Load-Store Reordering for Low-Power Multimedia Data Transfers

Woongki Baek
School of Computer Science & Engineering
Seoul National University
Seoul, Korea
wkb@davinci.snu.ac.kr

Jihong Kim
School of Computer Science & Engineering
Seoul National University
Seoul, Korea
jihong@davinci.snu.ac.kr

***Abstract*—We present a load-store reordering technique for low-power multimedia data transfers. The proposed technique is based on two common characteristics of multimedia applications: (1) Many multimedia streams have a strong spatial locality and (2) output data generated by many multimedia operations are significantly different from the input data. In this paper, we propose a compiler-level technique which combines loop unrolling and load-store scheduling to minimize both the self and coupled transition power on a data bus. Experimental results show that the total amount of the transition activities is reduced on average by 21.3% over the original code for many image processing applications.**

I. INTRODUCTION

As computation and communication have been steadily moving toward mobile and embedded platforms, realizing low power consumption has become a critical concern in designing modern embedded systems. It has been shown that a significant portion of the total power in digital CMOS circuits is dissipated on buses [1]. Thus, a considerable reduction in power dissipation of a whole system can be expected by optimizing power consumption on buses efficiently. In this paper, we attempt to reduce power dissipation in multimedia applications by exploiting inherent characteristics of multimedia streams and operations so as to minimize transition activity on a data bus.

According to [2], power dissipation on buses can be divided into two major types which are s*elf transition power* and the *coupled transition power*, respectively. *Self transition power* is referred to the dynamic power consumption which is proportional to the frequency of transition (i.e., *self transition activity*) on the each single line. On the other hand, with the smaller feature size, power dissipated by coupling capacitance becomes more important. For example, the lateral (i.e., coupling) component of capacitance in the metal 3 layer in a 0.35-μm CMOS process reaches five times the sum of fringing and vertical components when the substrate serves as a bottom plane [3]. *Coupled transition power* is defined as the power dissipation by coupling capacitance.

In this paper, we describe a load-store reordering algorithm for low-power multimedia transfers by reducing both self and coupled transition power on a data bus. Many multimedia applications have two common characteristics. First, many multimedia data streams have a strong spatial locality. Second, output data processed by many multimedia processing operations is very different from the input data. Due to these two characteristics, the amount of transition activities on a data bus can be changed significantly with different sequences of load-store instructions. We propose a low-power load-store reordering technique combined with loop unrolling which reduces both the self and coupled transition activity. Our experimental results show that the significant amount of the power dissipation on a data bus can be reduced by using the proposed technique.

The rest of the paper is organized as follows. Section II summarizes previous research efforts related to our work. In section III, we explain our power model and give a motivational example of the proposed technique. Section IV presents the proposed load-store reordering problem to minimize both the self and coupled transition power on a data bus. We show experimental results using image processing applications in Section V. Section VI concludes with a summary.

II. PREVIOUS WORK

There have been many investigations which attempted to minimize the power dissipation on buses at the various levels of the design abstraction from the gate level to the system level.

Panda and Dutt [4] focused on the reduction of power dissipation in memory intensive applications by reducing the number of transitions on the memory address bus. They exploited regularity and spatial locality in the memory access and determined the mapping of behavioral array references to physical memory locations to minimize address bus transitions. Dasgupta and Karri [5] presented algorithms to minimize on-chip data bus transitions by suitably binding and scheduling the data transfers of a Control Data Flow Graph (CDFG). Kim *et al*. [6] proposed a low-power bus encoding scheme to minimize the coupled transition

power.

The aforementioned approaches [4]-[6] are designed to minimize either the self transition power or the coupled transition power, but not both. On the other hand, Lyuh *et al*. [2] presented an on-chip bus synthesis algorithm to minimize the total sum of the self and coupled transition power in the micro-architecture synthesis. Unlike the previous approaches, they minimized both the self and coupled transition power in an integrated fashion.

In this paper, we propose the compiler and application level technique to minimize the total sum of the self and coupled transition power on a data bus. The aforementioned approaches [2], [4]-[5] can be applied only to the hardware synthesis level while our method can be still applicable when the hardware design is already determined. Bus encoding schemes (e.g., [6]) need extra hardware units such as bus encoder and decoder. However, our technique can be used to conventional embedded processors without any extra hardware.

III. BASIC IDEA

A. Power Model

According to [6], the dynamic power consumption by interconnects and drivers for the period of execution of T clock steps is given by

$$P_{dyn} = (X_T \cdot (C_s + C_l) + Y_T \cdot C_c) \cdot V_{dd}^2 \quad (1)$$

where C_s and C_l are self capacitances, C_c is coupling capacitance, and V_{dd} is the supply voltage. The capacitance ratio is defined as $\gamma = (C_c)/(C_s + C_l)$.

As for the coupled transition activities, there are four types of possible coupled transitions, as illustrated in Fig. 1. Type 1 and 3 result in C_c not being charged while Type 2 and 4 cause C_c being charged up to $\alpha C_c V_{dd}$ and $\beta C_c V_{dd}$, respectively [2]. [α and β are set to 1 and 2 in [2]].

Assuming that no Dynamic Voltage Scaling (DVS) techniques are applied (i.e., V_{dd} is constant.), optimizing power dissipation on a data bus is equivalent to minimizing the weighted sum of the self and coupled transition activities Z_T which is given by

$$Z_T = X_T + \gamma \cdot Y_T. \quad (2)$$

B. Motivational Example

We show a motivational example which illustrates how load-store ordering affects both the self and coupled transition activities on a data bus in Fig. 2. Two 8-bit image pixels which respectively have values of $01110111_{(2)}$ and $01110110_{(2)}$ will be inverted in bit-wise. We suppose that image pixels will be transferred via an 8-bit on-chip data bus. In Fig. 2(a), two pixels will be inverted by following the load-store sequence, which is `load` $pixel_0$, `store` *inverted* $pixel_0$, `load` $pixel_1$, and `store` *inverted* $pixel_1$. A different load-store scheduling, which is `load` $pixel_0$, `load` $pixel_1$, `store` *inverted* $pixel_0$, and `store` *inverted* $pixel_1$, is given in Fig. 2(b).

In Fig. 1(a), the amount of the self transition activities (marked with rectangular boxes in Fig. 2(a)) is 10 and the amount of the coupled transition activities is 21, respectively.

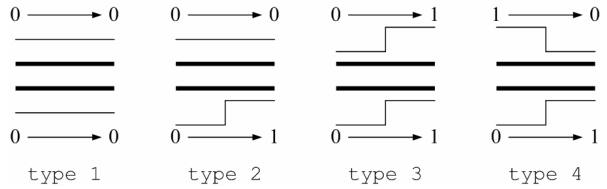

Figure 1. Four different types of coupled transition activities.

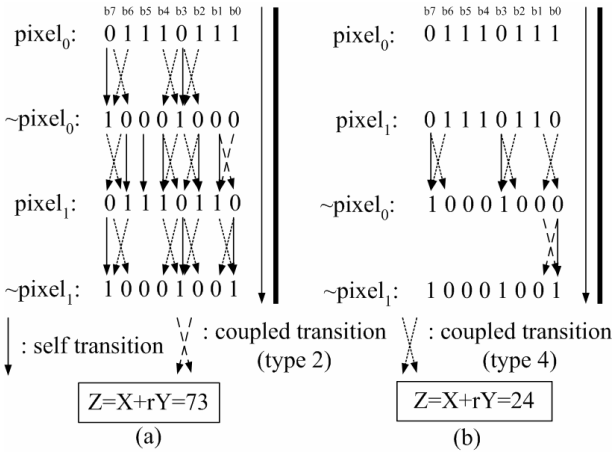

Figure 2. The total amount of the transition activity on a data bus while inverting two image pixels by following two different load-store sequences. (a) a load-store-load-store sequence. (b) a load-load-store-store sequence.

Assuming the capacitance ratio γ is 3, the total amount of transition activities Z_T will be $10 + 3 \cdot 21 = 73$. On the other hand, the amounts of the self and coupled transition activities are 3 and 7, respectively, in Fig. 2(b). Thus the total amount of transition activities becomes 24.

We can explain why the amount of transition activities occurred by two load-store sequences in Fig. 2(a) and Fig. 2(b) are significantly different by the following two reasons. First, there is a strong spatial locality in two input image pixels (only 1 bit is different.). Strong spatial locality usually can be found on many multimedia streams. If data which have very similar values are transferred consecutively via a data bus, small amount of transition activities can be expected. Second, the values of output data are very different from the values of the corresponding input data because the image processing operation in our motivational example is image inverting. This characteristic can be found many other multimedia processing operations. If the input data are transferred consecutively via a data bus, it will usually cause the large amount of transition activities. This motivational example clearly shows how important load-store scheduling is to minimize the power dissipation on a data bus.

IV. LOW-POWER LOAD-STORE REORDERING PROBLEM

To address the low-power load-store reordering problem formally, we need to consider two sub-problems. First, we should determine the optimal number of loop unrolling to reduce the power dissipation on a data bus under the given register budget. Second, we need to find the load-store scheduling which minimizes the transition activities on a

```
1.  Find the optimal load-store scheduling of the current loop by
    using the method described in Section IV-B.
2.  Calculate the transition activities caused by step (1).
3.  Unroll the loop once more. If extra register spills are occurred,
    set the optimal number of loop unrolling to the current number
    of loop unrolling and exit.
4.  Find the optimal scheduling of the loop of step (3).
5.  Calculate the transition activities caused by step (4).
6.  If (amount of step (2) > amount of step (5))
        goto step (3);
    else {
        optimal # of loop unrolling = current # of loop unrolling;
        exit;
    }
```

Figure 3. A greedy loop unrolling algorithm

data bus. We will discuss two sub-problems in Section IV-A and IV-B, respectively.

A. Loop Unrolling Problem

Many multimedia applications have two important characteristics. (1) Many multimedia streams have a strong spatial locality. (2) Most of multimedia operations change the value of input data value drastically. As we have seen in the motivational example, loop unrolling generally helps us to have more chances to exploit these characteristics to reduce the transition activities. To find the optimal number of loop unrolling, we take a greedy approach. We show the proposed algorithm in Fig. 3. Since no extra register spills will be caused by the proposed algorithm, there will be no side effect on both of the performance and the power dissipation on a data bus.

B. Load-Store Reordering Problem

To find a load-store scheduling which minimizes the transition activities on a data bus, we formulate the problem using a Load-Store Transition Activity Graph (LSTAG) $G = (V, E)$. LSTAG is a directed and weighted graph with a vertex set $V = \{v_1,\ldots,v_n\}$. Each vertex $v_i \in V$ corresponds to a load or store instruction in the given multimedia application. Each edge e_{ij} indicates the data dependency from vertex v_i to v_j. The weight w_{ij} of the edge represents the amount of the transition activities when instruction v_j is executed right after instruction v_i. Edge weights can be determined by profiling the amount of transition activities for the given multimedia data and operation. Fig. 4 shows an example of the LSTAG for an image inverting application.

Once the LSTAG graph G is constructed, we can find the optimal load-store scheduling by finding a Hamiltonian path on G which visits all the vertices exactly once with a minimum-cost. However, due to the data dependency, each load instruction should be ahead of the corresponding store instruction for the correctness of the program. Considering this constraint, the low-power load-store reordering problem can be reduced to solving a variant of the Asymmetric Traveling Salesman Problem (ATSP).

Although ATSP is an NP-complete problem, we can optimally solve the load-store reordering problem using a branch-and-bound-based approach. The reason is that the

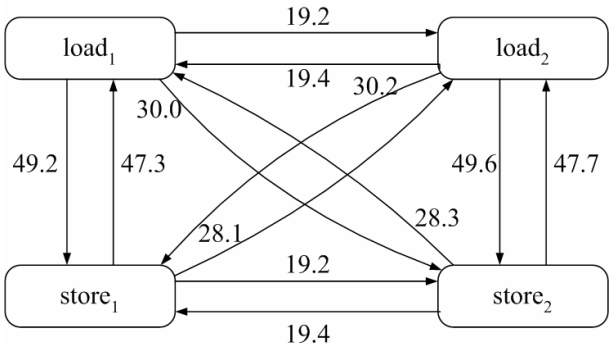

Figure 4. An example of the load-store transition activity graph.

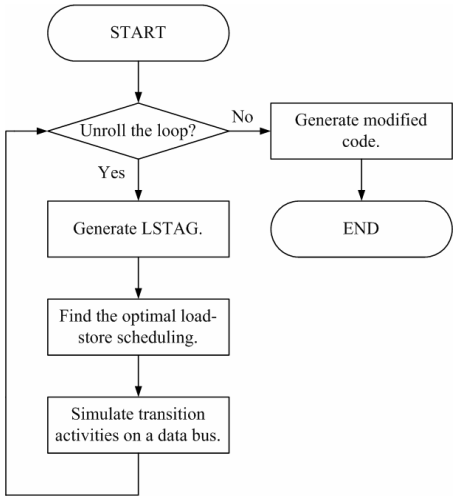

Figure 5. Overall flowchart for the load-store reordering problem.

number of load-store instructions in a typical multimedia application is not quite large (i.e., relatively small solution search space) because the number of loop unrolling is bound by the available register budget with which loop can be unrolled without extra register spills.

V. Experimental Results

We implemented all the components of the proposed load-store reordering algorithm in C and executed them on an Intel Xeon Pentium IV computer with 2.4GHz clock speed. The load-store scheduling solver which was implemented using a branch-and-bound-based algorithm found the optimal solution within a reasonable time (< 1s). We assumed two types of embedded micro-processors which respectively have 8-bit and 32-bit on-chip data buses with 32 general-purpose registers. To validate the effectiveness of the proposed algorithm, we performed experiments for a number of image processing applications such as inverting, thresholding, bit-plane slicing (BPS), high-pass filtering (HPF), and histogram equalization [7] on 256 × 256 8-bit images.

Table. 1(a) and (b) summarize the reduction in the amount of the total transition activities. We set the capacitance ratio (γ) to 3 and normalized all the results to the amount of total transition activities caused by the original code. As shown in Table. 1(a) and (b), the proposed scheme works well for the image inverting and thresholding pro-

application	the number of loop unrolling	reduction in transition activities (%)
inverting	4	54.9
thresholding	4	34.0
BPS	4	31.6
HPF	4	16.7
histogram	2	14.6
average		30.4

(a)

application	the number of loop unrolling	reduction in transition activities (%)
inverting	4	48.6
thresholding	4	23.1
BPS	2	17.8
HPF	2	9.9
histogram	1	7.1
average		21.3

(b)

Table 1. Reduction in the total transition activities ($\gamma = 3$). (a) on an 8-bit data bus. (b) on a 32-bit data bus

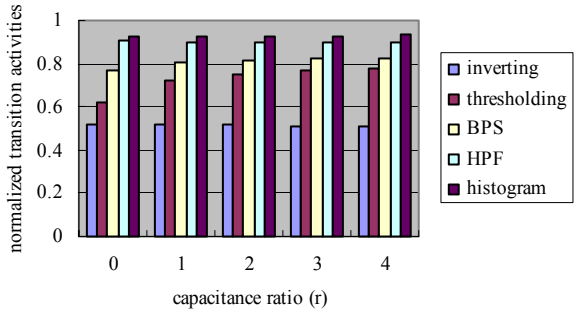

Figure 6. Effect of the capacitiance ratio on the total transition activities on a 32-bit data bus

grams. This is because the image inverting and thresholding make output data significantly different from the input data, giving more rooms for an improvement by load-store reordering. On the other hand, the proposed approach is less efficient for the histogram equalization because the output data are less different from the input data than the former image processing applications and loop can be unrolled only once due to the small budget of registers. On average, the proposed reordering reduces the amount of the total transition activities by 21.3% on a 32-bit data bus.

To study the effect of the size of data buses on the total transition activities, we performed experiments with two different micro-processors which have 8-bit and 32-bit data buses, respectively. As shown in Table. 1(a) and (b), reduction in the total transition activities on a 32-bit data bus is generally less than the one on an 8-bit data bus. We can explain this phenomenon by following two reasons. First, due to the interference between different pixels which are loaded and stored simultaneously on a 32-bit data bus, additional transition activities have been occurred as a side-effect. Second, with a 32-bit data bus, less number of loop unrolling can be done than with an 8-bit data bus due to the insufficient register budget. However, the amount of reduction in the total transition activities is still considerable as shown in Table. 1(b). Thus we can expect that the proposed technique will work well with modern embedded micro-processors which have large size of data buses.

We also performed an experiment to study the effect of the capacitance ratio on reducing the power dissipation (refer to Fig. 6). We changed the capacitance ratio from 0 (i.e., only the self transition activities are considered.) to 4. The proposed method works less efficiently with thresholding and bit-plane slicing as the capacitance ratio increases, while the other multimedia applications remained almost unaffected. However, even with the high capacitance ratio, our method still reduces the amount of the total transition activities considerably (21.3% with γ=3 and 22.5% with γ=4). According to [2] the capacitance ratio will be increasing with the smaller feature size. From the experimental result, we can expect that the proposed method will be still quite effective as the scale of process technology shrinks.

VI. CONCLUSION

We presented a load-store reordering technique for reducing the amount of transition activities on a data bus. Our method is based on the observation that many multimedia streams have a strong spatial locality and most of multimedia operations generate the output data which is significantly different from the input data. To exploit these characteristics, we combined loop unrolling and load-store instruction reordering to reduce the transition activities on a data bus. Experimental results show that the proposed approach is quite effective, achieving about 21.3% reduction in the amount of transition activities on average for many image processing applications.

ACKNOWLEDGMENT

This work was supported by University IT Research Center Project.

REFERENCES

[1] A. P. Chandrakasan, S. Shung, and R. Brodersen, "Low-Power CMOS Digital Design," *IEEE Journal of Solid-State Circuits*, vol.27, No.4, pages 473-484, April 1992.

[2] C. Lyuh, T. Kim, and K. Kim, "Coupling-Aware High-Level Interconnect Synthesis," *IEEE Trans. Computer-Aided Design*, vol. 23, pp. 157-164, Jan. 2004.

[3] Y. Shin and T. Sakurai, "Coupling-driven bus design for low-power application-specific systems," in *Proc. Design Automation Conf.*, 2001, pp. 750-753.

[4] P. R. Panda and N. D. Dutt, "Low-power memory mapping through reducing address bus activity," *IEEE Trans. VLSI Syst.*, vol. 7, pp. 309-320, Sept. 1999.

[5] A. Dasgupta and R. Karri, "Simultaneous scheduling and binding for power minimization during microarchitecture synthesis," in *Proc.* Int. Symp. Low Power Design, Apr. 1995, pp. 69-74.

[6] K. Kim, K. Baek, N. Shanbhag, C. L. Liu, and S. Kang, "Coupling-driven signal encoding scheme for low-power interface design," in *Proc. Int. Conf. Computer-Aided Design*, 2000, pp. 318-321.

[7] R. C. Gonzalez and R. E. Woods, Digital Image Processing, Addison-Wesley, 1992, pp. 161-249.