Low-Power Prediction Based Data Transfer Architecture

Maged Ghoneima [1], Ehsan Atoofian [2], Amirali Baniasadi [2], Yehea Ismail [1]
[1]ECE Department, Northwestern University [2]ECE Department, University of Victoria
{ ghoneima[1], ismail[1]}@ece.northwestern.edu, {eatoofia[2], amirali2}@ece.uvic.ca

ABSTRACT

The energy dissipation of on-chip buses is becoming one of the main bottlenecks in current integrated circuits. This paper proposes a prediction-based technique to reduce data- bus power dissipation. This technique uses value prediction to speculate the next value to transfer over the data bus. Two identical predictors are placed on the two ends of the bus. If the sender predictor accurately predicts the data to be transferred, data is not transmitted, and the bus energy dissipation is reduced. After implementing the proposed architecture in a 70nm CMOS technology, and using a representative subset of SPEC2K benchmarks, an average of 41% of the bus transitions were eliminated, which led to a 31% average overall energy reduction.

1. Introduction

With the continuous scaling of technology, increased die area and faster clock speeds, the delay and energy dissipation of on-chip buses are becoming one of the main bottlenecks in current integrated circuits. The average energy dissipation of an n-bit bus can be expressed as

$$E_{av} = 0.5 \cdot n \cdot AF \cdot C_{eff} \cdot V_{DD}^2 \qquad (1)$$

where AF is the bus activity factor, C_{eff} is the line effective capacitance, and V_{DD} is the supply voltage. As technology scales, the effective capacitance of a bus line is significantly increasing due to increasing coupling capacitance. Also due to the larger die areas, buses span longer distances, leading to higher interconnect lengths. Moreover, the bus width n is being increased by designers to increase the bus throughput. All of these factors lead to a dramatic increase in the energy dissipation of on-chip buses. As a result, among different processor resources interconnect wires account for a significant fraction (up to 50%[1]) of the energy consumption. This fraction is expected to grow in the future [2]. Thus, on-chip buses should be designed to consume reasonable power while providing sufficient performance.

Several architectures were proposed in the literature to reduce the energy dissipation of on-chip buses. Some of these low power bus architectures reduce the energy dissipation by targeting the effective coupling capacitance between adjacent bus lines [3]-[10]. These techniques use coding and circuit techniques to eliminate the occurrence of the worst-case switching scenario and hence reduce the effective line capacitance.

Other architectures concentrate on reducing the number of bus transitions [11]-[16], i.e. reducing the bus activity factor AF. Bus inversion encoding [11] sends either the data or its complement based on the hamming distance from the previously sent data. Adaptive encoding [12] relies on expensive hardware to dynamically adapt to the data stream by using data statistics. [13]-[14] reduce address bus power dissipation by using sequential and stride address patterns, where only the difference between consecutive addresses is sent instead of sending the entire address. [15] suggests frequent value encoding to reduce data bus power. They show that a small number of values occur frequently on data bus. By detecting and encoding frequent values dynamically, they reduce the number of data bus transitions. Basu et al. [16] use value caches at each end of data bus to reduce power. The value caches keep the recent values sent over data bus. If the data to be sent is found in value cache, the index of the data in value cache is sent instead of the actual data. Consequently, the number of active lines is reduced. Our work is different from previous work as we use data locality and rely on data addresses to index the value predictor.

Lipasti et al [17],[18] showed that values in registers and memory locations have high locality. Locality is the likelihood of a previously seen value reappearing in the same location. In this paper, we propose using value locality within memory locations to reduce the number of transactions over data buses. We use a small hardware to keep data transfer history. This history is used to predict future values to be sent over bus. The same history is maintained on both ends of the bus. Before sending data, the sender accesses its local history. If the real data and the predicted value match, nothing is sent over the bus. Consequently, in the case of correct prediction the number of transaction over data bus is reduced. Using a representative subset of SPEC2K benchmarks, 41% of bus transitions were reduced on average. This in turn resulted in 31% power reduction in a 70nm CMOS technology.

The rest of the paper is organized as follows. In section 2 we explain the concept of value locality. We discuss details of our method and hardware implementation in section 3. Section 4 includes experimental results. Finally, section 5 presents the conclusion.

2. Value Locality

Value or data locality, indicates the likelihood of the recurrence of a previously-seen value within a storage location. Value prediction has been used to exceed data flow limits in high performance processors [17]-[19]. Figure 1

presents value locality for data sent over the data bus for a representative subset of SPEC CPU2000 benchmarks. The bars show the probability of recurring previously seen values within a memory location. Here we report value locality for a history depth of one, *i.e.,* we report how often the retrieved value matches the most recently seen value.

On average, 45% of data accesses are just a repetition of the previous transaction. In other words, ideally and by using an oracle predictor, 45% of bus transactions could be eliminated. While ideal prediction is not possible, several studies have suggested relatively accurate value prediction techniques [17],[19]. In this work we use a power efficient and low-overhead value predictor to reduce the number of cache memory accesses, and hence reduce the traffic on the data bus.

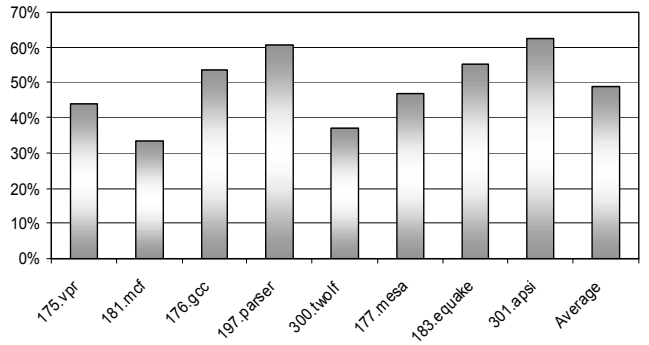

Figure 1. Data locality for values sent over the data bus

3. Prediction-Based Data Transfer

In this section we discuss the details and hardware implementation of our prediction-based data transfer technique. We later report how predictor size impacts our technique.

3.1 Design Methodology

Our proposed technique uses a value predictor to speculate the value being transmitted. Correct value predictions will save bus power dissipation by eliminating data transfer on the data bus. To explain this better, we present a schematic of predictor-enhanced data bus in Figure 2. Figure 2 shows a mirror pair of value predictors at the two ends of the bus. One is on processor side and the other one is on the cache side. The two predictors are virtually the same, and use the same mechanism to predict data values. The predictor on the sender side has access to the original data vector, and can verify the predicted value correctness using the predictor logic unit. If an accurate data prediction is detected the data transfer would not be necessary and therefore is avoided. Under such circumstances, the value predictor used on the other side will make the same exact prediction, and use the speculated data. Note that predictors on both sides are maintained consistent and produce the same data. The receiver assumes that the data predicted by its predictor is predicted accurately, unless informed otherwise by the sender. In the case of a misprediction, the sender sends the original value, and no power saving is achieved. A predict control bit is sent along with the data bus to show the correctness of prediction, as shown in Figure 2.

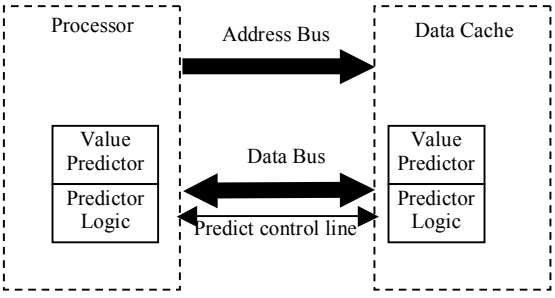

Figure 2. Mirror predictors for data bus between a microprocessor and a cache memory

3.2 Hardware Implementation

Figure 3 shows the schematic of last value predictor [17]. This predictor uses the most recent value to make future predictions. The last value predictor includes a decoder and a value history table (VHT). The decoder is used to select the proper VHT entry. Every VHT entry consists of two fields: tag and value. The decoder uses part of the address. The remaining address part is used as the tag field. Since multiple addresses may map to the same VHT-entry, the tag field is needed to distinguish them. The tag field is used to assure that the predicted data is associated with the right address. We use the VHT value field to record the last value of a memory location.

When a memory location is accessed for the first time, a VHT entry is allocated for the memory location and the data stored in the memory location is stored in the value field. Next time the same memory location is accessed, the value field is used to predict the data.

At the sender side, the predicted data is compared with real data. If both values are equal, the prediction is verified and data is not sent over the bus. In the case of a mismatch, the real data is sent over the bus, and both the sender and receiver update their VHTs using the real data. Therefore, sender and receiver use consistent information for prediction.

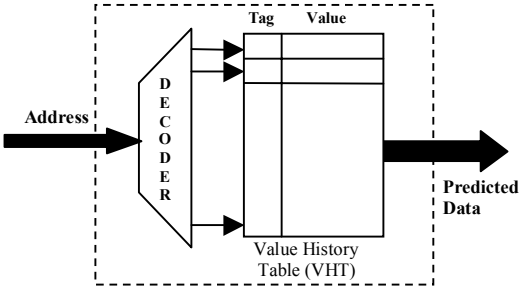

Figure 3. Last Value Predictor

The overhead of this prediction based data transfer architecture is the two value predictors, the predictor logic and the additional predict control line. It will be shown in the next section, that for buses longer than a certain value, the energy penalty of this overhead is compensated by the energy savings

due to the reduced data bus traffic, and an overall energy savings is achieved.

3.3 VHT Size Sensitivity Analysis

In this section, we report how value predictor size impacts value prediction accuracy. As explained earlier, multiple addresses may map to the same predictor entry. This is more likely to take place in smaller predictors. As such, we expect higher accuracy for larger value predictors.

Figure 4 shows predictor accuracy for different predictor sizes. Bars from left to right report for value predictor with 64, 256, 1024, 4096, and unlimited entries respectively. A 64-entry value predictor has the least accuracy. The average locality for 64, 256, 1024, and 4096 entries are 21%, 39%, 49% and 54% respectively.

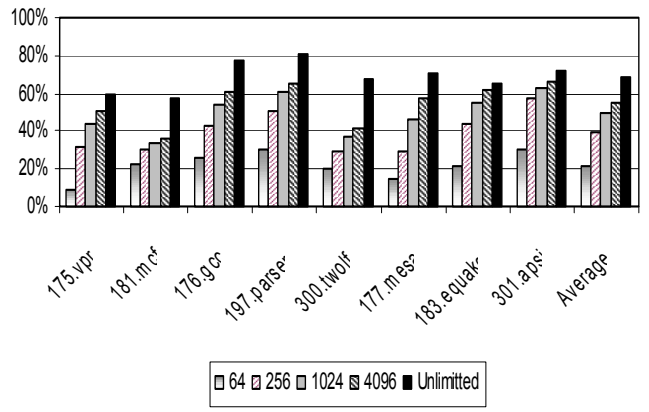

Figure 4. Locality for different value predictor size

4. Implementation and Simulation Results

4.1. Implementation:

The proposed predictor based data bus (PB) architecture was implemented using the 70nm BPTM technology [20],[21]. The last value predictor, explained in section 3.2, was custom designed with 1024 entries for a 32-bit data bus.

The bus length was varied between 1500 and 9000 µm. The number and the size of repeaters on the bus interconnects were optimized using the HSPICE optimizer. A common bus architecture (CB) was also implemented in the same technology, and for a fair comparison it was optimized for the same cycle time of the PB bus . Note that as the common bus does not have the overhead of the predictor, there is more available time for the data transfer on the bus. Thus, for the same cycle time, the repeaters of the common bus have smaller sizes compared to the predictor based data bus.

4.2. Methodology

In this Section, we report our analysis framework. We used SPEC CPU2000 suite benchmarks compiled for the MIPS-like PISA architecture used by the Simplescalar v3.0 simulation tool set [22].

HSPICE simulations of the overall predictor based data bus system are extremely slow. Thus, simulating a long stream of input vectors will be very time consuming. Instead, we simulate the overall bus with a certain set of vectors from which the energy dissipation of the different subsystems, for the different line's switching scenarios, are captured. The energy dissipation values are then stored in a lookup table. Lookup tables are generated for each bus length. The input vectors for the different SPEC CPU2000 benchmarks and a VHDL description of the predictor based data bus are supplied to ModelSim, which generates the datastream transmitted on the bus. Using the energy lookup tables, generated by HSPICE, and the datastream sent on the bus, MATLAB estimates the energy dissipation of the whole system. This process is shown in Figure 5.

4.3. Simulation Results

The simulation methodology described in section 4.2 has been applied for the common and the predictor based data bus architectures. The process was repeated for each bus length, and the energy dissipation (dynamic and leakage) of the overall bus architecture was recorded. This overall energy dissipation includes the energy dissipation of the predictor, the data bus interconnects, the driving circuitry and repeaters.

As the energy dissipation of the bus is proportional to the bus length, there exists a certain bus length at which the overhead energy penalty is exactly compensated by the energy savings due to the reduced switching of PB. Buses with lengths higher than this critical length will have lower energy dissipation compared to the common bus CB. This critical length depends on the switching statistics of the input datastream and how well the predictor works with it as shown in Figure 6. Taking the average of the benchmark energy dissipation curves in Figure 6, the PB bus starts saving energy compared to CB bus for bus lengths greater than 1500 µm, and the percentage average energy savings saturates at about 31%.

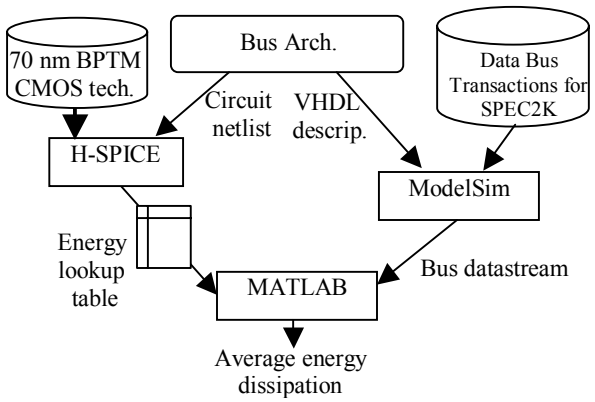

Figure 5. Simulation Methodology

Note that results presented in Figure 6 are consistent with those reported earlier in Figure 4. For example, as reported in Figure 4, *301.apsi* has the highest data bus value locality.

Similarly, *301.apsi* shows the lowest bus power dissipation in Figure 6. Similar trends are observed for other benchmarks.

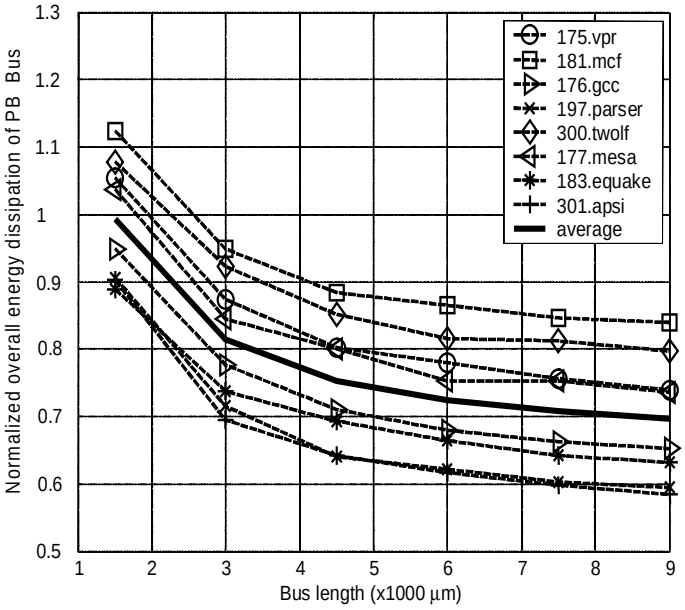

Figure 6. Energy Dissipation of the proposed predication based bus PB architecture relative to the common bus CB architecture

5. Conclusion

The energy dissipation of on-chip buses is becoming one of the main bottlenecks in current integrated circuits. This paper proposes a prediction-based technique to reduce data bus power dissipation. This technique uses value prediction to speculate the next value to transfer over the data bus. Two identical predictors are placed on the two ends of the bus. If the sender predictor accurately predicts the data to be transferred, data is not transmitted and the bus energy dissipation is reduced. After implementing the proposed architecture in a 70nm CMOS technology, and using a representative subset of SPEC2K benchmarks, an average of 41% of the bus transitions were eliminated, which led to a 31% average overall energy reduction.

References

[1] D. L. Liu and C. Svensson, "Power consumption estimation in CMOS VLSI chips", *IEEE Journal SSC*, volume 29, issue 6, pp. 663–670, June 1994.

[2] Semiconductor Industry Association, *International Technology Roadmap for Semiconductors*, 2003.

[3] P. Saxena and C. Liu, "A Postprocessing Algorithm for Crosstalk-Driven Wire Perturbation", *IEEE Transaction on CAD*, pp. 691-702, June 2000.

[4] L. Macchiarulo, E. Macii, and M. Poncino, "Wire placement for crosstalk energy minimization in address buses", *Proc. of DATE'02*, pp.158-162, Mar 2002.

[5] J. Cong, L. He, C.-K. Koh, and Z. Pan, "Interconnect Sizing and Spacing with Consideration of Coupling Capacitance," *IEEE Trans.on CAD*, vol. 20, no. 9, pp. 1164-1169, Sep. 2001.

[6] E. Macii, M. Poncino, and S. Salerno, "Combining Wire Swapping and Spacing for Low-Power Deep-Submicron Buses," *Proceedings of the IEEE Great Lakes Symposium on VLSI*, pp. 198-202, April 2003.

[7] P. Gupta and A. Kahng, "Wire Swizzling to Reduce Delay Uncertainty Due to Cap. Coupling," *Proc. of the International Conference on VLSI Design*, Jan. 2004.

[8] Y. Shin and T. Sakurai, "Coupling-Driven Bus Design for Low-Power Application-Specific Systems," *Proceedings of DAC'01*, pp. 750–753, June 2001.

[9] A. B. Kahng et al., "Interconnect Tuning Strategies for High Performance ICs", *Proceedings of DATE*, pp.471-478, 1998.

[10] K. Hirose and H. Yassura, "A bus delay reduction technique considering crosstalk," *Proceedings of DATE'00*, pp. 441-445, 2000.

[11] M. R. Stan. and Burleson,W. P., "*Bus-invert coding for lowpowerI/O*," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, pages 49–58, Vol. 3, 1995.

[12] L. Benini, A. Macii, E. Macii, M. Poncino, and R. Scarsi, "*Synthesis of Low-Overhead Interfaces for Power-Efficient Communication Over Wide Buses*," ACM/IEEE Design Automation Conference, pages 128–133, 1999.

[13] T. Givargis, D. Eppstein, "Reference Caching *Using Unit Distance Redundant Codes for Activity Redcution on Address Buses",* International Workshop on Embedded System Hardware/Software Codesign,September 2002.

[14] L. Benini, G. De Micheli, E. Macii, D. Sciuto, and C. Silvano, *"Asymptotic zero-transition activity encoding for address buses in low-power microprocessor-based systems,"* Great Lakes VLSI Symposium, pp. 77-82 Urbana IL, March 13-15, 1997.

[15] J. Yang, R. Gupta, "*FV Encoding for Low-Power Data I/O*," ACM/IEEE International Symposium on Low Power Electronic Design, pages 84–87, 2001.

[16] K. Basu, A. Choudhary, J. Pisharath, M. Kandemir , "Power Protocol: Reducing Power Dissipation on Off-Chip Data Buses," 35th Annual IEEE/ACM International Symp. on Microarchitecture, Nov. 2002.

[17] M. H. Lipasti and J. P. Shen, "Exceeding the data flow limit via value prediction," In *29th International Symp. On Microarchitectures*, pages 226–237, Dec. 1996.

[18] M. H. Lipasti, C. B. Wilkerson, and J. P. Shen, "Value locality and load value prediction," In *7th International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 138–147, Oct. 96.

[19] Y. Sazeides and J. E. Smith. The predictability of data values. In *30th International Symposium on Microarchitecture*, pages 248–258, Dec. 1997.

[20] Berkley Predictive Technology model: "http://www-device.eecs.berkeley.edu/~ptm".

[21] Y. Cao, T. Sato, D. Sylvester, M. Orshansky, and C. Hu, "New paradigm of predictive MOSFET and interconnect modeling for early circuit design," *Proceedings of CICC*, pp. 201-204, June 2000.

[22] D. Burger, T. M. Austin, and S. Bennett. "Evaluating Future Microprocessors: The SimpleScalar Tool Set". *Technical Report CS-TR-96-1308, University of Wisconsin-Madison,* July 1999.