2J-9

時相論理型言語Tokioによるハードウェア記述

中村　宏・藤田　昌宏*・田中　英彦・元岡　達

東大工学部　*現在富士通研

1．はじめに

我々は、時相論理型言語Tokioを中心としたハードウェア設計支援システムを考えている。このシステムでは、システムレベルの仕様から状態遷移レベルまで段階的に、Tokioで一貫して記述する。そして、この状態遷移レベルの記述は、C-MOSゲートアレイ等に合成する。後半部についてはすでに報告されている[1]。ここでは、Tokioによる段階的なハードウェア記述例を示す。

2．なぜTokioか？

ハードウェアを記述する際、検証系及び合成系の支援は不可欠であり、数学的基礎のしっかりした言語で記述する必要がある。そのようなプログラミング言語として、prologに代表される論理型プログラミング言語があるが、基礎となっている古典論理に時間の概念がないため、並列性・順序性の記述が容易ではない。

これに対し、時間の概念を含んだ論理として時相論理（temporal logic）がある。Tokioは時相論理に基づいているので、ハードウェア記述に必要な並列性・順序性の記述が容易にできる。Tokioの詳細については[2，3，4，5]を参照されたい。

3．Tokioによるハードウェア記述

現在PIEのUNIRED（unificationとreduction）及びpipeline merge sorterなどがTokioで書かれている。

ここでは、アレイ方式の2ビット×2ビットのmultiplier[6]を例に挙げる。

（1）論理構成

fig.1の基本セルをfig.2のように並べた構成である。fig.2の各部の入力値は、（X1 X0）×（Y1 Y0）の場合に対応している。

（2）Tokioによる記述

アルゴリズムに沿った記述がfig.3である。

fig.2でわかるように、この計算では、順序性を持つ3つの時区間（インターバル）それぞれで2つの基本セルでの並列処理が行なわれる。それがfig.3のmul2の部分である。カンマ´,´で区切られたものは、同時に並列に実行されることを表わし、チ

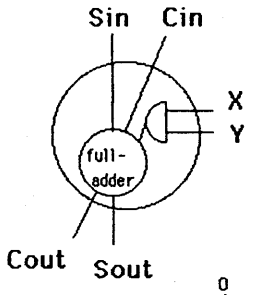

fig.1　基本セル

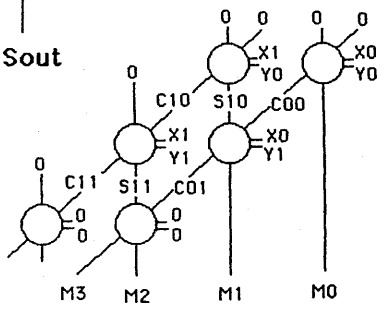

fig.2　アレイ方式multiplierの構成

```
mul2(X1,X0,Y1,Y0,M3,M2,M1,M0) :-
        cell(0,  0,  X0,Y0,M0,C00),
        cell(0,  0,  X1,Y0,S10,C10)
        &&
        cell(S10,C00,X0,Y1,M1, C01),
        cell(0  ,C10,X1,Y1,S11,C11)
        &&
        cell(S11,C01,0, 0,M2 , M3),
        cell(0,  C11,0, 0, _, _  ).


cell(Sin,Cin,X,Y,Sout,Cout) :-
        and(X,Y,L1),S1 <- Sin,C1<- Cin
        &&
        full_adder(S1,C1,L1,Sout,Cout).

full_adder(S1,C1,L1,Sout,Cout) :-
        half_adder(S1,C1,S2,C2),L2 <- L1
        &&
        half_adder(S2,L2,S3,C3),
        C4 <- C2
        &&
        or(C3,C4,Cout),
        Sout <- S3.

half_adder(A,B,S,C) :-
        length(1),
        A = 0, B = 0,
        @S = 0, @C = 0.
half_adder(A,B,S,C) :-
        length(1),
        A = 0, B = 1,      （一部省略）
        @S = 1, @C = 0.
```

fig.3　´&&´を用いたmultiplierの記述

ョップ゛&&゛はあるインターバルを2つのインターバルに分割することを表わす。従って、A&&BはAが終了してBが起きるという順序性を表わす。

1つのセルではand の後にfull-adderが実行されるという記述がcellの部分である。S 1<−Sin, C 1<−Cin, という記述は、and が開始される時刻のSinとCinの値をand が終了する時刻のS 1とC 1に転送することを表す。

実行される様子を時間軸上に書くとfig.4 のようになる。

1つのセルに着目すると、fig.5 のようなデータパスになる。それに従って記述したものがfig.6 である。

゛#゛はalwaysで、P:−#Q. は、Pの定義されたインターバルの中の毎時刻でQが実行されることを表す。従って、cellが定義されたインーヘバル内で毎時刻delay , half-adder, or, and が実行され、データが逐次流れていく。

fig.5 でわかるように1つのcellの実行には4クロックかかるので、mul2の実行には12クロック必要である。データパスを意識したfig.6 の記述は、fig.3 の記述より低位レベルの記述といえる。

4. おわりに

ハードウェアの記述を段階的にTokioでできることがわかったので、今後は検証系、合成系を作成していく予定である。

5. 参考文献

[1] 藤田：時相論理型言語Tokioによるアルゴリズム記述とC−MOSゲートアレイの自動合成, Logic Programming Conference 13.1 1985

[2] 青柳：Tokioかける言語, Logic Programming Conference 8.1 1985

[3] 河野：時相論理型言語Tokioの実装, Logic Programming Conference 8.2 1985

[4] 河野：情報処理学会第31回大会,2J-7 1985

[5] 青柳：情報処理学会第31回大会,2J-8 1985

[6] 須藤 ,中島 ,吉村：ハードウェアアルゴリズムとVLSI設計, 情報処理Vol.26 No.6

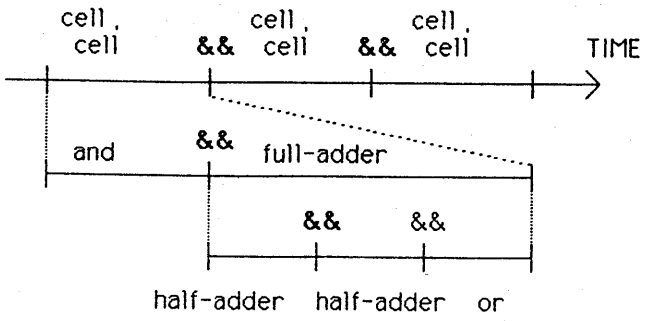

fig.4 時間軸上の実行の様子

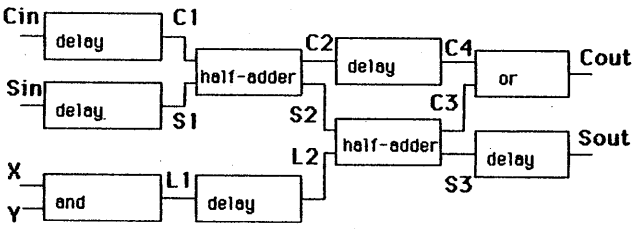

fig.5 セルのデータパス

```
mul2(X1,X0,Y1,Y0,M3,M2,M1,M0) :-
        cell(0,  0,  X0,Y0,M0, C00),
        cell(0,  0,  X1,Y0,S10,C10),
        cell(S10,C00,X0,Y1,M1, C01),
        cell(0  ,C10,X1,Y1,S11,C11),
        cell(S11,C01,0, 0, M2, M3 ),
        cell(0,  C11,0, 0, _, _   ).

cell(Sin,Cin,X,Y,Sout,Cout) :-
        #delay(Sin,S1),
        #delay(Cin,C1),
        #and(X,Y,L1),
        #delay(L1,L2),
        #half_adder(S1,C1,S2,C2),
        #delay(C2,C4),
        #half_adder(S2,L2,S3,C3),
        #delay(S3,Sout),
        #or(C3,C4,Cout).

delay(X,X1) :-
        @X1 = X.
```

fig.6 ゛#゛を用いたmultiplierの記述