Microprocessors and Microsystems 38 (2014) 485–495

Contents lists available at ScienceDirect

Microprocessors and Microsystems

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

Register swapping schemes for low power execution

Po-Yueh Chen [*], Chiung-Hsien Jen

Department of Computer Science and Information Engineering, National Changhua University of Education, Taiwan

ARTICLE INFO

Article history:
Available online 11 March 2014

Keywords:
Instruction bus
Low power
Greedy algorithm
Dynamic programming

ABSTRACT

For embedded systems, the power dissipation on buses has become an essential issue in recent years. Many real-time embedded processors, such as DSP processors, adopt the Harvard architecture in which the data and instruction buses are separated to avoid processing-speed degradation. The power dissipation on an instruction bus can be reduced if the switching activities between consecutive instructions on that bus are reduced. Two efficient algorithms, the greedy method and the dynamic programming based method, are proposed to swap commutative source register fields of adjacent instructions. The switching activities on the instruction bus are therefore reduced, without affecting the execution results. Experimental results show that the proposed schemes result in a reduction of as much as 21.43% in the switching activities of consecutive source register fields between commutative blocks. In addition, the proposed schemes can be conveniently integrated with other encoding schemes to further improve the power dissipation on an instruction bus.

1. Introduction

In recent years, various digital products significantly enhance human life. A variety of inexpensive digital products are implemented because of the rapid development of integrated circuits (ICs). From small-scale integration (SSI) to very large-scale integration (VLSI), the decreasing of chip area makes it feasible to integrate more functions on a single electronic product.

However, due to the increasing clock frequency and circuit integration on a chip, the power dissipation becomes substantial [1]. It results in enormous heat dissipation which decreases the performance and stability of a circuit. In order to have a well-operating system, many schemes on low power design have been proposed in recent years. On the other hand, portable electronic devices such as mobile phones, PDAs, digital cameras, and notebooks become globally widespread. Applying these devices, consumers always have a critical demand for their battery life, which can be extended by adopting appropriate low power schemes on the target systems/chips [2].

Another reason for low power design comes from environmental concerns. Energy consumption induces the emission of carbon dioxide (CO_2), a greenhouse gas (GHG) resulting in global warming. Therefore, many low power policies/programs, such as the Energy Star program [3], have been brought into action around the world. To a certain extent, low power designs benefit both energy conservation and carbon reduction.

In most CMOS circuits, the power dissipation can be categorized into two components: static power and dynamic power. The static power results from the leakage current. Although the CMOS leakage loss is negligible for older technologies, it increases and become respectable as the process dimension decreases [4]. The dynamic power consists of the short-circuit power and the switching power. The former comes from the short circuit currents that arise when pairs of PMOS/NMOS transistors are conducting simultaneously. The latter is due to the charging and discharging of the load capacitance driven by a circuit. It can be modeled as the following equation:

$$P_{switching} = \frac{1}{2} C_{load} \cdot V_{DD}^2 \cdot f \cdot E_{SW} \qquad (1)$$

where C_{load} denotes the load capacitance, V_{DD} is the supply voltage, f is the clock frequency, and E_{SW} represents the switching activity. The switching activity is defined as the number of gates which make a logic transition within one clock cycle [5]. For older technologies, the switching power dominates the total power in a CMOS circuit. For instance, the switching power accounts for 80% of the overall power in 0.35 μm process [6]. Therefore, compared with other types of power dissipation, the switching power is a critical issue.

For systems/chips with buses, the total power dissipation is dominated by the power dissipation on busses [7]. In general, a bus consists of an address bus and a data bus. Because of the nature

* Corresponding author.
E-mail address: pychen@cc.ncue.edu.tw (P.-Y. Chen).

http://dx.doi.org/10.1016/j.micpro.2014.03.001
0141-9331/© 2014 Elsevier B.V. All rights reserved.

of program execution, the addresses are normally sequential, whereas the data transferred via data bus relatively have higher randomness. In [8–10], various techniques are proposed to reduce the dynamic power dissipation on address buses. On the other hand, due to the randomness of data, it is a great challenge to reduce the power dissipation on data buses.

Many real-time systems such as DSP processors adopt the Harvard architecture [11], in which the data bus and the instruction bus are separated to avoid processing-speed degradation. Fig. 1 illustrates the structure of the Harvard architecture. Compared to the Von Neumann architecture [12], in which programs and data are sharing a common memory and therefore one instruction execution requires several cycles, the Harvard architecture allows users to access instruction and data the same time. Furthermore, it also allows different sizes for the data and instruction memory modules. Because the instruction and data memory modules are independent, two busses (an instruction bus and a data bus) are employed by the Harvard architecture. In this paper, exploiting the symmetry of some instructions, we effectively reduce the power dissipation on the instruction buses in the Harvard architecture.

The remainder of this paper is organized as follows. In Section 2, we briefly review various techniques for reducing the power dissipation on instruction buses. Section 3 formally defines the problem for swapping source register fields and proposes two efficient algorithms accordingly. In Section 4, experimental results are presented. The integration of the proposed schemes and another encoding scheme is described and evaluated as well. Finally, the conclusions are summarized in Section 5.

2. Literature review

The techniques for reducing the power dissipation on instruction buses can be mainly categorize into two classes: hardware-oriented and software-oriented. The former utilizes a codec, and even extra bus lines if necessary, at both ends of a bus. Without altering the hardware architecture, the later modifies the compiled codes to reduce the switching activities during execution.

2.1. Hardware-oriented techniques

The bus-invert code [13] utilizes an extra control line to invert the data transmitted over a bus. It guarantees that the Hamming distance of two consecutive data does not exceed one half of the bus width. If the data transmitted over a bus is uniformly distributed, the bus-invert code is a simple yet efficient approach for reduction of switching activity.

In application-specific systems, the real bus data are actually not uniformly distributed. The partial bus-invert code [14] selects some of the bus lines as a sub-bus, and applies the bus-invert code on them whereas the remaining lines are not coded. The advantage of the partial bus-invert code is that unnecessary inversion of inactive/uncorrelated bus lines is eliminated and therefore the codec areas are reduced. Moreover, [14] also proposed the multi-way partial bus-invert coding which divides a bus into some groups where each group has its own invert control line.

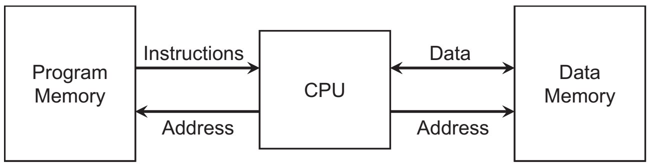

Fig. 1. The Harvard architecture.

The effect of multi-way partial bus-invert coding depends on the correlation between bits across groups. It is therefore not suitable for instruction streams. To improve the effect of partial bus-invert coding on instruction buses, Gu and Guo [15] proposed the segmental bus-invert coding, which divides an instruction into groups based on instruction fields.

Encoding for opcodes [16] can cut down the switching activities of opcodes in consecutive instructions during program execution. It decreases not only the power dissipation on instruction registers but also that on instruction buses.

In [17], the authors presented a low power framework for instruction bus encoding, in which programs are transformed after the phase of compilation. The power dissipation on an instruction bus is reduced accordingly. However, an extra decoder is required for program execution.

2.2. Software-oriented techniques

Petrov and Orailoglu [18] proposed a technique called register name adjustment, which renames the registers' indices to reduce bit transitions of instruction streams. For low power execution, Lee et al. [19] explored compiler transformation techniques to the problem of scheduling very long instruction word (VLIW) instructions. Similarly, Chabini and Wolf [20] achieved low power dissipation on instruction buses by reordering the instructions in basic blocks of target programs. In the study, the authors formulated the problem of reordering instructions as an integer linear programming problem and introduced two heuristic algorithms accordingly.

In next section, we will describe a problem of swapping source register names and propose two algorithms for the problem. Without any extra codec, the simple yet efficient algorithms preliminarily modify the target program and effectively reduce the corresponding bit transitions on the instruction bus.

3. The proposed methods

Commutative operations include addition, multiplication, and all logic operations. More specifically, a commutative operation produces the same result regardless of the order of its operands. While encountering a commutative operation, we can swap the source register fields in the instruction format to reduce the switching activities on an instruction bus. In the following, these instructions are referred as commutative instructions for convenience. An instruction is called non-commutative if it is not commutative.

3.1. Problem description

As shown in Fig. 2, assume that there are $n+2$ consecutive instructions $I_0, I_1, I_2, \ldots, I_n, I_{n+1}$ where $I_1, I_2, \ldots$, and I_n are commutative instructions, whereas I_0 and I_{n+1} are non-commutative ones.

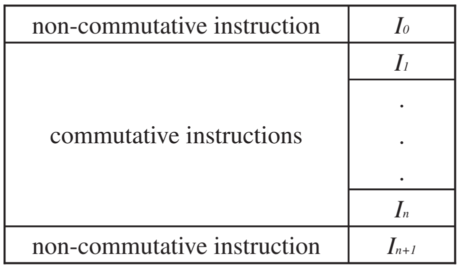

Fig. 2. A commutative block.