Low Power Address Encoding using Self-Organizing Lists *

Mahesh Mamidipaka
Center for Embedded Systems
University of California, Irvine
Irvine, CA - 92697
maheshmn@cecs.uci.edu

Dan Hirschberg
Dept. of ICS
University of California, Irvine
Irvine, CA - 92697
dan@ics.uci.edu

Nikil Dutt
Center for Embedded Systems
University of California, Irvine
Irvine, CA - 92697
dutt@cecs.uci.edu

ABSTRACT

Off-chip bus transitions are a major source of power dissipation for embedded systems. In this paper, new adaptive encoding schemes are proposed that significantly reduce transition activity on data and multiplexed address buses, that do not add redundancy in space or time and which have minimal delay overhead. These adaptive techniques are based on self-organising lists to achieve reduction in transition activity by exploiting the spatial and temporal locality of the addresses. Unlike previous approaches that focus on instruction address buses, experiments demonstrate significant reduction in transition activity of up to 54% in data address buses and up to 59% in multiplexed address buses. The average reductions are twice those obtained using current schemes on a data address bus and more than twice those obtained on a multiplexed address bus.

1. INTRODUCTION

Power dissipation has become a critical design criterion in most system designs, especially in portable battery-driven applications such as mobile phones, PDAs, laptops, etc. that require longer battery life. Reliability concerns and packaging costs have made power optimization even more relevant in current designs. Moreover, with the increasing drive towards System On a Chip (SOC) applications, power has become an important parameter that needs to be optimized along with speed and area. Power reduction techniques have been proposed at different levels of the design hierarchy from the algorithmic level[3] and system level[11] to the layout level[12] and circuit level[11]. The dominant source of power dissipation however, is due to the charging and discharging of node capacitances during transitions, referred to as capacitive power[17, 10].

*This work was partially supported by grants from NSF (MIP-9708067), DARPA (F33615-00-C-1632) and Motorola Corporation

ISLPED'01, August 6-7, 2001, Huntington Beach, California, USA.

A significant portion of total power dissipation is due to the transitions on the off-chip buses. It is estimated that power dissipated on the I/O pads of an IC ranges from 10% to 80% of the total power dissipation with a typical value of 50% for circuits optimized for low power[14]. This is because of the large switching capacitances associated with these bus lines. Therefore, various techniques have been proposed in the literature, which encode the data before transmission on the off-chip buses so as to reduce the average and peak number of transitions.

However, most techniques have focussed on reduction of transition activity on the instruction address buses and have not generated consistent improvement on data address buses, without incurring significant penalty through redundancy in space or time. In this paper, new encoding techniques are presented that exploit the notion of self-organizing lists to adaptively encode data and multiplexed addresses. This approach achieves significant and consistent transition activity reduction without adding redundancy in space or time. The paper is organized as follows: Section 2 reviews the related work and Section 3 discusses the techniques for data address buses and their implementations. The techniques proposed for multiplexed address bus and their implementation details are discussed in Section 4. Section 5 shows the reduction in transition activity obtained by applying these techniques on various programs. Furthermore, these results are compared with the existing techniques to demonstrate the efficacy of these techniques. Finally, conclusions and future work are presented in Section 6.

2. RELATED WORK

Since instruction addresses are mostly sequential, Gray coding[16] was proposed to minimize the transitions on the instruction address bus. The Gray code ensures that when the data is sequential, there is only one transition between two consecutive data words. However this coding scheme may not work for data address buses because the data addresses are typically not sequential. An encoding scheme called T0 coding[2] was proposed for the instruction address bus. This coding uses an extra bit line along with the address bus, which is set when the addresses on the bus are sequential, in which case the data on the address bus are not altered. When the addresses are not sequential, the actual address is put on the address bus. Bus-Invert (BI) coding[14] is proposed for reducing the number of transitions on a bus. In this scheme, before the data is put on the bus, the number of transitions that might occur with respect to the previously transmitted data is computed. If the transition count is more than half the bus width, the data is inverted and put on the bus. An extra bit line is used to signal the inversion on the bus.

Variants of T0, T0_BI, Dual T0, and Dual T0_BI[15] are proposed which combine T0 coding with Bus-Invert coding. Ramprasad et al. described a generic encoder-decoder architecture[13], which can be customized to obtain an entire class of coding schemes for reducing transitions. The same authors proposed INC-XOR coding, which reduces the transitions on the instruction address bus better than any other existing technique. An adaptive encoding method is also proposed by Ramprasad et al.[13] , but with huge hardware overhead. This scheme uses a RAM to keep track of the input data probabilities, which are used to code the data.

Another adaptive encoding scheme is proposed by Benini et al., which does encoding based on the analysis of previous N data samples[1]. This again has huge computational overhead. Mussol et al. propose a Working Zone Encoding (WZE) technique[9], which works on the principle of locality. Although this technique gives good results for data address buses, there is a huge delay and hardware overhead involved in encoding and decoding. Moreover this technique requires extra bit lines leading to redundancy in space. Recently, Cheng et al. have proposed coding techniques for optimizing switching activity on a multiplexed DRAM address bus[4].

Although the existing methods give significant improvement on instruction address buses, none of the encoding methods yield significant and consistent improvement on the data and multiplexed address buses without redundancy in space or time. This is because most of the proposed techniques are based on the heuristic that the addresses on the bus are sequential most of the time. However, data addresses are typically not sequential and hence the existing techniques fail to reduce transition activity. Many of the existing schemes add redundancy in space or time, which may be expensive in some applications.

3. DATA ADDRESS BUSES

Although data addresses may not be sequential, they still follow the principles of spatial and temporal locality[6]. That is, it is more likely that there will be an access to a location near the currently accessed location (spatial locality) and it is more likely that the currently accessed location will be accessed again in the near future (temporal locality). We propose adaptive encoding techniques that exploit the principle of locality for reducing the transitions on the data address bus.

We develop heuristics to minimize the number of transitions between the most frequently accessed address ranges by assigning them the codes with minimal Hamming distance. To achieve this, we employ the Move-To-Front (MTF) and TRanspose (TR) methods in self-organizing lists[7] for assigning codes that reduce transitions on the address bus.

MTF is a transformation algorithm that, instead of outputting the input symbol, outputs the index of the symbol in a table. The table has all possible symbols stored in it. Thus the length of the code is the same as the length of the symbol. Both the encoder and decoder initialize the table with the same symbols in the same positions. Once a symbol is processed, the encoder outputs the code corresponding to its position in the table and then the symbol is shifted to the top of the table (position 0). All the codes from position 0 until the position of the symbol being coded are moved to the next higher position. The TR algorithm is similar to MTF in that the code assigned to the symbol is the position of the symbol but, instead of moving the symbol to the front, the symbol is exchanged in position with the symbol just preceding it. If the symbol is at the beginning of the list, it is left at the same position.

Note that in both MTF and TR, the most frequent incoming symbols are moved to the beginning of the list and thus the indices of these locations are closer in terms of Hamming distance. Therefore the transition activity between the codes assigned to most frequent incoming symbols is minimized. These heuristics are very useful in data address buses since there is a greater likelihood of two different address sequences being sent on the bus (two arrays being accessed alternatively, reads from an address space and writes to a different address space, etc.). In such cases, it is desirable to keep the encoding of these address ranges as close as possible i.e., with minimal Hamming distance. The MTF and TR heuristics achieve the goal by self-organization. But storage of all the possible address ranges and managing them in the list is impractical in terms of area and delay overhead. Hence the address bus is partitioned into a set of smaller address buses and encoding is applied on each of these addresses separately. We now discuss the architecture and implementation of the self-organizing lists based encoder and decoder. The delay/area overheads for each of these techniques is minimal, as descibed here, and as demonstrated in our experiments presented in Section 5.

3.1 Encoder Implementation of Self-organizing Lists

The functional implementation of an encoder based on self-organizing can be split into two phases. During the first phase, for every incoming symbol, the index corresponding to it is extracted from the list and put on the bus. In the second phase, the list is organized based on the incoming symbol. While the first phase is common for both MTF and TR techniques, the techniques use different strategies for organizing the lists based on the incoming symbol.

A generic structure for the implementation of self-organizing lists is shown in Figure 1 for a bus of width 2. Note that the straightforward implementation of searching for the symbol in the list and sending the index corresponding to the symbol location has huge delay overhead in the critical path. A better way of implementing this would be to keep the location of the symbol fixed and, for every incoming symbol, update the codes of the symbols. As shown in Figure 1, the incoming signal is fed to the select lines of the multiplexer(SEL_MUX) which outputs the code corresponding to that symbol. The selected code(Y_1Y_0) is then fed back to combinational logic which organizes the codes of the symbols in the list. This combinational logic is different for the MTF and TR techniques.

For MTF encoder, the combinatorial logic will have the following functionality:

$$
\begin{array}{lcll}
N_{xy} & = & C_{xy} & \text{if } Y_0Y_1 < C_{xy} \\
 & = & C_{xy}+1 & \text{if } Y_0Y_1 > C_{xy} \\
 & = & 00 & \text{if } Y_0Y_1 = C_{xy}
\end{array}
$$

For TR encoder, the combinatorial logic will have the fol-

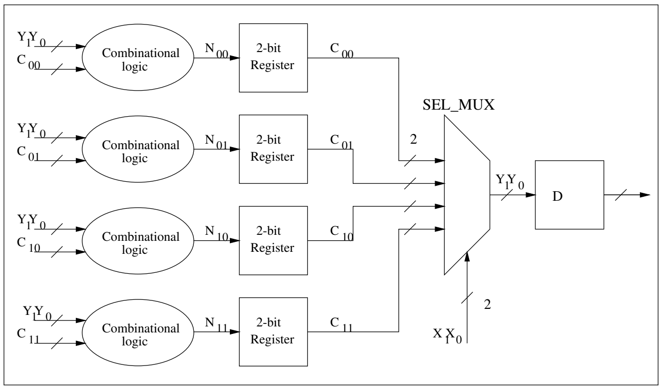

Figure 1: Generic Architecture for Self-organizing Lists based Encoder

lowing functionality:

$$
\begin{aligned}
N_{xy} &= C_{xy} - 1 && \text{if } Y_0Y_1 = C_{xy} \ \ and \ \ C_{xy} \neq 0 \\
&= C_{xy} + 1 && \text{if } Y_0Y_1 = C_{xy} + 1 \\
&= C_{xy} && \text{Otherwise}
\end{aligned}
$$

The encoder inserts a one-cycle delay between arrival of the address and output of the encoding. As indicated by Benini et al. [2], this is not an overhead because even if binary code (without encoding) were used, the flip-flops at the output of the bus would be needed. This is because the address would be generated by a very complex logic circuit that produces glitches and misaligned transitions. The flip-flops filter out the glitches and align the edges to the clock thereby eliminating excessive power dissipation and signal quality degradation.

The delay induced in the address path due to this encoding is the delay of the multiplexer(SEL_MUX). The size of the multiplexer is exponentially proportional to the bus-width. Since the buses are split into buses of smaller widths and the encoding is applied to each of them independently, the size of multiplexer and hence the delay overhead due to it is minimized. The other paths that arise due to the encoding, start and terminate within the module and hence should not add to any timing violations. Some other minor overheads would be involved at reset because the registers have to be initialized to the appropriate values. Since none of these overheads appear in the actual address generation path, these paths are not considered critical.

3.2 Decoder Implementation of Self-organizing Lists

Similar to the encoder, the functionality of the decoder can be split into two phases. In the first phase, the symbol corresponding to the code is extracted from the list and, in the second phase, the list is organized based on the extracted symbol. Figure 2 shows the architecture for the MTF based decoder.

Unlike the encoder where the codes are stored in the list, for the decoder the symbols are stored in the list. The incoming code(Y_1Y_0) is fed to the multiplexer(SEL_MUX) to extract the symbol from the list. The extracted symbol (X_1X_0) is then fed back to organize the list based on the strategy used. The inputs to the multiplexers in front of the registers shown in Figure 2 determine the symbol that

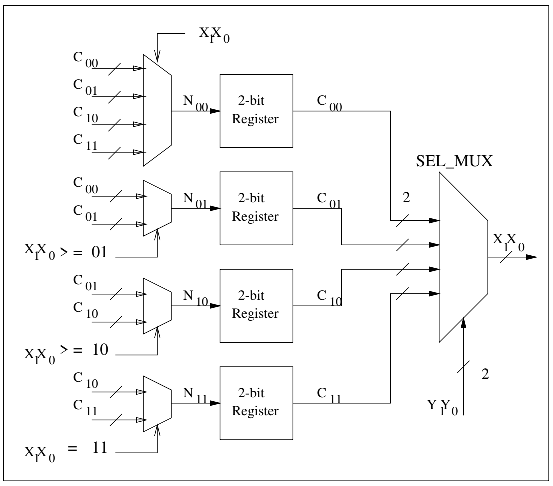

Figure 2: Architecture for MTF based Decoder

will replace the location corresponding to the index position. The decoder for the TR based implementation is similar to MTF based implementation except that the inputs to these multiplexers are different.

Similar to the encoder, the critical path in the decoder is the multiplexer(SEL_MUX) for extracting the symbol corresponding to the code. All other paths start and terminate within the module and hence are not considered critical.

The actual delay and area overhead for self-organizing lists based encoder and decoder are presented in Section 5.

4. MULTIPLEXED ADDRESS BUSES

In a multiplexed address bus, both instruction and data addresses are sent on the same bus. So while the addresses still exhibit the principle of locality, they are often sequential because of the characteristic of instruction addresses[6]. We propose heuristics that make use of both the sequential nature and the locality principle to reduce the transition activity on the multiplexed address bus. Fortunately, when the addresses are sequential, most of the transitions occur on a few of the least significant bits. Thus, we use techniques related to sequential data on the least significant bits and techniques that exploit the principle of locality on the higher significant bits.

When the addresses are sequential, the least significant bits account for the most number of transitions. More specifically, the 4 least significant bits contribute to approximately 93.75% of the total transitions in sequential addresses. Different encoding techniques could be applied on the least significant bits which significantly reduce the transition activity in instruction addresses[8]. We describe the INC-XOR and Delta-TS techniques below.

- The INC-XOR encoding technique[13] best reduces the transition activity in an instruction address stream. In this scheme, the data transmitted is the EXclusive-OR of the current address and previous address incremented by a constant. The technique was proposed to be applied on the whole address bus. However, by applying this technique to only the least significant bits, we still get significant reductions in transition activity. The comparison of reductions when INC-XOR is applied on the 4 least significant bits and when applied on the whole address bus is shown in Section 5.

- The Delta-TS encoding technique transmits the difference between the previous address incremented by a constant and the current address with Transition Signaling (TS). Since the subtractor involved in delta calculation is only 4 bits wide, efficient Look-Up-Table (LUT) based implementations could be used to lower delay overhead. The structure of a 4-bit Delta-TS based encoder is shown in Figure 3.

The actual delay and area overheads for encoding/decoding of these techniques, along with the reductions obtained by using these techniques are presented in Section 5. The self-organizing lists based techniques (MTF/TR), are applied to the least significant bits and reduce transitions due to sequential addresses; the Delta-TS/INC-XOR techniques are applied to the higher significant bits and exploit the principle of locality to reduce the transitions.

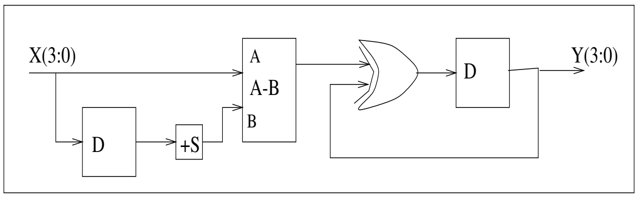

Figure 3: Delta-TS Encoder

5. EXPERIMENTS

We now present results of our low-power address encoding techniques. The programs used for the experiments are the UNIX compression and compiler utilities (gzip and cc), commonly used UNIX commands (ls, who, and date), and standard C programs (factorial and sort). The address traces of the programs were obtained by executing them on an instruction-level simulator, SHADE[5] on a SUN Ultra-5 workstation. The comparison is made in terms of the total number of toggles on the bus before and after the encoding is applied. We also present the actual area and delay overhead of the encoding and decoding through synthesis using the Synopsys Design Compiler.

Table 1. Transition activity reductions using self-organizing lists based encodings on data address bus

	# of addr	%Seq	Binary (in K)	MTF+TS (w=2)	TR+TS (w=3)	MTF+TS (w=3)	TR+TS (w=4)	MTF+TS (w=4)
cc	55496	11%	562.8	38%	38%	45%	41%	48%
gzip	905338	0.4%	9082.0	31%	47%	44%	54%	52%
ls	40704	4%	338.9	19%	29%	30%	36%	37%
who	71443	8%	638.2	18%	33%	31%	40%	37%
date	21032	8%	205.2	18%	31%	32%	39%	38%
factorial	3783	5%	35.8	14%	25%	27%	30%	33%
sort	23390	4%	232.9	16%	28%	29%	35%	38%

Table 1 shows the percentage reduction in transition activity for the self-organizing lists based encodings applied to the data addresses. As indicated in Section 3, the address bus is split into smaller bus widths and the encoding is applied to each of these buses independently. In the results shown, "w" indicates the width of the smaller buses. The first column indicates the programs to which the encodings have been applied. Column 2, "# of addr." indicates the total number of addresses to which the encodings have been applied. Column 3, "%Seq" indicates the percentage of addresses which are sequential. Column 4, "Binary" indicates the total transitions that occur on the bus without any encoding. Columns 5, 6, 7, 8, and 9 indicate the percentage reductions obtained when the MTF and TR techniques are applied for different values of w. It was observed that when Transition Signaling ($Y_i = Y_{i-1}\ \ xor\ \ X_i$, where Y is the outgoing bit stream and X is the incoming bit stream) is applied on top of these encodings, a greater reduction in transition activity is often achieved. The results shown in the table indicate the reduction after Transition Signaling(TS).

Note that the reductions increase with increasing bus width, but the delay overhead due to the encoding/decoding increases rapidly with higher bus widths as shown at the end of this section. So a configuration could be selected based on the desired transition activity reduction and tolerable delay overhead. For applications with tight delay constraints, the configuration with minimum delay overhead, w=2 could be used.

Table 2. Transition activity reductions using different encoding techniques on data address bus

	TR(+TS) (w=4)	MTF(+TS) (w=4)	Gray	Inc-Xor	Bus-Inv (w=8)
cc	39%(41%)	41%(48%)	20%	-4%	23%
gzip	43%(54%)	42%(52%)	41%	-8%	42%
ls	25%(36%)	31%(37%)	17%	-9%	17%
who	36%(40%)	39%(37%)	15%	-6%	20%
date	34%(39%)	36%(38%)	14%	-6%	16%
factorial	26%(30%)	32%(33%)	4%	-9%	7%
sort	28%(35%)	35%(38%)	17%	-8%	19%
Average Reduction:	33%(40%)	36%(40%)	18%	-7%	20%

Table 2 shows the comparison of these reductions with those obtained using existing techniques. A maximum reduction of 54% was achieved by TR+TS with w=4 for the gzip program. The values in the parenthesis for MTF(+TS) and TR(+TS) in columns 2 and 3, indicate the percentage reductions achieved by using TS on top of MTF/TR technique. Except for the 'who' program, there was an increase in the reductions by using TS on top of MTF/TR. This extra reduction on average, is considerable in case of TR(~7%), but is less significant for MTF ($<$ 4%). Columns 4, 5, and 6 show the reduction in transition activity for Gray coding, INC-XOR, and Bus-Invert coding respectively. As expected, since the addresses are not sequential, the INC-XOR technique fails to give any reduction. While Bus-Invert coding gives the best reduction among the existing techniques, this technique needs 4 extra bit lines for implementation which may not be tolerable. And clearly the self-organizing lists based techniques outperform the existing techniques. On average, the reduction with self-organizing lists based encoding is twice that of the best existing technique. Also, the self-organizing lists based encoding does not add any redundancy in space or time (no extra bit-lines or time slots are needed for implementation).

Table 3 shows the reduction in transition activity for various combination of encoding techniques on the multiplexed address bus. While the Delta-TS and INC-XOR are applied on the least significant 4-bits, the MTF and TR are applied on the higher significant bits. As in the data address bus, the multiplexed address bus is split into smaller buses(w=4) and the encodings are applied on each of them independently.

Table 3. Transition activity reductions for various encoding techniques on multiplexed address bus

	# of addr (in K)	%Seq	Binary (in K)	Inc-Xor +MTF	Inc-Xor +TR	Delta +MTF	Delta +TR
cc	387.3	60%	2100.7	49%	41%	50%	40%
gzip	4412.7	57%	35323.0	58%	56%	59%	57%
ls	443.7	57%	2415.5	46%	33%	47%	34%
who	819.6	58%	4445.7	47%	36%	48%	38%
date	153.2	60%	823.2	50%	39%	51%	41%
factorial	30.8	62%	142.8	49%	34%	51%	35%
sort	295.2	60%	1543.0	49%	38%	50%	39%

Table 4. Comparison of transition activity reductions for various encodings on multiplexed address bus

	Delta +MTF (w=4)	Gray	Inc-Xor	Bus-Inv (w=8)
cc	50%	22%	22%	21%
gzip	59%	47%	14%	50%
ls	47%	19%	20%	17%
who	48%	18%	23%	16%
date	51%	19%	24%	18%
factorial	51%	16%	27%	19%
sort	50%	18%	24%	17%
Average Reductions:	51%	23%	22%	22%

It can be noticed that MTF based encodings give better reductions than TR based encodings. Also, Delta based encodings give marginally better reductions than the INC-XOR ones. In all the cases in which the encodings have been applied, the Delta+MTF combination gave the best results. The best reduction obtained with these encodings was 59% on the gzip program. But if delay overhead is a major concern, then INC-XOR+MTF could be used which gives reductions marginally less than that of Delta+MTF. Table 4 shows the comparison of these reductions with the existing techniques: Gray, INC-XOR and Bus-Invert coding. Among the existing techniques Gray coding gives the best reductions. The reduction for INC-XOR is mainly due to the instruction addresses in the multiplexed address bus which are sequential. On average, Delta+MTF gives a reduction of 51% which is more than twice that of the best existing technique(Gray, 23%).

Each encoding scheme incurs some area and delay overhead. Table 5 compares the area(number of library cells) and the delay(ns) of encoders and decoders that are based on MTF and TR techniques with those based on other techniques. The designs were synthesized using Synopsys Design Compiler on a 0.6μm LSI_10K library and the synthesis was done for a 32-bit address bus. For MTF and TR the synthesis was done for w=4, and for Bus-Inv, synthesis was done for w=8 (i.e., the same parameters used in the previous experiments).

Table 5. Area and Critical Path Delay overheads for Self-organizing Lists based Encoder and Decoder

	Area(lib. cells)		Delay (ns)	
	Enc	Dec	Enc	Dec
MTF+TS	2582	2485	4.6	4.2
TR+TS	2436	2345	4.6	4.2
Bus-Inv	210*	40*	4.7	1.9
Gray	72	406	3.2	12.3
Inc-Xor	496	518	4.2	4.2

The asterisk for Bus-Inv indicates that the area overhead due to extra bit lines was not considered in its area evaluation. As can be noted, the delay overhead in the critical path for MTF/TR is comparable to that for the existing techniques. But the area overhead of these techniques is considerably more than that of other techniques. Considering the fact that the reduction in transition activity obtained with this technique is consistently more than twice the existing techniques, we think this extra overhead in area is acceptable. If the area overhead is a concern, it can reduced substantially by applying this technique on fewer number of bits (say w=3). Another overhead that needs analysis is the power dissipation due to encoder and decoder. For TR, because of the structure of implementation, for any given input, there can be a change in encoding for at most 2 symbols out of 16 symbols. So, approximately, only 1/8th of the gates could be active in any cycle for any input. Similarly for MTF, on average 1/2 of the gates would be active. Assuming a transition activity of 0.5, the possible number of transitions are approximately 600 for MTF and 150 for TR. Note that the I/O capacitance is at least 3 orders of magnitude more than that of the internal capacitance[9]. Hence the overhead due to internal power dissipation is still considerably less than the reduction obtained.

The delay induced in the critical path due to the encoding/decoding and the area overhead for the Delta-TS and INC-XOR techniques are shown in Table 6 for a 4-bit address bus. The delay overhead of Delta-TS is higher than that of INC-XOR technique because of the 4-bit subtractor needed for calculating the difference between the current address and the previous address. But the reduction in transition activity by using the Delta-TS technique is marginally more than that obtained by using INC-XOR technique. Also, it can be noted that the area overhead of these modules is minimal.

Table 6. Area and Critical Path Delay overheads for Delta-TS and INC-XOR Encoding/Decoding

	Delta-TS		INC-XOR	
	Encoder	Decoder	Encoder	Decoder
# of Cells	60	51	30	30
CP_Del(ns)	2.97	2.58	0.96	0.96

6. CONCLUSIONS AND FUTURE WORK

In this paper, we presented self-organizing list based encoding techniques (MTF and TR) for data address buses. For multiplexed address buses, we employ a combination of encoding techniques: while the Delta and INC-XOR are applied on the least significant bits, the self-organizing lists based encoding are applied on the more significant bits of the multiplexed address bus. This enables the exploitation of both the sequential nature of the instruction addresses as well as the locality of addresses in the multiplexed address buses. The proposed techniques consistently outperform the existing techniques in both data address bus and multiplexed address bus without adding the overhead of redundancy in space or time. Results show that TR with TS applied to various data address streams gives up to 54% reduction in transition activity. On a multiplexed address bus, Delta+MTF yields a reduction of up to 59%. Future work will involve the applicability of the proposed techniques to data buses.

7. REFERENCES

[1] L. Benini, A. Macci, E. Macii, M. Poncino, and R. Scarsi. Architectures and synthesis algorithms for power-efficient bus interfaces. *IEEE Transactions on Computer Aided Design of Circuits and Systems*, 19, 2000.

[2] 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. In *Great Lakes Symposium on VLSI*, Urbana, IL, 1997.

[3] A. P. Chandrakasan and R. W. Broderson. Minimizing power consumption in digital CMOS circuits. *Proceedings of the IEEE*, 83:498,523, 1995.

[4] W-C. Cheng and M. Pedram. Low power techniques for address encoding and memory allocation. *ASP-DAC*, 2001.

[5] R. F. Cmelik and D. Keppel. Shade: A fast instruction-set simulator for execution profiling. Technical Report UW-CSE-93-06-06, 1993.

[6] J. L. Hennessy and D. A. Patterson. *Computer Architecture: A Quantitative Approach.* Morgan Kaufmann, San Francisco, 1996.

[7] J. Hester and D. S. Hirschberg. Self-organizing linear search. *Computing surveys*, 17:295,311, 1985.

[8] M. N. Mahesh, D. Hirschberg, and N. Dutt. Encoding techniques for low power address buses. Technical Report #01-22, University of California, Irvine, 2001, http://www.ics.uci.edu/~maheshmn/encoding_tr.doc.

[9] E. Musoll, T. Lang, and J. Cortadella. Working-zone encoding for reducing the energy in microprocessor address buses. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 6, 1998.

[10] F. Najm. Transition density, a stochastic measure of activity in digital circuits. In *Design Automation Conference*, pages 644-649, 1991.

[11] M. Pedram. Power minimization in IC design: principles and applications. *ACM Transactions on Design Automation of Electronic Systems*, 1:3-56, 1996.

[12] M. Pedram and H. Vaishnav. Power optimization in VLSI layout: a survey. *The Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology.*, 15:221-232, 1997.

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

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

[15] M. R. Stan and W. P. Burleson. Low-power encodings for global communications in CMOS VLSI. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 5:444-455, 1997.

[16] C. L. Su, C. Y. Tsui, and A. M. Despain. Saving power in the control path of embedded processors. *IEEE Design and Test of Computers*, 11:24-30, 1994.

[17] N. Weste and K. Eshragian. *Principles of CMOS VLSI Design, A Systems Perspective.* Addison-Wesley, Reading, CA 1998.