

TRON仕様に基づく32ビットマイクロプロセッサ TX1のCPUアーキテクチャ

岸上秀哉、増渕美生、内海透、宮森高、宮田操
Ⓐ東芝 半導体技術研究所

TX1はTRONCHIP32の仕様に基づく、東芝の32ビットマイクロプロセッサであり、組込み制御を主要な用途としている。TX1の目標性能は5MIPS以上でありこの性能を得るためにTX1では内部を4つのブロックに分割し、それらができるだけ非同期に独自の処理を進めるパイプライン方式となっているのが特長である。

TX1の内部構造を評価するために、TX1の機能シミュレータ上で各種ベンチマークプログラムを実行した。本報告ではTX1の内部構造の特長について述べ、シミュレーションによる評価結果について検討する。

- CPU architecture of 32bit microprocessor TX1 based on TRONCHIP specification -(in Japanese)

*by Hidechika KISHIGAMI, Yoshio MASUBUCHI, Tohru UTSUMI,
Takashi MIYAMORI and Misao MIYATA*

Semiconductor Device Engineering Laboratory
Toshiba Corporation
580-1, Horikawa-cho, Saiwai-ku, Kawasaki, 210, Japan

The 32bit microprocessor TX1 is TOSHIBA's first implementation of the TRONCHIP32 specification and is designed suitable for built-in machine control. The target performance is more than 5 MIPS. This performance level is obtained by dividing TX1 into four major blocks and operating them independently in a loosely coupled pipelining fashion.

Several benchmark programs have been executed on a functional simulator for TX1 to evaluate the performance of the microprocessor. In this paper, we report the measurements obtained from these benchmarks and discuss the effectiveness of the TX1 design architecture.

1. はじめに

TRONCHIPは1990年代の半導体技術を生かした究極のVon-Neumann型マイクロプロセッサを目指しており、基本命令の高速実行やOS用高機能命令の具備等の特長がある[1]。

東芝では、このTRONCHIP仕様に基づくマイクロプロセッサとして、TX1とTX3の2つの“TXシリーズ”32ビット・マイクロプロセッサを開発中である[2]。本報告ではこのうちシリーズの下位機種であるTX1の内部構造について紹介し、シミュレーションによる評価結果について述べる。

TX1は組込み制御を主な用途とした32ビット・マイクロプロセッサであり、その内部は4つの大きなブロックから構成される。そして各ブロックはかっちりとしたパイプライン構造に組み込まれているのではなく、それぞれができるだけ非同期に独自の処理を進める方式となっているのが特長である。また分岐命令の高速実行を実現するために、デコード部での“先行分岐方式”を採用している。今回はこれらTX1の内部構造を詳細に評価するために、機能シミュレータFAL[3]を用いて各種ベンチマーク・プログラムを実行した。

以下では、まず“TXシリーズ”の概要およびTX1の外部仕様について簡単に紹介した後、TX1の内部構造の特長とその評価結果について述べる。

2. “TXシリーズ”の概要

“TXシリーズ”はTRONCHIP仕様に基づく東芝の新規32ビット・マイクロプロセッサ・ファミリの総称であり、現在TX1、TX3の2つの32ビット・マイクロプロセッサを開発中である。表1にTX1、TX3の概要を示す。TX1は組込み制御を主な用途とし、平均5MIPSの性能をもつ。命令は全部で88種類あり、32ビット・マイクロプロセッサの基本機能のみ持つ。これに対してTX3はパソコンやワークステーション等を主な用途とし、平均10MIPSの性能をもつ。命令は10進演算命令、浮動小数点演算命令を含め約130種類あり、メモリ管理機能、大容量キャッシュ・メモリをもつ、高機能・高性能32ビット・マイクロプロセッサである。

また“TXシリーズ”ではTX1、TX3のマイクロプロセッサの他に、直系周辺LSIとしてクロックジェネレータ(CG)、割込みコントローラ&タイマ(ICT)、DMAコントローラ(DMAC)を、さらに高機能周辺LSIとしてTX1をコアとしたトランクリング・LANプロセッサ(TRL1)を各々開発中である。

表1. TX1, TX3の概要

	TX1	TX3
性能	5MIPS	10MIPS
クロック周波数	25MHz	33MHz
製造プロセス	1.0 μ m	0.8 μ m
トランジスタ数	45万	120万
パッケージ	135pinPGA	135pinPGA
外部バス・サイクル	2サイクル	2サイクル 1サイクル(バースト)
基本命令	88種類	101種類
十進命令	-	11種類
浮動小数点命令	-	18種類
MMU	無	2レベルページング 4レベルリング保護
キャッシュ	無	8Kバイト(命令) 8Kバイト(データ)

3. TX1の内部構造

3.1 TX1の内部ブロック

TX1の内部は図1に示すように、IFU (Instruction Fetch Unit), DCU (Decode Unit), EXU (Execution Unit)およびOMU (Operand Management Unit)の4つの大きなブロックから構成される。各ブロックの動作概要は次の通りである。

●IFU (Instruction Fetch Unit)

IFUは命令のプリフェッチを行うブロックで16バイトの命令バッファを持つ。IFUは現在命令バッファに空きが4バイト以上ある場合に、OMUに対してプリフェッチ要求を行う。

一方命令バッファ中の命令コードはDCUからの要求に従って、オペコードは2バイト単位で、イミディエートやディスプレースメントは2バイトまたは4バイト単位でDCUに送られる。オペコードの転送バス(2バイト巾)とイミディエート/ディスプレースメントの転送バス(4バイト巾)は分離されているので、命令バッファ中に命令が存在する場合には、1サイクルで最大6バイトをDCUに送ることができる。

●DCU (Decode Unit)

DCUは命令のデコードを行うデコード部とオペランドの実効アドレスの計算を行うアドレス生成部から構成される。デコード部ではIFUに対して命令コードの要求を行い、IFUから送られてきた命令コードのうちのオペコードのデコードを行う。デコードの結果判別された命令の種類は、その命令に対するマイクロプログラムの先頭アドレスとしてEXUに送られる。また、デコードの結果得られたオペランド情報はアドレス生成部に送られる。

アドレス生成部では、このオペランド情報とIFUから送られてきたディスプレースメントをもとにオペランドの実効アドレスの計算を行い、その結果をOMUに送る。また、オペランドがレジスタの場合にはレジスタ番号を、イミディエートやリテラルの場合には各々その値をOMUに送る。

DCUではその他に、無条件分岐命令や条件付き分岐命令の分岐先アドレスを計算し、“先行分岐”の処理も行う。“先行分岐”のメカニズムについては後述する。

●EXU (Execution Unit)

EXUでは命令がマイクロプログラム制御で実行される。EXUにはALU、バレルシフト、レジスタファイル等があり、内部は2バス構造となっている。EXUでは基本命令は2サイクルで実行される。

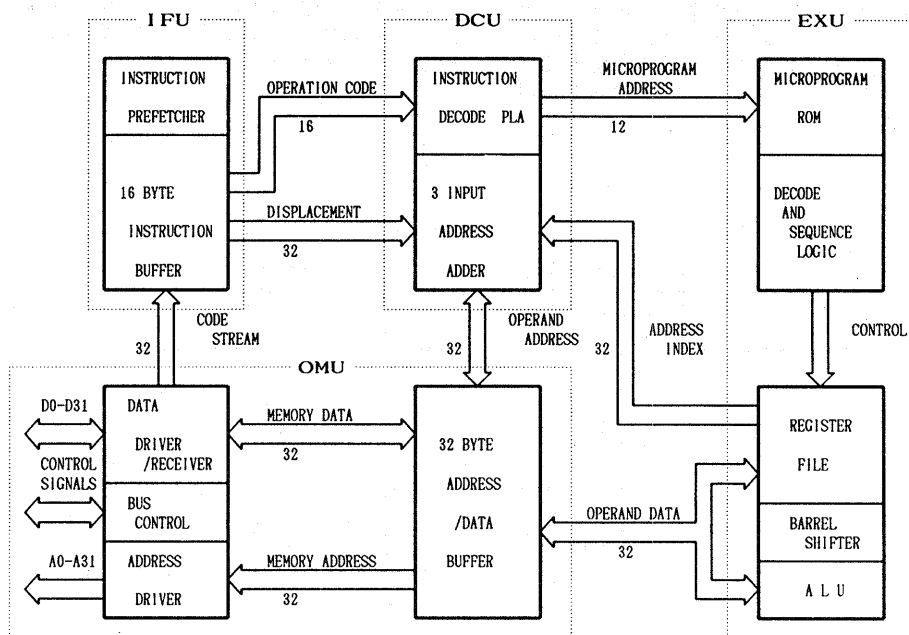


図1. TX1の内部ブロック

● OMU (Operand Management Unit)

OMUは、IFUからの要求による命令コードのプリフェッチ、DCUからの要求によるデコード中の命令のオペランドのプリフェッチ、EXUで実行中の命令のオペランドのフェッチ&ストアを行う。OMUにはDCU、EXUとの処理速度差を吸収するために、オペランドのアドレス&データを保持する4本のオペランドバッファが用意されている。オペランドバッファは、メモリオペランドのアドレス&データだけでなく、レジスタオペランドのレジスタ番号やイミディエートも保持する。そしてEXUで実行中の命令のオペランドをオペランドバッファを介してアクセスすることにより、命令を実行するマイクロプログラムをオペランドの種類に依存せず一本化することができる。またメモリライトもこのオペランドバッファを用いたストア・バッファ方式で実行される。

さらにOMUには外部バスとのデータ転送を制御するインタフェース機能もある。

3. 2 パイプライン構造

TX1は、4つのブロック (IFU, DCU, EXU, OMU) が各々できるだけ非同期に独自の処理を進める方式を採用しており、パイプライン構造は図2に示すようになっている。パイプラインの1つの流れは、命令フェッチ (担当ブロック; IFU)、命令デコード (DCU)、命令実行 (EXU) の3ステージからなり、この流れと並行して、オペランドの実効アドレスの計算 (DCU)、オペランドのプリフェッチ (OMU) の処理が進められる。また実行ステージで発生するメモリライトもパイプラインとは非同期にOMUで処理される。

このパイプライン構造の特長は、命令の実行と実効アドレスの計算/オペランドフェッチとが独立に進められるため、デコードにより命令の種類が判別すると、オペランドがフェッチされる前にEXUで命令の実行を開始することができることである (先行実行)。

また、命令のデコードが完全に終了する前にオペランド情報が揃えば、オペランドの実効アドレスの計算/オペランドフェッチの処理を開始することもできる。

3. 3 先行分岐

TX1は分岐命令の高速実行のために、DCUでの“先行分岐”方式を採用している。“先行分岐”は、DCUで分岐先アドレスを計算して、OMUに対して分岐先命令のフェッチを要求することにより実現される。例えば命令のシーケンスが

```

NOT @abs : 16
JMP LBL
...
...
LBL ADD Rn, Rm
    
```

の場合のパイプラインの流れは図3のようになる。この場合JMP命令 (B) はDCUで先行実行されるため、分岐先のADD命令 (X) のフェッチはJMP命令の実行開始前に行われる。条件付分岐命令の場合も、1つ前のEXUで実行中の命令により条件判断に用いられるフラグが変化しない場合は“先行分岐”が行われる。

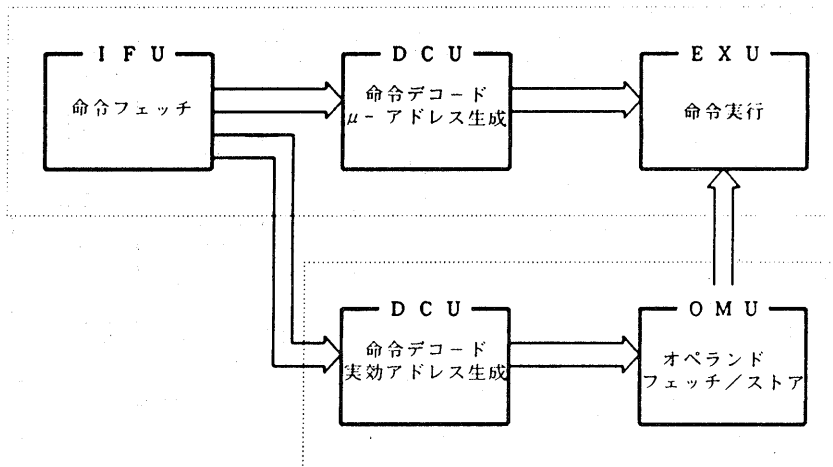


図2. TX1のパイプライン構造

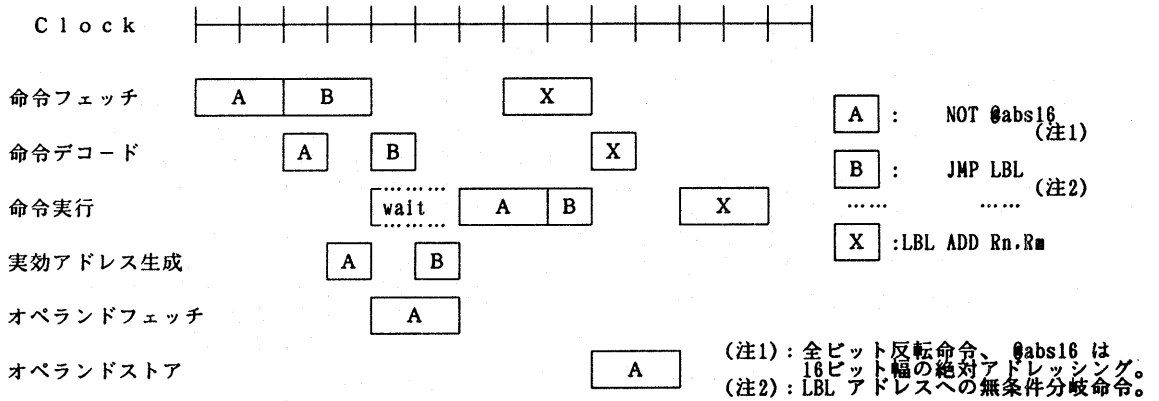


図3. 先行分岐のタイミング

3. 4 オペランドバッファ

TX1の特長の1つとして、オペランドのアドレス&データを保持するための4本のオペランドバッファがOMUに用意されていることがあげられる。オペランドバッファは図4に示すように、アドレスパート (AP 0-3)、データパート (DP 0-3) から構成される。同一番号のAP、DP、は同一のオペランドに対してのみ使用することができ、APの指すメモリアドレスの内容がDPに入る。このオペランドバッファにより、オペランドのプリフェッチの処理を、“命令フェッチ=>命令デコード=>命令実行”のパイプラインの流れから分離して、並行して行うことができる。

AP0	DP0
AP1	DP1
AP2	DP2
AP3	DP3

図4. オペランドバッファの構造

4. TX1内部構造評価

4. 1 方法

TX1の内部構造を詳細に評価するために、設計・検証用に機能記述言語H2DL[4]で記述されたTX1を用いて、機能シミュレータFAL上で各種ベンチマークプログラムを実行した。この方法は、従来行っていたGPSSを用いた評価[5]と比べて、次のような特長がある。

a) H2DLによる記述は、実際のTX1を“機能レベル”で完全に表現しているので正確なシミュレーションを行うことができる。

b) 入力として実際のプログラムを用いることにより、例えばオペランドのアドレス計算時のレジスタ干渉の影響等も考慮したシミュレーションを行うことができる。

このような利点がある反面、計算時間の制限からあまり大きなプログラムを実行することはできない。今回テストに使用したプログラムは次の5つである。

★Sieve

素数を求めるプログラムで、アセンブリ言語で作成したもの。今回は2~35の素数を計算した。

★Fibonacci

フィボナッチ数列f(n)を求めるプログラムで、C言語で作成されたプログラム[6]をSUN3上でコンパイルし、それを1対1対応でTX1の命令に変換したもの。今回はf(6)を計算した。

★Sort

クイックソートを行うプログラムで、C言語で作成されたプログラム[6]をSUN3上でコンパイルし、それを1対1対応でTX1の命令に変換したもの。今回は10個の数字をソートした。

★EDN-f

ビットのセット、リセットおよびテストを行うプログラムで、アセンブリ言語で作成したもの[7]。

★EDN-k

(7×7)の転置行列をもとめるプログラムで、アセンブリ言語で作成したもの[7]。

4. 2 評価項目

TX1の特長である、パイプライン構造、先行分岐、オペランドバッファの効果を評価するために、以下に示す値をシミュレーションにより求めた。なお外部メモリのアクセスは2サイクル(0wait)とした。

a) 実行サイクル数、実行命令数および総合性能の指標としてのMIPS値(0.25MHz)。
 b) 外部バス、DCU、EXUの稼働率およびDCU、EXUの要因別の待ち発生割合。(ただし“稼働率=1-待ち発生割合”、である。)

<DCUの待ち発生要因>
 ☆命令のデコードを終了したが、次の命令がIFUの命令バッファ(1B)にない。
 (待ちDCU/IFU)

☆命令のデコードを終了したが、前の命令がEXUで実行中である。(待ちDCU/EXU)

☆命令のデコードで生成したオペランドアドレス/データを、OMUのオペランドバッファ(OB)が詰まっているために格納できない。
 (待ちDCU/OMU)

<EXUの待ち発生要因>
 ☆命令の実行が終了したが、次の命令がDCUでまだデコードされていない。
 (待ちEXU/DCU)

☆命令の実行に必要なオペランドがOMUのオペランドバッファ(OB)にまだ格納されていない。
 (待ちEXU/OMU)

c) 命令バッファ(1B)の平均格納バイト数。オペランド・バッファ(OB)の平均使用本数。

d) オペランドリード、オペランドライト、命令フェッチの各メモリアクセス回数。

命令フェッチのためにオペランドリード/ライトが待たされたサイクル数の合計。
 (オペランドフェッチ衝突待ち)

e) 無条件分岐命令(BRA, JMP 命令)および条件付分岐命令(Bcc 命令)のEXUでの実行開始から次命令のEXUでの実行開始までの平均サイクル数。

(分岐命令実行サイクル)

f) DCUで命令のデコード終了前にその命令をEXUで実行したサイクル数の合計。ただしEXUでの待ち状態は含まない。

(先行実行サイクル)

5. シミュレーション結果と検討

シミュレーションで実行した命令の種類別数を表2に、またシミュレーション結果を表3に示す。以下結果の各項目について検討を行う。

a) 総合性能

総合性能の指標としてのMIPS値は、プログラムによりばらつきがあるが、4.4~7.5が得られた。SieveのMIPS値が高いのは、1命令あたりのメモリオペランドアクセスの割合が約14%と非常に低いからであり、逆にSortのそれは約99%と高いためにMIPS値が低くなっている。EDN-fのMIPS値があまり高くないのは、分岐する命令の割合が約34%と高いためと考えられる(他のプログラムでは13~21%)。EDN-kのMIPS値もあまり高くないが、この原因はビット操作命令や、比較とループを1命令で実行す

表2. 実行した命令の種類

	Sieve	Fibonacci	Sort	EDN-f	EDN-k
平均命令長 (byte)	2.4	2.4	3.3	2.8	2.7
実行命令数	330	234	673	89	239
MOV命令数	52	57	174	22	14
CMP命令数	74	15	106	15	21
算術演算命令数	85	22	115	0	54
論理演算命令数	0	0	0	0	0
シフト/ローテート命令数	0	0	62	0	0
PUSH/POP命令数	0	86	42	0	7
ビット操作命令数	0	0	0	9	66
無条件分岐命令数	45	39	38	18	8
条件付分岐命令 taken数	19	8	35	13	41
not taken数	55	7	71	11	28
その他	0	0	30	1	0

る命令など高機能命令を多く使用しているためと考えられ、この場合MIPS値は必ずしも実際の性能を表わしていない。このことはEDN-f やSortについてもある程度あてはまり、したがって今回求めたMIPS値は総合性能のひとつの“めやす”程度に考えるべきである。

b) 稼働率および待ち発生割合

どのプログラムの場合も外部バスの稼働率がやや高いが、DCU稼働率とEXU稼働率は同程度であり、まづまづバランスのとれたパイプライン構造といえる。

DCUの待ち発生要因では、待ちDCU/IFUの割合が高く、待ちDCU/EXUと待ちDCU/OMUの割合は低い。待ちDCU/IFUの割合が高いのは命令フェッチが命令デコードに追従できていないためであり、2サイクルのメ

モリアクセスでも“バスネック”の傾向にあることを示している。待ちDCU/EXUの割合が低いことは、EXUの処理能力が十分であることを示しているが、しかしこれは今回テストに用いたプログラムの基本命令の割合が高いためであり、実行サイクルの多い高機能命令の割合が高くなると待ちDCU/EXUの割合も上がることが予想される。待ちDCU/OMUの割合が低いことは、オペランド・バッファ(OB)の本数が十分であることを示している。

EXUの待ち発生要因のうち、待ちEXU/DCUの原因は、EXUとDCUの処理能力に差があるというよりは、むしろ待ちDCU/IFUが原因であると考えられる。また待ちEXU/OMUも“バスネック”に原因があると考えられる。

表3. シミュレーション結果

	Sieve	Fibonacci	Sort	EDN-f	EDN-k
実行サイクル数	1104	921	3533	506	1212
実行命令数	330	234	673	89	239
MIPS (@25MHZ)	7.5	6.4	4.8	4.4	4.9
外部バス稼働率	0.86	0.91	0.84	0.78	0.72
DCU稼働率	0.75	0.61	0.72	0.52	0.68
待ち DCU/IFU	0.25	0.27	0.20	0.41	0.19
待ち DCU/EXU	0	0.12	0.07	0.07	0.13
待ち DCU/OMU	0	0.001	0.015	0	0
EXU稼働率	0.62	0.57	0.51	0.56	0.58
待ち EXU/DCU	0.22	0.20	0.10	0.27	0.17
待ち EXU/OMU	0.16	0.23	0.39	0.17	0.25
平均IBバイト数	4.7	3.6	6.6	4.2	6.7
平均OB使用本数	0.8	1.2	1.5	0.9	1.1
オペランドリード数	17	71	489	18	71
オペランドライト数	28	71	174	15	28
命令フェッチ数	432	276	814	163	337
オペランドフェッチ衝突待ち (リード)	17	46	354	18	57
(ライト)	17	38	314	18	57
	0	8	40	0	0
分岐命令実行サイクル 無条件分岐命令(BRA/JMP)	5	3	2.8	0	7
条件付分岐命令(Bcc)	3.8	4.6	4.1	5.1	5.4
先行実行サイクル	106	40	143	34	83

c) IBの平均格納バイト数、OBの平均使用本数

Sieve, Fibonacci, EDN-f ではIBの平均格納バイト数は3.6 ~ 4.7 バイトと少ないが、逆にSort, EDN-kでは6.6 ~ 6.7 バイトと多くなっている。Sort, EDN-kの平均格納バイト数が多いのは、分岐する命令の割合が高くなく、高機能命令の割合が比較的高くEXUでの平均命令実行サイクルが大きいことが原因である。(EXUでの平均命令実行サイクルは、“実行サイクル数×EXU稼働率/実行命令数”で計算でき、Sieve, Fibonacci, Sort, EDN-f, EDN-k で各々 2.1、2.2、2.7、3.2、2.9となる。)この結果からみると、IBの容量は16バイトは多少大きい感じをうけるが、これも今回テストに用いたプログラムの基本命令の割合が高く平均命令長が短いこと等を考慮すると、16バイトの容量は適当であると考えられる。

OBの平均使用本数は0.8 ~ 1.5本であり、OBの数4本で十分である。このことは、待ちDCU/OMUの割合が低いことからわかる。ただし、SortではOBを4本全て使用している割合も結構高く、OBの本数が多すぎることはない。

d) メモリアクセスの回数、オペランドフェッチ衝突待ち

メモリアクセスの回数のうち、命令リード数が多く、これが“バスネック”の原因となっている。理想的な命令リード数は、“平均命令長×実行命令数/4”でありこの値を1とすると、実際の命令リード数の割合は1.9 ~ 2.5となり、“むだな”命令リード数が多いことがわかる。この原因はもちろん分岐命令にあるが、命令バッファ(IB)のプリフェッチ方式に改良を加えるなどにより、“むだな”命令リード数を減らすことは可能であると思われる。この点に関しては今後の検討課題である。

また命令リードのメモリアクセスのためにオペランドリード/ライトのメモリアクセスが待たされる割合(オペランドフェッチ衝突待ち)は2 ~ 10%程度あるが、これは“むだな”命令リード数を減らしたり、あるいは命令/データ分離型のキャッシュをチップ内部に持つこと等により改善される値である。

e) 分岐命令平均実行サイクル

TX1では分岐命令の高速実行のために、DCUでの先行分岐方式を採用しており、またオペランドのプリフェッチの処理をパイプラインの流れから分離して、非同期で行うことにより、実質のパイプライン段数を減らし、分岐によるペナルティの減少をはかっている。分岐命令実行サイクルは分岐先の命令長に依存するが、無条件分岐命令(BRA命令)では早い場合平均3サイクル程度であり、条件付分岐命令(Bcc命令)では3.8 ~ 5.4サイク

ルと比較的短いサイクルで実行している。先行分岐を行わない場合は、分岐に最低でも5サイクル要することを考えると、先行分岐の効果が表れているといえる。

f) 先行実行サイクル

TX1ではDCUでのデコードで命令の種類が判別すると、命令のデコードが終了する前にEXUでの命令実行を開始することができる。この効果を調べたのが先行実行サイクルであるが、先行実行により4.0%~9.6%の性能向上が得られたことになる。先行実行は命令長の長い命令や、実行サイクルの長い高機能命令の場合にその効果が大きいことを考えると、実際にはより効果が得られることが予想される。

6. おわりに

以上、TRONCHIP仕様に基づく東芝の32ビット・マイクロプロセッサ“TX1”の内部構造について紹介し、機能シミュレーションによる評価結果について報告した。総合性能としては、4.4 ~ 7.5MIPSという結果が得られ、またTX1の特長であるパイプライン構造、先行分岐、オペランドバッファに関する種々の情報を調べることで、これらの効果について確認することができた。今後より大きなプログラムによる性能評価を行っていく予定である。

7. 参考文献

- [1] K.Sakamura, "Architecture of the TRON VLSI CPU", IEEE Micro Vol.7.No2, April, 1987, pp17-31
- [2] K.Namimoto, T.Sato and A.Kanuma, "TX series based on TRONCHIP Architecture" TRON Project 1987, Springer-Verlag, pp.291-308
- [3] 武井、他、“機能論理シミュレータFALを核とした設計検証システム”, 電子情報通信学会 回路とシステム研究会, CAS86-184, pp79-86
- [4] 西尾、他、“階層的ハードウェア設計言語H2DLの言語仕様”, 情報処理学会自動化研究会 22-2(1984.7.17)
- [5] 宮田操, 増淵美生, 岸上秀哉, "TX1のパイプライン構造設計", 第2回リアルタイムOS TRON研究会資料, pp13-22, 電子情報通信学会, 1987年7月
- [6] BYTE Editorial Staff, "High-Tech Horsepower", BYTE, July, 1987, pp101-108
- [7] Robert D Grappel and Jack E Hemenway, "A tale of four μ Ps: Benchmarks quantify performance", EDN April 1, 1981, pp179-265