Vol.2014-EMB-34 No.2
2014/9/17

組込みプロセッサ用分離式BTBのFPGA実装と電力評価

福田 孝作[†1,a)]　孟 林[†1,b)]　熊木 武志[†1,c)]　小倉 武[†1,d)]

概要：近年，情報化社会の進化と共に，FPGA を用いたソフトウェアプロセッサが多く存在し，高性能な組込みプロセッサも求められる．分岐予測はプロセッサ性能向上の有効な手段である．しかし，分岐予測ミスの時に，ミスペナルティも生じる．そのため，分岐予測精度の向上が重要な課題となる．分岐予測は，分岐予測器を用いた分岐方向 (Taken，Not-Taken) の予測と，BTB (Branch Target Buffer) を用いた分岐先のアドレスの予測である．BTB 予測ミスの時に，分岐方向が正しく予測されても，継続して命令をフェッチできないため，プロセッサの性能向上に影響する．現在，我々は既存の BTB を条件分岐命令用 BTB と無条件分岐命令用 BTB に分離し，無条件分岐命令用 BTB を CAM(Content Addressable Memory) で実現する方式を提案し，SimpleScalar により性能向上を確認した．本論文では，FPGA を用いて，回路規模及び，電力評価を行い，BTB の最適化を行う．実験結果から，従来の BTB に無条件分岐用 BTB を 128 エントリ追加することにより，3.12% の性能向上が得られた．更に無条件分岐用 BTB の更新はローテーション法が有効であると分かった．

1. はじめに

近年は命令パイプライン段数を深くし，命令発行の幅を広くすることで動作周波数を向上させている．しかしこれらの傾向に伴い分岐予測ミスが増加傾向である．このため，分岐予測精度の更なる向上はプロセッサの性能向上にとって重要な課題となり，過去に BTB (Branch Target Buffer) を用いて、分岐予測の提案が行われた [1], [2], [3], [4]．その中の一つとして，条件分岐命令と無条件分岐命令の BTB を分けることにより，分岐予測精度の向上させる方式が提案されている．

また、FPGA のソフトウェアプロセッサも、組込みプロセッサとして幅広く使われる。高性能の組込みシステムが求められると同時に、組込み用プロセッサの性能向上が重要な課題となる．さらに、組込みプロセッサにおいて、BTB を用いた高速化の研究も行っている [6], [7].

本論文は提案されている分離式 BTB と既存の BTB を FPGA 上に実装し，回路規模及び，消費電力を比較することで，分離式 BTB の優位性を述べる．

†1 現在，立命館大学
Presently with Ritsumeikan University
a) ri0008xr@ed.ritsumei.ac.jp
b) menglin@fc.ritsumei.ac.jp
c) kumaki@fc.ritsumei.ac.jp
d) togura@se.ritsumei.ac.jp

2. 先行研究

2.1 ソフトウェアプロセッサ

FPGA は現在，急速的に高集積化，低価格化しており，様々な製品や環境で多く採用されている．FPGA を使えば開発費のコスト削減，開発期間の縮小に繋がる為，ASIC や ASSP，マイクロプロセッサのような既存のデバイスから置き換わりが進んでいる．ここで、ソフトウェアプロセッサとしては、xilinx 社の MicroBlaze, Altera 社の Nios が挙げられる。

表 1 FPGA と ASIC の比較

	FPGA	ASIC
プロトタイプ納期	即日	1.5〜2 週間
コスト (NRE)	0 円	100 万円から
量産期間	即時	最短 5 週間から
最低生産数量	1 個	100 個から
配置・配線	ユーザ側で可能	ベンダ側で実施
マスク管理	不要	プロジェクトごとに管理

表 1 に FPGA と ASIC の比較を示す．表から分かるように，組込み向けプロセッサを開発する上でも，FPGA は効率の良い設計が可能である．

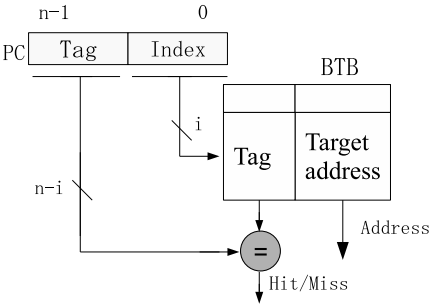

図 1 現在の BTB

2.2 BTB

分離式 BTB の比較対象として，基本形 BTB の構成を図 1 に示す．PC(Program Counter) には分岐命令のアドレスが格納されている．BTB は，キャッシュと同じ構成であり，分岐命令のアドレスの上位の一部がタグ（Tag）として保存され，分岐先のアドレスがターゲットアドレス（Target Address）として保存される．BTB を用いて，分岐先アドレスの予測と登録は以下のように行う．

- **フェッチ時の分岐先予測：**
  はじめに PC に格納されているアドレスの下位から ibit をインデックスを生成し，インデックスの指し示すエントリのタグを取り出す．次に，PC のインデックス部以外の (n-i)bit をタグとし，BTB から取り出したタグと排他的論理和によりパターンの比較を行う．パターンが一致する場合，予測成功 (Hit) となりインデックスが指し示すエントリのターゲットアドレスを分岐先のアドレスとして使用する．しかし一致しない場合は予測失敗 (Miss) となるため，正しい分岐先のアドレスが予測できず，パイプライン処理の実行ステージで分岐先のアドレスを得るまで，新たに命令をフェッチできない．
- **コミット時の BTB 更新：**
  BTB 内の履歴表を更新する時も，フェッチ時と同様に PC の下位 ibit をインデックスとして使用する．予測失敗の時に，インデックスが指し示すエントリに，タグとターゲットアドレスのペアを登録する．しかし，同じ BTB のエントリに異なる分岐命令がアクセスした場合，競合が発生する．その競合が原因で BTB の性能向上を制限している．

3. 分離式 BTB と IPC 評価

3.1 分離式 BTB の構成

先行研究 [15] により，予測の失敗は条件分岐命令と無条件分岐命令の関係が深いと判明したため，命令の種別に条件分岐命令用の BTB（CBTB：Conditional Branch Target Buffer），及び無条件分岐命令用の BTB（NBTB：NonConditional Branch Target Buffer）を持つ分離式 BTB を提案している．分離式 BTB の構成を図 2 に示す．

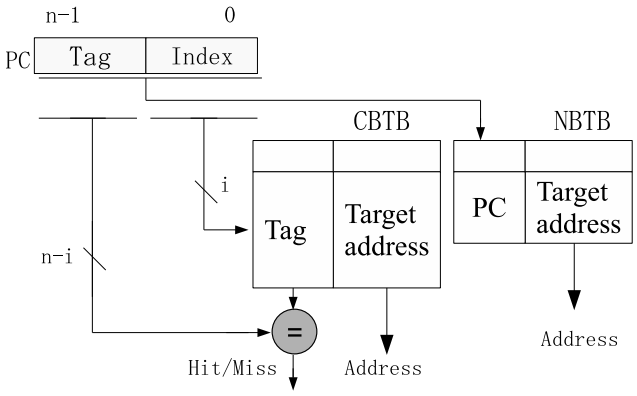

図 2 提案している分離式 BTB

CBTB は従来の BTB と同じ構成であり，ハードウェアは SRAM を用いる．NBTB は PC 内の無条件分岐命令のアドレスと，分岐先アドレスが格納される構成とし，無条件分岐命令のアドレスを格納する部分に CAM を用いる．予測するときに，CAM で分岐命令のアドレスを検索し，一致する場合は，CAM のエントリに保存された分岐先のアドレスを予測アドレスとして使用する．登録するときは，分岐命令のアドレスを用いて，CAM 内を検索して登録を行う．見つからない場合は空いているエントリに登録する．空きがない場合はローテーション法により上書きを行う．

3.2 SimpleScalar による IPC の評価

先行研究 [1] から分離式 BTB を SimpleScalar[16] を用いて，評価を行った結果を図 3 に示す．縦軸は IPC の向上率であり，上に行くほど性能が向上している．横軸はベンチマークのテスト種別である．図の Average 欄は全てのベンチマークの IPC の平均向上率である．追加された NBTB のエントリ数は 32，64，128，256，512，1024，2048 である．

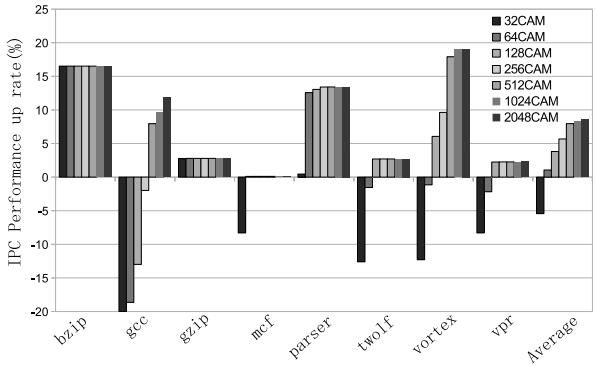

図 3 IPC による分離式 BTB の評価

図 3 から 64 エントリから IPC の向上率がプラスになり，128 エントリでは 3.12% の性能向上が得られた．以上のことからシミュレーション上では NBTB を 128 エントリ追加した分離式 BTB が最適であると確認した．

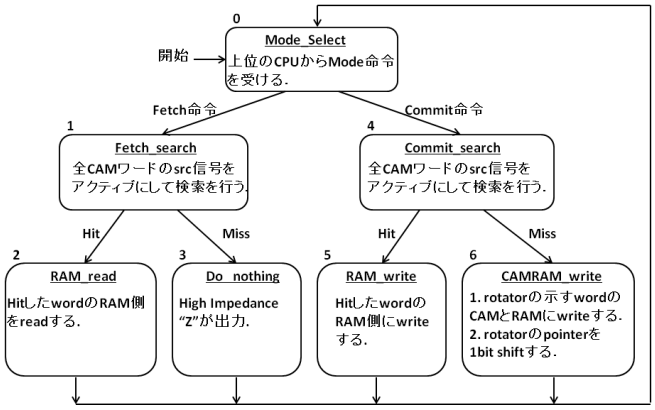

図 4　無条件命令用 BTB の動作フロー

4. FPGA 実装法

本章では分離式 BTB を FPGA へ実装する為の手順を紹介する．CBTB は RAM だけで構成される．NBTB は CAM と RAM を用いた構成であり，回路規模や消費電力を抑えるために，いくつかの工夫を行っている．アドレスを持たない．各エントリには CAM のヒットフラグとローテータによるアクセスを行う．以下で NBTB の詳細な構成を述べる．図 4 に NBTB に動作フローを示す．

NBTB の動作パターンは，何も動作を行わない **NOP(no operation) 命令**，フェッチ時に分岐先アドレスを予測する **Fetch 命令**，BTB 内の履歴表を更新する **Commit 命令**，Fetch 命令と Commit 命令を同時に行う **Fetch & Commit 命令**の 4 パターンある．各命令での動作を以下に述べる．

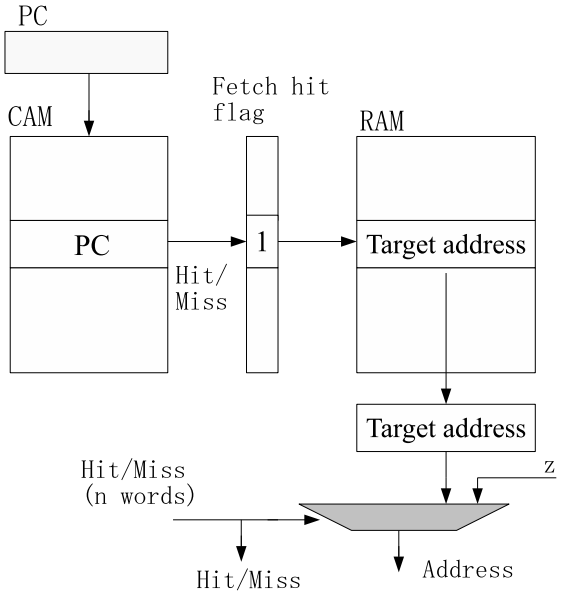

図 5　Fetch 時の動作

- **分岐命令がフェッチされた時：**
  図 5 はフェッチ命令の概略図である．フェッチ命令を受けると，CAM 内のデータと PC のデータの比較を行い，PC と同じデータが存在するかを検索する．そして，その検索結果がフェッチヒットフラグに書き込まれる．検索結果がヒットした場合，フェッチヒットフラグを使用し，CAM のヒットしたワードに対応する RAM 内の分岐先アドレスが読み出される．検索結果がミスした場合，処理としては何もされず，出力はハイ・インピーダンスとなる．

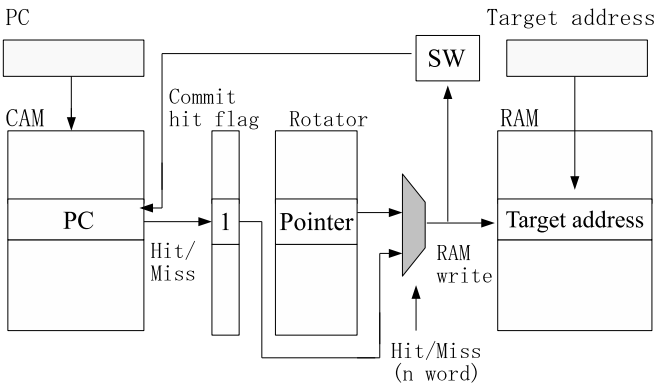

図 6　Commit 時の動作

- **分岐命令がコミットされた時：**
  図 6 はコミット命令の概略図である．コミット命令を受けると，CAM 内のデータと PC のデータの比較を行い，PC と同じデータが存在するかを検索する．そして，その検索結果がコミットヒットフラグに書き込まれる．検索結果がヒットした場合，コミットヒットフラグを使用し，CAM のヒットワードに対応する RAM のワードに分岐先のアドレスが書き込まれる．検索結果がミスの場合，ローテータを使用し，ローテータの

ポインタが示すワードに対して，新しい PC は CAM に，分岐先アドレスは RAM にそれぞれ書き込まれる．CAM と RAM に書き込んだ後に，ローテータのポインタを 1bit シフトする．

5. 実験環境と電力評価

5.1 実験環境

実験を行う環境を図 7 に示す．検証用のハードウェアは，Xilinx 製 FPGA と ATMEL 製 CPU を搭載した評価基盤を使用する．この基盤は，FPGA を評価するための各種 I/O，スイッチ，表示器，コンフィグメモリ，A/D コンバータ，PS/2 通信回路，USB コントローラを搭載している．FPGA の機能として，Spartan-6 の XC6SLX25〜XC6SLX150 まで実装することができる．CPU を評価するための各種 I/O，スイッチ，表示器，USB コントローラを搭載している．FPGA と CPU が通信する手段として，CPU のバス出力と FPGA が接続されている．FPGA の機能，容量について表 2 に示す．ハードウェア設計ツールは Xilinx ISE Design Suite 13.1 Project Navigator を使用する．消費電力の測定には，TEXIO 社製のマルチメータ DL-2060 を使用する．

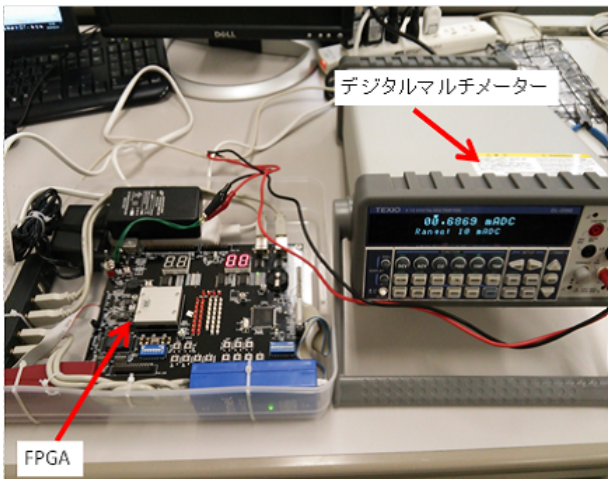

図 7 実験環境

表 2 使用する FPGA の機能

Family	Spartan6
Device	XC6SLX150
Package	FGG484
Registers	184304
LUTs	92152
Block RAM	4824
最大ユーザ I/O	576

5.2 実装結果

NBTB と CBTB を各ワードごとに表 2 の FPGA 上に実装し，使用されている回路規模を測定した．実験の結果を表 3 に示す．NBTB のメモリサイズは 32，64，128，512，1024，2048 を測定した．CBTB のメモリサイズは 2048 の 1-way と 4-way をそれぞれ測定した．回路規模を表す指標は使用されている Registers 数と LUTs 数とする．結果より NBTB はエントリ数の増やすことで，Registers，LUTs 共に回路規模が増大していることが分かる．また CBTB では 1-way と 4-way では 4 倍以上，回路規模が増大していることが分かる．

5.3 電力評価

消費電力の測定は図 7 に示すように，FPGA に直接マルチメータを接続し行った．電力の評価方法としては FPGA 上に NBTB と CBTB を実装し，その他に BTB へテストパターンを流すためのモジュールを同時に実装した．テストパターンの実装は BTB の回路規模に影響しないようにするため，FPGA 上の BlockRAM 内にテストパターンを格納するようにした．消費電力の比較方法は BlockRAM から BTB へテストパターンを流し，BTB が分岐先を予測し BTB と BlockRAM どちらも動作している**実行状態**と，BlockRAM は動作をしているが，BTB は分岐先の予測を行っていない**待機状態**を測定し，消費電力がどのように変化するかの評価を行った．表 4 に各エントリの実行状態と待機状態の電流値の差分を示す．測定したエントリは 32，64，128，256，512 である．この実験結果は各エントリ 200 ポイントずつ測定した値の平均である．

表 4 BTB の消費電力の差分

	Memory Config. (Words)	The average of the difference (A)	
		NBTB	CBTB
BTB	32	0.006346	0.002589
	64	0.013745	0.004832
	128	0.01877	0.008086
	256	0.039308	0.011982
	512	0.055326	0.021627

表 4 の結果から，CBTB の方が NBTB より差分値が全体的に低い．すなわち，CBTB の方が消費電力が少ないことが分かる．これは CBTB は RAM だけで構成されているが，NBTB には一部に CAM が使われている為，分岐先を予測する際に，ワンステップで全てのエントリにアクセスすることから，消費電力が大きくなっていると考えられる．

表 3　ハードウェア検証の結果

	Memory Config. (Words)	Circuit size		
		Registers	LUTs	LUT-FF pairs
NBTB	32	2036	1515	2369
	64	4055	3914	5472
	128	8090	6086	10474
	256	16157	12185	21917
	512	32288	24001	43791
	1024	64576	48002	87582
	2048	129152	96004	175164
CBTB(1-way)	2048	100388	170110	87939
CBTB(4-way)	2048	401552	680440	351756

5.4　総合的評価

図 8 に CBTB と NBTB の消費電力の差分を示す．横軸は BTB のサイズであり，縦軸は測定よって得られた BTB の電流値である．NBTB と CBTB 共に，128 エントリまでの消費電力の差分は 0.005(A) 程しか増加していないが，128 エントリ以降は NBTB では 0.02(A) 増加していることから，消費電力面を考慮した最適な BTB のサイズは 128 エントリであると分かる．

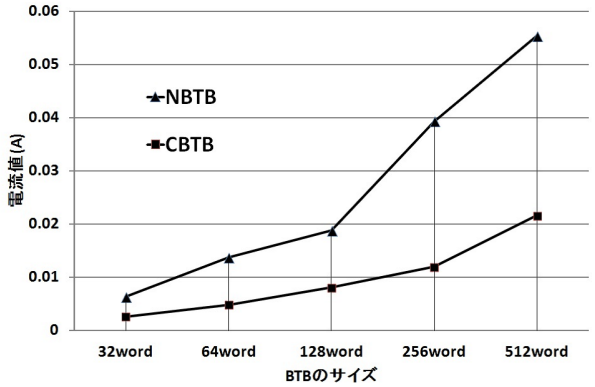

図 8　NBTB と CBTB の電力比較

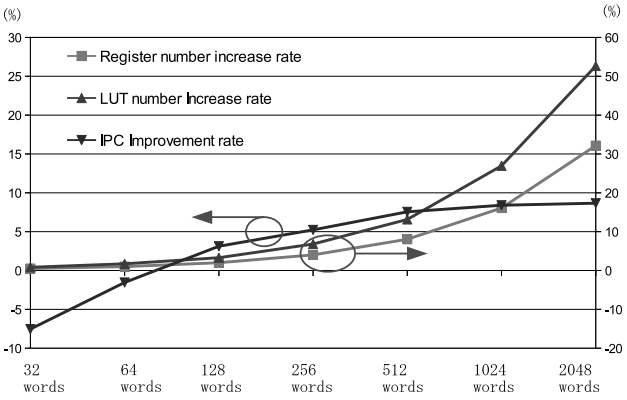

図 9　回路規模と IPC の比較

次に図 9 は各ワードで回路規模，IPC を比較したものである．横軸は BTB のサイズであり，縦軸は測定よって得られた BTB の回路規模と IPC の向上率である．回路規模はエントリ数が 128 エントリまでは緩やかに増加しているが，512 エントリ付近から急激に増大し，1024 エントリでは LUT は 14%，Register は 6% 増加し，2048 エントリでは LUT は 26%，Register は 16% 増加していることが分かる．IPC は 32 エントリ，64 エントリではマイナスであるが，128 エントリでは 3% 向上している．512 エントリ以降はエントリ数を追加しても，IPC の向上率はあまり変わらないことが分かる．

以上の結果より IPC，回路規模，消費電力を考慮した上で最も良い BTB は 128 エントリであると結論づけることができる．

6.　まとめ

本稿では，CAM を用いて，FPGA 上に従来の BTB と提案手法の分離式 BTB を実装し，IPC に加え回路規模及び，消費電力を測定し，分離式 BTB の評価を行った．その実験の結果，分離式 BTB は 128 エントリが最も良いと分かった．今後は分岐予測命令の種類やヒット率を変化させることで，どのように消費電力に影響を及ぼすかについての検証を行う予定である．さらに頻繁にミスをする分岐予測命令の種類を調べ，それに適した BTB の提案，低消費電力化について検討する必要がある．

参考文献

[1] C. H. Perleberg and A. J. Smith, “Branch Target Buffer Design and Optimization,” IEEE Transactions on Computers, Vol.42, No.4, 1993.
[2] L. Meng, K. Yamazaki and S. Oyanagi, “A Novel Branch Predictor Using Local History for Miss-Prediction Bias,” Proc. of the 2012 International Conference on Computer Design (CDES’12), pp77-83, Jul.2012.
[3] F. Saito and H. Yamana, “The Branch Predictor referring a BTB Entry Existence,” Transaction of IPSJ, Vol.45, No.7, pp.71-79, Otc.2004
[4] Brian K. Bray and M. J. Flynn, “STRATEGIES FOR BRANCH TARGET BUFFERS,” Technical Report No.

CSL-TR-91-480, 1991.
[5] S. Wang, J. Hu and S. G. Ziavras, “BTB Access Filtering: A Low Energy and High Performance Design,” Symposium on VLSI, pp81-86, 2008.
[6] Yen-Jen Chang, “An Energy-Efficient BTB Lookup Scheme for Embedded Processors”, IEEE Transactions on Circuits and system-II: EXPRESS BRIEFS, Vol.53, No.9, pp.817-821, 2006.
[7] Sunwook Kim, Eutteum Jo and Hyungshin Kim, “Low Power Branch Predictor for Embedded Processors”, 10th IEEE International Conference on Computer and Information Technology, pp.107-114, 2010.
[8] http://japan.xilinx.com/tools/microblaze.htm
[9] http://www.altera.co.jp/devices/processor/nios2/ni2-index.html
[10] T. Yamano, T. Kita, K. Murata, M. Nakanishi and T. Ogura, “A Study of the String Search System Using CAM for Network Security,”
[11] T. Ikenaga and T. Ogura,” CAM2: A Highly-parallel Two-dimensional Cellular Automaton Architecture,” IEEE Trans. Comput., Vol. 47, No. 7, 1998.
[12] Y. Ishikawa, J. Uchida, Y. Miyaoka, N. Togawa, M. Yanagisawa and T. Ohtsuki, “A CAM Processor Optimizing Method with Area Constraints,” IEICE Technical Report, Vol.103, No.705,pp.13-18, 2004.
[13] T. Kumaki, K. Iwai and T. Kurokawa, “A Flexible Multi-Port Content Addressable Memory,” IEICE Transactions on Information and Systems, Vol.J87-D-1, No.1, pp.12-21, 2004. (in Japanese)
[14] C.C.Kavar and S.S.Paramar, “Performance Analysis of LRU Page Replacement Algorithm with Reference to different Data Structure”, International Journal of Engineering Research and Application, Vol.3, No.1, pp.2070-2076, 2013.
[15] K.Fududa, L.Meng, T,Kumaki and T.Ogura, “A CAM-based Separated BTB for a Superscalar Processor,” 2013 First International Symposium on Computing and Networking, pp385-388,2013.
[16] D. Burger and T.M. Austin, “The SimpleScalar Tool Set Version2.0,” Technical Report, University of Wisconsin-Madison Computer Sciences Dept. July 1997.
[17] A. Hilton and A. Roth, “Ginger : Control independence using tag rewriting ,” Proc. 35th Int’ l Symposium on Computer Architecture, May 2007, 436-447, 2007
[18] Dong Ye and David Kaeli, “A reliable return address stack: microarchitectural features to defeat stack smashing”, ACM SIGARCH Computer Architecture News-Special issue: Workshop on architectural support for security and anti-virus (WASSA), Vol.33, No.1, pp73-80, 2005.