C-10

分岐機構を強化したＳＩＭD型並列コンピュータの開発
Development of an SIMD parallel computer with enhanced branch mechanisms

大部　雄也†　　高橋　義造‡
Katsuya Ohbe　　Yoshizo Takahashi

１．まえがき

ＳＩＭD型マシンは構造が簡単で高並列度の実現が容易である．しかし従来のものは単純な分岐機構のため多数の分岐命令を含む複雑なプログラムの処理は困難である．そこでＭＩＭD型マシンで通常使用されるＳＰＭD型プログラムをそのまま実行できるような新しい分岐機構をＳＩＭD型マシンに付加する事を考えた．従来のものでは分岐条件を満たさないプロセッサ要素（ＰＥ）は無条件にスリープ状態となり，制御プロセッサ（ＣＰ）から送られる命令を無視するようになっている．もしもすべてのＰＥが分岐条件を満たさなければ，その間ＣＰから送られるスレッドは実行されないのでその時間が無駄になる．このような事態をさけることのできる新しい分岐機構を開発した．

２．ＳＩMD型マシン

ＳＩＭD型並列マシンの構造はＭＩＭD型のものに比べ簡単であるため，高並列度の並列処理実現が容易である．しかし単一命令の並列同時処理という事で，処理能力に限度が生じる．また分岐命令に対する動作が貧弱である為に，多数の分岐命令を含むプログラムを処理する際，多くの無駄なスレッドを排出し多大なタイムロスが生じる事もある[1][2]. 今図１のプログラムを実行したとする．

```
Program{
        A;
        IF(条件 1){ B;
                IF(条件 2)   C;
                ELSE         D;
        } ELSE   E;
        F;
        WHILE(条件 3)   G;
        H;
}
```

図１　分岐命令を含むプログラム

A，B，C,...，Hは独立した命令群（スレッド）である．従来のＳＩＭD型ではＣＰからＰＥに対し(A)(B)(C)(D)(E)(F)(G)･･･(G)(H)という順にスレッドを送りＰＥはそれらを実行する．また Program 内に生じる条件分岐に対しては，条件を満たすＰＥのみが実行を行う．ここで１つ問題が生じる．例えばもしすべてのＰＥが（条件１）を満たしたならば(E)は無駄なスレッドになってしまう．またその逆にすべての PE が（条件１）を満たさなければ，(B)(C)(D)は無駄なスレッドとなってしまう．従来のＳＩＭD型では，このような状況を解決するために，タグやマスクレジスタでＰＥの状態を記録しそれぞれのＰＥとＣＰが通信を行う事で無駄なスレッドの発生を回避している[1]. この考え方は簡単なプログラム構造の際には効果的であるが，多大な数のＰＥが存在する場合や条件分岐が複雑に存在する場合の実現は非現実的である．そこで，どのような場合にも柔軟に対応可能な新しい分岐機構を考えた．

†大阪工業大学大学院情報科学研究科
Graduate School of Information Science, Osaka Institute of Technology
‡大阪工業大学情報科学部
Faculty of Information Science, Osaka Institute of Technology

３．新しい分岐機構

3.1　アーキテクチャ

図２は新しい分岐機構を持つＣＰとＰＥのアーキテクチャである．

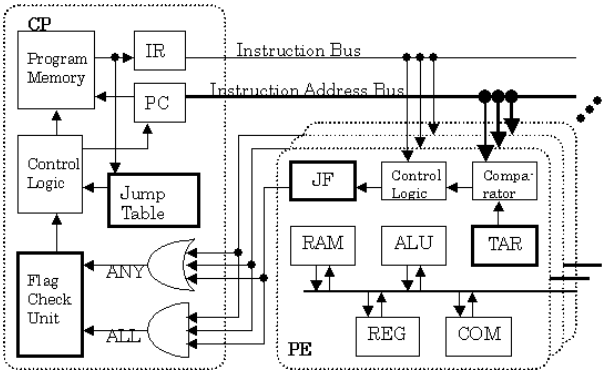

図２　新しい分岐機構をもつＣＰとＰＥのアーキテクチャ

図２において JumpTable（JT），FlagCheckUnit（FCU），JumpFlag（JF），TargetAddressRegister（TAR）が新しく付加したものである．更に CP が現在実行している命令のアドレス番地を Instruction Address Bus に送信するので PE が実行アドレスを確認する事が可能になる．

3.2　分岐動作

分岐動作には前方分岐と後方分岐が存在する．表１はそれぞれの動作について表したものである．表１のように前方分岐と後方分岐では動作が異なる．前方分岐の場合には，PE は分岐条件を満たすと TAR に分岐後のアドレスを格納し JF を立ち上げスリープ状態に入る．その後 Instruction Address Bus により CP の実行しているアドレスを確認し TAR の値と一致すれば JF を下げ通常動作に戻る．また CP は前方分岐の際，JF の数を確認し

表１　分岐命令におけるＣＰとＰＥの動作

CP/PE	分岐先	状態	動作
CP	前方分岐	ALL=1	PC=JT(MIN); *1
		ALL=0	PC++;
	後方分岐	ANY=1	PC=operand;
		ANY=0	PC++;
PE	前方分岐	分岐条件を満たす	JF=1; TAR=operand; sleep;
		分岐条件を満たさない	JF=0;
	後方分岐	分岐条件を満たす	JF=1;
		分岐条件を満たさない	JF=0; TAR=PC+1; sleep;

すべてが立ち上がっていれば（ALL=1）前方全体分岐を行う．その際の分岐先アドレスは JT によって決められる．これに対し後方分岐の場合には，PE は分岐条件を満たすと JF を立ち上げるだけでよい．そして CP は１個でも JF が立ち上がっていれば（ANY=1）後方全体分岐を行う. また条件を満たさない PE は TAR に現在のアドレスに'1'を加えた値を格納しスリープ状態となる．スリープの間の動作は前方分岐の際と同じで CP の実行アドレスを確認し, TAR の値と一致すれば動作を再開する. つまり，後方分岐の際は前方分岐とは逆に１つでも PE が後方分岐を必要とすれば全体で必ずその位置に分岐することになる.

このように動作する事で動作全体の無駄を可能な限り省略し，最適な処理動作を行う分岐機構である.

3.3 Jump Table

条件分岐の際，分岐中に分岐が生じる二重分岐などの多重分岐の場合がある．その中でも特に前方分岐中の前方分岐の際には特に注意が必要である．例えば外側の前方分岐の際に特定の PE が条件を満たし JF を立ち上げたとする．次に内側の前方分岐に残りすべての PE が条件を満たすと，この時点ですべての JF が立ち上がる（ALL＝１）. この場合 CP は無条件に２つ目の分岐先に全体分岐を行なう．しかし１つ目の分岐先が２つ目の分岐先より前のアドレスであると，矛盾が生じてしまう．そこでこのような状況にならないように，分岐先アドレスを格納する JT というレジスタを PE の数だけ付加してある．そして，前方分岐の度に JF を確認し，１個でも JF が立ち上がっていれば（ANY＝１）分岐先アドレスを JT に記録しておく．また常に最小値にポインタを指定しておき，全体分岐の際にはその値を使用する(表１*1). この JT により多重分岐であっても確実な分岐先アドレスが確保される.

４．実行

実際の動作について考えてみる．今，PE２台の SIMD 型マシンで図３のプログラムを，３台で図４プログラムを実行したとする．図３における 3.・6.は前方分岐，図４における 4.は後方分岐である．表２は図３のプログラムを実行した場合の，表３は図４のプログラムを実行した場合の結果である.

```
1.        ld    x
2.        cmp   y
3.        jm    els
4.        ld    b
5.        st    a
6.        jmp   fin
7. els    ld    d
8.        st    c
9. fin:   equ   *
```

図３　前方分岐を含むプログラム

```
1.        ld    x
2. do:    sub   y
3.        cmp   y
4.        jp    do
5.        st    x
```

図４　後方分岐を含むプログラム

表２

(1)PE1 のみが分岐条件を満たす

命令	PE1	PE2	JF
1	EXE	EXE	0
2	EXE	EXE	0
3	EXE	EXE	ANY
4	—	EXE	0
5	—	EXE	0
6	—	EXE	ANY
7	EXE	—	0
8	EXE	—	0
9	EXE	EXE	0

(2)PE1,PE2 共に分岐条件を満たす

命令	PE1	PE2	JF
1	EXE	EXE	0
2	EXE	EXE	0
3	EXE	EXE	ALL
7	EXE	EXE	0
8	EXE	EXE	0
9	EXE	EXE	0

(3)PE1,PE2 共に分岐条件を満たさない

命令	PE1	PE2	JF
1	EXE	EXE	0
2	EXE	EXE	0
3	EXE	EXE	0
4	EXE	EXE	0
5	EXE	EXE	0
6	EXE	EXE	ALL
9	EXE	EXE	0

表３ PE1,PE2,PE3 で図４のプログラムを実行した場合の結果

命令	PE1	PE2	PE3	JF
1	EXE	EXE	EXE	0
2	EXE	EXE	EXE	0
3	EXE	EXE	EXE	0
4	EXE	EXE	EXE	ANY
2	—	EXE	EXE	0
3	—	EXE	EXE	0
4	—	EXE	EXE	ANY
2	—	—	EXE	0
3	—	—	EXE	0
4	—	—	EXE	0
5	EXE	EXE	EXE	0

表２(1)は分岐条件３を PE1 が満たし PE2 が満たさなかった場合である．この時，CP はすべての命令を送らなければいけないので通常の SIMD 型マシンと同等の動作となる．表３(2)は分岐条件３を PE1,PE2 共に満たした場合である．この場合４から６までの命令はどれも実行しないので CP は全体分岐で避ければよい．表３(3)は逆に PE1,PE2 共に分岐条件３を満たさない場合である．この場合７から８までの命令はどれも実行しないので CP は全体分岐で避ければよい．つまり前方分岐の際，新しく考えた分岐機構では表３(2)(3)のような場合，通常より最適な動作が可能となる．後方分岐の場合は表４のように，分岐の必要な PE がある限りその場で全体分岐をおこなう．表４の場合，１回目の分岐 PE１が条件を満たし，２回目で PE2 が３回目ですべての PE が条件を満たし，後方分岐のループを抜けるという動作である.

５．まとめ

SIMD 型マシンの分岐機構を強化し高速で柔軟な処理が可能なプロセッサの開発を行った．今後２０万ゲート FPGA に８台程度のプロセッサを搭載することを目標に，VHDL による設計を進める予定である.

参考文献

[1]Takahashi Yoshizo,et al, “An Enhancement of SIMD Machine for Executing SPMD Programs”, ParallelComputing:Fundamentals,Applications and New Directions,North-Holland,pp.203-206,1997

[2]W.Daniel Hillis(喜連川優訳), “コネクションマシン”, パーソナルメディア, pp.24-26, 1990