A Low-Energy Adaptive Bus Coding Scheme

Benjamin Bishop Anil Bahuman[1]
Department of Computer Science
{bishop@cs.uga.edu, abahuman@ai.uga.edu}
University of Georgia
Athens, GA 30602

Abstract

We have extracted run-time memory access traces from the Mediabench benchmark set. These traces exhibit a high degree of repetition. We propose an adaptive bus coding scheme that will reduce transition activity by exploiting value repetition. For this scheme, we introduce an extra bitline similar to bus-invert coding.

1. Background

A considerable amount of energy is consumed in on-chip interconnect and I/O pads driving off-chip busses. A number of techniques have been proposed to reduce this energy consumption.

By reducing the voltage swing of interconnect lines, it is possible to dramatically reduce the energy consumption. However, some of these schemes lack robustness. A number of schemes are reviewed and presented in [8].

It is also possible to apply adiabatic techniques to reduce bus energy consumption. However, these schemes tend to be slow. So, it can be difficult to use them in practice. [1] and [2] are examples of work in this area.

A number of schemes exist to code the value that is to be sent in a form that consumes less energy than the traditional twos complement representation.

- In a signed magnitude representation, the value is separated into an unsigned magnitude portion and a 1 bit sign.

- The Gray code scheme [4] takes advantage of the fact that values in a sequence tend to be near. Gray codes of consecutive values differ by only one bit. This makes them especially useful for address busses.

- In Bus-invert coding [3], redundancy is used to reduce transition activity. An extra bitline is added. If more than half of bitlines are making a transition, the extra line is asserted and the inverse of the value is sent instead. Otherwise, the extra line is not asserted and the value is sent as usual.

It is noted that transition activity does not necessarily indicate energy consumption in [5]. The authors also propose a coding scheme that is tuned to reduce energy lost to interwire capacitance.

In [6], a coding scheme is proposed that uses decorrelation and entropy coding. In addition, the authors also examine an adaptive coding scheme. However, the adaptive scheme is not well described and the authors report mixed results for the data examined.

2. Our Work

We began our work by extracting run-time bus activity. We used the Mediabench multimedia benchmark set [7] since this is very representative of mobile applications. The bus activity was generated by modifying the SimpleScalar simulator [9]. The simulator also had to be modified to ignore unimplemented system calls. In addition, the keyring for PGP had to be generated by a host compiled version.

In the bus traces generated this way, we observed a great deal of bus value repetition. Figure 1 shows the extent of this repetition averaged over all Mediabench benchmarks. Look Up Table (LUT) size refers to the number of previous transaction values that were stored. The *hit probability* is calculated as the proportion of matches [with a value stored in the LUT] over all transactions. Unique indicates that the LUT does not store repeated values (as in non-unique). It is possible to achieve a value hit rate of 50% with a LUT as small as 16 entries.

Figures 2 through 4 show results for individual benchmarks. A total of 21 benchmarks were used to generate bus

[1]Anil Bahuman is affiliated with the Artificial Intelligence center at UGA

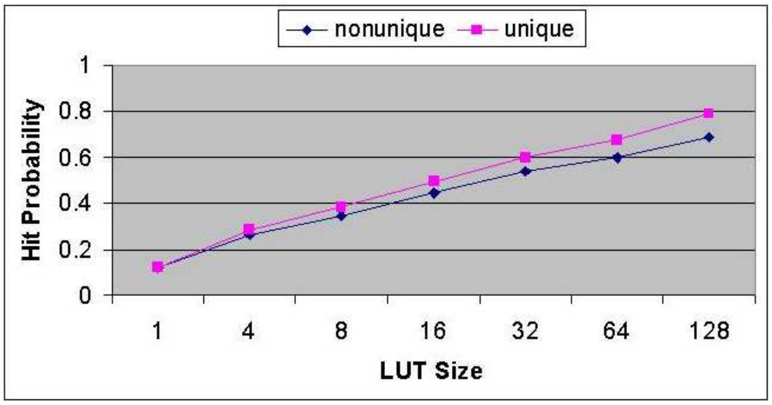

Figure 1. Performance of Unique vs. Non-Unique LUT Storage

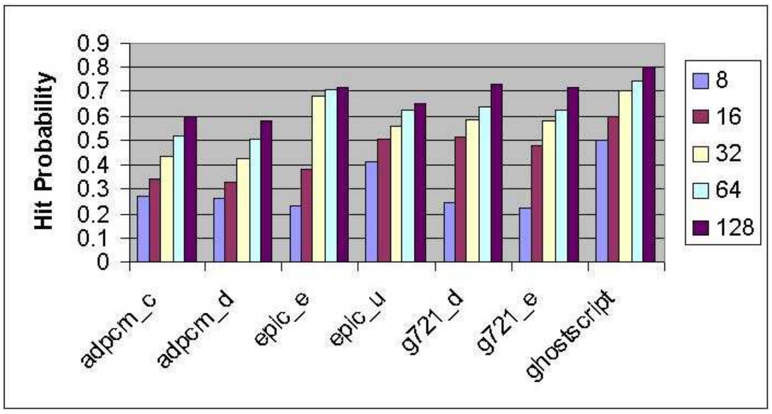

Figure 2. Value Hit Rate for Various LUT Sizes

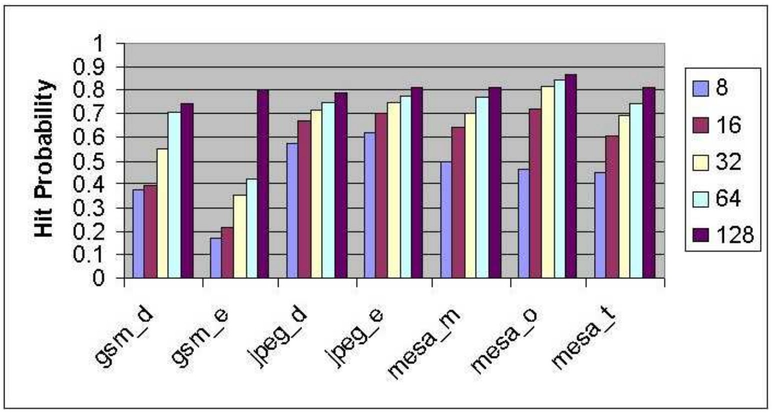

Figure 3. Value Hit Rate for Various LUT Sizes (cont.)

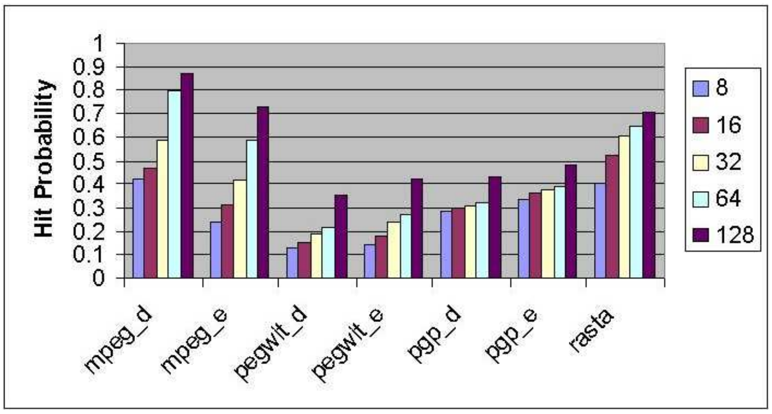

Figure 4. Value Hit Rate for Various LUT Sizes (cont.)

data traces for our simulator. These benchmarks are a subset of the MediaBench benchmark set [7].

In both the Unique and Non-Unique LUT replacement strategies, the entry with the most time since it has been referenced is replaced. We believe that this replacement strategy can be implemented efficiently in hardware.

We propose an adaptive bus coding scheme that will reduce transition activity by exploiting value repetition. For this scheme, we introduce similar LUTs at both sending and receiving ends that store the previous n values on the bus and an extra bitline similar to bus-invert coding.

2.1. Hardware Description

Figure 5 shows a block diagram of the proposed coding hardware. Both the sender and the receiver are equipped with similar LUTs. At the sending end, a look up is performed on the sender value. If it is present, the address is found and the hit flag is set. If the hit flag is set, the mux passes the one hot coding of the address XOR the previous bus value. Otherwise, the mux passes the sender value.

At the receiving end, if the hit flag is asserted, the bus value XOR the previous bus value is used to find the one hot coding of the LUT address. This is then used to look up the receiver value. If the hit flag is not set, the mux selects the bus value as the receiver value.

At the end of the transaction, the sender adds the sender value to its LUT and the receiver adds the receiver value to its LUT. The proposed coding scheme guarantees that the sender and receiver values are identical.

3. Results

In order to evaluate the merit of the proposed scheme, software was developed to compare transition activity for various coding schemes. This software is available for review at [10].

The transition activity for all coding schemes studied is presented in Figure 6. In addition to the adaptive coding scheme for twos complement numbers (AD-2C), we also considered applying the adaptive scheme to Gray coded values (AD-GC).

4. Conclusions

We have proposed an adaptive coding scheme that has proven to be very effective on data extracted from realistic applications. In fact, our adaptive scheme beats more traditional approaches with just an 8 word look up table. Further, our scheme is simple to understand and implement.

5. Future Work

In the future we would like to extend our work by:

- Incorporating the inter-wire capacitance model described by [5].
- Considering the effect of caching on post cache busses.
- Evaluating the energy overhead of the coding/decoding logic using circuit simulation.
- Using traces that reflect the memory access patterns of the operating system.
- Considering ways to extend the coding scheme to be able to express values that are not in the LUT but differ by only a few bits.

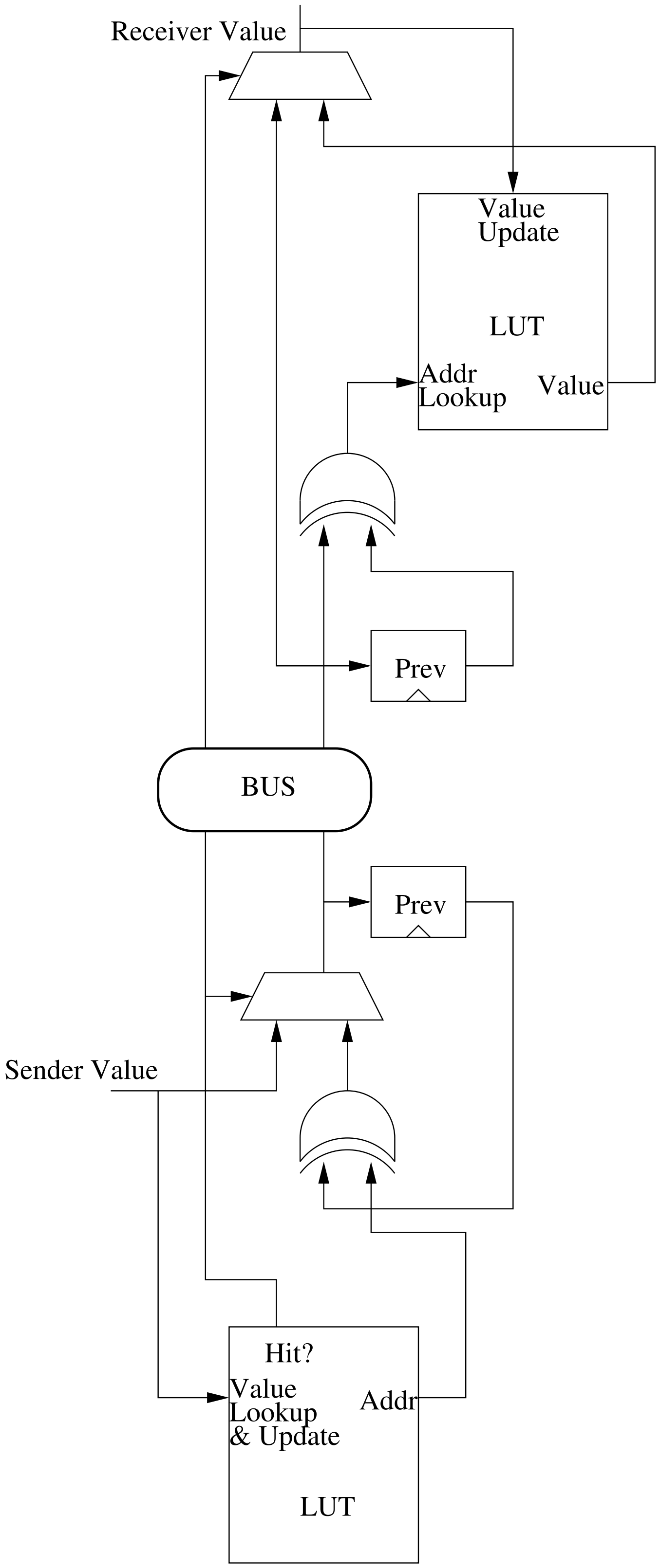

Figure 5. Block diagram of the coding hardware

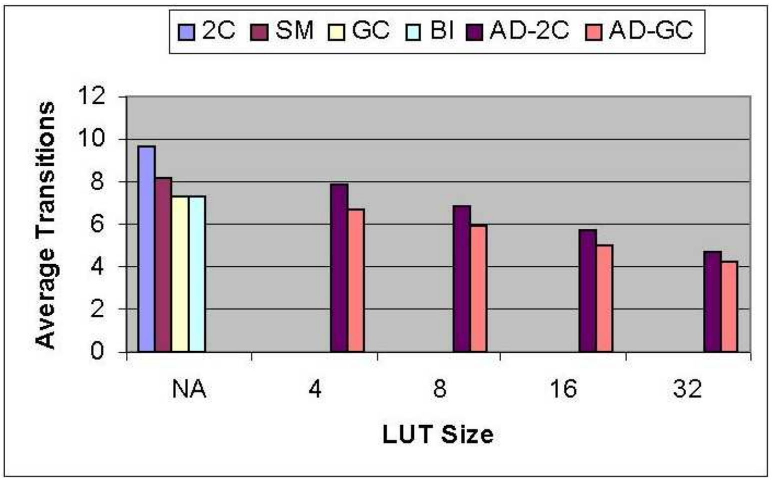

Figure 6. Average transition count for various coding schemes

References

[1] K. Khoo, A. Willson, "Charge Recovery on a Databus", *Proceedings of International Symposium on Low Power Electronics and Design*, pp. 185-189, 1995.

[2] B. Bishop, M.J. Irwin, "Databus Charge Recovery: Practical Considerations", *Proceedings of International Symposium on Low Power Electronics and Design*, pp. 85-87, 1999.

[3] M. Stan, W. Burleson, "Bus-Invert Coding for Low-Power I/O", *IEEE Transactions on VLSI Systems*, vol. 3, no. 1, pp. 49-58, March 1995.

[4] F. Gray "Pulse Code Communication", U.S. Patent 2,632,058, March. 1953.

[5] P. Sotiriadis, A. Chandrakasan, "Low-Power Coding Techniques Considering Inter-wire Capacitances", *Proceedings of the IEEE Custom Integrated Circuits Conference*, pp. 507-510, 2000.

[6] S. Ramprasad, N. Shanbhag, I. Hajj, "A Coding Framework for Low-Power Address and Data Busses", *IEEE Transactions on VLSI Systems*, vol. 7, no. 2, pp. 212-221, June 1999.

[7] C. Lee, M. Potkonjak, W. Mangione-Smith, "Mediabench: A Tool for Evaluating and Synthesizing Multimedia and Communications Systems", *Proceedings of the 30th Annual International Symposium on Microarchitecture*, pp. 330-335, 1997.

[8] Hui Zhang, Jan Rabaey, "Low-Swing Interconnect Interface Circuits", *Proceedings of International Symposium on Low Power Electronics and Design*, pp. 161-166, 1998.

[9] T. Austin, D. Burger, "The SimpleScalar Architectural Research Tool Set, Version 2.0", http://www.cs.wisc.edu/˜mscalar/simplescalar.html

[10] http://www.cs.uga.edu/˜bishop/munch.tar.gz