マイクロコンピュータ 48－9
計算機アーキテクチャ 69－9
（１９８８．１．２１）

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に、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をコアとしたトークンリング・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	１．０μｍ	０．８μｍ
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 十進命令 浮動小数点命令	８８種類 － －	１０１種類 １１種類 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、バレルシフタ、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。

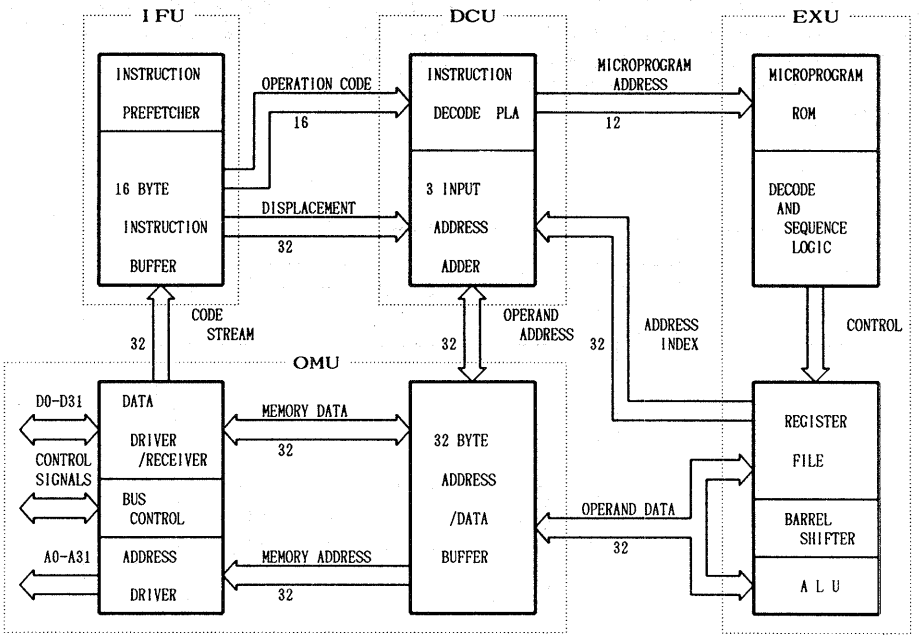

図1．　ＴＸ１の内部ブロック

●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に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。

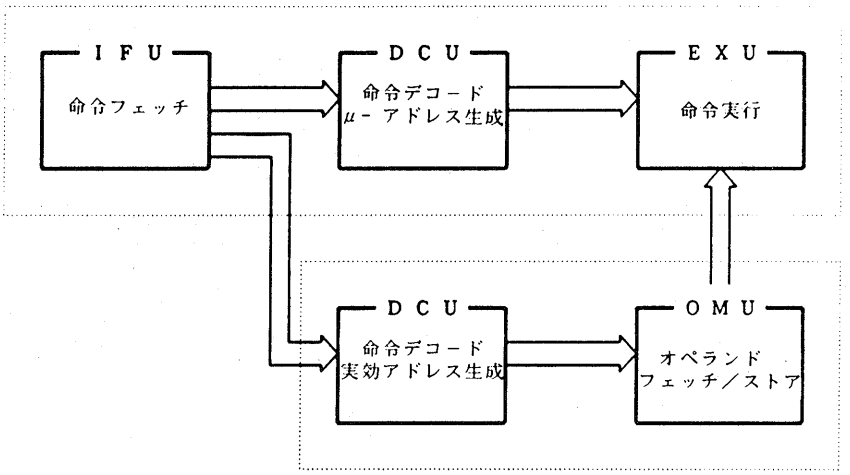

図2.　ＴＸ１のパイプライン構造

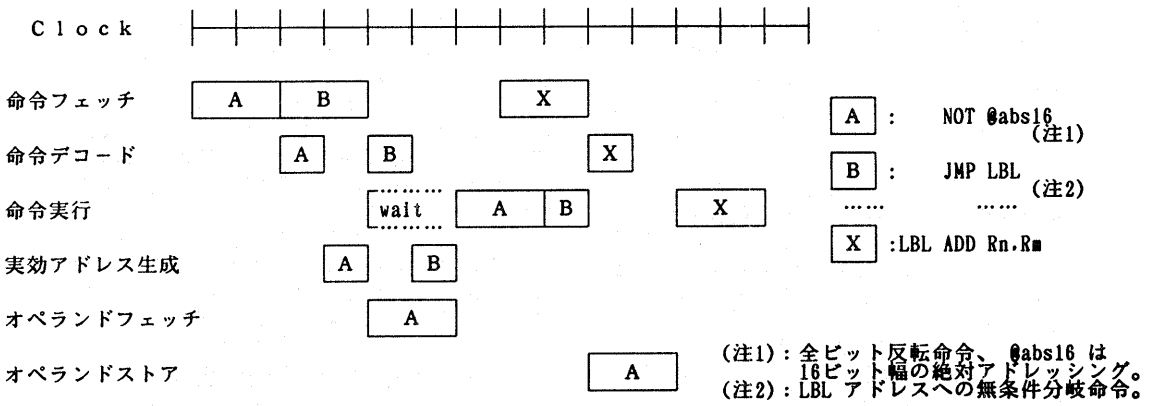

図３．　先行分岐のタイミング

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(@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の割合。（ただし“稼働率＝１－待ち発生の割合”、である。）

＜ＤＣＵの待ち発生の要因＞

☆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（ＩＢ）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の待ち発生の要因＞

☆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 ）

☆命令の実行で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（ＩＢ）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いが約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いが約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の割合い

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の待ち発生の要因では，待ちDCU/IFU の割合いが高く、待ちDCU/EXU と待ちDCU/OMU の割合いは低い。待ちDCU/IFU の割合い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 の割合いが低いことは、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いが高いためであり、実行サイクルの多い高機能命令の割合いが高くなると待ちDCU/EXU の割合いも上がることが予想される。待ちDCU/OMU の割合いが低いことは、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の待ち発生の要因のうち、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	17	46	354	18	57
（リード）	17	38	314	18	57
（ライト）	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いが高くないこと、高機能命令の割合いが比較的高く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,Fibonaci,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い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 の割合いが低いことからもわかる。ただし、SortではＯＢを４本全て使用している割合いも結構高く、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いは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い（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、実行サイクルの多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により、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，増渕美生，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 μPs: Benchmarks quantify performance”,EDN April 1, 1981, pp179-265