時相論理型言語Tokio の処理系に関する考察

5F-4

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 の目的

Tokio は，時相論理に基づき，ハードウェア設計，並列アルゴリズム開発をサポートすることが目標である。Tokio では，様々なレベルからハードウェア，アルゴリズムを記述でできる。これらの記述に対してサポートするのは，次のようなツールである。

①アーキテクチャ評価支援ツール
②パイプライン化支援ツール
③タイミング検証ツール
④Tokio 言語インタプリタ・コンパイラ

これらは，specfication，description，implementation などを連結するツールになっている。（図1 ）

ここでは，④について必要な実行方式について検討する。Tokio は基本的には，プログラミング言語であるが，この座標軸に対応して，それぞれことなる形式の実行をおこなうことが必要である。ここで要求されるのは，

a.仕様の直接実行
b.決定的な記述のシミュレーション
c.柔軟なプログラミング言語としての実行

などである。

a.は，ハードウェア，アルゴリズムの開発時に，非決定的な仕様記述を含んだ形で記述されたものの直接的な記述である。この時には，絶対的な速さよりも，作成実行サイクルと実行の正確さが問題となる。b.では，仕様を実際の回路に落していく際にできる大規模な回路の高速シミュレーションが要求される。c.では，回路設計，アルゴリズム設計の際に，詳細な記述を必要としない部分や，まだ設計の終っていない部分，さらには，実行状態を表示するためのシステムインタフェースなどの記述に使われる。

	motivation	Tokio Tools
Specification	experiments	execution
Descripton	validation	verification
Implementation	integration	simulation

図1　Tokio の視点

2.Tokio の構成

Tokio のプログラムは，各時刻の値をもつ変数と，実行される時刻を時相演算子によって指定した述語からなる。Tokio は，Prologの拡張であり，ＡＮＤ，ＯＲ，の論理関係が各述語間に存在する。@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。プログラムは，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。

Tokio の非決定性を生成する部分は，二つあり，一つは，直接的にＯＲ項からくる部分で，もうひとつは，時区間を分割する時相演算子からくる部分である。

3.Tokio の実行

Tokio の実行は，clausal formに書かれた時相論理式のresolutionである。このときＡＮＤ，ＯＲ，時相演算子（とくに，@，＆＆）で作られる証明木をどの順に実行していくかが問題となる。逐次実行の場合，

①　ＯＲ方向のdepth first
②　ＡＮＤ方向のdepth first
③　時間方向のbredth first

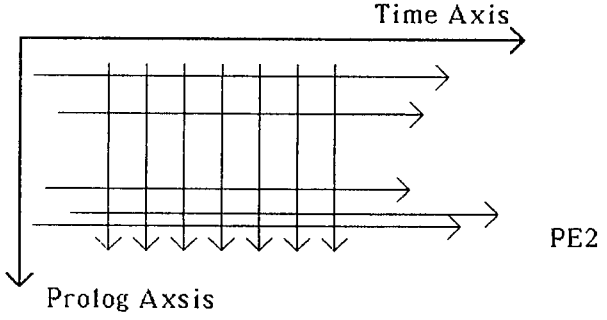

図2　Tokio の実行

Some difficult points in executing
Temporal Logic based Language Tokio
Shinji KONO, Hiroshi NAKAMURA　Masahiro FUJITA＊, Hidehiko TANAKA
The Tokyo University,　＊Fujituu Laboratory

④ 時区間の短い順

が，まず考えられる（図2）。現在のTokio は，この方法である。この方法では，記述された時相論理式と同等の順に，実行され，自然なシミュレーションができる。

反面，③のbredth firstが，④の時区間の長さの決定を遅らせる効果と相まって，実行効率を落している。この方法は，a.の用途には向いているが，大規模なシミュレーションには向かない。また，①②④の方法では，実際に到達できる解が実現できない可能性がある。したがって，次のような実行方法がよい。

① ＯＲ方向のbredth first
② ＡＮＤ方向のbredth first
③ 時間方向のbredth first
④ Linear Programmig による時区間の解決

この方法では，シミュレーションのためには，効率的でない。通常詳細なシミュレーションに入る場合には，1.決定的な実行系列，2.時区間の固定，等の条件が満たされているので，次のような戦略がよい。

① ＯＲ方向のdepth first
② ＡＮＤ方向のdepth first
③ 時間方向のdepth first
④ 既知の時区間の長さ

この場合，depth fristにより，無限長の証明木の枝に入らないように，プログラムする必要がある。このようなプログラミングは，一般には，難しいので，①②③をデータ駆動型のdepth first 実行にするとよい。

○ データ駆動による実行

これにより，既存のシミュレータと同等の効率をえることができる。但しこれには，Tokio のプログラムからのコンパイルが更に必要である。一つの方法は，発表5F-2のように回路に落して，既存のシミュレータを用いる方法がある。

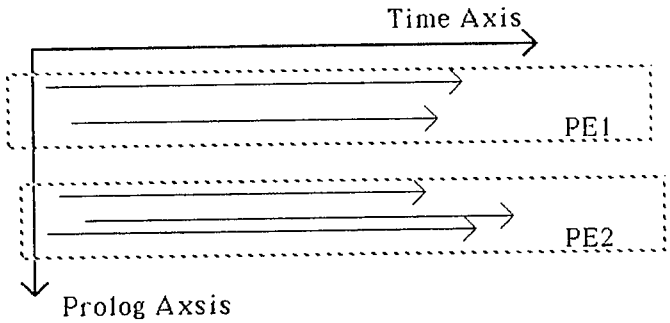

図3 時間方向逐次，現在軸方向並列

4.Tokio の並列実行

3.で考察した，実行順序は，もちろんそれぞれ並列に実行することもできる。この時に問題になるのは，プロセッサ間の共有変数への参照を押さえること，大域的な

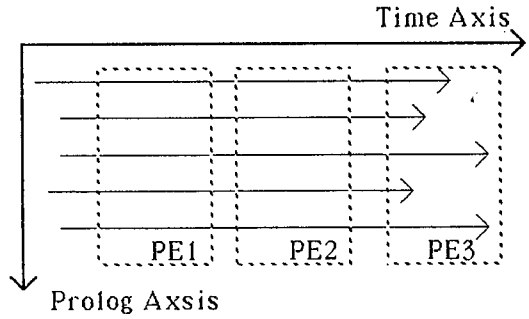

図4 時間軸方向並列，現在軸方向逐次

バックトラックの回避である。①②は，Prologと同じなので，Tokio 自身は，③④をどのように，実行するかが問題となる。最初の選択は，③④のみ並列におこない，①②を，並列に行うものである。

(1)時間方向逐次，現在軸方向並列　図3

この場合，①②をＡＮＤ並列として実行することが，シミュレータとして望ましい。時間軸方向のバックトラックは，大域的なバックトラックとなるので許さない。この方法は，全体として，決定的であることを要求するので，データ駆動的な方法により実現するのがよいと考える。

(2)時間軸方向並列，現在軸方向逐次　図4

この方法では，プロセッサ台数Ｎの時に，プロセッサnに対してmod n の時刻のclauseを割り当てる。データの流れは，リング状に流れることになる。この場合，シミュレータとしては，実行順序が，時間と関係なくなるで，奇妙な動作になるが，各時刻の独立性が良いことと，様相演算子により，動作させるプロセッサを指定できると言う利点がある。シストリックアレイを実行させた場合に，パイプライン状に並列に実行される。

それぞれの場合に，時間軸方向の制限を利用して，限定的なバックトラックを導入することができる。

参考文献

情報処理学会第35回全国大会（1987）（5f-2）