マルチメディア通信と分散処理　82－21
（１９９７．　４．　２５）

仕様記述言語LOTOSからハードウェア記述言語VHDLへの変換

花屋 雅貴、稲富 猛、西垣 正勝、佐藤 文明、水野 忠則
静岡大学情報学部

プロトコルの仕様記述言語LOTOS[1]を用いた応用としてテスト系列の生成、ソフトウェアへの応用など様々な研究がなされている。近年高速なネットワークの研究、開発によりプロトコルはより高速な動作を求められている。そのためプロトコルをハードウェアで実装することがしばしば行われている。そこで本稿ではLOTOSからハードウェア記述言語VHDL[2]への変換方針を提案する。これによりプロトコルのハードウェアによる実装の際、より早いプロトタイプの作成及び計算機上での動作確認が可能になると思われる。また、LOTOSからVHDLへの変換のために「同期ゲート」、「セレクタ」ハードウェアモジュールを導入する。
キーワード：LOTOS、VHDL、同期、ハードウェア構成

A Translation Method of FDT LOTOS into HDL VHDL

Masataka Hanaya, Takeshi Inatomi, Masakatsu Nishigaki, Fumiaki Sato, Tadanori Mizuno
Faculty of Information, Shizuoka University

FDT(Formal Description Techniques)LOTOS is applied to generating test sequences , implementing software and so on . Nowadays high speed networks are researched and developed , so protocols are asked for higher speed behavior . Therefore protocols are implemented hardware in order to satisfy this demand . Now this paper proposes translating FDT LOTOS into HDL(Hardware Description Language) VHDL . We think that it is possible for this proposal to make a prototype quickly and to confirm the behavior on computers . And we introduce「synchronism gate」and「selector」which are made as hardware module in order to translate LOTOS into VHDL.
key word : LOTOS , VHDL , synchronism , hardware composition

1　はじめに

近年、プロトコルの開発において仕様記述言語が用いられるようになってきている。しかしながら、その仕様をテストしたりする環境は完全に整っているとは言い難い。

また、ハードウェア開発環境においては、設計及び検証の能率化、様々なユーザレベルでハードウェアが理解できるように、ハードウェア記述言語(Hardware Description Language：HDL)が用いられるようになってきた。

そして高速なネットワークが開発され、発展するにともないプロトコルをハードウェアで実装することで高速な動作をさせるような研究が注目されている。このときプロトコル仕様の記述から、早急なプロトタイプの試作およびその動作確認が求められている。

そこで我々は仕様記述言語としてISO標準のLOTOSを、HDLとしてIEEE標準のVHDLを選びLOTOSからVHDLへの自動変換手法の提案、実装を目的とする。これによりプロトコルのハードウェアによる実装の際、システムの動作確認及び開発コストの削減などにおいて有益であると考える。

本稿では、LOTOSからVHDLへの変換のための問題点とアルゴリズムについて提案する。ただし今回対象とするLOTOS記述は(1)データを扱わない基本LOTOSで(2)再帰呼び出しはそのプロセス自身を呼び出すもののみ(3)多重同期は扱わないの制限を加えている。

2 LOTOSとVHDL

2.1 LOTOSの概念

LOTOSは基本的に次の考え方に基づいて言語設定が行なわれている。

「システムは、そのシステムのやりとりする外部から観測可能なイベント間の時間順序を規定することで仕様化可能である。」

ここでイベントとは

1. 瞬時発生である
2. それ以上細分化できない
3. 他のイベントとの時間的重なりがない、つまり、ある時間に起こるイベントは唯一つに限定される。

ものであり、これをイベントのアトミック性という。この性質によりLOTOS記述は各プロセスのイベント木(イベント列を木で表したもの)の関係により表せる。

またLOTOSはシステムをプロセスという実行単位で階層化して表現できる。

2.2 LOTOSの同期

LOTOSにおける重要な意味を持つ概念として、イベントの同期というものがある。LOTOSの同期とは、通信する二つ以上のプロセスが、同時に、あるイベントの発生に関与するということである。つまり二つ以上の全てのプロセスにおいて同期すべきイベントが生起可能な場合において、はじめてそのイベントが起こりうる。

2.3 LOTOSオペレータ

LOTOSではイベント列によってシステムの動作を記述する。このイベント間の関係をLOTOSオペレータが表す。以下に主要なオペレータについて説明する。以下A、Bはプロセスもしくは動作式(イベント列)を表しa、bはイベントを表す。

・プレフィクス

a ; b と記述しイベント間の順次性を表す。この場合イベントaが起きたのちに、イベントbが起こる。

・選択

a [] b と記述しイベント間の選択を表す。この場合イベントaもしくはbが起こる。どちらが起こるかは、その他のプロセスと同期の取れたものが選択される。

・並列オペレータ

非同期(独立)並列は A ||| B と記述し、このオペレータの両側のプロセスはお互いに関与せず非同期に独立に動作することを表す。

完全同期並列は A || B と記述し、両側のプロセスは全てのイベントに関して完全に同期することを表す。

一般同期並列は A |[イベント名]| B と記述し両側のプロセスは指定されたイベントに関しては同期をすることを表す。

・割り込み

A [> B と記述しプロセスの割り込みを表す。これはプロセスAの振舞いの任意の時点で、プロセスBの初期イベントが起こり得ることを意味している。プロセスBの初期イベントが起こったならば制御はAを離れBに移る。

・隠蔽

hide a , b in A と記述しプロセスAの外へイベントa、bを見えないようにする。このときa、b以外のイベントに関してはその他のプロセスから見ることができる。

2.4 VHDLの特徴

VHDL はハードウェア記述言語の一つである。VHDL の特徴は、

- 階層的にハードウェアを記述できる
- 様々な抽象レベルで記述できる
- シミュレーションによるテストが可能である

という点が挙げられる。具体的な記述としては、各ハードウェアコンポーネントの入出力ポートの指定を行うエンティティ宣言と、各コンポーネントの動作を規定するアーキテクチャ宣言からなる。階層化されたハードウェアを記述するには、アーキテクチャ宣言中で使用するコンポーネントを指定し、ポートマップ宣言でそれぞれの接続情報を記述する。

3 変換における問題点

LOTOS から VHDL に変換するにおいていくつかの問題点がある。並列オペレータによるプロセスの並列性の解釈。選択オペレータが階層にまたがった場合の解釈。これらは VHDL でいかに記述するかと言う問題ではなく、一般に LOTOS を解釈する場合においてもしばしば問題になる点である。

3.1 並列性の解釈

並列オペレータには先に述べたように非同期のものと同期を取るものとがある。一般に非同期並列ではオペレータの両側のプロセスの全てのイベントの組合せを考えるため、合成されたプロセスのイベントの数が増えてしまう。従ってそのままハードウェアにしたのではハードウェアの規模も大きくなってしまう。またイベントの全ての組合せを解析することは両側のイベントが増えれば増えるほど困難になる (解釈 1)。そこで非同期並列の場合はオペレータの両側で別々のプロセスとして各々をハードウェアとして構成する (解釈 2)。図 1参照。ただし、これ

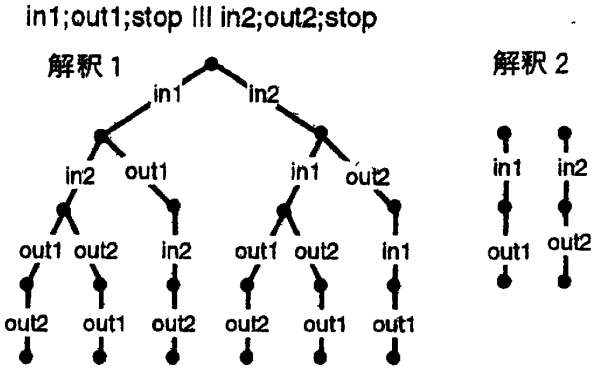

図 1: 非同期並列の問題

により LOTOS の一つの特徴である「イベントはある時刻にはただ一つしか起こらない」というイベントのアトミック性が崩れてしまう。同期並列の場合オペレータの両側のイベントで同期が取れるかどうかの解析をしなくてはならない。これはイベントの枝刈りと言われるがオペレータの両側のイベントが多かったり、階層化されて記述されている場合にはその解析は困難である [3][4]。

これら二つの問題は LOTOS を解釈するうえで大変な負担となるため、オペレータの両側のプロセスのイベントや階層に制限を与えることがある。

3.2 選択の解釈

選択オペレータによって生起するイベントの選択を行なえる。ここで選択オペレータが 2 段以上になった場合には問題が生じる。図 2のように選択オペレータを状態遷移の分岐として表現することでプロセス B、C はハードウェアとして構成できる。ついで B、C を選択するプロセス A を同様に構成しようとすると B、C を合成した状態遷移 A をつくらなければならない。これでは階層が深くなった場合にその解析 (合成) は困難である。このように選択を状態遷移の分岐として解釈すると、下位のモジュールを合成し一つのモジュールとしてハードウェア化しなくてはならないために、ハードウェアとし

てはLOTOSの階層を保存できない。

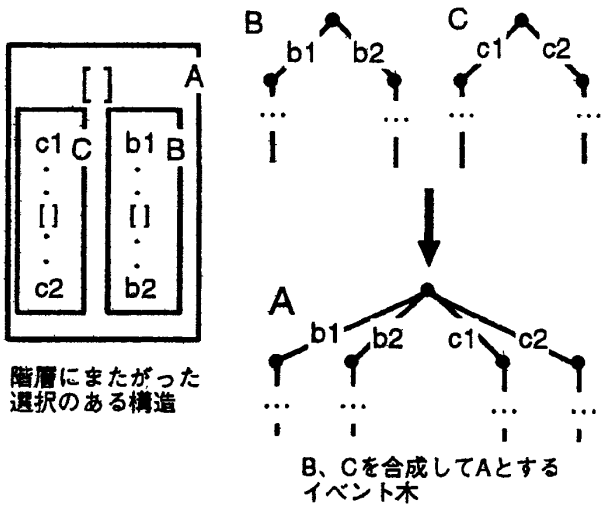

図 2: 階層化された選択の問題

この問題に対しても選択における制限を与えることで回避している [5]。

4 変換手法

本稿では前節で述べた問題点を解決するために、同期を判定するハードウェア (同期ゲート) と選択を制御するハードウェア (セレクタ) を提案した。これからこれらのハードウェアコンポーネントの仕様とそれらを用いた変換アルゴリズムについて述べる。

4.1 同期ゲート

次に挙げるのが同期ゲートに要求される機能である。

- 複数の同期要求を同時に受け付ける。
- 同期がとれるかを判定する。
- 判定結果をプロセスに返す。
- 複数の要求が同期可能な場合はランダムに一つを選ぶ。

最初の項目は並列プロセスはイベントのアトミック性が保証されていれば、ある時刻には一つの動作 (同期要求) しかないため必要ない。最後の項目を満たすことでLOTOS解釈の曖昧な部分といえる非決定性の問題も解決できる。全体の動作の流れは、一方のプロセスからのいくつかの同期要求の中から優先度 (クロック毎に変化する) の高いもの選ぶ。その要求ともう一方プロセスの要求全てとのマッチング (同期) をとる。複数の同期要求と同期が取れた場合には優先度の高いものを一つ選ぶ。同期の取れた要求に対してその要求を出したプロセスに同期判定信号を出力する。二つ以上同時に同期をとる多重同期の場合はマッチングを繰り返すことで対応できる。

4.2 セレクタ

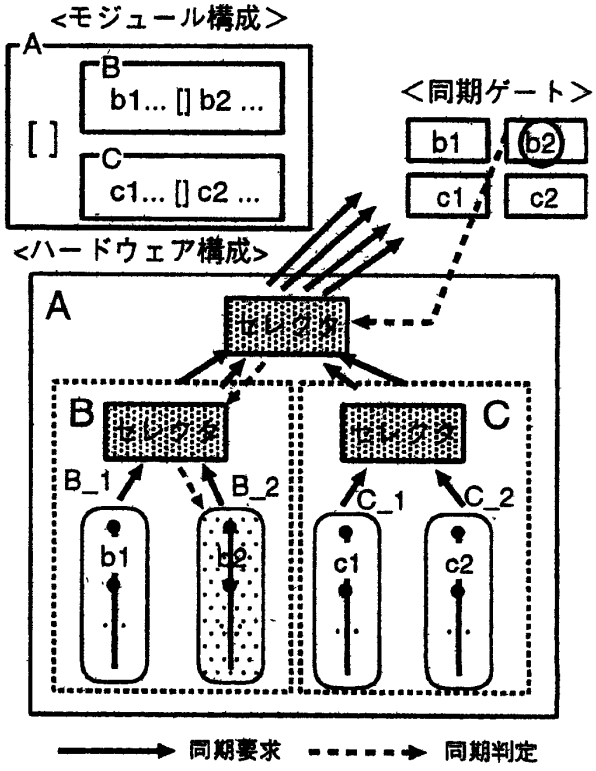

図 3: セレクタを用いたハードウェア構成

セレクタを用いたときのハードウェア構成は図3のようになる。このようにセレクタを用いることでLOTOSの階層をハードウェアに保存できる。

以下がセレクタの動作である。

- 複数の同期要求をゲートに出力する。
- 一番最初に戻ってきた同期判定のみプロセスに出力する。
- 同期判定が一つ戻ってきた後は判定結果を全て受け付けない。
- 同期判定が一つ戻ってきた後それ以外の要求をキャンセルさせるように同期ゲートに信号を送る。
- 複数の要求が同時に戻ってきた場合には優先度の高いものを選択する。

選択されるプロセスの内の最初のイベントだけがセレクタを通して同期をとる。

4.3 変換アルゴリズム

4.3.1 基本方針

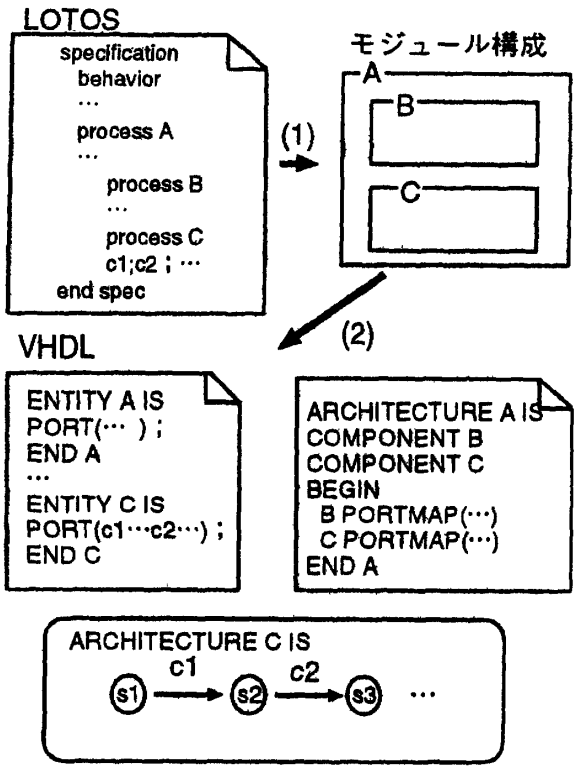

図 4: 基本方針

変換手順としては、(1)LOTOS 記述された仕様の分割、(2)VHDL コードの生成、の二つに分かれる。この流れを図 4に示した。

(1)では、プロセス呼び出しによるモジュール分割と LOTOS オペレータによるモジュール分割を行う。そして、分割された各モジュール及びモジュール間の関係の情報を取り出す。(2)では(1)で取り出した情報に従い VHDL コードを生成する。まず、分割された各モジュールに対応するコンポーネントのエンティティ(インタフェイス)宣言を行う。次に、各コンポーネントの動作をアーキテクチャ宣言で記述する。このとき上位階層では使用するコンポーネントを指定し(同期ゲート、セレクタも指定される)、それらの接続関係を明示する。一方、最下層ではイベント列を考慮した状態遷移記述を行う。

4.3.2 オペレータとハードウェア構成

各オペレータをどのようにハードウェアとして構成するかについてこの節で述べる。

まずイベントの種類の数だけ同期ゲートを用意する。

a;b のようなプレフィクスは同期ゲート a に同期要求信号を出力し、同期判定が戻ってきた後に同期ゲート b に同期要求信号を出力する状態遷移記述を行う。また記述の中に再帰呼び出しがある場合には最後の状態から最初の状態に状態を遷移させる。ただし自分自身を再び呼び出す再帰のみを対象とする。

選択と同期並列に関しては前章の図 3のように適当な場所に同期ゲートとセレクタを配置する。

順次合成は最初のプロセスのハードウェアが正常終了したならば、制御を続くプロセスのハードウェアに移すべく制御信号を前のプロセスから後のプロセスに出力する。

割り込みは割り込み部分と割り込みされる部分のハードウェアを別々に構成する。そして各々が独立して同期要求を同期ゲートに出す。割り込みをする方の同期が取れた時点で、割り込みされる方のハードウェアへ動作を終了するように制御信号を送る。

イベントの隠蔽は同期ゲートの配置を図5のようにすることで、他のプロセスから隠蔽されるべき同期ゲート(イベント)を見えなくする。(プロセスAとBがゲートg1で同期をとり、プロセスAの中でA1とA2がゲートg2で同期をとる。このときhide g2 in Aによりゲートg2が隠蔽される例)

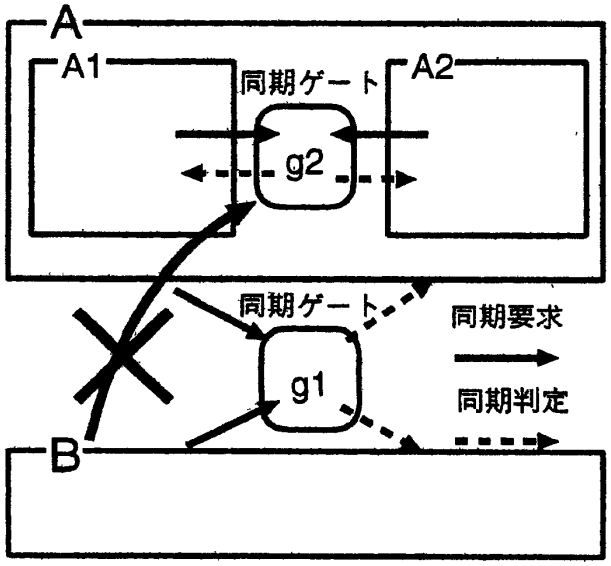

図 5: 隠蔽の実現

4.3.3 VHDL記述

VHDLではまず、ENTITY宣言で各ハードウェアコンポーネントの入出力ポートを定義する。プロセスのなかでイベントaとbが起こる場合、ポートとしてa_in(入力)、a_out(出力)、b_in、b_outを宣言する。inには同期ゲートからの判定が入力され、outは同期ゲートへの同期要求が出力される。この他にクロック入力ポート、順次合成、割り込み、再帰を実現するために制御用入出力ポートが必要な場合がある。

具体的な動作はARCHITECTURE宣言で行なう。上位階層では使用するコンポーネントの指定(COMPONENT宣言)と、それらの入出力ポートがどのように接続されるか(PORT MAP記述)を記述する。

最下層ではイベント列に従った状態遷移を記述する。

5 まとめ

LOTOS記述の同期表現、選択表現をハードウェアで実現可能にするために「同期ゲート」及び「セレクタ」ハードウェアを導入した。これらの提案によりLOTOSの解析の負担を軽減し、ハードウェア化するとき元の仕様の階層を保存することができる。現在、LOTOSの解析の部分のプログラムが終了し、「同期ゲート」と「セレクタ」はVHDLコードで記述中である。今回はデータを扱わない基本LOTOSで変換手法を提案したが今後データを扱えるフルLOTOSへの拡張を行なう。

参考文献

[1] 高橋、神長著:仕様記述言語LOTOS、カットシステム

[2] Z・ナバビ著:VHDLの基礎、日経BP出版センター

[3] 黄、安本、北道、東野、谷口:LOTOS風言語で書かれた同期式順序回路の要求仕様記述と回路自動合成、マルチメディア通信と分散処理, 66-9, (1994.7)

[4] 水野、安本、森岡、東野、谷口:通信プロトコルのLOTOS仕様から並行EFSM群への変換の一手法、マルチメディア通信と分散処理, 80-14, (1997.1.30)

[5] A.M.Lopes
: HARPO, gopher://sanson.dit.upm.es:70/00/pub/lotos/tools/HARPO