時相論理型言語Tokio によるハードウェア記述

— 時間に依存するfactによる同期記述 —

6Q-9

河野真治　中村宏　藤田昌宏＊　田中英彦
東大　工学部
＊富士通研

1.時相論理型言語Tokio

論理回路設計を支援する場合に，ハードウェア記述を様々なレベルから，formalにおこなうことが重要である。Tokio は，時相論理（LITL）〔1〕にもとづき，高度に抽象的なレベルからハードウェア記述をおこなうことができる。我々は，これまで各種のハードウェアをTokioにより実際に記述し，いくつかの記述に対して検証をおこなってきた。

しかし，論理式による記述は，形式的に完全であるが，記述の階層化に問題がある。論理式自体が平担な為に全体の記述が平担になってしまう。この問題は，Cや，Prolog，Lispなどのプログラミング言語にも共通した問題である。

ここでは，Tokio のプログラムの基本形であるClausal Formに，命題論理式を付け加えて，ハードウェアの同期部と機能部の記述を分離することを考察する。これにあわせて，Tokio の記述をモジュール化するために，オブジェクト指向の記述を導入する。

これにより，平担な記述を避けることができ，より実用的な記述が可能となる。

2.Clausal Form of Tokio

Tokio のプログラムは，LITLの論理式となっている。LITLの真偽値は，時間の区間（時区間）に対して定まる。LITLは，基本的な時相演算子として，Next（@），Chop（＆＆）の二つを用いる。@Pは，次の時刻でPが真であることを表す。P＆＆Qは，ある時区間の前半でPが成立し，後半でQが成立することを表す。

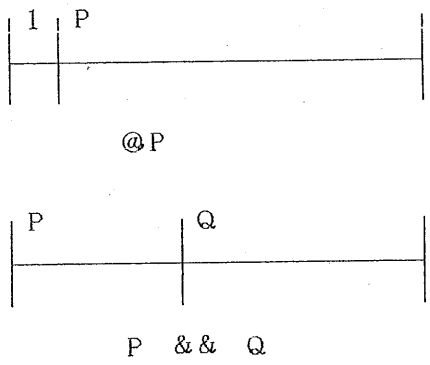

Fig.1 LITL時相論理演算子

Tokio は，一階述語時相論理に基づいている。LITL上のあるresolutionがTokio の実行に相当する。Tokio のプログラムは，任意の時相論理式ではなく，通常のPrologのProgram clauseの拡張になっている。Clausal Formは，HeadとBodyからなり，Headには，引数をもった原子述語がくる。Bodyには，任意の時相演算子と論理AND による原子述語の組合せが来る。論理変数は，Headに現われる変数は，Universal Quntifier がつき，Bodyにのみ現われる変数は，Existential Quntifier がつく。

```
p (X)  :-  q (X, Y), r.
```

この式は，次の論理式に対応する。

$$\#\,(\forall X \exists Y.\ p\,(X) \vee \sim q\,(X,\ Y) \vee \sim r)$$

#は，任意の時区間でこの式が成立することを表す。Tokio のプログラムは，この型の論理式の論理積である。任意の論理式がTokio のプログラムに変換できるのではないが，ハードウェアの機能部の記述には，これで充分である。これは，機能部の記述に以下の性質があるからである。

① 否定は，時相演算子には，かからない。
② 記述は，決定的である。

この性質は，実際のハードウェアの性質に対応していて，未来のデータは知りえないこととハードウェア記述が決定的であることに対応している。

しかし，同期部の記述は，直接のハードウェア記述でなく，仕様記述であることが望ましい。同期部では，任意の論理式が記述できることが望まれる。

さらに，機能部の記述は，機能単位に分割することが容易であるが，同期部の記述は，本質的にシステム全体に依存する。一般に同期部の記述には，次のような性質がある。

① 同期に関係する変数は，大域的である。

Hardware Description in Tokio - Synchronization in time-dependent Fact
Shinji KONO, Hiroshi NAKAMURA Masahiro FUJITA＊, Hidehiko TANAKA
The Tokyo University,　＊Fujituu Laboratory

② 同期に関係する変数は，2値である。

③ 同期の記述は，宣言的であることもある。

このような記述には，命題論理を用いるのがよい。次の章では，命題論理について考察する。

3.命題Tokio と状態遷移

LITLでは，任意の論理式を状態遷移図に変換することができる〔2〕。この状態遷移図は，LITLの論理式を展開規則によって時間順に展開していくことによって得られる。fig.2 は，展開の例である。

得られた状態遷移図は，非決定的な部分を含んでいる。

4.時間に依存するfactによる同期記述

一旦同期部の記述を状態遷移図に変換してしまえば，それをもとに同期部を実行することは容易である。同期部は，命題論理で記述されているので，Tokio の変数をもたないfactに対応させることができる。Tokio のfactの記述は，これまでは，時間的に値が変化することはないので，これはTokio の拡張になっている。

変数をもたないfactに命題変数を対応させることにより，この命題変数はTokio のなかで大域的になる。この方法により，同期部の記述と機能部の記述を適切に分離することができる。

5.モジュール単位の記述

さらに記述のモジュール化のためには，機能部を実際の要素単位に分割することが望ましい。Tokio のモジュールは，ハードウェアに対応するので，生成は，実行の最初におこなわれ消滅することはない。モジュール間の通信は，同期信号に相当する時間に依存するfactと，バスまたは，データパスに相当する共有された時相論理変数による。通常のメッセージパッシングによって通信をおこなうオブジェクトとは異なる。同期部の記述は，モジュールの記述とは別に記述する。

```
module ALU
synch   rcv,ack ;

alu  (Data,Reply) :-
      #rcv, add (Reply,Data) && ack;
alu  (Data,Reply) :-
      #~rcv;
```

Fig.3 Factによる記述例

6.Temporal Prolog との比較

Temporal Prolog 〔3〕では，論理変数は状態をもたない。したがって，状態は原子述語の真偽値のみよって構成される。もしTemporal Prolog の論理変数のとる値が固定されている場合は，命題論理で問題を記述することと同じである。ハードウェアの記述の場合は，同期信号の種類は，有限であるので，Tokio の方法で充分である。特にTokio では，データ転送の記述に状態をもつ変数を用いるので，よりハードウェアに近い記述が可能である。

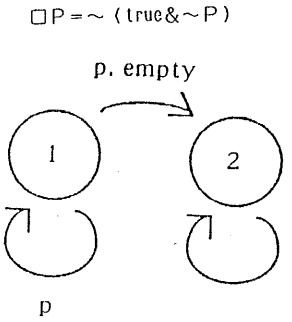

Fig.2 命題Tokio と状態遷移

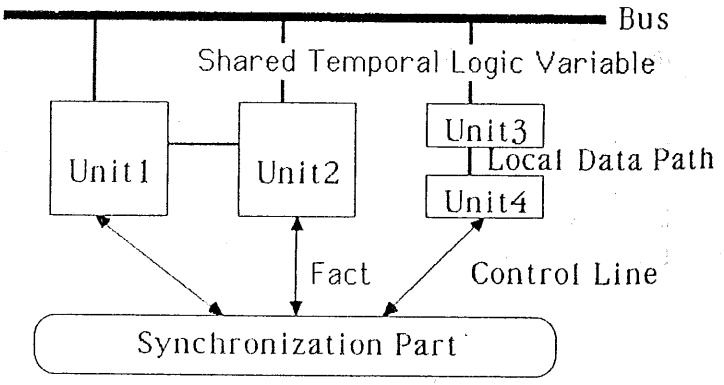

Fig.4 時相論理変数によるオブジェクト間通信

参考文献

〔1〕B.Moszkowski, "A Temporal Logic for Multilevel Reasoning about Hardware", IEEE Computer Magazine,February 1985

〔2〕河野，青柳，藤田，田中，"時相論理型言語TOKI 0 の検証"，Proc. of The Logic Prog. Conf. June 1986

〔3〕桜川，"Temporal Prolog"，ソフトウェア基礎論15—2 December 1985