数理解析研究所講究録
第458巻 1982年 142-151

ハードウェアアルゴリズムの記述法について

京都大学　工学部　　大井　康
安浦　寛人
矢島　脩三

1. まえがき

VLSI等のハードウェア技術の発達に伴い，論理設計の対象となるハードウェアの大規模化・複雑化が進み，本質的に並列動作するようなハードウェアアルゴリズムの記述法の確立が重要な課題となってきている[(1)(2)(3)]。ここでは，アルゴリズムの複雑さの理論的立場から，ハードウェアアルゴリズムの記述法について考察する。

ハードウェアとハードウェアアルゴリズムの関係においては，1つのハードウェアアルゴリズムに対する幾通りものハードウェア実現が考えられるし，逆に，幾つかのハードウェアアルゴリズムがその上で動くようなユニバーサルハードウェアを考えることもできる。すなわち，個々の問題を解くハードウェアアルゴリズムの抽象的概念は，具体的なハードウェアを記述するために開発された，現存するハードウェア記

述言語[4][5]では記述しにくい，あるいは記述できない性質のものである。

しかし，ハードウェアアルゴリズムが，それを実現する実体としてのハードウェアを意識しながら設計されている限り，両者に記述上の類似点は見出せると考えられる。ここではハードウェアアルゴリズムを，内部の動作仕様と外部に対する入出力仕様を記述することによって表現し，またこのようなアルゴリズム表現を要素として含む，新たなアルゴリズムも定義できるような，階層的な記述も扱えるようにする。

ここでは，このようなハードウェアアルゴリズム記述言語（以下 HADL と略す）の設計目的や設計指針について考察し，また，HADL を一つ提案して具体的なアルゴリズムの記述例を示す。

2. HADL の設計目的

ハードウェアアルゴリズムを，図等の直観的なイメージでなく，形式的記述で表現することは，それ自身のあいまいでない仕様を与え，ひいては正当性の検証や各種の解析を可能とする。特に計算量の理論の立場からは，アルゴリズムの使用ハード量や実行時間のめやすが，その記述を解析することで得られるのが望ましい。すなわち，HADL 記述を行なうことによって，次のような検証・解析が可能になると考えられ

る。

(1) シミュレーション等によるアルゴリズム自身の検証

(2) 使用ハード量，計算時間等の解析

これに対し，実際面からの要求としては，ハードウェアアルゴリズムの実現に対する指標を，その記述がある程度与え得ることが望まれよう。さらに，ある種の実現法に固定したときの使用ハード量，計算時間等のより正確な再評価がたやすいことも重要な点である。

3. HADLの設計指針

ここでは，ハードウェアとハードウェアアルゴリズムの差異を意識しながら，2節で示した目的に合うようなHADLを設計しようとする際の指針について考察する。

HADLは次のような特徴を備えているのが望ましいと考えられる。

(1) 記述の階層性

アルゴリズムを一度に細部にわたって書くのではなく，幾つかのレベルに分けて書く。アルゴリズムの，記述からの把握が容易になり，また，アルゴリズム全体の計算量に関する解析に柔軟性を与える。（この場合，アルゴリズムはモジュールとしてとらえられている）

(2) 厳密な入出力仕様

実データがどのようなタイミングにどこから入り，または出ていくのかという時間／空間的なデータ分布は，ハードウェアアルゴリズムの動作の本質的な部分である。

(3) サブモジュールの発生・消滅

アルゴリズムのダイナミックな動作をとらえるため，時間の経過あるいは隣接するモジュールの要請に基づく，実行中のモジュールの起動状況を記述する。これにより，アルゴリズムの重畳の可能性等が検討できる。

(4) サブモジュールの結合を直接記述しない

サブモジュール間の相互関係は，結線のような静的な結合ではなく，動的なデータ転送に重点を置く。これにより，物理的な回路構造と論理的なアルゴリズムを分離して解析することができる。

(5) 同期記述と非同期記述を区別する

あるモジュール内でなら常に有効であるクロックの設定ができる同期記述，本質的に非同期であるような独立したサブモジュール群の結合を，サブモジュールを単位として書けるような非同期記述の両者がHADLには必要である。

4. ハードウェアアルゴリズムの記述例

HADLの具体的な仕様を一つ提案し，それによる記述例を

2つ次に示す。記述したのはNビット加算のアルゴリズムである。前者は順序機械を用いたもの，後者は順次桁上げ加算器である．後者については，特にそのパイプライン性に着目した記述を行なっている。

記述全体は1つのモジュールと見なされ，大きく入出力仕様宣言部とアルゴリズム動作記述部に分けられる。入出力仕様宣言部では，仮想的な外部モジュール端子（変数）を宣言し，そこを実データがいかに流れるかを記述する。一方，アルゴリズム動作記述部は，機能的動作記述，構造的動作記述の2種に大別され，前者は，アルゴリズムの基本オペレーションとして許容できる範囲の動作を有限オートマトンのレベルで直接記述するものであるのに対し，後者は，1つのモジュールをいくつかのサブモジュールの集まりとみなし，それらの相互通信状況を，外部モジュール端子も含めて記述するものである。

図a)において，その動作は機能的に表現されている。すなわち，<< BEHAVIOUR >> 部では，1ビット直列加算器を表わす有限オートマトンが記述されている。それに対して，図b)における動作は構造的である。すなわち，<< BLOCK >> 部で，図a)のN=1の場合をサブモジュールとして含むよう定義し，それを << BEHAVIOUR >> 部で，動的に発生/実行/消滅させる

```
((MODULE))  SQ_ADDER.FUN(N);
  ((CONST))  N=16;
  ((INPUT DATA))  AUGEND(N), ADDEND(N), ICARRY;
  ((OUTPUT DATA)) SUM(N), OCARRY;
  ((INPUT ENTRY))  A, B, CARRY_IN;
  ((OUTPUT ENTRY)) C, CARRY_OUT;
  ((INPUT SPEC))
     CARRY_IN:=ICARRY;
     DO I=1 TO N;
        A:=AUGEND(I);
        B:=ADDEND(I);
        CLOCK;
     END;
  ((OUTPUT SPEC))
     DO I=1 TO N;
        SUM(I):=C;
        CLOCK;
     END;
     OCARRY:=CARRY_OUT;
  ((BEHAVIOUR))
     DECLARE FF;
     FF:=CARRY_IN;
     DO I=1 TO N;
        C= A ⊕ B ⊕ FF;
        FF=A·B+B·FF+FF·A;
        CLOCK;
     END;
     CARRY_OUT:=FF;
((MEND))
```

図a)　順序機械を用いた
加算アルゴリズム

```
((MODULE))  RC_ADDER.STR(N);
  ((CONST))  N=16;
  ((INPUT DATA))  AUGEND(N), ADDEND(N), ICARRY, CNTSGNL;
  ((OUTPUT DATA)) SUM(N), OCARRY;
  ((INPUT ENTRY))  A(N), B(N), CARRY_IN, CONTI;
  ((OUTPUT ENTRY)) C(N), CARRY_OUT, CONTO;
  ((INPUT SPEC))
     CONTI:=CNTSGNL;
     CARRY_IN:=ICARRY;
     DO I=1 TO N;
        A(I):=AUGEND(I);
        B(I):=ADDEND(I);
        CLOCK;
     END;
  ((OUTPUT SPEC))
     DO I=1 TO N;
        SUM(I):=C(I);
        CLOCK;
     END;
     OCARRY:=CARRY_OUT;
  ((BLOCK))
     FADD(*)=SQ_ADDER.FUN(1);
  ((BEHAVIOUR))
     GENERATE FADD(1) (CONTI);
       FADD(1).A=A(1),
       FADD(1).B=B(1),
       FADD(1).C=C(1),
       FADD(1).CARRY_IN=CARRY_IN;
     END GENERATE;
```

```
    DO I=1 TO N-1;
       GENERATE FADD(I+1) (EXIST(FADD(I)));
         FADD(I+1).A=A(I+1),
         FADD(I+1).B=B(I+1),
         FADD(I+1).C=C(I+1),
         FADD(I+1).CARRY_IN=FADD(I).CARRY_OUT;
       END GENERATE;
       IF I=N-1 THEN GOTO ENDPROCESS;
       EXEC FADD(I);
       DELETE FADD(I);
    END;
    ENDPROCESS:
    EXEC FADD(N);
    CONTO=EXIST(FADD(N));
    CARRY_OUT=FADD(N).C;
    DELETE FADD(N);
((MEND))
```

図 b) 順次桁上げによる
加算アルゴリズム

プロセスを表現している。

同期記述のためにはCLOCK文を導入している。あるモジュールの一単位時間の経過は，1つのCLOCK文で示される。この記述では，時刻の値という概念を用いていないが，もしこれを用いる場合には，制御との兼ねあいを考慮せねばならない。すなわち，時刻を判断し伝達する作業には，ハード量や時間のコストが課せられる場合があり得るからである。

2つのCLOCK文の間のデータ転送は，その順序に関係せず同時に生ずるとみなしてよい。

5. あとがき

HADLについての若干の考察について述べた。しかし未解決の問題はまだまだ多い。例えば，ハードウェアアルゴリズムの入出力仕様がどの程度自然か，それを作ったり利用したりするプリプロセス，ポストプロセスの複雑さをどう考え，アルゴリズムの計算量に関連づけるかは，そういった問題の一例である。

謝辞

御討論いただいた上林彌彦助教授をはじめとする本学矢島研究室の諸氏に感謝致します。

参考文献

(1) H.T. KUNG : The Structure of Parallel Algorithms
ADVANCES IN COMPUTERS. VOL.19 pp. 65-112 Academic Press 1980

(2) 都倉信樹 : VLSIアルゴリズムおよび面積時間複雑度
情報処理 VOL.23 NO.3 pp. 176-186 1982年3月

(3) 矢島・安浦・上林 : ハードウェアアルゴリズムの設計とその問題点
信学技報 AL81-86 pp. 49-58 1981年12月

(4) Su, S.Y.H. : A Survey of Computer Hardware Description Language in USA
Computer, IEEE Dec. 1974

(5) 坂村 健 : ハードウェア記述言語
情報処理 VOL.22 NO.6 pp. 570-573 1981年 6月