プロセス記述による非同期式制御回路合成システム

4E−6

籠谷裕人　Hong Minh Nhut　南谷 崇
東京工業大学 工学部

1 はじめに

近年の素子技術はスイッチング遅延が1ピコ秒にせまる高速なデバイスを実現しつつある。しかし従来の同期式プロセッサ回路はチップ全体へのクロック分配が必要であり、配線遅延の相対的な増大によるクロックスキューのため、こうした素子を活用できるような高速のクロックを用いることができない[5]。

素子の高速性を有効に活用する一つの方法は、プロセッサを非同期式に構成することである。非同期式回路は、同期式回路の設計にあるような論理設計とチップ設計の相互依存性を排除でき、また、回路を拡張する場合のタイミング設計のやり直しも不要となり拡張性に富むといった利点を持つ。

我々は文献[6]で、非同期式制御回路の自動合成手法を提案した。本稿では、この手法を用いた自動合成システムのプロトタイプの作成に関して述べる。まず設計の対象とする非同期回路のモデルについて述べ、次に仕様の記述と合成方式の概要、さらにシステムの構成について述べる。

2 非同期回路モデル

本稿で設計対象とする制御回路を構成する各機能モジュールは、他の機能モジュールや制御対象とのやりとりを、**要求信号**と**応答信号**を用いたハンドシェークによって実現する(図1)。こうした個々の機能モジュールは、他と通信する独立した**プロセス**とみなすことができるので、その機能モジュールの動作仕様を、対応するプロセスの通信動作としてプロセス記述言語で記述する。

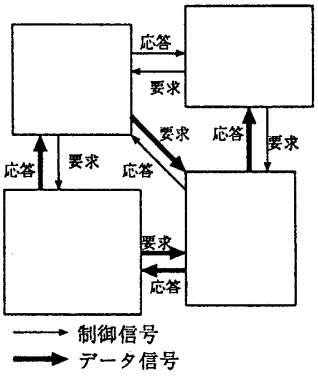


図1: 機能モジュール間の通信

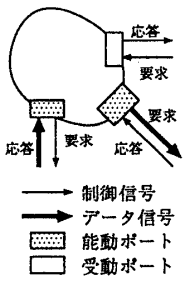


図2: プロセスの外観

プロセスは通信路に対して、ポートを介して信号を送り、また信号を受けとる。各ポートは名前を持ち、同名のポートは同一の通信路に接続される。ポートには**能動ポート**と**受動ポート**があり、能動ポートは要求信号を出して応答信号を受け、受動ポートは要求信号を受けて応答信号を出す。それぞれの通信を**能動通信**、**受動通信**と呼ぶ(図2)。またデータ通信はデータに何らかの符号化を行い、要求信号や応答信号として用いることによって、制御信号と同等に扱う。符号化はタイミング情報を含めることができ、回路構成が容易な**二線二相符号**を用いる。信号の種類をまとめると以下のように分けられる。

制御信号 単線。値1は要求または応答があることを示す。

データ信号 二線。二線符号語(01または10)の時は0または1の値を持つことを示し、スペーサ(00)はデータがないことを示す。

3 記述と合成の概要

3.1 記述方式

Martinらは HoareのCSP[1]を元にしたプロセス記述言語を提案し[2]、さらにBerkelらは能動通信と受動通信を記述の上で明示的に区別するように改良した[4]。本方式では基本的に同様な記法を用いる。

表1はこの記述言語の各記述要素とその意味を表す。演算子の優先順位は次のようになる(左側ほど強い)。

$$!,?,:= \;>\; * \;>\; \bullet \;>\; ; \;>\; \rightarrow,: \;>\; |,\|$$

表1: プロセス記述言語の概要

表記	意味
A, A'(ポート名)	能動制御通信、受動制御通信
$A!expr$, $A'!expr$	(能動、受動) データ送信
$A?v$, $A'?v$	(能動、受動) データ受信
$A!expr?v$, $A'?v!expr$	(能動、受動) データ送受信
:=	代入
$\langle v_1, v_2 \rangle$	データの連結
;	直列実行
•	並列実行
$[expr \rightarrow \cdots \| \cdots \| expr \rightarrow \cdots]$	条件選択
$\{expr \rightarrow \cdots\}$	条件反復
:	要求受付
|	調停
*	反復

3.2 合成手順

合成は図3に示す流れに従って行う。

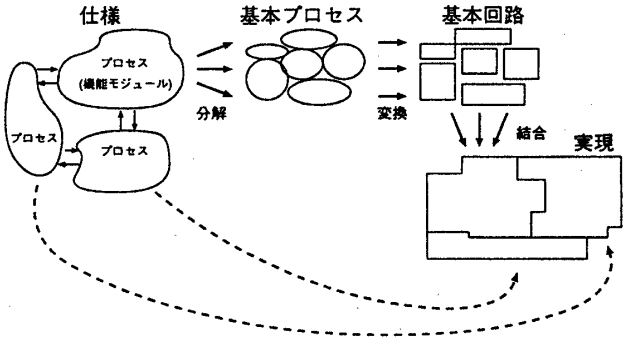


図3: 合成の流れ

先の記述言語で表したプロセスの仕様は、一旦十分に小さな基本プロセスに分解する。基本プロセスは単一の機能のみを持つプロセスで、表2の種類を用意する。分解の手続きは、記述の構文解析木を作り、その各ノードを基本プロセスに対応させることによって行

Synthesis System of Asynchronous Control Circuits based on Process Description

Hiroto Kagotani, Hong Minh Nhut, Takashi Nanya
Faculty of Engineering, Tokyo Institute of Technology

われる [6]。これらの分解された基本プロセスのうち、同じ関数や変数から構成され同じ動作を行うものは一つにまとめ、上位のプロセスで共有するように修正する。

表 2: 基本プロセス

$*A$	能動プロセス
$*A'$	空プロセス
$*(A' : B; C)$	直列実行プロセス
$*(A' : B \bullet C)$	並列実行プロセス
$*((A' \bullet B') : C)$	受動同期プロセス
$*(A' :[expr_1 \rightarrow B_1 \parallel \cdots \parallel expr_n \rightarrow B_n])$	条件選択プロセス
$*(A' : \{expr \rightarrow B\})$	条件反復プロセス
$*(A'_1 : B_1 \mid \cdots \mid A'_n : B_n)$	調停プロセス
$*((A'_1 : B_1); C_1 \parallel \cdots \parallel (A'_n : B_n); C_n)$	ノンブロッキングプロセス
$*(A' : B!expr)$	能動送信プロセス
$*(A' : B?v)$	能動受信プロセス
$*(A' : B!expr?v)$	能動送受信プロセス
$*(A'!expr : B)$	受動送信プロセス
$*(A'?v : B)$	受動受信プロセス
$*(A'?v!expr : B)$	受動受送信プロセス

分解された基本プロセスは種類が限られており、それに対応する回路はすべてあらかじめ用意することができる。これらを基本回路と呼ぶ。関数を含む基本プロセスに対応する基本回路は、その関数を実現する組合せ回路を含むが、その組合せ回路の実現方法には触れず、本システムでは関数名によって表現する。

各基本回路の接続は、一般には同名ポートの要求信号と応答信号を接続することによって行う。プロセスの共有によって複数の能動ポートと一つの受動ポートを接続しなければならない場合は、そのための回路を介して接続する。また受動側プロセスに関数処理や変数へのアクセスが含まれ、動作に時間がかかる場合、自掃モジュール [6] を挿入して効率化する。

4 システム構成

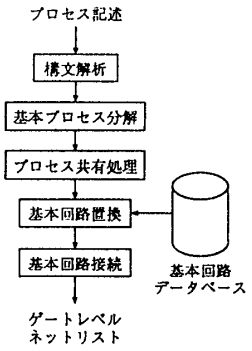
本合成システムは、現時点では図 3 のうち単一の機能モジュールのみを扱い、この流れに沿って図 4 の手順で処理を行う。

将来、非同期式制御回路の設計全体を支援するシステムを作成するには、フローチャートなどの視覚的な仕様記述からプロセス記述を生成する処理や、与えられた関数を実現する組合せ回路の合成システム、さらに全機能モジュールのネットリストを合成してチップレイアウトを生成する処理を付加することによって行うことができる。本システムはそうした設計支援システムの核となる部分である。

図の処理手順のうち構文解析部は、記述言語の文法から yacc によって生成する。それ以外の部分は C 言語によって記述し、約 2000 行である。

一方、ネットリストによる表現では、合成された回路を目で見て確認するには不適当であり、こうした目的には見易くレイアウトされた回路図を出力することが望ましいが、一般に回路図を見易くレイアウトするという問題は困難である。本システムによって合成される単一の機能モジュールは、基本回路の上位と下位の関係が半順序となる。しかし半順序関係を見易くレイアウトするアルゴリズムにも適当なものがない。

そこで回路図を出力する目的には、プロセスの共有処理を行わないことにより、木構造の基本回路接続図を生成することにした。共有処理を行わない回路が木構造となることは、構文解析の結果が



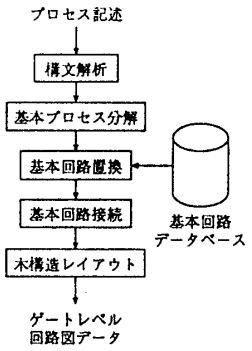


図 4: 合成システムの処理手順　図 5: 回路視覚化用の処理手順

木構造であることから明らかである。木構造の描画アルゴリズムは [3] を用いる。このアルゴリズムは、あるノードの全 sub tree の外形を多角形として生成し、これを重ならないように隣接させて、そのノードの外形 (長方形) を含む全体の外形を生成するという手順を再帰的に繰り返す。この処理を加えたシステム全体の流れは図 5 となる。生成される回路図データは、現在は X11 ウィンドウシステム上の描画プログラム tgif のオブジェクトファイルである。こうした理由は、描画プログラムによる手動での修正が容易であり、またこれを PostScript に変換するプログラムによって容易にプリントアウトできるためである。

5 まとめ

プロセス記述の分解によって非同期式制御回路を自動合成するシステムの作成について報告した。本システムは 20 程度の基本プロセスからなる回路も、Sun3/60 において 1 秒以下で合成することができる。より精密な性能評価は今後の課題であるが、大規模な回路においても $O(n)$ の時間で合成できると予想される。またプロセス共有を行った後の回路を見やすく回路図として描画する方法は、今後の検討課題である。

本研究の一部は (財) 大川情報通信基金 1991 年度研究助成によるものである。

参考文献

[1] C.A.R. Hoare. Communicating Sequential Processes. *Communications of the ACM*, Vol. 21, No. 8, pp. 666–677, August 1978.

[2] Alain J. Martin. Synthesis of asynchronous VLSI circuits. In J. Staunstrup, editor, *FORMAL METHODS FOR VLSI DESIGN*, chapter 6, pp. 237–283. Elsevier Science Publishers B. V., 1990.

[3] Sven Moen. Drawing dynamic trees. *IEEE Software*, Vol. 7, No. 4, pp. 21–28, July 1990.

[4] C.H. van Berkel and Ronald W.J.J. Saeijs. Compilation of communicating processes into delay-insensitive circuits. In *IEEE International Conference on Computer Design*, pp. 157–162. IEEE Computer Society Press, 1988.

[5] 南谷崇. 同期式プロセッサの限界と非同期式プロセッサの課題. 信学技報, December 1990. FTS90–45.

[6] 籠谷裕人, 南谷崇. プロセス記述による非同期式制御回路合成の一手法. 情処研報, December 1991. DA60–10 (信学技報 VLD91–98 に同じ).