接続されており、(ab) はそれらと等価な合成ネットワークを示している。つまり、(a) の n1.x の実体は (b) の d3.x であり、(b) の n2.x の実体は (a) の g1.y ということになる。系列的にみると、(b) の d3 がプリミティブ遷移したときに (a) の n1.x が外因子遷移し、その後、(a) の g1 がプリミティブ遷移したときに (b) の n2.x が外因子遷移することになる。

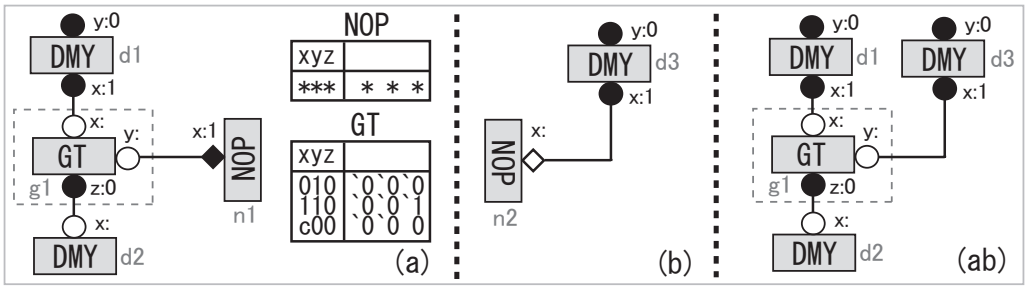

図４．外因子を含むAPEC+ネットワーク(a)(b)とその合成ネットワーク(ab)

（３）gAPEC+ モデルの形式的定義

論文［５］で gAPEC（general APEC）モデルから APEC モデルを定義したのと同様に、まず一般的な計算モデルとして gAPEC+（general APEC+）モデルを形式的に定義し、次にそれを制限した形として APEC+ モデルを定義する。なお、本論文では、集合演算の × 記号は２つの組あるいは要素を結合し、また、入れ子になった組は暗黙的に結合されることを前提とする。例えば、集合 {(p, t)} × {n} の要素は（(p, t) , n）で（p, t, n）となり、また、２つの入れ子組（(p, t) , (q, u)）は（p, t, q, u）となる。gAPEC+ モデルのネットワーク M は、APEC モデルの（P, T, V, R, S_I）に１項追加した形の M=（P, T, V, R, E, S_I）で示す６項組によって定義され、また、各要素の意味は次の通りである。

P はプリミティブの有限集合、T は端子の有限集合、V はキャリアが取る値の有限集合

$R \in R_{all}$、R は各プリミティブが持つルールの有限集合、

$R_{all}=\{ X \subseteq P_T \times \mathbf{N} \times (2^V\text{-}\phi) \times V_A : \forall \alpha\ (\in P_T \times \mathbf{N})\ [|(\{\alpha\} \times (2^V\text{-}\phi) \times V_A) \cap X| \leqq 1] \}$

R の要素は１つのプリミティブルールの一部であり、その構成要素の意味は次の通りである。

P_T は P × T の意味で、プリミティブと端子の組の集合である。**N** は {0, 1, 2, ...} として定義されるルールの順位のための自然数で、小さい数ほど優先順位が高い。$(2^V\text{-}\phi)$ を「条件値集合」と呼び、ある順位のある端子における条件値の全ての組み合わせを意味する。条件値集合の要素が１つの値を持つ場合は、その値がルールの条件値となる。複数の値を持つ場合は、いずれかの値に合致する条件を意味し、これを OR 条件と呼ぶ。例えば OR 条件 {0, 1} ならば、0 と 1 のいずれかに合致すれば条件が満たされる。３章（１）において任意値の意味で使用した「c」は、例えば V={ 0, 1 } とした場合には、条件値に V を指定することで簡潔に記述できる。V_A は条件値が合致したときに変更するキャリアの値と行動状態を指定する組の集合で、この要素を行動値と呼ぶ。ここで $V_A=V \times A$、$A=\{\tau_{rm}, \tau_{lv}\}$ として、τ_{rm} は滞在状態、τ_{lv} は出発状態を意味する行動状態である。例えば $\{(p, x, 0, \{0\}, 1, \tau_{rm}), (p, y, 0, \{0, 1\}, 0, \tau_{lv})\} \subseteq R$ は、「p.x が 0 でかつ p.y が 0 または 1 のとき、p.x の値を 1 で滞在かつ p.y の値を 0 で出発にする、p の順位 0 のルール」という意味である。なお、V={ 0, 1 } とした場合には、２番目の要素を（p, y, 0, V, 0, τ_{lv}）のように記述してもよい。

$E \subseteq P_T$、E は外因子の集合

$S_I \in S_{all}$、S_I はネットワーク M の初期状態、

$S_{all}=\{X\subseteq P_T^{\,2}\times V_A:\ \forall\alpha(\in P_T)\forall\beta(\in P_T)[|(\{(\alpha,\beta),(\beta,\alpha)\}\times V_A)\cap X|\leqq 1\ \wedge$
$|(\{\alpha\}\times P_T\times V_A)\cap X|\leqq 1\ \wedge\ |(P_T\times\{\alpha\}\times V_A)\cap X|\leqq 1]\}$

$S\in S_{all}$ をネットワークの状態と呼び、Sの要素をキャリアと呼ぶ。キャリアを構成する最初の P_T の組は送信先、２番目の P_T の組は送信元（現在の端子）、V_A の組は現在のキャリアが保持する値と行動状態を示している。集合 S_{all} を定義する条件部分は、２つの端子は高々１つのキャリアによって接続され、かつ１つの端子は高々１つの接続しか持たないように制約している。

（４）gAPEC+ モデルにおける遷移の形式的定義

ネットワーク M=（P, T, V, R, E, S_I）の状態 $S\in S_{all}$ を操作して次状態 $S'\in S_{all}$ を得ることを遷移と呼び、プリミティブ遷移、キャリア遷移、外因子遷移の３種類がある。プリミティブ遷移は $f_p: S_{all}\times P\rightarrow S_{all}$ によって写像される。$S\in S_{all}$、$p\in P$ とするとき f_p（S, p）の結果を次のように定義する。

$f_p(S,p)=\{x:(p,t,n,V_c,v_a,a)\in R,(q,u,p,t,v,\tau_{rm})\in S,$
$\alpha=\{y:\forall(p,t',n,V_c',v_a',a')(\in R)[y\in((P_T\times\{(p,t')\}\times V_c'\times\{\tau_{rm}\})\cap S)]\},$
$\alpha\neq\phi,\ \neg\exists n'(\in\mathbf{N})[\forall(p,t',n',V_c',v_a',a')(\in R)[$
$((P_T\times\{(p,t')\}\times V_c'\times\{\tau_{rm}\})\cap S)\neq\phi]\ \wedge\ n'<n],$
$x\in((S-\alpha)\cup\{(q,u,p,t,v_a,a)\})\}$

ここで、プライム記号「 ´ 」を伴う変数は束縛変数であり、例えば$\forall(p,t',n,V_c',v_a',a')$の中で$\forall$によって束縛されている変数は $t'\in T$、$V_c'\in(2^V-\phi)$、$v_a'\in V$、$a'\in A$ である。遷移後の状態は、状態Sのときのプリミティブpにおいて、端子 p.t に関する順位 n のルールが存在し、p.t に滞在状態のキャリアが存在し、p の任意の端子について順位 n のルールの条件を満たすキャリアの集合 α が空ではなく、p の任意の端子について n より高い順位で条件を満たすものがなく、これらを満たした上でSから α を除去して p.t 上の変更後のキャリアを加えた状態の要素の集合である。

キャリア遷移は $f_c: S_{all}\times P_T\rightarrow S_{all}$ によって写像される。ここで、$S\in S_{all}$、$(p,t)\in P_T$ とするとき f_c（S, p, t）の結果を次のように定義する。

$f_c(S,p,t)=\{x:(q,u,p,t,v,\tau_{lv})\in S,x\in((S-\{(q,u,p,t,v,\tau_{lv})\})\cup$
$\{(p,t,q,u,v,\tau_{rm})\})\}$

遷移後の状態は、状態Sのときの端子 p.t において、p.t に出発状態のキャリアが存在し、これを満たした上でSから p.t 上の出発状態のキャリアを除去して、q.u 上の滞在状態のキャリアを加えた状態の要素の集合である。

外因子遷移は $f_e: S_{all}\times E\times V_A\rightarrow S_{all}$ によって写像される。ここで、$S\in S_{all}$、$(p_e,t_e)\in E$、$(v,a)\in V_A$ とするとき f_e（S, p_e, t_e, v, a）の結果を次のように定義する。

$f_e(S,p_e,t_e,v,a)=\{x:(q,u,p_e,t_e,w,\tau_{rm})\in S,x\in((S-\{(q,u,p_e,t_e,w,\tau_{rm})\})\cup$
$\{(q,u,p_e,t_e,v,a)\})\}$

遷移後の状態は、状態Sのときの外因子 $p_e.t_e$ において、$p_e.t_e$ に滞在状態のキャリアが存在し、これを満たした上でSから $p_e.t_e$ 上の滞在状態のキャリアを除去して、$p_e.t_e$ 上の値vを保持する行動状態aのキャリアを加えた状態の要素の集合である。

もし３種類の遷移のいずれかによって写像される状態が ϕ でなければ、その前状態を「遷移可能」

と呼び、一方、ϕに写像するような遷移を「無効遷移」と呼ぶ。ここで、上述の写像形式による遷移を簡潔に記述できるようにするために、等価な他の表記法を導入する。まず、$S \in S_{all}$、$(p, t) \in P_T$として、$f_p(S, p) \equiv S \rightarrow p$および$f_c(S, p, t) \equiv S \rightarrow p.t$と定義する。次に、$S \in S_{all}$、$(p_e, t_e) \in E$、$v \in V$として、$f_e(S, p_e, t_e, v, \tau_{rm}) \equiv S \rightarrow p_e.t_e<v>$および$f_e(S, p_e, t_e, v, \tau_{lv}) \equiv p_e.t_e<\grave{}v>$と定義する。これらの表記法は前述の写像形式によって写像された状態を意味し、例えば$S \rightarrow p \rightarrow p.t \rightarrow p_e.t_e<\grave{}v>$は$((S \rightarrow p) \rightarrow p.t) \rightarrow p_e.t_e<\grave{}v>$と解釈されて、その遷移後の状態は$f_e(f_c(f_p(S, p), p, t), p_e, t_e, v, \tau_{lv})$によって写像された状態と等しくなる。このような連続的に記述した遷移のことを、遷移系列と呼ぶ。もし遷移系列内に1つでも無効遷移が含まれていれば、前述の定義により、その系列の遷移後の結果状態はϕになる。もし遷移系列の結果状態がϕでなく、かついかなる遷移も可能ではない場合、これを完了状態と呼ぶ。また、もし遷移系列の結果状態がϕでなく、かつ外因子遷移だけが可能である場合、これを準完了状態と呼ぶ。

（5）APEC+ モデルの形式的定義

gAPEC モデルと APEC モデルの関係と同様に、gAPEC+ モデルを3端子4ルールのプリミティブによるビットレベル並行計算に特化させるために制限することで、APEC+ モデルを容易に定義できる。具体的には、gAPEC+ モデルを「$|T|=3, |V|=2, \forall(p', t', n', v_c', v_a', a') (\in R)[n' < 4]$」で制限し、これを APEC+ モデルの形式的定義とする。また、特別な事情が無い限り、APEC+ ネットワークの端子名および状態値はそれぞれ$T=\{x, y, z\}$、$V=\{0, 1\}$として固定的に定義して用いることとする。

ところで、各プリミティブ個別にルール部を定義するのではなく、ルール部だけを抽出したものを各プリミティブに組み込んだほうが、同じ機能を持つ複数のプリミティブを使用する際に便利である。実際、APEC+ モデルの図式表現では、JOIN や DUP といったニーモニックをプリミティブの枠内に表記することを前提としている。形式的定義上でニーモニックに相当する機構を実現するには、ルール表に相当する$R \in R_{all}$の端子・順位・条件値・行動値の部分を抽出した$G_{MNI} \subset T \times \mathbf{N} \times (2^V - \phi) \times V_A$を定義してから、$(\{p_1, p_2, p_3\} \times G_{MNI}) \subseteq P \times G_{MNI}$のようにして合成すればよい。

4．APEC+ モデルによる機能部品の簡単化

（1）論理演算プリミティブの責任化

ある演算が終わるまで次の演算を開始しないような機構を実装する場合には、演算が完了するまでキャリアを入力側に返却しないように経路をロックする必要があるが、これは同期プリミティブを入力の前および出力の後に付加して演算部を構成することで実現できる。このように、機能部品などに同期機構を付加することを責任化と呼び、責任化されたものには「責任ある」（Responsible）を冠して呼ぶことがある。演算部の責任化は構成を煩雑化させる要因の一つだが、APEC+ モデルでは論理演算プリミティブの実現に必要なルール数は最大3個で済むため、ルール部を1個使用して同期機構を埋め込むことで、煩雑さを軽減させることができる。図5の（a）（b）は EXOR 演算部を責任化した APEC+ ネットワークの例であり、（a）は RET プリミティブ、（b）は RESP プリミティブを同期機構として使用している。（a）の RET は双方向通信 x を送出 y と返却 z に分離するので、e1 への入力 r1.x と r2.x は

r1.z と r2.z にキャリアが到着するまでロックされる。RDUP は DUP を責任化したもので、r1.z と r2.z のキャリアが返却されるまで d1.x をロックする。r3.y は out.x からの返却キャリアが到着するまでロックされるので、破線で囲まれた部分は責任ある EXOR として働くことがわかる。（b）の RESP は x の値を z から送出し、y に完了通知が届くまで x をロックする。y と z にキャリアが到着すると、x と y から０を返却してロック解除になる。RET との違いは、返却値が０であることを前提とした単方向通信を担う点である。（c）は EXOR を責任化した REXOR プリミティブを用いて構成した（a）（b）と等価なネットワークで、（a）（b）では５個のプリミティブで構成していた部分を（c）では１個で実現していることがわかる。REXOR の x と y にキャリアが到着すると、それらをロックして z から計算結果を出力し、z に０が返却されると x と y から０を返却してロック解除になる。上位３番目までのルールで計算処理を定義し、最後のルールで x と y のロックを解除して初期状態に戻す処理を定義している。

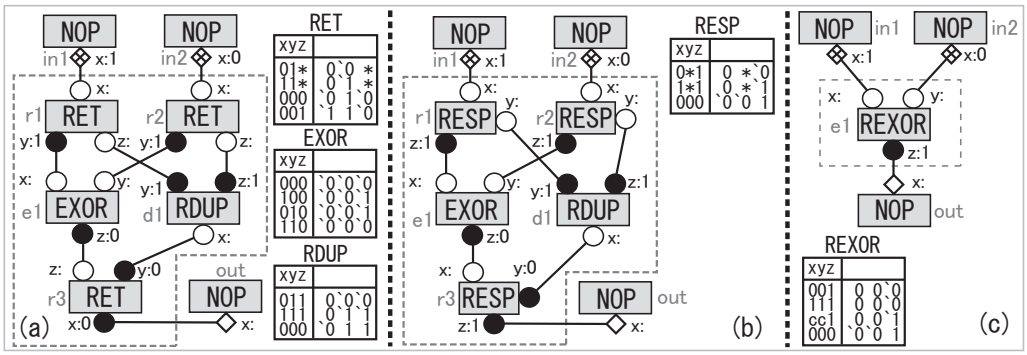

図５．EXOR演算を責任化したAPEC+ネットワーク（１と０を入力して１を算出）

（2）MEM プリミティブ

図６の（a）は OR 条件を使わずにメモリ機能を実現する APEC+ ネットワークの例であり、RGT と RNT は GT と NT（GT の逆で、y の値が０のときに x の値を通過させる）をそれぞれ責任化したバージョンである。APEC モデルの論文［５］では GT と NT でメモリ部品を構成していたが、本論文のメモリ部品（a）は JOIN がキャリアを返却するまで全ての入力経路がロックされることから、より安全な設計となっている。cmd からコマンド（０は読込、１は書込）を、data から書込む値を入力すると、コマンドに従って内部状態を更新した後、保持する値を data に返却する動作を行う。（b）は MEM プリミティブを用いて（a）と同等の機能を実装したネットワークであり、（a）では６個のプリミティブを用いて構成していた部分を（b）では１個で実現していることがわかる。（b）の m1.y にはメモリ値を保持するキャリアが滞在しており、これは（a）の g1.x に滞在するキャリアの値に対応している。（c）はコマンドによるモード選択を行わず、読込・書込を別々の端子で行う DMEM プリミティブを用いたネットワークの例である。x 端子から値の読込、z 端子から値の書込を行うことができるが、両方の端子にキャリアが到着した状態でプリミティブ遷移が起こった場合は、ルール順位により読込が優先されることになる。

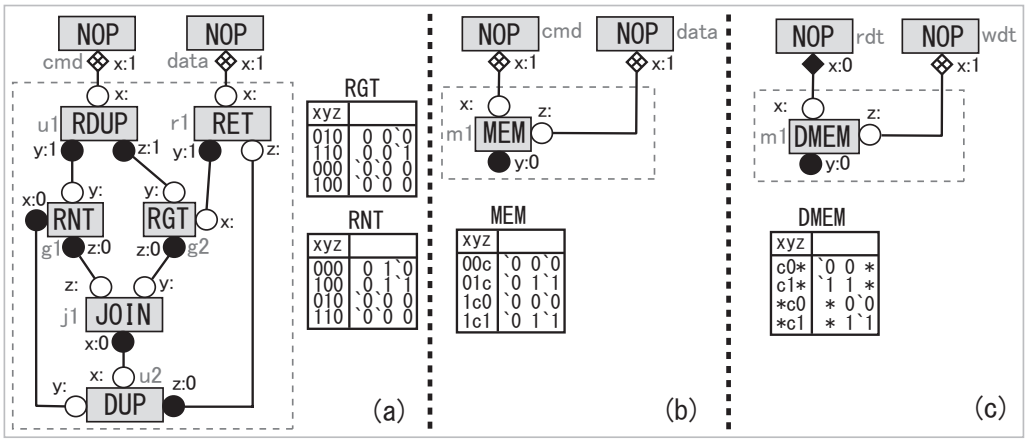

図６．メモリ機能を実現するAPEC+ネットワーク（メモリに１を書き込む）

５．まとめ

本論文では、まず２章で APEC モデルの基本概念とネットワークの図記法について説明し、またネットワーク例を用いてビットレベル並行計算の進行過程を遷移系列で示した。次に３章では、APEC モデルに OR 条件と外因子を導入したものを APEC+ モデルと命名し、その２つの機能の説明および gAPEC+ モデル～ APEC+ モデルの形式的定義を記述した。そして４章では、APEC+ モデルの OR 条件を用いることで、同期機構を埋め込んだ演算部品およびメモリ部品を１個のプリミティブで構成できることを示した。今後の展望としては、VPL APECbits を改良して APEC+ モデルに対応させることなどを考えている。

【参考文献】

［１］ K. Compton, S. Hauck: “Reconfigurable Computing: A Survey of Systems and Software”, ACM Computing Surveys, vol.34, no.2, pp.171-210, June 2002.

［２］ H. Ito, R. Konishi, H. Nakada, H. Tsuboi, Y. Okuyama, A. Nagoya: “Dynamically Reconfigurable Logic LSI: PCA-2”, IEICE Trans. Inf. & Syst., vol.E87–D, no.8, pp.2011-2020, August 2004.

［３］ 網代孝, 土田賢省 : “ビジュアルプログラミング言語 A-BITS”, 信学技報, vol.103, no.318, pp.15-20, September 2003.

［４］ T. Ajiro, K. Tsuchida: “A Bit-Level Concurrent Visual Programming Language (A-BITS) and a Base Computation Model (APC) for its Development”, IEEE Symposium on Visual Languages & Human-Centric Computing, pp.269-271, September 2005.

［５］ T. Ajiro, K. Tsuchida: “A Model of Computation for Bit-level Concurrent Computing and Programming: APEC”, IEICE Trans, vol.E91-D, no.1, pp.1-14, January 2008.

［６］ T. Ajiro, K. Tsuchida: “Visual Programming Language for Bit-level Concurrent Programming: APECbits”, IEEE Symposium on Visual Languages & Human-Centric Computing (VL/HCC'08), pp.113-116, September 2008.