INTEGRATION, the VLSI journal 43 (2010) 49–57

Contents lists available at ScienceDirect

INTEGRATION, the VLSI journal

journal homepage: www.elsevier.com/locate/vlsi

Reducing leakage power with BTB access prediction

Roger Kahn, Shlomo Weiss *

Department of Electrical Engineering—Systems, Tel Aviv University, Tel Aviv 69978, Israel

ARTICLE INFO

Article history:
Received 15 June 2008
Received in revised form
13 May 2009
Accepted 15 May 2009

Keywords:
Computer architecture
Microprocessors
Branch prediction
Dual voltage scaling
Leakage power

ABSTRACT

This paper investigates three architectural methods to reduce the leakage power dissipated by the BTB data array. The first method (called here *window*) periodically places the entire BTB data array into drowsy mode. A drowsy entry is woken up by the first access in the time interval and remains active for the remainder of the interval (window). There is an associated performance loss which is related to the size of the window, since there is a delay when a specific line must be woken up. The second method, *awake line buffer* (ALB), limits the number of active BTB entries to a predetermined maximum. While this reduces power dissipation it comes with a performance penalty that is relative to the size of the buffer. ALB, however, reduces the power dissipation of the data array more than the window method. The third method, 2-*level ALB* (2L-ALB), uses a two level buffer with the identical number of combined entries as the previous method. This method exploits the fact that many branches operate numerous times in a fixed sequence. By predicting the next BTB access, 2L-ALB achieves further reduction in leakage power without incurring any further performance loss, compared to the ALB method.

© 2009 Elsevier B.V. All rights reserved.

1. Introduction

Since more powerful microprocessors are being put into devices with small form factors such as laptop computers and various consumer electronic appliances, power dissipation will continue to be a driving force in computer architecture research. In modern CMOS technology there are two elements of power consumption, dynamic (or switching) power and leakage power caused by leakage current from CMOS transistors. Leakage power dissipation is a significant factor in the total power dissipated by microprocessors [10]. As a result, a substantial research effort has been spent in investigating methods that reduce leakage power [29,31,33].

Branch target buffers (BTB) [13,19] provide early target information in speculative processors (during instruction fetch) and are an integral part of the branch prediction mechanism found in modern microprocessors. Power used by branch predictors is significant [9]. The branch prediction mechanism typically dissipates about 7% and as much as 10% of the processor's power, and the BTB typically dissipates about $\frac{7}{8}$ of the power dissipated by the branch predictor overall [18].

In our work we studied three architectural methods for reducing leakage power in the BTB data array. Since the BTB is organized as a cache memory we use the drowsy cache circuit primitive from [5,12]. This circuit, called dual voltage scaling (DVS), reduces the leakage power dissipated by the cache. Two supply voltages are provided to the cache line, one for the awake mode and one for the drowsy mode. The supply voltage is changed when changing between states for the specific cache line. In this design the high leakage power is only relevant when the line is awake and not when the line is drowsy. The reduction of the leakage power to the cache line is around 90%. When a cache line is placed into drowsy mode the data contents are not lost; however, there is a cost of one clock cycle involved in waking up the cache line to enable an access and some power dissipated to wake up the line. According to [5] the power dissipation to wake up the cache line is very low. All three of the architectural methods that we studied are based on this circuit primitive.

We studied the effects of reducing the leakage power of the data array of the BTB using the three methods described below. In order to minimize the performance impact we keep the entire tag array awake. The data array consumes a majority of the power of the BTB so the power reduction is still significant.

This paper makes three contributions toward leakage power reduction in a BTB.

First we present the effect of using a fixed size window for a BTB. With this method (called here *window*) we simply turn off at fixed intervals all the lines of the data array of the BTB. This method was first introduced in [12] for a cache memory. The effect of this policy on performance when applied to a BTB was not presented in [12] however. Hence our first contribution is the evaluation of the window policy when used in a BTB.

Our second contribution is the introduction of a new method, which we call *awake line buffer* (ALB), for reducing the BTB leakage power. ALB employs a buffer of size n that represents the maximum number of entries in the data array that can be awake.

* Corresponding author. Tel.: +972 3 640 7400; fax: +972 3 640 7095.
E-mail address: weiss@eng.tau.ac.il (S. Weiss).

0167-9260/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.vlsi.2009.05.001

This buffer contains a list of the locations in the BTB that are awake. In the event that the buffer is full and another line needs to be woken up, a victim is selected randomly among the lines already awake to be placed into drowsy mode. We evaluate the performance and power impact of this method.

Our third contribution is the introduction and evaluation of another new method, which we call 2-*level ALB* (2L-ALB), for further reducing the BTB leakage power. The 2L-ALB uses a two level buffer. The maximum number of data lines that are awake is the same as in ALB. However, due to the structure of the buffer our simulations show that on average the number of lines awake is from 25% to 33% less than for ALB without any appreciable change in performance. This method introduces the concept of *access prediction* for a BTB, where a prediction is made of the next BTB line that will be accessed. The exact sequence and logic used for this type of access prediction is explained in more detail in Section 2.3. This is different from the traditional function of the BTB which is to provide a predicted target address for a specific branch address.

The rest of the paper is organized as follows. Section 3 contains our simulation parameters and methodology. Sections 2.1, 2.2, 2.3 present the *window*, *ALB*, and 2*L-ALB* methods, respectively, and their power performance results. Section 4 contains a discussion comparing the impact of the methods that we studied and the impact of combining two of these methods that we studied. Section 5 reviews related work on the subject of leakage power reduction, cache prefetching prediction. Our conclusions are in Section 6.

2. Reducing leakage power dissipation of the BTB

In this section we present the three methods that we investigated for reducing the leakage power of the BTB.

2.1. Window

We studied the effect of switching all of the BTB data lines into drowsy mode at fixed intervals. We did this test at intervals of 500, 1000, 2500, 5000, 10 000, and 20 000 cycles. We see that increasing the interval reduced the performance penalty, however, the power dissipation increases since the number of lines that are awake on average are higher at the end of each window. Fig. 1 shows the performance penalty and the percentage power reduction on average of the BTB data array for this method. Measurements are taken by counting the number of data array lines that are awake at the end of every window.

2.2. Awake line buffer

We studied the effect of limiting the maximum number of BTB lines that are awake at any time to a predetermined maximum. In the event that the maximum number of entries are already awake and there is a need to wake up a new entry, then another entry already awake must be placed into drowsy state. An SRAM buffer is maintained whose size is equal to the maximum number of lines that are to be left on. The width of each entry is $\log_2 m$ bits for an m-entry BTB. In all the experiments reported in this paper m was set to 512. We studied the effects of setting the maximum number of awake lines at 32, 64, and 128 entries. We discuss the results in the next section after introducing the 2-level ALB.

2.3. Predicting BTB access with the 2-level awake line buffer

In this section we introduce a novel method 2-(*level awake line buffer* or 2*L-ALB*) that reduces power dissipation without sacrificing performance compared to the ALB method described above. The 2L-ALB is similar to ALB in the overall concept of limiting the maximum number of BTB lines that are awake at any time to a prescribed maximum. However, the similarity between the two methods ends here. The 2L-ALB method exploits the fact that there are often sequences of branch instructions that are repeated (like program loops). If access prediction can be realized (predicting which BTB entry will be accessed next), then we can prevent that BTB entry from being placed into the drowsy state during such a sequence. When a line that is likely to be used shortly is not placed into drowsy mode then the one cycle penalty necessary (in the circuit primitive that we used [5,12]) to wake up the line when needed again is eliminated. This improvement makes up for the fact that although less lines in the BTB are left awake due to the nature of this design that will be described shortly, performance as measured in instructions per cycle (IPC) on average is not sacrificed.

The buffer of size n, which is a power of two, used in the implementation of this method is divided into two parts as shown in Fig. 2. The number n is the maximum number of lines that can be awake. The buffer contains an "upper" level part (*upper level buffer* or *ULB*) consisting of $n/2$ direct-mapped entries which contain a pointer to BTB entries that are currently awake. A BTB entry that is pointed to by an entry in the ULB cannot be placed into drowsy mode. The format of the entries in this buffer is shown in Fig. 3. Each entry in the ULB maps to a specific set of BTB entries. The number of BTB entries mapped to by a single ULB entry depends on the ratio between the size of the BTB and the size of the ULB.

The buffer also contains a "lower level" component (*lower level buffer* or *LLB*) consisting of $n/2$ SRAM entries which are $\log_2 m$ bits

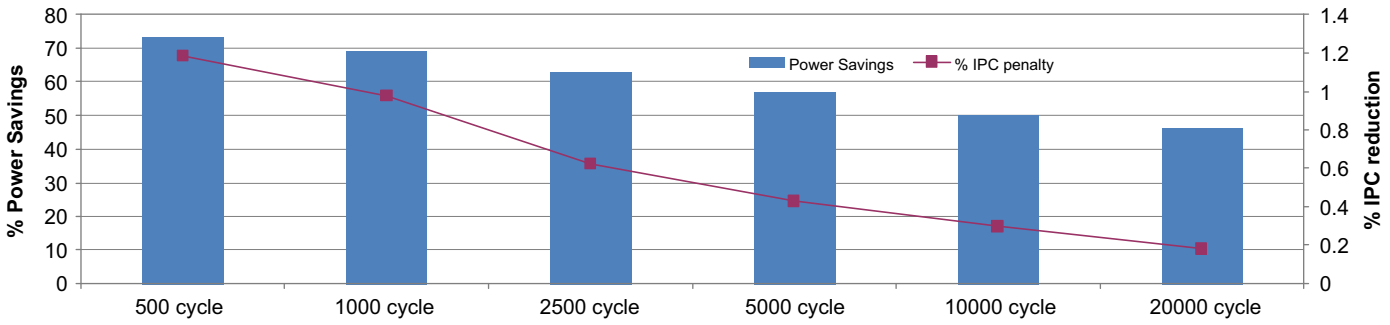

Fig. 1. Performance penalty and power reduction for the BTB data array when using the window method to reduce leakage power dissipation.

Download English Version:

https://daneshyari.com/en/article/538488

Download Persian Version:

https://daneshyari.com/article/538488

Daneshyari.com