Power-Efficient Memory Bus Encoding Using Stride-Based Stream Reconstruction

Kuei-Chung Chang, Tsung-Ming Hsieh and Tien-Fu Chen
Department of Computer Science and Inofrmation Engineering
National Chung Cheng University
Chiayi, Taiwan 621, ROC
ckj@cs.ccu.edu.tw

Abstract

With the rapid increase in the complexity of chips and the popularity of portable devices, the performance demand is not any more the only important constraint in the embedded system. Instead, energy consumption has become one of the main design issues for contemporary embedded systems, especially for I/O interface due to the high capacitance of bus transition. In this paper, we propose a bus encoding scheme, which may reduce transitions by reconstructing active address streams with variable cached strides. The key idea is to obtain the variable strides for different sets of active addressing streams such that the decoder reconstructs these interlaced streams with these strides. Instead of sending the full address, the encoder may only send partial address or stride by using either one-hot or binary-inversion encoding. To exploit the locality and dynamically adjust the value of stride of active address streams, we partially compare the previous addresses of existing streams with the current address. Hence, the data transmitted on the bus can be minimally encoded. Experiments with several MediaBench benchmarks show that the scheme can achieve an average of 60% reduction in bus switching activity.

Keywords: Bus Encoding, Low Power, Interconnection, SOC

1 INTRODUCTION

As designers try to integrate multimedia and communication applications on a system chip, the design issues of power dissipation has become as important as that of area and speed. In today's processors, a large number of input/output pins are dedicated to interface the processor core to the external memory through high-speed address and data buses. The portion of input/output (I/O) pads on a system chip consumes significant energy of an overall system [18], because the embedded processor has much fewer transistors integrated on the chip than a general-purpose high-performance processor. The complexity and the physical length of bus systems will lead to an increased contribution to the total power consumption of a SOC.

As submicrometer technology matures, interwire parasitic capacitance is no longer negligible and even becomes a major issue. The closer geometrical proximity of adjacent bus lines will form a parasitic capacitance between them. This effect not only leads to crosstalk and delay effects, it also leads to an increased power consumption since the parasitic capacitance is charged and discharged when there is a voltage swing between two or more bus lines. Consequently, lowering down the switching activity impacts the reduction of the overall system power consumption significantly.

There are several ways to reduce the problem of interwire capacitances, including widen the distance between bus lines, using place & route (P&R) tools to avoid side-by-side routing of bus lines, change the geometrical shape of bus lines, and bus encoding techniques. Although many encoding techniques for instruction-address buses have been reported, there are not as many encoding methods for data address or multiplexed address buses. In the case of instruction address bus encoding, a high correlation between consecutive addresses is exploited. In the case of data addresses, there is much less correlation between consecutive data addresses, and the offsets are usually much larger. It is difficult to reduce the transitions on a data address by bus encoding.

For multiplexed address buses, compared to data only, there is more correlation between addresses because of the presence of the instruction address; thus, more reduction in activity can potentially be obtained when compared to the data address. However, the presence of two different address streams (i.e., instruction and data ad-

dresses) with different characteristics makes the encoding even more complex.

In this paper, we focus on run-time hardware-based bus encoding technique by dynamically reconstructing address streams with variable cached strides to reduce the address bus transition. It can adjust stride values according to different application characteristics. The key idea is that the variable strides for different sets of active addressing streams are recorded, and the decoder reconstructs these interlaced streams with those strides obtained earlier. Instead of sending the full address, the encoder may only send partial addresses or strides by using either one-hot or binary-inversion encoding. There is a lot of locality in the memory access patterns of digital signal processing (DSP) applications compared to general applications. It can predict the upcoming address references by exploiting the property of consecutive address references. The experimental results show that our proposed encoding scheme can decrease the switching activity of the multiplexed address bus by an average of 60%.

The rest of the paper is organized as follows: Section 2 summarizes the related work. In Section 3, we describe the bus encoding scheme using stream reconstruction. Section 4 gives the experimental results and discusses the implementation issues and the overhead. Finally, we conclude in Section 5.

2 RELATED WORK

There are several kinds of encoding schemes proposed recently. Bus-invert coding [18], a hardware-based encoding, inverts the transmitted data if the transitions are larger than half of the bus width. It selects between the original and the inverted patterns in a way that minimizes the switching activity on the bus. This technique is quite effective for reducing the number of ones in addresses with random behavior, but it is ineffective when addresses exhibit some degree of locality.

Working zone [13, 14] only sends the reference of working zone address and offset. When a new address arrives, the offset of the address is calculated with respect to all zone registers. The address is thus mapped to the working zone with the smallest offset. The working-zone method uses one extra line to show whether encoding has been done or the original value has been sent. A stride is a constant offset that occurs between multiple consecutive addresses repeatedly and if detected, can be used to completely eliminate the switching activity for such addresses. The working-zone method has a large area and power dissipation overhead due to the complexity of the decoder and encoder logic. In addition, it can completely be ineffective with some address traces. Consider a data-address bus where address offsets are not small enough to be mapped to one-hot code; in such a case, the original address is sent over the bus, which usually causes many transitions on the bus.

In [11, 12], Mamidipaka et al. proposed an encoding technique based on the notion of self-organizing lists. They use a list to create a one-to-one mapping between addresses and codes. For multiplexed address buses, they used a combination of their method and INC-XOR [16]. In INC-XOR, which is proven to be quite effective on instruction address buses, each address is XORed with the summation of the previous address and the stride; the result is then transition-signaled over the bus. The size of the list in this method has a significant impact on the performance. To achieve satisfactory results, it is necessary to use a long list. However, the large hardware overhead associated with maintaining long lists makes this technique quite expensive. Furthermore, the encoder and the decoder hardware are practically complex and their power consumption appears to be quite large.

In [3], it partitions the source-word space into a number of sectors with unique identifiers called sector heads. These sectors can, for example, correspond to address spaces for the code, heap, and stack segments of one or more application programs. Each source word is then dynamically mapped to the appropriate sector and is encoded with respect to the sector head. In general, the sectors may be determined a priori or can dynamically be updated based on the source word that was last encountered in that sector. These sector-based encoding techniques are quite effective in reducing the number of inter-pattern transitions on the bus, while incurring rather small power and delay overheads.

In [10], it introduces the bus encoding technique, adaptive dictionary-encoding scheme "ADES" that minimizes the power consumption of data buses through a dictionary-based encoding technique. Based on exploration of data properties on buses, it saves on average more than 25% of bus energy compared to the raw cases for both address and data buses.

Reference caching using UDRC [6] exploits the fact that address reference are likely to be made up of an interleaved set of sequential address stream and convey the index of active stream in UDRC format. Static bus encoding decreases the switching activity by profiling the address stream

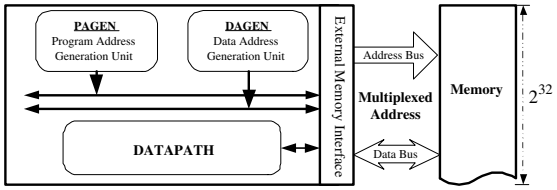

Figure 1: System architecture using multiplexed address bus

and mapping into hardware. Address level bus power optimization (ALBORZ) [1], which calculates the weight of offset and encodes and address level bus power optimization is an example of static bus encoding scheme. BEAM [2] is based on the support of a function unit called "ISA aware unit" that can predict the upcoming address. So, we would not toggle the address bus because we can get the current address from instruction. AMBA is the most popular bus protocol, and there are still some encoding schemes adapted to it. In [17], a modified bus-invert coding scheme is proposed to support AMBA-based SOC platform.

3 THE PROPOSED METHOD

Overview

The target system architecture of this paper is illustrated in Figure 1, where multiple multiplexed address streams are considered in our method. The data and programs are both stored in external memory, and there is only one port for memory address. There are two types of I/O buses; one is the address bus, where the memory references are most correlated, and the other one is the data bus, where bus transaction data sent over the bus are less related. There is a lot of locality in the memory access patterns of digital signal processing (DSP) applications compared to general applications. Take the processor TMS320C54x (a DSP produced by Texas Instruments) for an example, several addressing modes are provided to accelerate sequential memory access. In these memory-intensive applications, energy consumption of buses is significant. Although data addresses may not be sequential, they still follow the principles of spatial and temporal locality.

As shown in Figure 2, there are three common active address streams, including one instruction stream and two data streams. The accesses of instruction stream and data arrays are regular with constant strides. Based on the above observation, we propose an encoding mechanism,

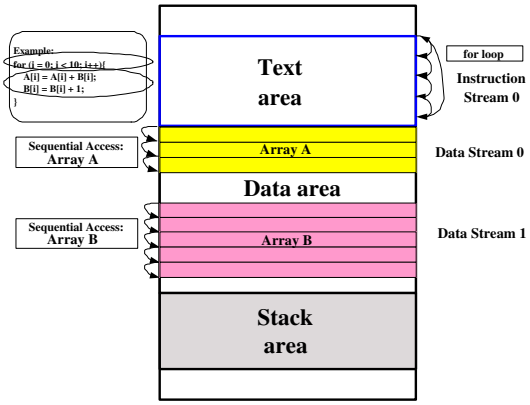

Figure 2: Memory allocation and three address streams

named *stride-based stream reconstruction encoding (SBSRE)*, to take advantages of consecutive property of memory references, and the encoding mechanism reconstructs the interlaced sets of instruction and data address streams with variable strides. In addition, we consider a finite state machine to dynamically adjust the value of strides in order to exploit the locality and regulate these address streams.

An encoding scheme for low power buses generally consists of a predictor, a decorrelator, a selector and its corresponding decoder [8]. The predictor generates a prediction of the current value of the received input based on the past value. Subsequently, the decorrelation phase is used to encode the results from predictor and produce the codeword for reducing the switching activity. Our prediction scheme consists of a *predicted address table (PAT)*, two *previous reference address tables (PRAT)*, and two *stride tables (ST)*, as shown in Figure 3. The *PAT* is used to keep the next predicted addresses, the PRAT is used to keep the previous referenced addresses, and the *ST* is used to keep the offset between two consequence accesses for next prediction. We may predict the value of upcoming address of a certain stream based on a stride by taking advantages of the sequential property of memory references. Because we store those strides in a stride table, the encoder may only send the index of the *PAT* or the stride value of the address stream over the address bus by different encoding formats according to different prediction results.

To make this technique effective, the total number of the frequently occurring stride patterns should be much smaller than the number

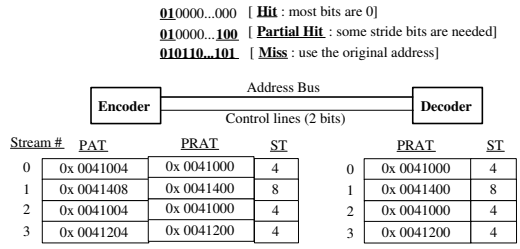

Figure 3: The proposed mechanism

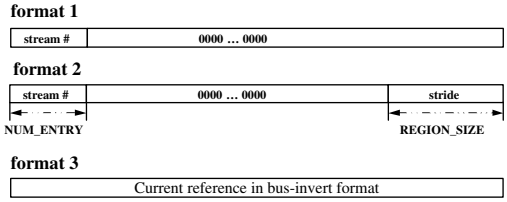

Figure 4: Encoding formats

of all possible patterns and a stride pattern in the source should have a very high probability to be in the ST. In other words, the source must have a small amount of patterns that cover most part of the address word. As our experiments indicate, high-frequency stride patterns make $SBSRE$ technique effective for bus encoding. It is not hard to understand that our proposed method technique can take advantage of temporal locality. For this reason, in some sense, the widely used cache in memory systems can be seen as a special case of a $SBSRE$ technique application.

Endocing and Decoding Algorithms

The block diagram of the proposed mechanisms is shown in Figure 3. The encoder contains a PAT, a $PRAT$, and a ST. The decoder contains a $PRAT$ and a ST. Initially, the contents of the tables are all set to zero. When an address word is sent the current address and the addresses stored in the PAT will be compared. The compared results are introduced as follows.

1. *Exactly Hit (EH).* If the current input address hits an entry in the PAT, the simplest *format 1* shown in Figure 4 can be sent over the bus by one-hot encoding. It will only transmit the index of the table to the decoder, and much fewer ones will be transmitted. The control lines will send the signal indicating a hit. Using the index part received, the decoder can recover the referenced address by adding the address in the $PRAT$ and the corresponding stride value in the ST.

2. *Partially Hit (PH).* If the current input address partially hits (within the near regions) an entry in the PAT, a stride is obtained by subtracting the current address from the last corresponding reference address in $PRAT$. If the current stride is not equal to the stride in the corresponding entry of the ST, the current stride for the corresponding address stream has to be dynamically adjusted. In this case, both the index of the active stream in the PAT and its new stride value are required to be sent over the bus by *format 2* shown in Figure 4. The control lines will send the signal indicating a partial hit. In this case, some bits transmitting the actual stride values will be needed.

   To avoid unnecessary glitch for an array stream with a regular stride, we further propose a *stride-regulation policy* as shown in Figure 5 to keep the maximum possibility of the stride re-usage. In the finite state machine, *'EH'* and *'PH'* means *exact hit* and *partial hit* respectively. We would update the stride value when a partial hit occurs. However, when the state changes from the *'Firm'* state to the *'Init'* state, the stride value will be kept unchanged to avoid a small portion of irregular strides. An example is that the program counter sometimes has an unexpected large offset caused by the *'branch'* instruction in an instruction stream. For example, in Figure 6, after the first iteration of the inner loop, the active stream of *array b* becomes *'Firm'* state with a stride value of *'0x10'*. When the program steps into next iteration of the inner loop, the memory reference of *array b* changes from *0x00410048* to *0x00410000*, where the stride becomes *0x48*. To prevent such an irregular update glitch, we do not update the stride value.

3. *Miss Predicted.* If any of the addresses in the PAT does not fully match or partially match with the current address, a miss occurs. In this case, we cannot reconstruct the current address according to the corresponding information stored in the decoder on the receiver side. To start a new address stream, we need to transmit the current address by *format 3* shown in Figure 4. At the same

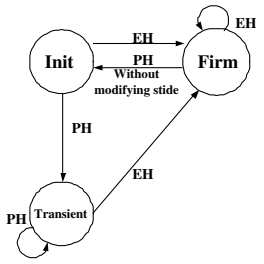

Figure 5: The finite state diagram of the proposed stride-regulated policy

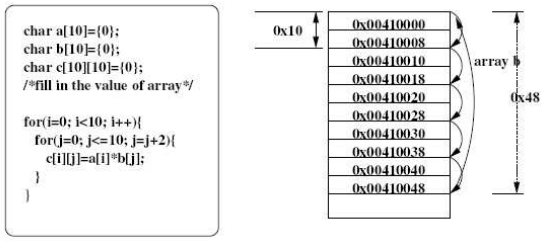

Figure 6: Stride-regulated policy example

time, a default stride value (8 or 4) is assigned to the address stream. The control lines will send the signal indicating a miss.

Control Signals

We can observe that there are many zero bits in format 1 and format 2 as shown in Figure 4. In this case, the one-hot encoding can be employed to encode the stream number, and the stride value is transmitted in binary format. Also, exclusive-or operations are applied to format 1 and format 2 in order to further reduce the switching activity. On a miss, the current address is encoded by using bus-invert format. In addition to n bits of the original address bus, additional two control lines are required (*ctrl_info*) to notify the receiver side. We explain the corresponding meanings of these signals as follows.

- *EH*. The upcoming address is exactly hit, and the index of the active stream is sent over the bus by encoding format 1 with one-hot encoding.
- *PH*. The upcoming address is partially predicted. In other words, the address locates at a reasonable region, and we need to transmit both the index of the PAT and the actual stride value by encoding format 2.
- *BI_P0*. The current address is not pointed through any active stream in the PAT, and the value of polarity bit is '0'. They will be sent by format 3.
- *BI_P1*. The current address is not pointed through any active stream in the PAT, and the value of polarity bit is '1'. They will be sent by format 3.

Design Choice of Comparison Region Size

As discussed in the previous section, the switching activity due to the encoding format 2 is noticeable. If we enlarge the size of comparison region of the stride in format 2 in Figure 4, we would have more misses caused by irregular accesses. Enlarging the comparison region would decrease the exact hit rate. However, it would increase the partial predicted rate and introduce relatively less switching activity than bus-invert format. It is helpful to find a reasonable comparison region that can efficiently reduce the switching activity with slightly lowering the exact hit rate, which would be discussed in our experiments.

4 EXPERIMENTAL RESULTS

We implemented our proposed encoding scheme by SimpleScalar [4], and simulated the traces of MediaBench benchmarks to obtain the total switching activity. The address bus width in SimpleScalar is 32-bit, and the instruction word occupies 64-bit width. Our set of benchmark is composed of six applications from MediaBench. In experiments, we first show the occurrence frequency of each encoding format and the corresponding effect on total switching activity. In experiments, the number of maximum active address streams in the PAT is four with the least recently used (LRU) replacement policy [5]. Then, we will show the comparison results for different entries of the PAT. In addition to the proposed method, we have implemented bus-invert [18], asymptotic zero transition [9], working zone [13, 14], reference caching with UDRC [6] for comparison purposes. We also simulate the power savings of our proposed optimization mechanism. Finally, we discuss the implementation issues and the overhead of our proposed method.

The effect of the comparison region size

In Figure 7, the left-hand side shows the occurrence frequencies of 'EH', 'PH', and 'miss' with different region sizes. The right-hand side shows the corresponding effect on total switching activity with different region sizes. From the results of these figures we can find that there is a trade-off between the size of comparison region (stride

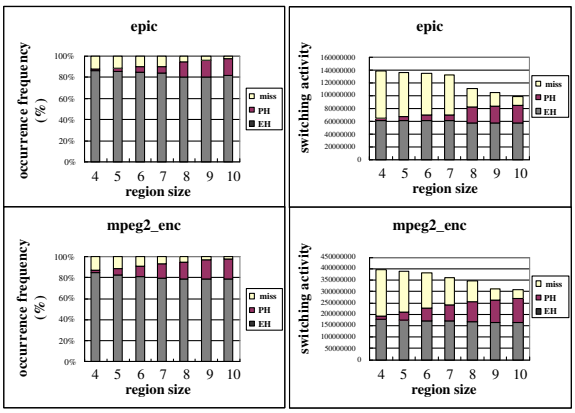

Figure 7: The occurrence frequency and the switching activity with different region sizes

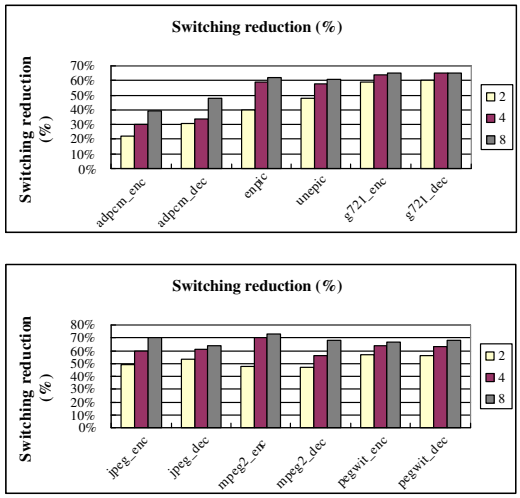

Figure 8: Comparison with different entry numbers of the table

bits) and the switching activity. If we increase the size of comparison region, the exact hit (EH) rate will be slightly decreased due to the irregular accesses. In addition, the partial hit (PH) rate will be significantly increased when the size of comparison region is increased. In this situation, part of the miss rate will be replaced by the partial hit rate. Instead of transmitting the encoding format in miss, we transmit the same data in partial hit (PH) format due to the larger region size, and the switching activities can be decreased. However, we cannot unlimitedly increase the region size due to the heavy hardware implementation cost. Therefore, finding a balanced point that achieves good savings in switching activity with slightly decreasing the occurrence frequency of exact hit (EH) is important. Based on our experimental results, eight bits would be a good choice of region size to achieve good results.

The effect of different entry numbers of the table

There is a trade-off between the design cost and switching reduction. As shown in Figure 8, the switching reduction with two entries in the PAT is about 50%. The switching reduction with four table entries can achieve about 60%. If we allow at most eight active address streams in each instant of time, we can get 65% switching reduction. Based on these results we can find that more entries of the table, more switching reduction can be achieved. However, the switching reduction with four entries may be a good choice for implementation because of the lower hardware cost.

Comparison with other encoding schemes

In this section, several encoding schemes were implemented for comparison, and the results are shown in Figure 9. Bus-invert encoding scheme can restrict the maximum transition to $n/2$ regardless of the locality of transmitted data. In our experimental results, bus-invert can achieve 15% reduction. The approach of Tzero is to exploit the consecutive property. However, there are several active address streams at each instant of time. So, it will miss predicted caused by only one address stream maintained. The average reduction in Tzero is about 15%. The switching activity reduced by working zone is 50%, and the average reduction in reference caching is about 25%. On average, the switching activity of our proposed approach can achieve 60%. In this experiment, the maximum number of active address stream is four, and the comparison region size is eight bits.

5 CONCLUSION

Interwire parasitic capacitance is no longer negligible and even becomes a major issue. The closer geometrical proximity of adjacent bus lines will form a parasitic capacitance between them. This effect not only leads to crosstalk and delay effects, it also leads to an increased power consumption since the parasitic capacitance is charged and discharged when there is a voltage swing between two or more bus lines. In this paper, we propose a bus encoding scheme for reducing the switching activity on address buses. It predicts the

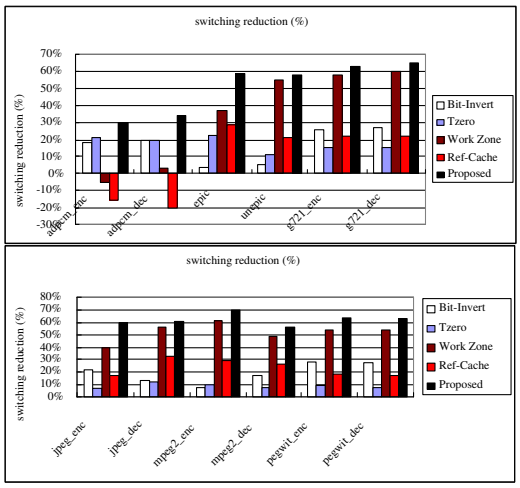

Figure 9: Comparison of different encoding schemes

upcoming reference address by run-time reconstructing address streams with previous cached strides. Based on the prediction results, we can efficiently achieve about 60% reduction of switching activity.

References

[1] Y. Aghaghiri, F. Fallah, and M. Pedram. ALBORZ: Address Level Bus Power Optimization. In *Proceedings of the International Symposium on Quality Electronic Design*, pages 470–475, 2002.

[2] Y. Aghaghiri, F. Fallah, and M. Pedram. BEAM: bus encoding based on instruction-sef-aware memories. In *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 3–8, 2003.

[3] Y. Aghaghiri, F. Fallah, and M. Pedram. Transition reduction in memory buses using sector-based encoding techniques. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 23(8):1164–1174, 2004.

[4] Doug Burger and Todd M. Austin. The simplescalar tool set, version 2.0. http://www.simplescalar.com, June 1997.

[5] P. J. Denning. The working set model for program behavior. *Communications of the ACM*, 11(5):323–333, 1968.

[6] D. Eppstein and T. D. Givargis. Reference caching using unit distance redundant codes for activity reduction on address buses. In *Proceedings of the International Workshop on Embedded System Codesign (ESCODES '02)*, pages 43–48, 2002.

[7] K. Flautner, N. S. Kim, S. Martin, D. Blaauw, and T. Mudge. Drowsy caches: simple techniques for reducing leakage power. In *Proceedings of the International Symposium on Computer Architecture*, pages 219–230, 2002.

[8] W. Fornaciari, M. Polentarutti, D. Sciuto, and C. Silvano. Power optimization of system-level address buses based on software profiling. In *Proceedings of the Eighth International Workshop on Hardware/Software Codesign*, pages 29–33, 2000.

[9] E. Macii D. Sciuto L. Benini, G. De Micheli and C. Silvano. Asymptotic zero-transition activity encoding for address busses in low-power microprocessor-based systems. In *Proceedings of the 13th Annual Intl. Symp. on Computer Architecture*, 1986.

[10] T. Lv, J. Henkel, H. Lekatsas, and W. Wolf. A dictionary-based en/decoding scheme for low-power data buses. *IEEE Transactions on Very Large Scale Integration Systems*, 11(5):943–951, 2003.

[11] M. Mamidipaka, D. Hirschberg, and N. Dutt. Low power address encoding using self-organizing lists. In *Proceedings of the International Symposium on Low Power Electronics and Design*, pages 188–193, 2001.

[12] M. Mamidipaka, D. Hirschberg, and N. Dutt. Adaptive low-power address encoding techniques using self-organizing lists. *IEEE Transaction on Very Large Scale Integration Systems*, 11(5):827–834, 2003.

[13] E. Musoll, T. Lang, and J. Cortadella. Exploiting the locality of memory references to reduce the address bus energy. In *Proceedings of International Symposium on Low power Electronics and Design*, pages 202–207, 1997.

[14] E. Musoll, T. Lang, and J. Cortadella. Working-zone encoding for reduce energy in microprocessor address buses. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 6(4):568–572, 1998.

[15] M. Powell, S.-H. Yang, B. Falsafi, K. Roy, and T. N. Vijaykumar. Gated-vdd: a circuit technique to reduce leakage in deep-submicron cache memories. In *Proceedings of the International Symposium on Low Power Electronics and Design*, pages 90–95, July 2000.

[16] S. Ramprasad, N.R. Shanbhag, and I.N. Hajj. A coding framework for low-power address and data busses. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 7(2):212–221, 1999.

[17] T. Arslan S. Osborne, A.T. Erdogan and D. Robinson. Bus encoding architecture for low-power implementation of an AMBA-based SoC platform. *IEE Proceedings-Computers and Digital Techniques*, 149(4):152–156, 2002.

[18] M. R. Stan and W. P. Burleson. Bus-Invert coding for low-power I/O. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 3(1):49–58, 1995.

Received: Apr. 2006. Accepted: Mar. 2007.