International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 6, No. 3 (May-2017)

http://dx.doi.org/10.12785/ijcds/060302

Generalized Worst Case Estimation of Misprediction Counts for Dynamic Branch Predictors

Marwa A. Elmenyawi[1], **Cherif Salama**[2], **Mostafa E. A. Ibrahim**[1] **and I. M. Hafez**[2]

[1]*Benha Faculty of Engineering, Benha University, Benha, Egypt*
[2]*Faculty of Engineering, Ain Shams University, Cairo, Egypt*

Received 21 Feb. 2017, Revised 18 Mar. 2017, Accepted 20 Apr. 2017, Published 1 May 2017

Abstract: Estimating the number of mispredictions is critically important for estimating the Worst-Case Execution Time for real-time systems. This paper generalizes and improves over previous attempts to provide a safe and tight mispredication count estimate for dynamic branch predictors. The paper gives closed formulas to compute mispredictions in case of simple and nested loops applicable to all variations of two-level adaptive branch predictors in addition to the gshare and gselect predictors. The given formulas are general enough to accommodate predictors with any counter size.

Keywords: Real-time system, WCET, Branch prediction.

1. Introduction

Embedded real-time software guarantees the commitment to system time restrictions [1], [2]. This is particularly critical for hard real-time systems, in which the lack of commitment to time restriction is a direct cause of system failure [3], [4].

The Worst Case Execution Time (WCET) estimation definitely differs from real execution time. Thus, the WCET is either an under or an over estimation [5], [6]. For hard real-time systems the WCET underestimation might lead to system failure. However, the overestimation of WCET significantly increases the software development cost [3]. WCET estimation techniques typically presents a trade-off process between accuracy and complexity.

To make an accurate WCET estimation, the architectural features such as; instruction/data caching, out-of-order execution, pipelining, fine-grained chip multi-threading, and dynamic branch prediction have to be taken into account. In this paper, we focus on dynamic branch predictors which are essentials in modern processors with high clock frequencies and deep pipelines. Li et al. [7] showed that if branch prediction is not modeled, WCET can be overestimated by up to 70% for some benchmarks. One of the major factors that influences the overestimation is how the analysis for the estimation is carried on [8].

The first step in modeling branch prediction for WCET estimation is to estimate the number of mispredictions of the used predictor. This paper presents a novel approach to limit the WCET overestimation by giving an upper bound to the number of branch mispredictions of loop structures. Moreover, the proposed approach generally and accurately formulates the estimation of branch mispredictions for any two-level adaptive predictor. Finally, the given formulation is applicable for l-bit saturating counters.

The existing works on branch prediction for WCET estimation are reviewed in Section 2. In Section 3, we describe our approach and in Section 4, the results of applying it on benchmarks are presented. Finally, Section 5 concludes the paper.

2. Related Work

Branch prediction is a technique that supports speculative execution to improve the pipeline performance. Branch prediction schemes can be divided into static and dynamic schemes. In static schemes, the branch is always predicted to the same direction while dynamic schemes predict a branch depending on its execution history [7].

Here, we examine all two-level adaptive branch prediction schemes in addition to the popular gshare and gselect predictors. There are nine variations of two-level adaptive branch prediction schemes. Each variation is distinguished by the way the branch history information is maintained in the first level (G for global, P for per-

E-mail address: marwa.elmenyawi@bhit.bu.edu.eg, cherif.salama@eng.asu.edu.eg, mustafa.ibrahim@bhit.bu.edu.eg, ismail_hafez@eng.asu.edu.eg

address, or S for set) and also by the way of association between the second level and the first level. The history information can be maintained in a single global branch history register (BHR), in separate per-address registers where each address is a branch instruction, or in separate per-set registers. Moreover, the global pattern history table (PHT) may be a single table that contains the second-level history information, or multiple tables where each branch instruction identified by its address has its own second-level pattern table [9]. The PHT contains *l*-bit counters which represent the value according to which the branch is predicted. Experiments showed that using more than 2-bits gives a minimal improvement at the expense of increased storage [10]. However, the storage cost has significantly dropped due to technological advances. As a result, predictors using counters of 3 or even more bits are now affordable. A 3-bits saturating counter is used in Alpha 21264 Microprocessor [11]. Fig. 1 illustrates the state diagram of a 3-bits counter while Table I shows the states of a 4-bits counter. The state names reflect the direction of the prediction (N for Not taken, and T for Taken) and the degree of confidence (S for Strong and W for Weak). For example, the state SN indicates a strong confidence that the branch is not going to taken. For the

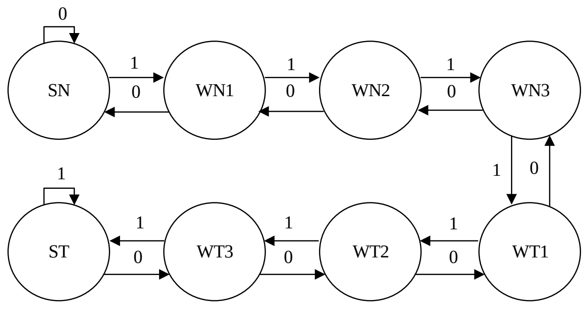

Figure 1. The State diagram for 3-bits counter.

gselect and gshare [12] predictors, the PHT is indexed by a combination of the BHR bits and some lower order bits of the branch address. To combine them, gselect uses simple concatenation, while gshare uses a bitwise exclusive OR operation.

Shaw [13] presented the original work on timing schema. He considered only high-level languages constructs and did not take in his account the micro-architectural features like branch prediction or instruction/data cache as these features were not used or known at this time.

Colin and Puaut [14] presented the first work on worst case execution time analysis using dynamic branch prediction techniques. They computed the abstract state of the BHT at the prediction time of branch by using static simulation. They only considered local branch predictor techniques.

TABLE I. Four-bit prediction scheme.

States	Bits	Direction
SN	0000	N
WN1	0001	N
WN2	0010	N
WN3	0011	N
WN4	0100	N
WN5	0101	N
WN6	0110	N
WN7	0111	N
WT1	1000	T
WT2	1001	T
WT3	1010	T
WT4	1011	T
WT5	1100	T
WT6	1101	T
WT7	1110	T
ST	1111	T

A similar approach to Colin and Puaut [14] was presented by Bate and Reutemann [15], [16] but they considered global branch predictor techniques. They classified branches according to their semantic contexts. In [15] they computed the maximum number of mispredictions for only bimodal and global branch predictor, moreover they used only 2-bit counters.

Mitra, Roychoudhury, and Li [7], [17], [18] used integer linear programming (ILP) to compute the WCET for global branch predictors. They derived a set of constraints from control flow graph (CFG) and solved them using an ILP solver. They only considered a branch history pattern of two and four bits.

The approach by Mitra, Roychoudhury, and Li was extended in [19]. Burguière and Rochange modeled the complexity of different branch predictors. Maïza and Rochange [20] also computed WCET of branch predictors using an ILP. Additionally, they modeled the complexity and demonstrated that there exists a relation between the branch history length and the complexity of the ILP problem.

Puffitsch [21] used a persistence notion to model the WCET of local branch predictors. He extended his work in [22] by applying the persistence notion to global branch predictors.

3. Methodology

This section introduces a method that accurately calculates the number of mispredictions of dynamic branch predictors for loop structures in the worst case. The novelty of the proposed method comes from supporting all variations of two-level adaptive branch predictors in addition to the gshare and gselect predictors and also for supporting *l*-bit counters.

Here, we discuss the branch predictor behavior for nested loops where the inner loop iterates n time while the outer iterates m times. Such a nested loop implies that the branch of the inner loop is executed several times with the following outcome: $(T^{n-1}N)^m$. We propose to classify all studied branch predictors into two categories based on whether the PHT is indexed using the **branch history pattern** or using the **branch address**. Branch prediction techniques that depend basically on the branch history pattern for indexing the PHT are: GAg, GAp, GAs, gshare, and gselect. Branch prediction techniques that depend basically on the branch address for indexing the PHT are: PAg, PAp, PAs, SAg, SAp, and SAs.

Using the branch address leads to accessing the same location in the PHT for all iterations. The situation is different when the branch history pattern is used since the BHR is shifted every time the branch is predicted implying a potentially different pattern in each iteration. We assume the following for simplicity:

1) The upper bound of the number of loop iterations is known.
2) No interference between branch instructions; i.e., the same entry in the BHT is not shared by two branches.
3) The loop does not contain any conditional branches that may end or change the loop index value.

As our aim is to derive a general form for the maximum number of mispredictions for branch predictors using l-bit counters, we first estimate the number of mispredictions for the two branch prediction categories using 3- and 4-bit counters as illustrated in subsections 3-A and 3-B. From the two estimates, we are able to generalize a form for l-bit counters.

A. Branch Address Based Category

a) Simple Loop

First, we compute the maximum number of mispredictions for simple loops having branch outcomes following the pattern $T^{n-1}N$ using 3- and 4-bit counters. Tables II and III list the number of mispredictions for 3-bit and 4-bit counters respectively depending on the number of loops iterations n and depending on the initial state of the saturating counter stored in the PHT at the location indexed by the branch. Each entry in the PHT table is decremented or incremented depending on prediction direction; taken or not-taken.

For initial states ST to WT1 when 3-bit counters are used, the maximum number of mispredictions is always 1 since branch outcome is always taken except for the last iteration when the loop is exited. Only the last iteration will only be mispredicted. The number of mispredictions for not-taken initial states, WN3 to SN, can be broken down as follows;

- For $n = 1$, there are no mispredictions since the branch outcome is not taken.
- For $n = 2$, the number of mispredictions for the states WN2 to SN is 1 since these states remain not taken when incremented which results in a correct prediction in the second iteration when the loop is actually exited. Thus, the first iteration is the only one that gets mispredicted. The number of mispredictions for state WN3 is 2 as this state is not-taken leading to a misprediction in the first iteration and then it is incremented to WT1 which leads to another mispredition in the second iteration when the loop is exited.
- For $n = 3$, the number of mispredictions for the initial state WN3 is 2 as it is mispredicted for the first and last iterations as follows: The outcome of the branch in the first iteration is taken while the state WN3 is not-taken after that the state is incremented to the taken state WT1 which is consistent with the outcome of the branch in the second iteration then, WT1 is incremented in the third iteration to the taken state WT2 while the third iteration's outcome is not-taken. However, the initial state WN2 has 3 mispredictions as it needs two increments to reach a taken state and when it reaches one, the actual outcome becomes not-taken. Finally, the number of misprediction for initial states WN1 and SN are 2 as these states are incremented three times to reach a not-taken state and as such only the first two iterations are mispredicted.
- For $n = 4$, the worst case occurs for the initial state WN1. When this state is incremented 4 times, it remains a not-taken state for the first 3 times and become a taken state only for the last loop iteration while the branch outcome is not-taken. This scenario leads to 4 mispredictions.
- For $n \geqslant 5$, the worst case is 5 mispredictions which happens for initial state SN. In such case, the state is incremented 4 times until it reaches a taken state for the last iteration. This scenario leads to a misprediction in the first 4 iterations and in the last iteration.

The maximum number of mispredictions for 3-bit counters in case of a simple loop, $mp_{loop}(n, 3)$, can be given by:

$$mp_{loop}(n,3) = \begin{cases} n, & if\ n < 5. \\ 5, & otherwise \end{cases}$$

TABLE II. Number of branch mispredictions for branch address based category using 3-bit counters.

n	ST	WT3	WT2	WT1	WN3	WN2	WN1	SN	Max
1	1	1	1	1	0	0	0	0	1
2	1	1	1	1	2	1	1	1	2
3	1	1	1	1	2	3	2	2	3
4	1	1	1	1	2	3	4	3	4
$\geqslant 5$	1	1	1	1	2	3	4	5	5

TABLE III. Number of branch mispredictions for branch address based category using 4-bit counters.

n	ST	WT7	WT6	WT5	WT4	WT3	WT2	WT1	WN7	WN6	WN5	WN4	WN3	WN2	WN1	SN	Max
1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1
2	1	1	1	1	1	1	1	1	2	1	1	1	1	1	1	1	2
3	1	1	1	1	1	1	1	1	2	3	2	2	2	2	2	2	3
4	1	1	1	1	1	1	1	1	2	3	4	3	3	3	3	3	4
5	1	1	1	1	1	1	1	1	2	3	4	5	4	4	4	4	5
6	1	1	1	1	1	1	1	1	2	3	4	5	6	5	5	5	6
7	1	1	1	1	1	1	1	1	2	3	4	5	6	7	6	6	7
8	1	1	1	1	1	1	1	1	2	3	4	5	6	7	8	7	8
$\geqslant 9$	1	1	1	1	1	1	1	1	2	3	4	5	6	7	8	9	9

A similar analysis to the one conducted for 3-bit predictors can be made for 4-bit predictors. From Table III, it can be concluded that the maximum number of mispredictions using 4-bit counters for simple loop, $mp_{loop}(n, 4)$, can be given by:

$$mp_{loop}(n,4) = \begin{cases} n, & if\ n < 9. \\ 9, & otherwise \end{cases}$$

In general, the maximum number of mispredictions for branch address based category using l-bit counters for simple loop, $mp_{loop}(n, l)$, can be given by:

$$mp_{loop}(n,l) = \begin{cases} n, & if\ n < 2^{l-1} + 1 \\ 2^{l-1} + 1, & otherwise \end{cases}$$

where l is the number of bits of the saturating counter.

Alternatively, the maximum number of mispredictions can be given by:
$mp_{loop}(n, l) = min(n,\ 2^{l-1} + 1)$

It can be expected that the maximum number of mispredictions for nested loops would be $m \times mp_{loop}(n)$ but we will show next that this value would be an overestimate.

b) Nested Loop

Second, we derive a general formula for the maximum number of mispredictions for nested loops using l-bit counters by estimating the number of mispredictions for 3- and 4-bit counters. Tables IV and V show the state transitions causing the highest number of mispredictions for 3-bit and 4-bit predictors respectively. The state in the inner loop is incremented according to the loop pattern $(T^{n-1}N)^m$ while it is decremented when transitioning to the next iteration of the outer loop. The number of mispredictions for each n inner loop are as follows;

- For $n = 1$, the maximum number of mispredictions using 3- and 4-bit counters are 4 and 8 respectively in the worst case initial state ST. Here, the loop iterates for one time with a not taken outcome and taken prediction. The loop reaches the correct prediction after 4 iterations of the outer loop in case of a 3-bit predictor since the first not taken state WN3 (011) can be reached from ST (111) after 4 decrements. Similarly, the 4-bit counter needs to be decremented 8 times to reach the first not-taken state WN7 (0111) from the 4-bit ST (1111). From that it can be concluded that for $n = 1$ the maximum number of mispredictions does not exceed 2^{l-1} nor the number of outer loop iterations m. As such it can be given by $\min(m, 2^{l-1})$.
- For $n = 2$, the two iterations are always mispredicted if the WN3 or WN7 are the initial states for 3- and 4-bit predictors respectively as the increment lead the state to be a taken state which should be not-taken. It is obvious that in this case, the number of mispredictions for l-bit counters is $2 \times m$.
- For $n = 3$, the inner loop branch counter is incremented twice and decremented once in each outer loop iteration resulting in an effective increment by one. So starting from a 3-bit SN (000) state, we need 4 outer loop iterations to be incremented 4 times and reach the WT1 (100) state to start

TABLE IV. The execution for branch address based category using 3-bit counter for nested loop .

	Outer Loop Iteration Number				
n	1	2	3	4	5
1	ST	WT3	WT2	WT1	WN3
2	WN3 ↦ WT1	WN3 ↦ WT1	WN3 ↦ WT1	WN3 ↦ WT1	WN3 ↦ WT1
3	SN ↦ WN1 ↦ WN2	WN1 ↦ WN2 ↦ WN3	WN2 ↦ WN3 ↦ WT1	WN3 ↦ WT1↦ WT2	WT1 ↦ WT2 ↦ WT3
4	SN ↦ WN1 ↦ WN2 ↦ WN3	WN2 ↦ WN3 ↦ WT1 ↦ WT2	WT1 ↦ WT2 ↦ WT3 ↦ ST	WT3 ↦ ST ↦ ST ↦ ST	ST ↦ ST ↦ ST ↦ ST

TABLE V. The execution for branch address based category using 4-bit counter for nested loop .

	Outer Loop Iteration Number								
n	1	2	3	4	5	6	7	8	9
1	ST	WT7	WT6	WT5	WT4	WT3	WT2	WT1	WN7
2	WN7 ↦ WT1	WN7 ↦ WT1	WN7 ↦ WT1	WN7 ↦ WT1	WN7 ↦ WT1	WN7 ↦ WT1	WN7 ↦ WT1	WN7 ↦ WT1	WN7 ↦ WT1
3	SN ↦ WN1 ↦ WN2	WN1 ↦ WN2 ↦ WN3	WN2 ↦ WN3 ↦ WN4	WN3 ↦ WN4 ↦ WN5	WN4 ↦ WN5 ↦ WN6	WN5 ↦ WN6 ↦ WN7	WN6 ↦ WN7 ↦ WT1	WN7 ↦ WT1 ↦ WT2	WT1 ↦ WT2 ↦ WT3
4	SN ↦ WN1 ↦ WN2 ↦ WN3	WN2 ↦ WN3 ↦ WN4 ↦ WN5	WN4 ↦ WN5 ↦ WN6 ↦ WN7	WN6 ↦ WN7 ↦ WT1 ↦ WT2	WT1 ↦ WT2 ↦ WT3 ↦ WT4	WT3 ↦ WT4 ↦ WT5 ↦ WT6	WT5 ↦ WT6 ↦ WT7 ↦ ST	WT7 ↦ ST ↦ ST ↦ ST	WT7 ↦ ST ↦ ST ↦ ST

providing a taken prediction. Starting from the fifth outer iteration, we get only one misprediction per iteration. The number of mispredictions in the first 4 loop iterations would be 2, 2, 3, and 2. In total for m outer loop iterations, the total number of 3 + 2 + 2 + 2 + m - 4 = m + 5. Similarly, in the case of the 4-bit counter, the number of mispredictions can be shown to be m + 9. In general, it can be shown that for l-bit counters, the number of mispredictions is $m + 2^{l-1} + 1$.

- For $n = 4$, the loop needs two outer iterations in case of a 3-bit counter to reach a loop with a taken entry state since the entry state is incremented twice in each outer loop iteration. From the third outer iteration, the only mispredicted state is the exit state. The number of mispredictions are 3 + 3 + m - 2 = m + 4. Similarly, the number of mispredictions in case of a 4-bit counter is equal to m + 8 which leads to a total number of mispredictions for l-bit counters equal to $m + 2^{l-1}$.

It can be seen that for $n \geqslant 3$, the number of outer loop iterations needed to reach correct predictions can be given by the formula: $\frac{2^{l-1}}{n-2}$ and that the number of mispredictions before that will not exceed n for each outer loop iteration. In addition, it can be seen that the formula that applies on n = 4 is applicable to $n > 4$ as well.

The maximum number of mispredictions for the branch address category of predictors using l-bit counters can be summarized by :

$$mp_{loop}(n, m, l) = \begin{cases} min(m, 2^{l-1}), & if\ n = 1 \\ 2 \times m, & if\ n = 2 \\ m + 2^{l-1} + 1, & if\ n = 3 \\ m + 2^{l-1}, & if\ n \geqslant 4 \end{cases}$$

B. Branch Pattern Based Category

This section continues the analysis of the maximum number of mispredictions for nested loops when l-bit predictors from the branch history pattern category are used.

As in the previous cases, the outer loop iterates m times and inner loop iterates n times with a k-bit history pattern and a l-bit counter. Predictors from this category work as follows at each branch prediction the history pattern is shifted left by one or zero and also the counter is incremented or decremented by one according to branch direction.

It is important to note that for this category of predictors when the same branch instruction is encountered, the branch history does not necessarily point to the same location in the PHT table since the branch history pattern is shifted to the left each time. However, since we are assuming a k-bit history pattern and a regular branch direction pattern of $(T^{n-1}N)^m$, the history pattern will stabilize to a repeating sequence of patterns after the first k-1 iterations provided that k is greater than n. This behavior is illustrated in Tables VI and VII.

Table VI provides a worst-case execution trace when n = 3, k = 7, and l = 3. To get the worst case, we assume an initial history pattern of 0000000 and an initial value of 000 (SN) for the 3-bit predictors corresponding to taken branches and an initial value of 111 (ST) corresponding to not-taken branches. The pattern needs 6 times to reach the first repeated pattern of 0110110. Thus, these 6 non-repeated patterns are mispredicted so the number of mispredictions in these non-repeated patterns is equal to k-1. In Table VI example, 3 different patterns are repeated and each pattern needs to be encountered 4 times to reach the correct prediction. As a result, we get an additional 12 mispredictions. In general, the number of repeated patterns is equal to n and the number of times each each of these patterns is mispredicted is equal to 2^{l-1}. The total number of mispredictions is therefore $n \times 2^{l-1}$. Each pattern needs 4 times to reach the correct state so the total number of mispredictions are 12 which equal $4 \times n = 2^{l-1} \times n$ as each taken state needs 2^{l-1} to reach the not-taken and vice versa.

TABLE VI. The execution for branch pattern based category using 3-bit counter for nested loop with n=3 and k=7.

Outer iter #	Pattern	Direction	Counter	Prediction
1	0000000	T	000	N
	0000001	T	000	N
	0000011	N	111	T
2	0000110	T	000	N
	0001101	T	000	N
	0011011	N	111	T
3	0110110	T	000	N
	1101101	T	000	N
	1011011	N	111	T
4	0110110	T	001	N
	1101101	T	001	N
	1011011	N	110	T
5	0110110	T	010	N
	1101101	T	010	N
	1011011	N	101	T
6	0110110	T	011	N
	1101101	T	011	N
	1011011	N	100	T
7	0110110	T	100	T
	1101101	T	100	T
	1011011	T	011	N
8	0110110	T	101	T
	1101101	T	101	T
	1011011	N	010	N

TABLE VII. The execution for branch pattern based category using 3-bit counter for nested loop with n=4 and k=2 .

Outer iter #	Pattern	Direction	Counter	Prediction
1	00	T	000	N
	01	T	000	N
	11	T	011	N
	11	N	100	T
2	10	T	000	N
	01	T	001	N
	11	T	011	N
	11	N	100	T
3	10	T	001	N
	01	T	010	N
	11	T	011	N
	11	N	100	T
4	10	T	010	N
	01	T	011	N
	11	N	011	N
	11	T	100	T
5	10	T	011	N
	01	T	100	T
	11	T	011	N
	11	N	100	T
6	10	T	100	T
	01	T	101	T
	11	T	011	N
	11	N	100	T

Table VII provides a worst-case execution trace when $n = 4$, $k = 2$, and $l = 3$. Again, to get the worst case, we assume that an initial history pattern of 00 and an initial value of 000 (SN) for the 3-bit predictors corresponding to taken branches and an initial value of 111 (ST) corresponding to not-taken branches. The pattern needs one time to reach the first repeated pattern 01 so this one time is mispredicted again resulting in $k-1$ mispredictions. Since n is greater than k in this example, the number of repeated history patterns is not equal to n. In fact, the only patterns that may appear and get repeated are the ones that have a single 0 corresponding to a single not-taken branch at the end of each execution of the inner loop. As such, there are k patterns with a single 0 and a single pattern (all ones) that might get repeated. Each of the k patterns with a single zero needs to be encountered 2^{l-1} in order to reach the correct prediction leading to $2^{l-1} \times k$ misprediction. For the all-ones pattern, the same rules used in subsection 3-A can be applied since the predictor associated with this pattern is always decremented once and incremented $n-k-1$ times.

We can summarize that the history patterns can divided into three groups; non-repeated, repeated with a single zero, and repeated without zeros where in each group the number of mispredictions can be given by :

- Non-repeated
  $mp_{loop\_n\_rp}(m,n,k,l)= k- 1$
- For repeated patterns with a single zero:

$$mp_{loop\_rp\_z}(m,n,k,l) = \begin{cases} 2^{l-1} \times n, & if\ n \leqslant k \\ 2^{l-1} \times k, & if\ n > k \end{cases}$$

- For repeated patterns without zeros:

$$mp_{loop\_rp\_nz}(m,n,k,l) = \begin{cases} 0, & if\ n \leqslant k \\ 2^{l-1}, & if\ n = k+1 \\ 2 \times m, & if\ n = k+2 \\ m + 2^{l-1} + 1, & if\ n = k+3 \\ m + 2^{l-1}, & n \geqslant k+4 \end{cases}$$

The following formula gives the final misprediction count as the sum of the three preceding mispredictions:

$$mp_{loop}(m,n,,k,l) = \begin{cases} 2^{l-1} \times n + k - 1, & if\ n \leqslant k \\ 2^{l-1} \times k + k + 2^{l-1} - 1, & n = k+1 \\ 2^{l-1} \times k + k + 2m - 1, & n = k+2 \\ 2^{l-1} \times k + k + m + 2^{l-1}, & n = k+3 \\ 2^{l-1} \times k + k + m + 2^{l-1} - 1, & n \geqslant k+4 \end{cases}$$

4. Results

In this section, we evaluate the proposed method and formulas using the SimpleScalar simulator [23]. We extended the simulator to support different number of saturating counter bits. Since our focus is counting the number of mispredictions thus we configured the SimpleScalar simulator to zero data/instruction cache misses. We used the following three branch predictors: a GAg predictor with a single 4-bit BHR indexing a 16-entry PHT, a gshare predictor with a single 4-bit BHR XORed with the 4-least significant bits of the branch address indexing a 16-entry PHT, and a bimodal predictor with a 16-entry BHT. Figures 2 and 3 show the results for the upper

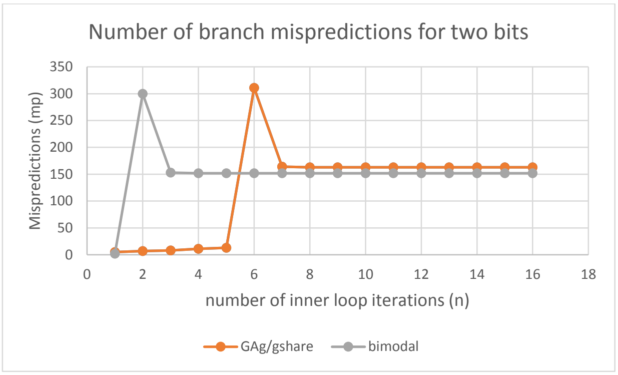

Figure 2. The maximum number of branch mispredictions for two bits counter.

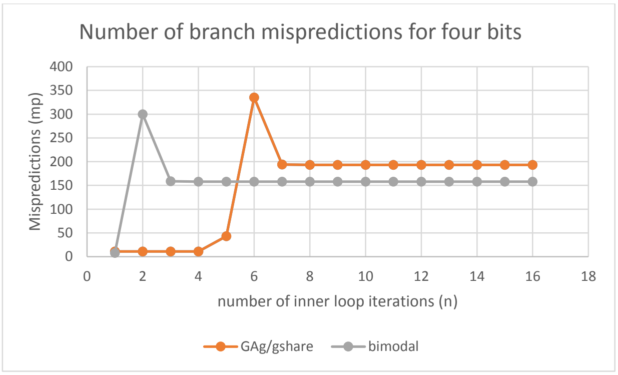

Figure 3. The maximum number of branch mispredictions for four bits counter.

bound of number of mispredictions for the GAg, gshare, and bimodal predictors two and four bits saturating counters. Note that the Gshare and GAg predictors produced identical results and were merged for that reason. These results are obtained from SimpleScalar for a C program of the following form:

```
for (int a=0; a < 150; a++)
   {
    for (int b=0; b < n; b++)
     {
       // do something
     }
    }
```

Figures 2 and 3 show that the inner loop with two iterations give the maximum number of branch mispredictions for the bimodal predictor. Moreover, the number of mispredictions for the bimodal predictor remain constant after four inner iterations as discussed in section 3. For GAg and gshare predictors, the maximum number of branch mispredictions occurs when the difference between the number of inner iterations and the number of history pattern bits is two. Similarly, the number of branch mispredictions remains constant when the difference is greater or equal to four. All the above observations are consistent with the equations we derived in Section 3.

5. Conclusion

In this paper, we provide general and accurate formulas for the estimation of mispredictions for two-level adaptive branch predictors. The generality of our formulation comes from that it can be applied to any two-level branch predictor and to all sizes of saturating counters. Previous work for estimating the number of mispredictions can only be used for specific predictors and only when the saturating counter size does not exceed 2 bits. The provided formulas are both proven and evaluated empirically using SimpleScalar. As such the formulas are shown to provide accurate estimates for the number of mispredictions and as such they can be used towards computing a tight worst-case estimation of the execution time of realtime systems.

References

[1] H. Abdelhamid, B. Mostefa, and C. Abdallah, "Embedded systems design using event-b theories," *International Journal of Computing and Digital Systems*, vol. 5, no. 2, pp. 173–187, March 2016.

[2] T. Perera and J. Collins, "Novel embedded system based species recognition system for pest control," *International Journal of Computing and Digital Systems*, vol. 5, no. 5, pp. 387–393, September 2016.

[3] V. P. Kozyrev, "Estimation of the execution time in real-time systems," *Programming and Computer Software*, vol. 42, no. 1, pp. 41–48, 2016. [Online]. Available: http://dx.doi.org/10.1134/S0361768816010059

[4] Y. Ismail, "A fast diamond motion estimation search algorithm for real time video applications," *International Journal of Computing and Digital Systems*, vol. 3, no. 2, pp. 101–110, May 2014.

[5] R. Wilhelm, J. Engblom, A. Ermedahl, N. Holsti, S. Thesing, D. Whalley, G. Bernat, C. Ferdinand, R. Heckmann, T. Mitra, F. Mueller, I. Puaut, P. Puschner, J. Staschulat, and P. Stenström, "The worst-case execution-time problem&mdash;overview of methods and survey of tools," *ACM Trans. Embed. Comput. Syst.*, vol. 7, no. 3, pp. 36:1–36:53, May 2008. [Online]. Available: http://doi.acm.org/10.1145/1347375.1347389

[6] M. Sharma, H. Elmiligi, and F. Gebali, "Performance evaluation of real-time systems," *International Journal of Computing and Digital Systems*, vol. 4, no. 1, pp. 43–52, January 2015.

[7] A. R. Xianfeng Li, Tulika Mitra, "Modeling control speculation for timing analysis," *The International Journal of Time-Critical Computing Systems*, vol. 29, no. 1, pp. 27–58, January 2005.

[8] H. Cassé, H. Ozaktas, and C. Rochange, "A Framework to Quantify the Overestimations of Static WCET Analysis," in *15th International Workshop on Worst-Case Execution Time Analysis (WCET 2015)*, ser. OpenAccess Series in Informatics (OASIcs), F. J. Cazorla, Ed., vol. 47. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2015, pp. 1–10.

[9] T.-Y. Yeh and Y. N. Patt, "A comparison of dynamic branch predictors that use two levels of branch history," in *Proceedings of the 20th Annual International Symposium on Computer Architecture*, ser. ISCA '93. New York, NY, USA: ACM, 1993, pp. 257–266.

[10] J.-L. Baer, *Microprocessor Architecture: From Simple Pipelines to Chip Multiprocessors*, 1st ed. New York, NY, USA: Cambridge University Press, 2009.

[11] R. E. Kessler, "The alpha 21264 microprocessor," *IEEE Micro*, vol. 19, no. 2, pp. 24–36, Mar. 1999. [Online]. Available: http://dx.doi.org/10.1109/40.755465

[12] S. McFarling, "Combining branch predictors," WRL Technical Notes TN-36, DigitalWestern Research Laboratory, Tech. Rep., 1993.

[13] A. Shaw, "Reasoning about time in higher-level language software," *IEEE Transactions on Software Engineering*, vol. 15, no. undefined, pp. 875–889, 1989.

[14] A. Colin and I. Puaut, "Worst case execution time analysis for a processor with branch prediction," *Real-Time Syst.*, vol. 18, no. 2/3, pp. 249–274, May 2000. [Online]. Available: http://dx.doi.org/10.1023/A:1008149332687

[15] I. Bate and R. Reutemann, "Worst-case execution time analysis for dynamic branch predictors," in *Proceedings of the 16th Euromicro Conference on Real-Time Systems*, IEEE Computer Society. Catania, Italy: IEEE, July 2004, pp. 215–222.

[16] R. D. Reutemann, "Worst-case execution time analysis for dynamic branch predictors," Ph.D. dissertation, University of York, 2008.

[17] T. Mitra, A. Roychoudhury, and X. Li, "Timing analysis of embedded software for speculative processors," in *Proceedings of the 15th International Symposium on System Synthesis*, ser. ISSS '02. New York, NY, USA: ACM, 2002, pp. 126–131.

[18] T. Mitra and A. Roychoudhury, "A framework to model branch prediction for wcet analysis," June 2002, short version of: A Framework to Model Branch Prediction for WCET Analysis, Tulika Mitra, Abhik Roychoudhury, 2nd Workshop on Worst Case Execution Time Analysis (WCET), Austria, June 2002. Also available as NUS Technical Report 11-01. One of Author's homepage: http://www.comp.nus.edu.sg/ abhik/. [Online]. Available: http://www.comp.nus.edu.sg/~abhik/pdf/wcet02.pdf

[19] C. Burguiere and C. Rochange, "On the Complexity of Modelling Dynamic Branch Predictors when Computing Worst-Case Execution Times," in *ERCIM/DECOS Workshop on Dependable Embedded Systems, Lübeck, Germany, 28/08/2007*. http://www.ercim.org/: ERCIM, septembre 2007, p. (on line).

[20] C. Maïza and C. Rochange, "A framework for the timing analysis of dynamic branch predictors (regular paper)," in *International Conference on Real-Time and Network Systems (RTNS), Nantes, 29/09/2011-30/09/2011*. http://www.irccyn.ec-nantes.fr/: IRCCyN, septembre 2011, pp. 65–74.

[21] W. Puffitsch, "Persistence-based branch misprediction bounds for WCET analysis," in *Proceedings of the 30th Annual ACM Symposium on Applied Computing, Salamanca, Spain, April 13-17, 2015*, 2015, pp. 1898–1905.

[22] W. Puffitsch, "Efficient worst-case execution time analysis of dynamic branch prediction," in *28th Euromicro Conference on Real-Time Systems, ECRTS 2016, Toulouse, France, July 5-8, 2016*, 2016, pp. 152–162.

[23] D. Burger and T. M. Austin, “The simplescalar tool set, version 2.0.” *SIGARCH Computer Architecture News*, vol. 25, no. 3, pp. 13–25, 1997. [Online]. Available: http://dblp.uni-trier.de/db/journals/sigarch/sigarch25.html

Marwa A. Elmenyawi Marwa Abdelrazik Elmenyawi was born in Benha, Egypt, in 1983. She received B.Sc. degree in 2004, in computer engineering from High Institute of Technology - Benha University, Benha, Egypt. She received her M.Sc. degree in 2009 from the same university. She is now an assistant lecturer at Faculty of Engineering - Benha University, Benha, Egypt. Also, she is an Ph.D. candidate at Ain Shams University, Cairo, working on Worst Case Execution Time (WCET) analysis.

Cherif Salama Cherif Salama was born in Cairo, Egypt, in 1979. He received the M.Sc. and B.Sc. degrees in electrical engineering from Ain Shams University, Cairo, Egypt in 2001 and 2006 respectively. He received his Ph.D. degree in computer science from Rice University, Houston, Texas in 2010. Since 2010, he was appointed as assistant professor in the Computer and Systems Engineering Department of the Faculty of Engineering of Ain Shams University. He is also the unit head of the Computer Engineering and Software Systems program in the same faculty. His research interests and publications span a relatively wide spectrum that includes hardware description languages, programming languages, parallel computing, and artificial intelligence.

Mostafa E. A. Ibrahim Mostafa Ibrahim received his BS degree in 1997, in computer engineering from High Institute of Technology - Benha University, Benha, Egypt. He received his MSc degree in 2004 from the same university. In November 2009, he received his PhD in electronics and communications engineering from Cairo University, Cairo, Egypt in conjunction with Vienna University of Technology, Vienna, Austria under a channel supervision grant. He spent two years at the Christian Doppler laboratory for design methodology of signal processing algorithms since August 2007 till the end of July 2009 working on his PhD dissertation with Prof. Markus Rupp; regarding estimating and optimizing the power consumption of embedded systems software. He is now an assistant professor at Faculty of Engineering - Benha University, Benha, Egypt. His research interests include Embedded Systems, Software Defined Radio, Image and Video processing, and Computer Architecture.

I. M. Hafez Ismail Mohamed Hafez was born in Cairo, Egypt, in 1961. He graduated from the Faculty of Engineering, Ain Shams University with a B.Sc. in Electronics and Communications in 1983. He got his professorship in 2003. He is currently the Vice Dean for environmental affairs and community service of the Faculty of Engineering of Ain Shams University. His fields of interest include devices, analog and digital circuits, and computer architecture.