時相論理型言語Tokio の実装

河野真治　青柳龍也　藤田昌宏＊　田中英彦
東大工学部
＊現在，富士通研

Tokio is a hardware description language which directly executes first order Linear Time Temporal Logic (LTTL). The parallel data flow paths of a design are correctly defined in LTTL. Tokio can simulate these descriptions. Besides the LTTL operators, Tokio also contains the operators of Local Interval Temporal Logic (LITL). Using these operators, the temporal ordering of processes is declared easily.

The execution of Tokio is a kind of resolution of LITL. The resolution consists of three parts.

The unification of temporal variables is the first one. In Tokio the meanings of variable is defined in the same way as in LTTL. Temporal variables have many values corresponding to different times. In our implementation of temporal variables, we have the stream-like representation. There are two kinds of unification in Tokio. One unifies the value of variables only at a time. The other unifies these in full time.

The second part is called 'reduction to the future'. In LTTL, meaning of predicates are not determined in a single clock. The LTTL operator is developed into following time.

The last element of Tokio is interval splitting. The LITL operator needs the interval concept. An interval consists of its fin time and flag of ending. For example, The 'chop' operator creates new intervals with new interval parameters.

In the last section of this paper we present a set of compiler code of Tokio. This code is an extension of D. Warren's Prolog abstract instruction set. The execution speed of Tokio is almost the same as that of Prolog, because Tokio execution in a clock period is exactly the same.

Tokio is a good hardware description language and it is considered to be a natural extension of Prolog, its implementation is as efficient as Prolog.

1　時相論理型言語Ｔｏｋｉｏ

Tokio はFirst Order Linear Time Temporal Logic (LTTL) 〔1〕に基づくハードウェア記述を目的とした擬似並列論理型言語である。この種の言語としてはMoszkowskiのInterval Temporal Logic (ITL) 〔2〕に基づくTempura 〔3〕が知らているが，Tokio はLTTL上に時区間の概念を構成し，ITL と同等の記述力を持っている。LTTLは一階述語論理を完全に包含しており，Tokio 自身もPrologを包含している。

最近のLSI 技術の進歩はより大規模なハードウェアを可能にしたが，それを人手により設計する事は不可能であり，CAD に対する要求は非常に大きい。ハードウェア記述はその基礎であり，DDL，CDL 等のハードウェア記述言語，さらには，Prolog 〔4〕が用いられてきた。論理型言語は数学的基礎がはっきりしており，検証，合成等の支援が行い易いと考えられる。ハードウェアにはもともと高度な並列性を含んでおり，より容易な並列性の記述が望まれる。LTTLはその様な要求にそった論理であるが，順序性の記述を美しく行うことはむずかしい。またシミュレーションは設計の重要な要素だが，LTTL を直接実行するものは従来なかった。また大規模な設計に対しては，階層的な記述ができることが望ましい。Tokio ではITL とLTTLの二重の記述を使い階層的な記述を実現し，その実行は仕様のシミュレーションである。

これまでの並列論理型言語では並列に走る計算間の同期をとる機構が論理とは別に必要であった。それらは例えばRead Only Annotation 〔5〕であり，Guarded Horn Clause 〔6〕であり，変数の宣言〔12〕であった。これらの同期機構は論理自体に時間の概念を持つTokio では，時相論理内で自然な形で記述する事ができる。また順序性の記述は時区間の分割を行うchop演算子により容易に行うことができる。chopはITL の演算子であるがTokio ではLTTLの演算子と時区間を表す変数により実現される。Tokio の記述は，手続き的な記述から宣言的な記述まで連続的なスペクトルを持っているので，アルゴリズムの記述が主体のシステムレベルの記述から，デバイスの接続を主体とした低レベルの記述までを階層的に記述することができる。またLTTLに基づく言語であるのでLTTL上の検証系や合成系〔4〕と結びつけることが可能である。

Tokio の仕様記述をハードウェアに落すシリコン・コンパイラがTokio を最も高速に実行することができる。しかし，今回実装した言語は記述したハードウェアのシミュレーションを目標とする。しかし処理系としてのTokio は，再帰呼出しと時間方向への非決定的な実行を含むハードウェア記述より広い範囲を実行できる。

Tokio の実行はPrologの実行が単一化と縮退であるのと同様に以下の３つの要素からなる。

・時刻毎に異なる値を持つ時相論理変数の単一化

・通常の縮退，及び未来への縮退

・時区間の分割

以下の章では，まずLTTLの拡張としての ITLについて簡単に要約し，つぎに上記の３つの要素の実装法を考察する。さらにTokio に特有な機能についてのべる。最後にTokio コンパイラの実行コードを提案する。

2 Tokio の論理

Tokio はITL （Interval Temporal Logic ）をLTTL （Linear Time Temporal Logic）に基づき実行する言語である。ここでは，ITL と共にTokio の論理を説明する。ITL は命題の真理値が時間の区間にたいして定まる論理である。ここでは,ITLといえばLocal ITL （LITL）を指すことにする。”Local ”は命題変数の真理値が時刻（時区間の始まり）のみで定まり，時区間の終りの時刻によらないことを表す。これに対しLTTLでは命題の真理値も時刻のみにより決まる。ITL では時間を区間として考えることにより，順序性の記述がLTTLよりも容易となる。しかし，LTTL上には，既に論理回路の自動合成系および検証系が存在するので〔4〕，LTTLを拡張することによりITL の記述を可能にすることを考える。ITL ,LTTL の相互変換の為には新しい変数を導入することが必要であり，それらは区間変数とよ呼ばれる。

区間変数はITL の時区間をLTTLで表現するために時区間に対応するように導入した変数である。Tokio では命題Ｐに対し必ず区間変数Ipが附属するものと考える。Ip は，ある時刻Ip.begと，ある時刻Ip.finでＴ（真）である変数であり，＜Ip.beg,Ip.fin ＞で表わされるIp.begからIp.finまでの時区間を表す。Tokio の真理値は命題ＰをLTTLの命題と考えたときの真理値とIpの組で決まる。時区間Ipでは次の関係が成り立つ。

$$Ip.beg =< Ip.fin \qquad (2-1)$$

ITL の主要な時相演算子である@ (next) と　&& (chop) の定義は次のようになる。ここでの定義はITL に近い定義であり，LTTL上の解釈は言語処理系がつける。LTTL上でのTokio の解釈はここでは述べないが，Tokio インタプリタの実行そのものがLTTL上でのITL の解釈となっている。

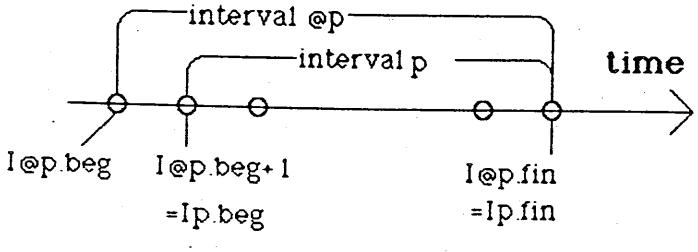

Fig.2.1 interval of ‘@’

@P

ＰはTokio の命題であり，区間Ｉ@ｐの次の区間Ipが存在して，そこでＰが成立することを表す。このとき，次の関係が成り立つ。Fig.2.1.

$$I@p.beg+1 = Ip.beg \qquad (2-2)$$

$$I@p.fin = Ip.fin \qquad (2-3)$$

時区間の長さとはfin-beg のことである。Ipの長さが0であるとIpの次の区間は存在しない。このような場合に真となるnextをweak next と呼び，偽となるnextをstrong next と言う。@は，strong next を表す。weak next は○で表す。LTTLのnextに対応するのはweak next である。

P　&&　Q

P，QをTokio の命題として，その区間Ｉｐ＆＆ｑを前半Ipと後半Iqに分割する。このときIp上で，Ｐが，Iq上でＱが成立する。このとき，次の関係が成り立つ。Fig.2.2.

$$Ip\&\&q.beg = Ip.beg \qquad (2-4)$$

$$Ip\&\&q.fin = Iq.fin \qquad (2-5)$$

$$Ip.fin = Iq.beg \qquad (2-6)$$

Ｐ＆＆Ｑは，最初にＰを実行し，次にＱを実行することであり，順序化の宣言である。LTTLでは，各時刻毎に宣言的な記述をするが，Ip を導入することにより，より手続き的な記述が可能となる。

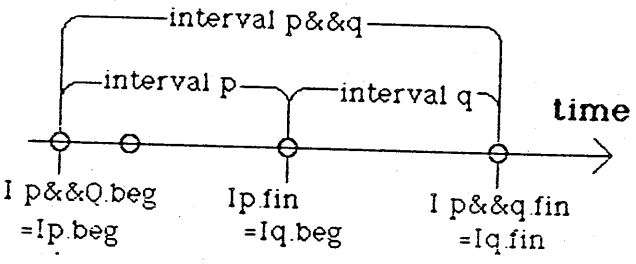

Fig.2.2 interval of ‘＆＆’

Tokio の命題は，一般的にはIp，即ち時区間に対して定まるが，時相論理の少数の演算子を除けば，その命題の真理値はIp.beg，即ち開始時刻だけで定まる。つまり時相演算子以外はlocal である。

Tokio は一階述語論理であり，変数の値は各時刻毎に存在する。これはITL ではLocal と呼ばれる性質である。命題論理ならばLocal な ITLには決定手続きが存在するが，local でないITL には存在しない。Tokio はLocal な ITLと同等な論理であり，LTTL上で解釈する点のみが異なるので，命題論理としてならば決定手続きは存在する。

一階の述語はその引数の全ての時間での意味から真偽が決定する。変数の現在の値のみを取り出したいときは，‘=’演算子を用いる。変数の未来の値を取出したいときには，@ (next) 関数を用いる。関数としてのnextは次の時区間が存在しない場合は任意の値をとるものとする。例えば次の様に定義される論理変数Ａを考える。