ISSN: 2472-3703 DOI: 10.6911/WSRJ.201908_5(8).0007

Study of New Method for Low Data Bus Dynamic Energy

Mingquan Zhang[1, a]

[1]School of Control and Computer Engineering, North China Electric Power University, Baoding 071003, China

[a]2898637276@qq.com

Abstract

In the paper, the author proposed a method named BI-FV (Bus Invert-Frequent Value) encoding that reduces the bus dynamic energy further than the conventional bus invert coding. In the proposed scheme, only a small cache and one line are added to the bus. Experimental results show that the BI-FV encoding reduces the bus dynamic energy by an average of 18.2%, compared to which without the method, and by an average of 7.1%, compared to conventional bus invert coding.

Keywords

Bus energy, frequent value, bus invert coding.

1. INTRODUCTION

For multi-core chip the energy consumption of per unit area and per unit time is significantly increased, which brings many problems. Firstly, the rapid increase of energy consumption is bound to hinder the performance further improved; studies have shown that now in multi-core the urgent problem to be solve is dynamic energy consumption control [1]. Secondly, the high bus energy consumption will increase the cost of the chip design, such as additional cooling devices and heating protection circuits are need to bring in, making the package costs of the chip increased significantly [2]. Finally, the increase of the bus energy consumption makes the temperature of the chip higher, which leads to a series of problems, such as the reduction of system reliability, the shortening of the supply time of battery power and the increase of the execution cost [3]. So optimizing the data bus dynamic energy saving becomes the research object for software and hardware designers.

Bit switching activity (SA) is a major source of power consumption in bus-based systems. Power consumption for on-chip driving can reach up to 30% of the total chip power, where the bit SA is the most dominant factor [4]; this is because of the large bus capacitances. Therefore, the reduction of bus SA has a large impact on the reduction of the system energy consumption.

With the rapid increase in the complexity and speed of integrated circuits and the popularity of portable embedded systems, power consumption has become a critical design criterion [2]. In today's processors, a large number of data needs to be sent through high-speed data bus. Compared to a general-purpose high-performance processor, an embedded processor has much fewer transistors integrated on the chip. Therefore, the amount of the energy dissipated on the data bus of an embedded processor is significant when compared with the total energy consumption of the general-purpose processor. On the other hand, there is a lot of value locality of the data for transmission on the bus. Consequently, low-power bus encoding techniques are often more attractive in the context of embedded processors. It is desirable to encode the values sent over these buses to decrease the SA and thereby reducing the bus energy consumption. An encoder on the sender side does this encoding, whereas a decoder on the receiver side is required to restore the original values.

ISSN: 2472-3703 DOI: 10.6911/WSRJ.201908_5(8).0007

In this paper, we propose a new method named BI-FV encoding which can reduce the bus SA and bus dynamic energy efficiently, and introduce low overhead. Our methods are irredundant, meaning that they require a small cache and one line to be added to the bus.

2. BUS INVERT CODING AND FREQUENT VALUE

2.1. Bus Invert Coding

Our approach is closely related to the bus invert (BI) coding [5]. Therefore, we first study the effect of BI coding on SA reduction. SA reduction rate with BI coding is as follows: For a sequence of w-bit code words, assume that their Hamming distances are $h_1, h_2, \ldots, h_n$,

$$h_i = \sum_{j=1}^{w} S_{(i-1)j} \oplus S_{ij}, \ i = 1,2,3,\cdots,n \tag{1}$$

Where n is the length of the code sequence, si j the jth bit of word i (denoted by si) in the sequence, and $\oplus$ the logic XOR operation.

Without any bus encoding, the total number of bit SA for the sequence of data when it is transferred on the bus is

$$\mathrm{SA} = \sum_{i=1}^{n} h_i \tag{2}$$

When BI is applied to this sequence, some words will be bit-inverted, if their Hamming distances are larger than w/2, the half of word width. The associated Hamming distances will be changed accordingly. When BI encoding is taken into account, the Hamming distance of a word, si, can be generalized as

$$H_i = \begin{cases} h_i & c_{i-1} = 0, \\ w - h_i, & c_{i-1} = 1, \end{cases} \tag{3}$$

Where ci−1 is the invert control of the previous transfer; when it equals 1, the previous transferred value is bit inverted.

2.2. Frequent Value Encoding

The principle of frequent value (FV) cache encoding[2] is as follows: the data with low frequency is transmitted on the bus in the form of original value, and the high frequency data is transmitted on the bus in the form of encoded value, which has little SA. In order to ensure correct decoding, additional bits are required to control the signal line. Because the high frequency data is encoded value with little SA, the total SA of the transmitted data is largely reduced and so the bus dynamic energy is reduced.

3. THE PROPOSED ENCODING SCHEME

Combined with the characteristics of the above two encodings, we propose a method to increase the auxiliary cache, that is, increasing frequent value caches for frequent transmission values. With the features of FV coding and value locality, we add FV cache at sender and receiver ends. When the sent value is searched in the FV cache of the sender end, it is transmitted as the index in the FV cache, and the receiver end is prompted with an indicator line. When the sent value is not in the sender FV cache, the original value will be sent. At the same time, we combine with BI coding, the SA is further reduced, and the receiver end decides to deal with the received values according to the indicator lines.

ISSN: 2472-3703 DOI: 10.6911/WSRJ.201908_5(8).0007

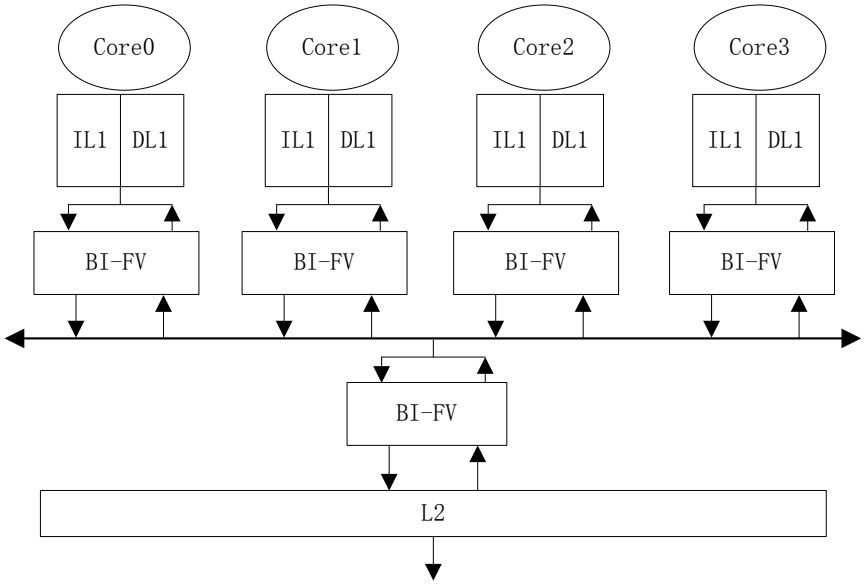

Figure 1. Multi-core architecture with BI-FVC based on bus

Our multi-core structure based on bus is as shown in Figure 1. It is four-core CMP structure, each core has 4-way set associative private first level of instruction cache (IL1) and data cache (DL1) with a size of 32KB, and 16-way shared second level cache (L2) with a size of 1MB. The cache row size at all levels is 64B, and it is included cache. The specific parameters are shown in Table 1. In the structure, there are five FV cache modules - one at each of the cores and one at the L2 end as shown in Figure 1. Each of the private L1 split caches is write-through. The shared L2 cache is write-back and maintains inclusion with respect to the L1 cache. In our multi-core system, the cores and L2 cache are connected by bus. The data bus is alternately used by different cores, so as to achieve the purpose of access the shared L2 cache.

4. EXPERIMENT AND RESULTS ANALYSIS

The default parameters are shown in Table 1. To verify the efficiency of bus energy saving, some of the parameters are adjusted in the experiments. We select three storage intensive programs from Olden [6] and CPU2006 [7] for performance evaluation benchmarks: mst_ht, em3d_ht and 429.mcf_ht. The programs are cross-compiled with GCC for MIPSII executable files. In order to reduce the impact of code optimization on application performance during program design, and make the program reach its peak as far as possible when the program runs, the GCC optimization option is the best optimized -O3.

Table 1. Three Scheme comparing

parameters	value
process technology	70nm
clock speed	500MHz
cores	4
IL1/DL1 size	32KB
L1 associativity	4-way
FVC	4entries,fully-associative
FVC energy/access	18.6pJ
bus width	32+2 lines
bus energy/access	11.6pJ/line

ISSN: 2472-3703 DOI: 10.6911/WSRJ.201908_5(8).0007

The number of bit SA on the bus refers to the number of charge and discharge caused by the "0" and "1" transformation when the bus transfers data. After the measures are adopted, the amount of communication and the number of SA on the bus lines are effectively reduced. Figure 2 shows the ratio of the number of bit SA with different measures to the number of original bit SA on the bus. S_BI indicates the ratio with using the BI coding alone. S_F1, S_F4, S_F8 respectively indicate the ratio with using FV alone and FV cache stores 1, 4, and 8 data values. S_BIF1, S_BIF4, S_BIF8 respectively indicate the ratio with using BI-FV encoding and FV cache contains 1, 4, 8 data values. The number of bit SA is reduced by an average of about 8% by using BI coding alone. When only one value is stored by FV cache alone (S_F1), the effect of reducing the number of bit SA is not obvious, while the number of SA can be greatly reduced (more than 15%) when the 4 values are stored (S_F4), and the continued increase of the entries of the FV cache cannot further reduce the number of bit SA further. This is because the increase of the entries of the FV cache will also increase the number of indexes, and it offsets some of the gains from these measures.

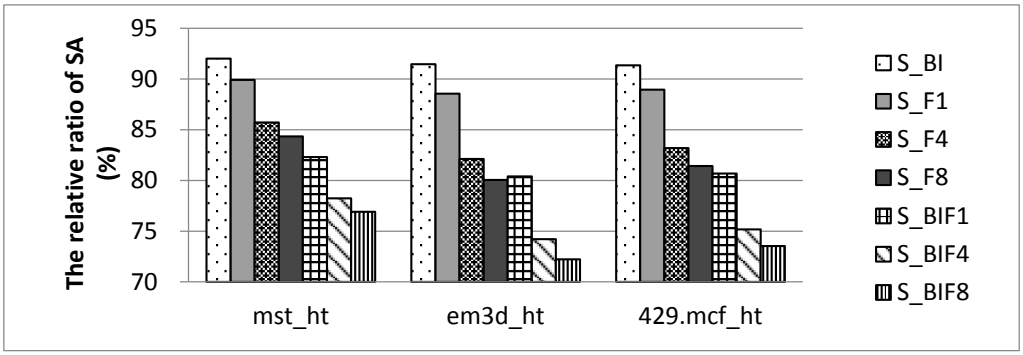

Figure 2. The relative ratio of bus SA with different measures

The number of bit SA is further reduced when using BI-FV encoding than using BI alone. This is due to the use of the FV coding algorithm can further reduce the number of bit SA.

As is known to all, bit SA directly determines the on-chip bus energy consumption. After using our method, the number of bit switching activity of the benchmarks is reduced in varying degrees, which brings advantages to bus energy saving. For bus energy saving there are the same trend as shown in Figure 3. Figure 3 is a comparison diagram of the energy saving effect on bus with different measures.When the measure is only FV cache, the bus energy consumption is reduced with the increase of n, and the maximum ratio of energy saving can be achieved 13.5% . When invert coding is used with the FV cache, the change trend of FV cache and bus energy consumption is consistent with the single use of FV. The greater the value of n, the greater the energy consumption of FV cache, and the limited space on the chip, so the value of n should not be too large. Considering the BI-FV with the FV cache taking with n=4 as a measure to optimize the energy consumption of the bus, the energy saving effect can be maximized.

ISSN: 2472-3703 DOI: 10.6911/WSRJ.201908_5(8).0007

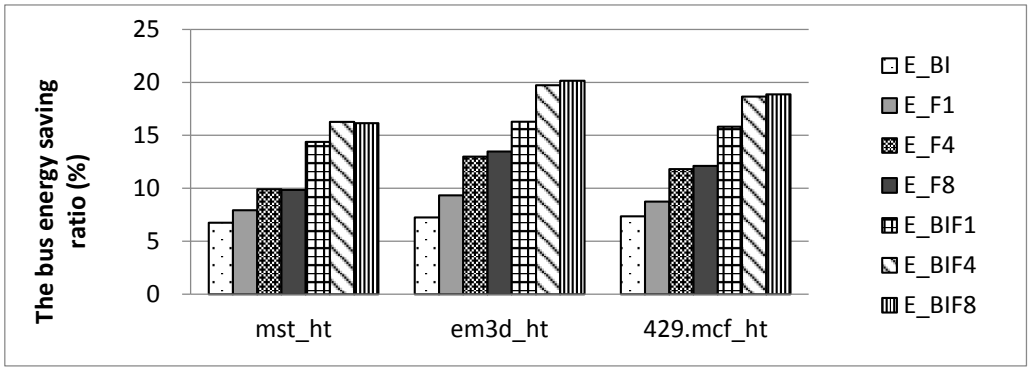

Figure 3. The ratio of bus energy saving with different measures

5. SUMMARY

In this paper we proposed a bus energy saving mothed BI-FV, which efficiently reduces the number of bus SA and so the bus dynamic energy is reduced. The encoding scheme is particularly suitable for memory intensive special-purpose applications. However, the method is general enough to be used in other types of buses. Experimental results show that the proposed encoding method is efficient.

ACKNOWLEDGEMENTS

This paper was financially supported by “the Fundamental Research Funds for the Central Universities (2018MS073)”.

REFERENCES

[1] Wang, S. N., Luo, B., Shi, W. S., Tiwari, D., Application configuration selection for energy-efficient execution on multicore systems[J], Journal of Parallel and Distributed Computing, 87(1),43-54 (2016).

[2] LIU, C., Sivasubramaniam, A., Kandemir, M. Optimizing bus energy consumption of on-chip multiprocessors using frequent values [J]. Journal of Systems Architecture, 52(2):129-142(2006).

[3] Niu, L. W., Energy efficient scheduling for real-time embedded systems with QoS guarantee [J], Real-Time Systems, 47(2), 75-108 (2011).

[4] Chiu, Ching-Te, Huang, Wen-Chih, Lin, Chih-Hsing, et al. Embedded transition inversion coding with low switching activity for serial links [J]. IEEE Transactions on Very Large Scale Integration (VlSI) Systems, 21(10):1797-1810 (2013).

[5] Zhang M, Gan Z, Zhang J, et al. “Joint Hybrid Frequent Value Cache and Multi-Coding for Data Bus Energy Saving”, IEEE, International Conference on Parallel and Distributed Systems. IEEE, 869-876(2017).

[6] Rogers, A., Carlisle, M. C., Reppy, J. H., Hendren, L. J., Supporting dynamic datastructures on distributed-memory machines[J], ACM Transactions on Programming Languagesand Systems, 17(2), 233-263 (1995).

[7] 17. Henning, John L., SPEC CPU2006 benchmark descriptions [J], SIGARCH Computer Architecture News, 34(4),1-17 (2006).