Adaptive Low-Power Address Encoding Techniques Using Self-Organizing Lists

Mahesh N. Mamidipaka, Daniel S. Hirschberg, and Nikil D. Dutt, *Senior Member, IEEE*

***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. These adaptive techniques are based on self-organizing lists to achieve reduction in transition activity by exploiting the spatial and temporal locality of the addresses. Also the proposed techniques do not require any extra bit lines and have minimal delay overhead. The techniques are evaluated for efficiency using a wide variety of application programs including SPEC 95 benchmark set. 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.**

***Index Terms*—Bus encoding, low power, transition activity.**

I. Introduction

POWER dissipation has become a critical design criterion in most system designs, especially in portable battery-driven applications such as mobile phones, personal digitial assistants (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 toward system-on-chip (SOC) applications, power has become an important parameter that needs to be optimized along with speed and area. The main sources of power dissipation in VLSI circuits [18] are leakage currents, the standby current (due to continuous dc current drawn from $V_{\rm dd}$ to ground), short-circuit current (due to a dc path between supply and ground lines during transitions), and capacitive current (due to charging and discharging of node capacitances during transitions). Power reduction techniques have been proposed at different levels of the design hierarchy from algorithmic level [3] and system level [12] to layout level [13] and circuit level [12]. The dominant source of power dissipation is due to the capacitive current (referred to as capacitive power [11], [18]) and is given by

$$P = \sum C_L * V_{\rm dd}^2 * E({\rm SW}) * f_{\rm clk}$$

where

P	capacitive power dissipation;
C_L	physical capacitance at the output of the node;
$V_{\rm dd}$	supply voltage;
$f_{\rm clk}$	clock frequency;
$E({\rm SW})$	average number of output transitions per $1/f_{\rm clk}$ time.

Most research efforts have focused on reducing the dynamic power consumption by reducing the number of transitions in the circuits. In particular, researchers have focused on reduction of power dissipation on off-chip buses since a significant portion of total power dissipation is due to the transitions on the off-chip buses. This is because of the large switching capacitances associated with these bus lines. 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 [15]. 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 focused 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 II reviews related work and Section III discusses techniques for data address buses and their implementations. The techniques proposed for multiplexed address bus and their implementation details are discussed in Section IV. Section V shows the reduction in transition activity obtained by applying these techniques on various programs. These results are compared with those from existing techniques to demonstrate the efficacy of these techniques. Finally, conclusions and future work are presented in Section VI.

II. Related Work

Since instruction addresses are mostly sequential, Gray coding [17] 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

Manuscript received February 19, 2002; revised September 4, 2002. This work was supported in part by the National Science Foundation under Grant MIP-9708067, in part by Defense Advanced Research Projects Agency under Grant F33615-00-C-1632, and in part by the Motorola Corporation.

The authors are with the Center for Embedded Computer Systems, Department of Information and Computer Science, University of California, Irvine, CA 92697 USA (e-mail: maheshmn@cecs.uci.edu; dutt@cecs.uci.edu; dan@ics.uci.edu).

Digital Object Identifier 10.1109/TVLSI.2003.814325

[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 [15] 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 [16]) are proposed which combine T0 coding with BI coding. Ramprasad *et al*. described a generic encoder–decoder architecture [14], 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 more than any other existing technique. An adaptive encoding method is also proposed by Ramprasad *et al.* [14] , 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 the previous N data samples [1]. This also has huge computational overhead. Musoll *et al.* propose a working zone encoding (WZE) technique [10], which works on the principle of locality. Although this technique gives good results for data address buses, there is a significant 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.* 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 (no extra bit lines) or time (no extra time slots). 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 typically are 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.

III. 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 [9].

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 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 to 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 described here and as demonstrated in our experiments presented in Section V.

A. 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.

The straightforward implementation searches for the symbol in the list, sends the index corresponding to the symbol location, and then updates the list by organizing the symbols. This implementation has huge delay overhead in the critical path for the symbol search and for encoding the position of the symbol. A better way to implement this is to keep the location of the symbol fixed and to vary the coding corresponding to each symbol. A generic structure for the implementation of self-organizing lists is shown in Fig. 1 for a bus of width

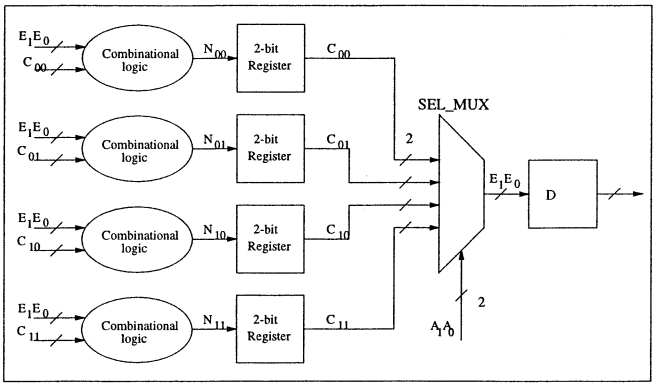

Fig. 1. Generic architecture for self-organizing lists based encoder.

2. Since there are four possible input symbols, the implementation shows four 2-bit registers that store the encodings corresponding to a possible input symbol (00, 01, 10, and 11). Let xy represent each possible input symbol. C_{xy} represents the encoding corresponding to symbol "xy" in current cycle. N_{xy} represents the encoding for "xy" in the next cycle. Effectively, C_{xy} represents the current position of the symbol in the imaginary list and N_{xy} represents the position of the symbol in the list after the organization of the list, that organization being based on the encoding of the input symbol. The encoder works in the following way. For every new incoming address (A_1A_0), the multiplexer (SEL_MUX) enables the selection of the corresponding address encoding E_1E_0. The encoding is put on the bus through a flip-flop indicated by "D." The encoding E_1E_0 is fed back to the combinational logic of each symbol register for changing its encoding to reflect the new organized list. The combinational logic also takes the current encoding C_{xy} as its input. The selection logic is same for both the MTF and TR techniques, but the combinational logic is different because of the difference in their organizational strategy.

For the MTF encoder, the combinational logic has the following functionality:

$$\begin{aligned} N_{xy} &= C_{xy}, && \text{if } Y_1Y_0 < C_{xy} \\ &= C_{xy} + 1, && \text{if } Y_1Y_0 > C_{xy} \\ &= 00, && \text{if } Y_1Y_0 = C_{xy}. \end{aligned}$$

The combinational logic signifies that the next cycle encoding for "xy" (N_{xy}) will be same as it is in the current cycle (C_{xy}) if the symbol encoding (E_1E_0) is less than C_{xy}. N_{xy} would be C_{xy} incremented by 1 if symbol encoding (E_1E_0) is greater than C_{xy} and N_{xy} will be 0 if the symbol encoding is C_{xy}, because for MTF, in the imaginary list, the incoming symbol needs to be moved from its current location to the top of the list (index 0) and the symbol positions between the current location and index 0 are incremented by 1.

For the TR encoder, the combinational logic has the following functionality:

$$\begin{aligned} N_{xy} &= C_{xy} - 1, && \text{if } Y_1Y_0 = C_{xy} \text{ and } C_{xy} \neq 0 \\ &= C_{xy} + 1, && \text{if } Y_1Y_0 = C_{xy} + 1 \\ &= C_{xy}, && \text{otherwise.} \end{aligned}$$

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, 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 do not add to any timing violations. Some other minor overheads would be involved at reset because the registers need to be initialized to appropriate values. Since these do not appear in the actual address generation path, these paths are not considered critical.

B. Decoder Implementation of Self-Organizing Lists

The decoding for the self-organizing lists based encoding is achieved by maintaining another list at the decoder end. The list at the decoder maintains consistency with that at the encoder by organizing itself as per the encodings received at the decoder. We assume that the initial positioning of the symbols in the lists at both encoder and decoder are the same. 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 symbol coding. Fig. 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 (E_1E_0) is fed to the multiplexer (SEL_MUX) to extract the corresponding symbol (A_1A_0) from the list. Since the encoding symbol (E_1E_0) reflects the index of the symbol, it 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 Fig. 2 determine the symbol that will replace the corresponding location. For MTF, the symbol at the index needs to be moved to the top of the list. The multiplexer at the top of the list has four inputs because the symbol at any index can move to the top of the list. The remaining multiplexers have have two inputs because for MTF the corresponding registers either retain the value or obtain the value from the preceding position. The decoder for the TR based implementation is similar to the 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 V.

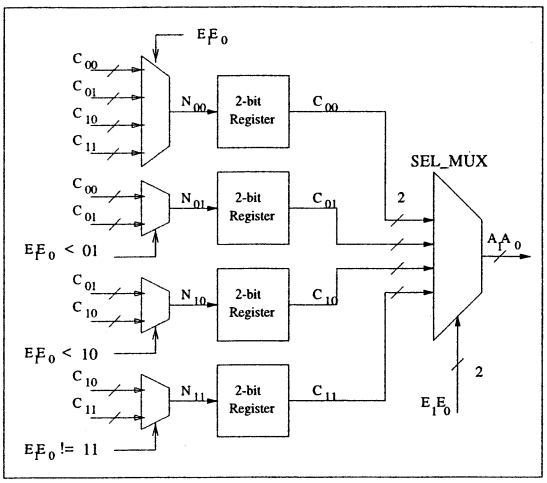

Fig. 2. Architecture for MTF-based decoder.

IV. 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. When the addresses are sequential, most of the transitions occur on a few of the least significant bits. Accordingly, 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 most of the transitions. More specifically, the four least significant bits contribute to approximately 93.75% of the total transitions in sequential addresses. So we propose a heuristic to use self-organizing lists based techniques (MTF/TR) on the higher significant bits to reduce transitions by exploiting the principle of locality in address streams, and use Delta-TS/INC-XOR techniques on the least significant bits to reduce the transitions in sequential addresses. We describe the INC-XOR and Delta-TS techniques as follows. While INC-XOR technique is proposed by Ramprasad *et al.*, we propose an alternative technique, Delta-TS in this paper. This heuristic would not be applicable to data address buses because data addresses typically are nonsequential and hence Delta-TS and INC-XOR would not yield any reductions.

- The INC-XOR encoding technique [14] 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 obtain significant reductions in transition activity. The comparison of reductions for INC-XOR (applied on four least significant bits) along with MTF/TR, against INC-XOR applied on the whole address bus is shown in Section V.

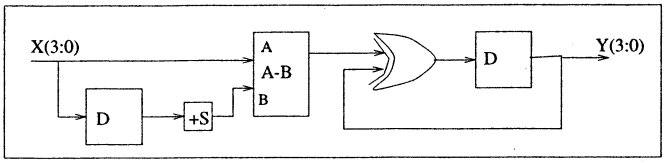

Fig. 3. Delta-TS encoder.

- The Delta-TS encoding technique transmits the difference between the previous address incremented by a constant and the current address with transition signaling (TS), TS being the exclusive or of the previous transmitted vector and the current input vector. Since the subtractor involved in delta calculation is only four bits wide, efficient look-up-table (LUT) based implementations can be used to lower delay overhead. The structure of a 4-bit Delta-TS based encoder is shown in Fig. 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 V.

V. Experiments

We now present results of our low-power address encoding techniques. The SPEC95 benchmark set is used for evaluating the encoding techniques. Also experimental results are presented over a set of typical application programs, denoted by TAP, contains the UNIX compression and mpeg audio decoder utilities (gzip and mpg123), 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.

Tables I and II show the percentage reduction in transition activity for the self-organizing lists based encodings applied to the data addresses for SPEC95 and TAP programs, respectively. While the address trace length for SPEC95 programs were in the range of 12.4 to 106.6 million addresses, for TAP programs the trace lengths were in the range of 31 000 to 4 400 000 addresses. As indicated in Section III, 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 two, "%Seq" indicates the percentage of addresses which are sequential. Column three, "actual (Tr/Ad)" indicates the average transitions per address that occur on the bus without any application of bus encoding on the address stream. Columns four through eight 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

TABLE I
TRANSITION ACTIVITY REDUCTIONS USING MTF AND TR FOR SPEC95 BENCHMARKS ON DATA ADDRESS BUS

	%seq	Actual (Tr/Ad)	MTF(+TS) (W=2)	MTF(+TS) (W=3)	TR(+TS) (W=3)	MTF(+TS) (W=4)	TR(+TS) (W=4)
Compress	1.4	7.5	8(15%)	12(24%)	13(22%)	18(30%)	20(26%)
Perl	0.5	7.9	21(18%)	24(28%)	24(29%)	25(31%)	29(33%)
M88ksim	0.0	4.2	6(11%)	4(17%)	14(22%)	11(20%)	14(26%)
Ijpeg	2.6	7.5	19(21%)	27(28%)	25(28%)	28(31%)	27(33%)
Gcc	0.2	8.4	20(21%)	24(29%)	23(30%)	29(33%)	32(37%)
Vortex	0.2	10.5	28(23%)	32(35%)	32(35%)	40(39%)	35(39%)
Go	0.0	10.2	14(18%)	19(25%)	18(27%)	27(34%)	25(35%)
Average	0.7	8.0	17(18.1%)	20(26.6%)	21(27.6%)	25(31.1%)	26(32.7%)

TABLE II
TRANSITION ACTIVITY REDUCTIONS USING MTF AND TR FOR TYPICAL APPLICATION PROGRAMS ON DATA ADDRESS BUS

	%seq	Actual (Tr/Ad)	MTF(+TS) (W=2)	MTF(+TS) (W=3)	TR(+TS) (W=3)	MTF(+TS) (W=4)	TR(+TS) (W=4)
Gzip	0.4	10.1	23(31%)	34(44%)	36(47%)	40(52%)	42(54%)
Mpg123	0.1	7.6	21(24%)	31(33%)	30(32%)	35(40%)	32(37%)
Ls	4.0	8.5	19(19%)	26(30%)	22(29%)	32(37%)	27(36%)
Who	8.2	8.9	24(18%)	33(31%)	31(33%)	36(37%)	33(40%)
Date	8.4	9.7	16(18%)	31(32%)	27(31%)	36(38%)	31(39%)
Factorial	5.3	9.6	14(14%)	27(27%)	21(25%)	32(33%)	26(30%)
Sort	4.6	10.0	20(16%)	28(29%)	22(28%)	33(38%)	26(35%)
Average	4.4	9.2	20(20.0%)	30(32.3%)	27(32.1%)	35(39.3%)	31(38.7%)

in transition activity is often achieved. The results shown in the table indicate the reductions before and after TS. The value in the brackets are the reductions when TS is applied on top of the corresponding encoding scheme.

Note that the reductions increase with increasing bus width in both program sets. However, with increasing bus-widths, the delay overhead due to the encoding/decoding increases rapidly. 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. Importantly, it can be observed that the reduction in transition activity obtained on the data address buses is consistent for both the larger benchmarks and smaller application programs. Also the reductions were observed to be more than average for programs with higher average transitions per address. This could be because of frequent switching between different data address zones with higher Hamming distance between them. Since this kind of data address bus activity moves the frequently accessed address spaces to the top of the self organizing lists, the reductions seem to be higher. A maximum reduction of 54% was achieved by TR + TS with $W = 4$ for the gzip program. Except for few cases, the use of TS on top of MTF/TR yielded further reductions. This reduction on average for $W = 4$, is considerable in case of TR(∼8%), but is less significant for MTF (∼5%).

Fig. 4 shows the comparison of these reductions with those obtained using existing techniques. The bars indicate the reduction in transition activity for MTF + TS($W = 4$), TR + TS($W = 4$), Gray coding, INC-XOR, and bus-invert coding ($W = 8$). Since bus-invert coding would be effective on smaller buses, the 32-bit address bus is split into four 8-bit buses and applied to them independently. As expected, since the addresses

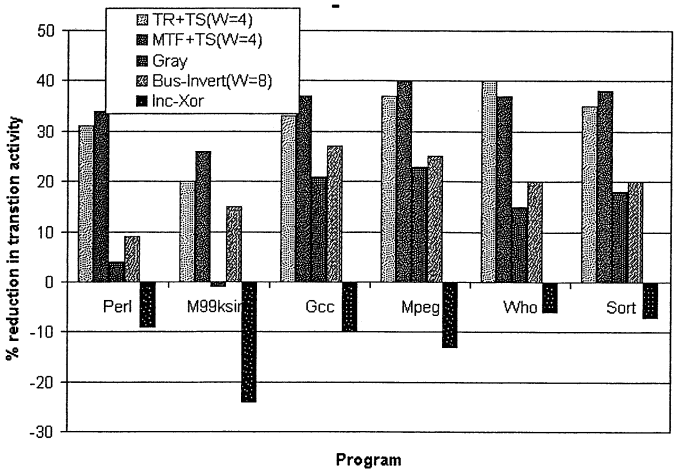

Fig. 4. Comparison of transition activity reductions for various encodings on data address bus.

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 four extra bit lines for implementation which may not be tolerable. It is observed that the self-organizing lists based techniques are consistent and 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).

Tables III and IV show the reduction in transition activity for various combination of encoding techniques on the multiplexed address bus in SPEC95 and TAP programs, respectively. Column two in these tables indicates the percentage of

TABLE III
TRANSITION ACTIVITY REDUCTIONS OF VARIOUS ENCODING TECHNIQUES FOR SPEC95 BENCHMARKS ON MULTIPLEXED ADDRESS BUS

	%Instr. Addr.	%seq	Actual (Tr/Ad)	MTF+ Inc-Xor	TR+ Inc-Xor	MTF+ Delta	TR+ Delta	MTF Only	TR Only
Compress	61	19	9.4	40%	37%	41%	38%	51%	42%
Perl	79	50	5.9	44%	34%	43%	34%	18%	15%
M88ksim	79	59	4.5	45%	30%	45%	29%	22%	7%
Ijpeg	80	55	6.8	49%	48%	49%	48%	37%	36%
Gcc	80	52	7.3	51%	46%	51%	46%	32%	29%
Vortex	78	52	7.8	51%	46%	51%	46%	33%	30%
Go	81	54	5.8	45%	39%	45%	40%	15%	13%
Average	76.8	48.7	6.8	46.4%	40.0%	46.4%	40.2%	29.7%	24.5%

TABLE IV
TRANSITION ACTIVITY REDUCTIONS OF VARIOUS ENCODING TECHNIQUES FOR TYPICAL APPLICATION PROGRAMS ON MULTIPLEXED ADDRESS BUS

	%Instr. Addr.	%seq	Actual (Tr/Ad)	MTF+ Inc-Xor	TR+ Inc-Xor	MTF+ Delta	TR+ Delta	MTF Only	TR Only
Gzip	79	57	8.1	58%	56%	59%	57%	49%	47%
Mpg123	73	43	7.4	48%	46%	48%	46%	32%	30%
Ls	84	57	5.4	46%	33%	47%	34%	23%	12%
Who	86	58	5.3	47%	36%	48%	38%	26%	18%
Date	87	60	5.4	50%	39%	51%	41%	25%	17%
Factorial	87	62	4.6	49%	34%	51%	35%	26%	13%
Sort	85	60	5.2	49%	38%	50%	39%	25%	17%
Average	83.0	56.7	5.9	49.6%	40.3%	50.6%	41.4%	29.4%	22.0%

instruction addresses in the total number multiplexed addresses. As expected, the percentage of instruction addresses dominate the data addresses in the multiplexed address stream. 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.

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 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. Fig. 5 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%). Similar to the data address bus, the reductions on multiplexed address bus due to the encoding techniques seem to be greater for applications with a greater number of transitions per address.

Each encoding scheme incurs some area and delay overhead. Table V 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 0.6 μm LSI_10K library and the synthesis was done for a

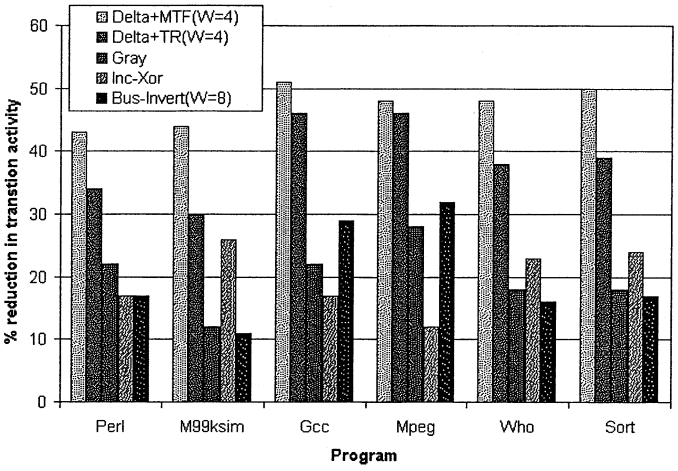

Fig. 5. Comparison of transition activity reductions for various encodings on multiplexed address bus.

32-bit address bus. For MTF and TR the synthesis was done for $W = 4$, and for bus-invert, synthesis was done for $W = 8$ (i.e., the same parameters used in the previous experiments).

The asterisk for bus-invert 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 obtained using existing techniques, we believe that this extra overhead in area is acceptable. If the area overhead is a concern, it can be substantially reduced by applying this technique on a fewer number of bits (say $W = 3$). Another overhead

TABLE V
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

TABLE VI
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

that needs analysis is the power dissipation due to encoder and decoder. For TR, because of the structure of the implementation, for any given input, there can be a change in encoding for at most two 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 is approximately 600 for MTF and 150 for TR. Note that the I/O capacitance is at least three orders of magnitude more than that of the internal capacitance [10]. 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 VI for a 4-bit address bus. The delay overhead of Delta-TS is higher than that of the INC-XOR technique because of the 4-bit subtractor needed for calculating the difference between the current address and the previous address. However, the reduction in transition activity by using the Delta-TS technique is marginally more than that obtained by using INC-XOR technique. Also, note that the area overhead of these modules is minimal.

VI. 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 to the least significant bits, the self-organizing lists based encoding are applied to the more significant bits of the multiplexed address bus. This enables exploitation of both the sequential nature of instruction addresses as well as the locality of addresses in multiplexed address buses. The proposed techniques consistently outperform the existing techniques in both data address 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%. An analysis of the synthesized encoder and decoder show that the delay overhead of the proposed encoding techniques is comparable to that of existing techniques, but the area overhead and power overhead seem to be more for the proposed techniques than for the existing techniques. However, the huge reductions in transition activity compared to the existing techniques compensates for the overhead in power. Future work will involve more efficient design of encoder and decoder with smaller area and power overheads. To quantify the reduction in power dissipation on address buses using different encoding techniques for a given application, it is necessary to have power models for the encoders/decoder. We plan to develop such models which can estimate power dissipation in address bus encoders and decoders. We also plan to look at the applicability of the proposed techniques to data buses.

ACKNOWLEDGMENT

The authors wish to thank the anonymous reviewers for their helpful comments on the manuscript.

REFERENCES

[1] L. Benini, A. Macii, E. Macii, M. Poncino, and R. Scarsi, "Architectures and synthesis algorithms for power-efficient bus interfaces," *IEEE Trans. Computer-Aided Design*, vol. 19, pp. 969–980, 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 *Proc. Great Lakes Symp. VLSI*, Urbana, IL, 1997, pp. 77–82.
[3] A. P. Chandrakasan and R. W. Broderson, "Minimizing power consumption in digital CMOS circuits," *Proc. IEEE*, vol. 83, p. 498, 523, 1995.
[4] W.-C. Cheng and M. Pedram, "Low power techniques for address encoding and memory allocation," in *Proc. Asia and South Pacific Design Automation Conf. (ASP-DAC)*, 2001, pp. 245–250.
[5] R. F. Cmelik and D. Keppel, "Shade: A fast instruction-set simulator for execution profiling," Univ. Washington, Seattle, Tech. Rep. UW-CSE-93-06-06, 1993.
[6] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*. San Mateo, CA: Morgan Kaufmann, 1996.
[7] J. Hester and D. S. Hirschberg, "Self-organizing linear search," *Computing Surveys*, vol. 17, p. 295, 311, 1985.
[8] M. Mamidipaka, D. Hirschberg, and N. Dutt. (2001) Encoding techniques for low power address buses. Univ. Calif. Irvine, Irvine, CA. [Online] Tech. Rep. #01-22, http://www.ics.uci.edu/~maheshmn/encoding_tr.doc.
[9] ——, "Low power address encoding using self-organizing lists," in *Proc. Int. Symp. Low-power Electron. and Design (ISLPED)*, 2001.
[10] E. Musoll, T. Lang, and J. Cortadella, "Working-zone encoding for reducing the energy in microprocessor address buses," *IEEE Trans. VLSI Syst.*, vol. 6, pp. 568–572, Dec. 1998.
[11] F. Najm, "Transition density, a stochastic measure of activity in digital circuits," in *Proc. Design Automation Conf.*, 1991, pp. 644–649.
[12] M. Pedram, "Power minimization in IC design: Principles and applications," *ACM Trans. Design Automation Electron. Syst.*, vol. 1, pp. 3–56, 1996.
[13] M. Pedram and H. Vaishnav, "Power optimization in VLSI layout: A survey," *The J. VLSI Signal Process. Syst. Signal, Image, Video Technol.*, vol. 15, pp. 221–232, 1997.
[14] S. Ramprasad, N. R. Shanbag, and I. N. Hajj, "A coding framework for low power address and data busses," *IEEE Trans. VLSI Syst.*, vol. 7, pp. 212–221, 1999.
[15] M. R. Stan and W. P. Burleson, "Bus-invert coding for low-power I/O," , vol. 3, pp. 49–58, 1995.
[16] ——, "Low-power encodings for global communications in CMOS VLSI," *IEEE Trans. VLSI Syst.*, vol. 5, pp. 444–455, 1997.
[17] C. L. Su, C. Y. Tsui, and A. M. Despain, "Saving power in the control path of embedded processors," *IEEE Des. Test Comput.*, vol. 11, pp. 24–30, 1994.
[18] N. Weste and K. Eshragian, *Principles of CMOS VLSI Design, A Systems Perspective*. Reading, MA: Addison-Wesley, 1998.

Mahesh N. Mamidipaka received the B.Tech. degree in electronics and communication engineering from the Regional Engineering College, Andhra Pradesh, India, in 1997 and the M.E. degree from the Indian Institute of Sciences, Bangalore, in 1999. He is currently working toward the Ph.D. degree at the University of California, Irvine.

From 1999 to 2000, he was an IC Design Engineer with the Digital Signal Processing Group of Texas Instruments, Bangalore, India. His research interests include high-level power estimation of VLSI circuits and low-power architectures for embedded systems.

Mr. Mamidipaka is a Student Member of ACM and SIGDA. He received the Best Student Paper Award at the International Conference on VLSI Design in 1999.

Daniel S. Hirschberg received the B.E. degree in electrical engineering from City College of New York in 1971, and the M.S.E., M.A., and Ph.D. degrees from Princeton University, Princeton, NJ, in 1973, 1973, and 1975, respectively.

From 1975 through 1981, he was an Assistant Professor in the Department of Electrical Engineering at Rice University, Houston, TX. Since 1981, he has been with the University of California, Irvine, where he is currently Professor in both the Department of Information and Computer Science (ICS) and the Department of Electrical and Computer Engineering. For several years, he served as Associate Chair of Graduate Studies and then of Undergraduate Studies with the ICS Department. His research interests include the design and analysis of combinatorial algorithms, and data structures. He is author of numerous articles in these research areas and has been a Consultant for public and private industry. From 1988 to 1990, he was an Associate Editor of the *ACM Transactions on Mathematical Software*. In 1996, he served as Program Co-chair for the Combinatorial Pattern Matching Conference. He has served on the technical program committees for Data Commpression Conference (DCC), Combinational Patern Matching (CPM), and String Processing and Information Retrieval (SPIRE).

Dr. Hirschberg is a Member of the Association of Computing Machinery (ACM) and Special Interest Group on Algorithms and Computing Theroy (SIGACT).

Nikil D. Dutt (S'82–M'84–SM'96) received the Ph.D. in computer science from the University of Illinois at Urbana-Champaign in 1989.

He is currently a Professor in the Center for Embedded Computer Systems at the University of California, Irvine, with academic appointments in the Department of Information and Computer Science (ICS) and the Department of Electrical and Computer Engineering (ECE). His research interests include embedded computer systems design automation, computer architecture, and optimizing compilers. He is coauthor of *High-Level Synthesis: Introduction to Chip and System Design* (Norwell, MA: Kluwer, 1992), *Memory Issues for Embedded Systems-on-Chip* (Norwell, MA: Kluwer, 1999), and *Memory Architecture Exploration for Programmable Embedded Systems* (Norwell, MA: Kluwer, 2002).

Dr. Dutt has served as an Associate Editor for IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION SYSTEMS and currently serves as an Associate Editor for *ACM Transactions on Design Automation of Electronic Systems*. He has served as Topic Chair for architectural synthesis for DATE and ICCAD, and has served on the program committees of several design automation and embedded systems conferences, including the International Federation for Information Processing Working Group (ICCAD), DATE, ASPDAC, International Symposium on System Synthesis (ISSS), Languages, Compilers, and Tools for Embedded Systems (LCTES) and CASES. He currently serves on the Advisory Board of ACM-SIGDA and also serves as Vice Chair of IFIP WG 10.5.