On-Going Research on Address Bus Encoding for Low Power: A Status Report

L. Benini [$] G. De Micheli [$] E. Macii [*] M. Poncino S. Quer D. Sciuto [‡] C. Silvano [#]

[‡] Politecnico di Milano
Dip. di Elettronica e Informazione
Milano, ITALY 20133

[*] Politecnico di Torino
Dip. di Automatica e Informatica
Torino, ITALY 10129

[$] Stanford University
Computer Systems Laboratory
Stanford, CA 94305

[#] Università di Brescia
Dip. di Elettronica per l'Automazione
Brescia, ITALY 25123

Abstract

This paper describes our research on bus encoding for low power dissipation. We present two encoding schemes, namely, the T0 code and the Beach code, that sensibly reduce the switching activity on address busses. The T0 code exploits the high sequentiality of the streams traveling on the address busses of general purpose, microprocessor-based systems. The Beach code, on the other hand, targets the reduction of the number of transitions on the address bus lines of special purpose systems, where streams usually have reduced sequentiality, and exploits the concept of block correlation that may exist between patterns being transmitted over the communication channel.

1 Introduction and Motivation

It is well known that the intrinsic capacitances of system-level busses are usually several orders of magnitude higher than for the internal nodes of a circuit [1]. Consequently, a considerable amount of power is required at the I/O pins of a processor when binary patterns have to be transmitted over the communication channels. Significant power savings can thus be achieved by reducing the number of transitions (i.e., the switching activity) at the processor's I/O interface.

One way of accomplishing this task consists of encoding the information transmitted over the busses. Depending on the type of information to be exchanged, some low-power encoding schemes have been introduced in the recent past.

The *Bus-Invert code* of [2] is a simple, yet effective, low-power encoding scheme. It works as follows: The Hamming distance between two successive patterns is computed; if it is larger than $N/2$, where N is the bus width, the current address is transmitted with inverted polarity; otherwise, it is transmitted as is. Obviously, a redundant bus line is required to signal to the receiving end of the bus which polarity is used for the transmission of the incoming pattern. The method guarantees a maximum of $N/2$ transitions per clock cycle, and it has shown to perform well when patterns to be transmitted are randomly distributed in time and no information about their mutual correlation is available; therefore, it is appropriate for data bus encoding.

Concerning address busses, the well-known fact that the addresses generated by processors in ordinary computing systems are often consecutive has suggested the use of the *Gray code* [3, 4] as encoding strategy. Unfortunately, it has been experimentally observed that, while addresses generated by real microprocessors running general purpose programs are usually characterized by high sequentiality, streams for special purpose software applications (e.g., image processing, matrix calculus) have a much smaller percentage of in-sequence addresses.

The above observations have motivated a deeper investigation of possible bus encoding techniques that may improve the efficiency of the existing approaches. In this paper, we report on the status of our current research in this area.

We first consider the case of microprocessor-based, general purpose systems, and we discuss possible schemes that achieve better performance than the simple Gray code. The encoding mechanisms we propose rely on the idea of avoiding the transfer of consecutive addresses on the bus by using a redundant line, *INC*, to transfer to the receiving sub-system the information on the sequentiality of the addresses. When two addresses in the stream are consecutive, the *INC* line is set to 1, the bus lines are frozen (to avoid unnecessary switchings), and the new address is computed directly by the receiver. On the other hand, when two addresses are not consecutive, the *INC* line is driven to 0 and the bus lines operate normally. The scheme above, called *T0 code* [5], guarantees an asymptotic performance of zero transitions under the hypothesis of infinite streams of consecutive addresses. Several variants of the T0 code are available [6], some of which incorporate the Bus-Invert principle to exploit distinctive spectral characteristics of the streams being transmitted.

We then move to special purpose systems, where the use of codes such as the Gray and the T0 are ineffective, due to the reduced sequentiality of typical address streams. In spite of this, it may still be the case that other types of temporal correlations exist between the patterns that are being transmitted. In particular, we have noted that time-adjacent addresses usually show remarkably high block correlations. Therefore, we propose to exploit such correlations to come up with a scheme, called in the following the *Beach code* [7], which minimizes the average address bus switching activity. Starting from typical traces of the N-bit address bus of the system being designed, we collect statistical information identifying possible block correlations. We then group bus lines in clusters according to their correlations, that is, lines belonging to the same cluster are highly correlated. For each cluster of size k we automatically generate an encoding function, namely, a one-to-one Boolean function $E : B^k \rightarrow B^k$. Each configuration of bits in the original cluster is translated into a new bit configuration. The algorithm which finds function E targets the minimization of the switching activity; thus, well established technology, initially developed for low-power FSM state assignment and re-encoding, can be successfully exploited. The output of the transformation is an encoded stream for which the average number of bus line transitions between two successive k-bit patterns is minimized. At the receiving end of the bus, the original encoding is obviously required. Then, the inverse function, E^{-1}, must also be calculated.

Since the motivation for using a bus encoding scheme is a reduction of the global power consumed by the system as a whole, it is mandatory to guarantee that power savings achieved through a decrease in the bus switching activity are not offset by the extra power dissipated by the encoding and decoding circuitry which is required at the bus terminals. In addition, bus latency is usually a critical design constraint. Therefore, simultaneous power and timing optimization must be targeted during the synthesis of the logic for address encoding/decoding. We have proposed fast and low-power implementations for the encoders and decoders related to the encoding techniques discussed in this paper. However, for space reasons, we do not comment on such implementations here.

2 General Purpose Systems

The Gray code achieves its asymptotic best performance of a single transition per emitted address when infinite streams of consecutive addresses are considered. However, the code is optimum only in the class of irredundant codes, that is, codes that employ exactly N-bit patterns to encode a maximum of 2^N data words. If we allow the addition of some redundancy to the code, better performance can be achieved by adopting the T0 solution, which requires a redundant line, *INC*, to signal with value one that a consecutive stream of addresses is output on the bus. If *INC* is high, all other bus lines are frozen to avoid unnecessary switchings. The new address is computed directly by the receiver. On the other hand, when two addresses are not consecutive, the *INC* line is low and the remaining bus lines are used as standard binary codes for the new addresses.

For infinite streams of consecutive addresses, the T0 code enjoys the zero transition property. Therefore, it outperforms the Gray code since, under the same assumption, Gray addressing requires one line switching per each pair of patterns. In addition, experimental results have shown the superiority of the T0 code even in the more realistic case of streams of consecutive addresses of limited length.

The T0 encoding scheme can be formally described by the following equation:

$$(\mathbf{B}^{(t)}, INC^{(t)}) = \begin{cases} (\mathbf{B}^{(t-1)}, 1) & \text{if } t > 0 \wedge \mathbf{b}^{(t)} = \mathbf{b}^{(t-1)} + \mathbf{S} \\ (\mathbf{b}^{(t)}, 0) & \text{otherwise} \end{cases}$$

where $\mathbf{B}^{(t)}$ is the value on the encoded bus lines at time t, $INC^{(t)}$ is the additional bus line, $\mathbf{b}^{(t)}$ is the address value at time t, and $\mathbf{S}$ is a constant power of 2, called *stride*.

The corresponding decoding scheme can be formally defined as follows:

$$\mathbf{b}^{(t)} = \begin{cases} (\mathbf{b}^{(t-1)} + \mathbf{S}) & \text{if } INC = 1 \wedge t > 0 \\ \mathbf{B}^{(t)} & \text{if } INC = 0 \end{cases}$$

For architectures like the MIPS [8], two streams, α and β, with quite different spectral characteristics, are time-multiplexed on the same address bus. Stream α, corresponding to instruction addresses, has high probability of having two consecutive addresses on the bus in two successive clock cycles; on the contrary, stream β, corresponding to data addresses, has almost no in-sequence patterns. The control signal, *SEL*, available in the standard bus interface to de-multiplex the bus at the receiver side, is asserted when stream α is transmitted; otherwise, *SEL* is de-asserted. An extension of the T0 approach, called *Dual_T0_BI code*, can be effective in these cases. This scheme provides the application of the T0 code and the updating of the encoding/decoding registers whenever *SEL* is asserted; otherwise, it uses the Bus-Invert code.

3 Special Purpose Systems

When the sequentiality of the address streams is low, solutions as the Gray and the T0 are not effective. However, it may well be the case that other types of correlations exist between patterns being sent on the bus. To identify such correlations, we propose the Beach code, which differs from available low-power encoding schemes in that it is strongly application oriented. In fact, the encoding and decoding functions are properly determined for a given program based on the analysis of the address streams produced by one or more executions of such program. Therefore, it is particularly suitable for special purpose machines, where the same application code is executed repeatedly by an off-the-shelf core processor or microcontroller. Since the use of components of this type as basic blocks for the development of digital systems is becoming a well-established design strategy in the microelectronics industry, we believe that the Beach code could provide a valuable option when power minimization has to be achieved. A high-level block diagram of the basic operations required to determine the Beach code is depicted in Figure 1.

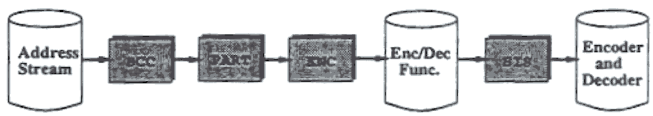

Figure 1: The Beach Encoding.

The entry point is the address stream produced by one or more runs of the code executed by the embedded processor. Such stream is fed to the tool, called BCC (Bit Correlation Computer), whose task is to perform the statistical analysis of the patterns appearing in the stream. In this phase, the target is to extract the information on the correlations that may exist between groups of bits. For obvious reasons, it is impossible to compute this information exactly, since the length of the stream can be in the order of millions; therefore, we measure the block correlations using a pairwise approximation.

Three different types of correlations are of interest to us:

- *Spatial* correlation, expressing the likelihood of correctly predicting the value of one bit of a pattern knowing the value of another bit in the same pattern;
- *Spatio-temporal* correlation, expressing the likelihood of correctly predicting the value of one bit of a pattern by observing its value in the previous pattern and by knowing the value of another bit in the same pattern;
- *Switching* correlation, expressing the likelihood of correctly predicting the occurrence of a transition of one bit of the bus by observing the occurrence of a transition on another bit when a pair of patterns is transmitted.

For each of these measures, a matrix C is constructed, whose entries c_{ij} represent the pairwise correlation between bit i and bit j.

The pairwise correlation information is then processed by program PART, whose objective is to cluster together bits that have a high pairwise correlation, since this is an indication that the probability distribution of bit patterns in the cluster is highly non-uniform. Clearly, we cannot allow excessively large clusters, because the hardware cost of the encoding/decoding logic rapidly increases with the cluster size, and so does the complexity of the data collection and the encoding procedure. Clusters are currently computed either as strongly connected components of the graph G representing matrix C, or with a greedy algorithm that allows to specify a bound on the size of the clusters.

Each cluster, represented by a graph G_i, is finally processed by the EXC program, whose purpose is to encode the bus lines belonging to the clusters so that the number of bus transitions occurring when the embedded code is executed again gets minimized. The encoding algorithm we have adopted is the one proposed in [9] for re-encoding the states of a finite state machine. The procedure is fully based on implicit representations of Boolean and pseudo-Boolean (i.e., real-valued) functions by means of BDDs [10] and ADDs [11], and it solves the re-encoding problem, whose exact solution is NP-hard, in a heuristic way; this is acceptable, since our purpose is to handle graphs whose sizes are larger than the ones that can be managed by traditional (i.e., based on explicit representation of the graph) methods, rather than obtaining an exact solution. Two heuristics are available, both pursuing the objective of pairing together the vertices of G_i having highest edge weights, and assigning them pairs of codes at the closest possible Hamming distance. The first heuristics is based on maximum weighted matching [12], the second one relies on a recursive variant of the Kernighan-Lin mincut partitioning algorithm [13].
The output of the EXC program is a set of encoding and decoding functions, one for each bus line, whose implementation in logic originates the encoder and decoder circuitry.

4 Experimental Results

We have applied the codes introduced in Sections 2 and 3 to the address streams generated by the MIPS microprocessor, and we have simulated the encoded traces to determine the total number of bus transitions. Concerning the Dual_T0_BI code, the selected benchmarks are common utilities, such as data compressors and word processors, as well as logic synthesis tools. For the Beach code, we have chosen a set of software functions which are often implemented in hardware as parts of dedicated systems for image processing, automotive control, DSP, robotics, plant control, and so on.
Tables 1 and 2 show the experimental data. For each example, we give the length of the address streams considered for the experiment, the percentage of in-sequence addresses, the number of bus transitions when no encoding is used, the number of bus transitions after encoding, and the savings achieved with respect to the unencoded case. For the general purpose programs, whose address streams are characterized by quite high sequentialities, results obtained by using the Gray code are also reported.
The results are highly satisfactory; in fact, an average savings in switching activity of 23.3% has been obtained for general purpose programs using the Dual_T0_BI code, and an average switching activity reduction of 41.9% has been achieved for special purpose applications using the Beach code.

5 Directions for Future Research

Although the savings in switching activity that can be obtained using the proposed encoding schemes are satisfactory, several issues are still open, in particular for what concerns the Beach code.
The procedure for finding clusters of bits suitable for encoding is highly heuristic and driven by approximate information (only pairwise correlations are considered). Several experiments revealed that clustering is paramount to achieve good results. If clusters are not chosen properly, the quality of the results is poor independently from the effort spent in computing the best codes. We conjecture that more powerful clustering strategies may greatly increase the power savings.
The synthesis procedure for encoder and decoder can be improved; one direction could be resorting to ZDDs [14] for directly incorporating some synthesis-oriented criteria in the translation from the functional representation to the circuit description, as done in [15].
The applicability of the code to symbolic encoding problems, such as the selection of power-optimal op-codes for instructions or microcode power minimization deserves some attention. In fact, in these applications, a power-conscious choice of which binary codes are to be assigned to symbolic instructions may greatly influence the overall power dissipation of the instruction decoding logic and the registers in the instruction processing pipelines.
The T0 code is based on a fixed scheme and is less relevant from the synthesis point of view. However, it points to an interesting direction of investigation. In fact, it proves that the introduction of a limited amount of redundancy can be helpful in reducing the average switching. The usefulness of redundant codes is well known to researchers working on state assignment for finite state machines [16]. We conjecture that redundant codes may achieve very low switching activity. However, only limited redundancy should be allowed in bus encoding, because pin count on modern VLSI chips is usually highly constrained.

6 Conclusions

Bus encoding for low power has high practical relevance in today's VLSI design because off-chip bus lines are loaded with capacitances that can be orders of magnitude larger than the internal ones. For this reason, it may be worthy to encode (and decode) the data transmitted on the bus to reduce its average switching activity. The area, speed, and power cost of the encoder and decoder are compensated by a sizable reduction of the overall chip power consumption.
We have described two different encoding schemes that reduce the switching activity on the lines of address busses. The first one, called T0 code, is based on the introduction of limited redundancy and exploits the high sequentiality of address streams. The second one, called Beach code, exploits the presence of block correlations between bit lines. While the T0 code adopts a fixed encoding/decoding scheme, the Beach encoding synthesizes dedicated encoding/decoding circuitry based on the statistics of the streams to which it will be applied. Hence, T0 is more suitable to general-purpose applications, while Beach targets specialized applications like the ones running on embedded processors and microcontrollers.
There are strong analogies between bus encoding and the classical state encoding problem that has been studied for a long time by the synthesis community [17, 18, 19]. We believe that several concepts and intuitions developed for state encoding can be fruitfully exploited in the new domain of application.

Acknowledgments

We wish to thank Luciano Lavagno for providing us with the code of the dashboard example [20].

References

[1] P. R. Panda, N. D. Dutt, "Reducing Address Bus Transitions for Low Power Memory Mapping," *EDTC-96: IEEE European Design and test Conference*, pp. 63-67, Paris, France, March 1996.

[2] M. R. Stan, W. P. Burleson, "Bus-Invert Coding for Low-Power I/O," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 3, No. 1, pp. 49-58, March 1995.

Benchmark	Stream Length	In-Sequence Addresses	Binary Transitions	Gray		Dual_T0_BI	
				Trans.	Savings	Trans.	Savings
espresso	2322423	54.4%	15147444	13411543	11.5%	11883056	21.6%
ghostview	517284	58.2%	3067635	2999453	2.2%	2785341	14.3%
gunzip	78486	52.8%	482093	370443	23.2%	343335	28.8%
gzip	153495	57.4%	833799	814787	2.3%	709769	14.9%
jedi	20835298	57.8%	152427486	132793451	12.1%	126132486	17.3%
latex	900426	55.5%	7002828	4802295	31.4%	4002340	42.9%
Average		56.0%			13.8%		23.3%

Table 1: Experimental Results for the Dual_T0_BI Code.

Application	Stream Length	In-Sequence Addresses	Binary Transitions	Beach	
				Trans.	Savings
dashboard	**84918**	**39.1%**	**619690**	**443115**	**28.4%**
dct	**13769**	**45.4%**	**48917**	**31472**	**35.6%**
fft	**23441**	**43.4%**	**138526**	**85653**	**38.1%**
mat_mult	**22156**	**40.4%**	**105947**	**60654**	**42.7%**
q-sort	**20347**	**38.8%**	**182673**	**96306**	**42.2%**
vxv_mult	**18417**	**45.1%**	**133272**	**46838**	**64.8%**
Average		**42.0%**			**41.9%**

dashboard:
dct:
fft:
mat_mult Matrix Multiplication.
q-sort: Quick Sort for Vectors of Integers.
vxv_mult Vector by Vector Scalar Multiplication

[3] C. L. Su, C. Y. Tsui, A. M. Despain, "Saving Power in the Control Path of Embedded Processors," *IEEE Design and Test of Computers*, Vol. 11, No. 4, pp. 24-30, Winter 1994.

[4] H. Mehta, R. M. Owens, M. J. Irwin, "Some Issues in Gray Code Addressing," *GLS-VLSI-96: IEEE 6th Great Lakes Symposium on VLSI*, pp. 178-180, Ames, IA, March 1996.

[5] L. Benini, G. De Micheli, E. Macii, D. Sciuto, C. Silvano, "Asymptotic Zero-Transition Activity Encoding for Address Busses in Low-Power Microprocessor-Based Systems", *GLS-VLSI-97: IEEE 7th Great Lakes Symposium on VLSI*, pp. 77-82, Urbana, IL, March 1997.

[6] L. Benini, G. De Micheli, E. Macii, D. Sciuto, C. Silvano, "Address Bus Encoding Techniques for System-Level Power Optimization," *EuroDAC-97: IEEE European Design Automation Conference*, Dusseldorf, Germany, November 1997, Submitted for Publication.

[7] L. Benini, G. De Micheli, E. Macii, M. Poncino, S. Quer, "System-Level Power Optimization of Special Purpose Applications: The Beach Solution," *ISLPED-97: ACM/IEEE International Symposium on Low Power Electronics and Design*, Monterey, CA, August 1997, Submitted for Publication.

[8] G. Kane, J. Heinrich, *MIPS RISC Architecture*, Prentice Hall, 1994.

[9] G. D. Hachtel, M. Hermida, A. Pardo, M. Poncino, F. Somenzi, "Re-Encoding Sequential Circuits to Reduce Power Dissipation," *ICCAD-94: IEEE/ACM International Conference on Computer-Aided Design*, pp. 70-73, San Jose, CA, November 1994.

[10] R. Bryant, "Graph-Based Algorithms for Boolean Function Manipulation," *IEEE Transactions on Computers*, Vol. C-35, No. 8, pp. 79-85, August 1986.

[11] R. I. Bahar, E. A. Frohm, C. M. Gaona, G. D. Hachtel, E. Macii, A. Pardo, F. Somenzi, "Algebraic Decision Diagrams and their Applications," *ICCAD-93: IEEE/ACM Intl. Conf. on Computer-Aided Design*, pp. 188-191, Santa Clara, CA, November 1993.

[12] T. H. Cormen, C. E. Leiserson, R. L. Rivest, *An Introduction to Algorithms*. McGraw-Hill, 1990.

[13] B. W. Kernighan, S. Lin, "An Efficient Heuristic Procedure for Partitioning Graphs," *Bell System Technical Journal*, Vol. 49, pp. 291-307, February 1970.

[14] S. Minato, "Zero-Suppressed BDDs for Set Manipulation in Combinatorial Problems," *DAC-30: ACM/IEEE Design Automation Conference*, pp. 272-277, Dallas, TX, June 1993.

[15] B. Kumthekar, I. H. Moon, F. Somenzi, "A Symbolic Algorithm for Low-Power Sequential Synthesis," *ISLPED-97: ACM/IEEE International Symposium on Low Power Electronics and Design*, Monterey, CA, August 1997, Submitted for Publication.

[16] G. De Micheli, "Symbolic Design of Combinational and Sequential Logic Circuits Implemented by Two-Level Logic Macros," *IEEE Transactions on Computer-Aided Design*, Vol. 5, No. 4, pp. 597-616, October 1986.

[17] G. De Micheli, *Synthesis and Optimization of Digital Circuits*, McGraw-Hill, 1994.

[18] S. Devadas, A. Ghosh, K. Keutzer, *Logic Synthesis*, McGraw-Hill, 1994.

[19] G. D. Hachtel, F. Somenzi, *Algorithms for Logic Synthesis and Verification*, Kluwer Academic Publishers, 1996.

[20] C. Passerone, L. Lavagno, C. Sansoé, M. Chiodo, A. Sangiovanni-Vincentelli, "Trade-Off Evaluation in Embedded System Design via Co-simulation," *ASP-DAC-97: IEEE Asia South-Pacific Design Automation Conference*, pp. 291-297, Chiba, Japan, January 1997.