Two-Dimensional Codes for Low Power

Mircea R. Stan
EE Department, U. of Virginia
m.stan@ieee.org

Wayne P. Burleson
ECE Department, U. of Massachusetts
burleson@galois.ecs.umass.edu

Abstract| **Coding was previously proposed for reducing power consumption in CMOS. The original formulations use extra redundancy in *space* (number of bus lines) for reducing the bus transition activity (and consequently the dynamic power and simultaneous switching noise). This paper proposes several new coding techniques for low power. First it looks at codes in which redundancy in *time* is used for reduced bus activity. Two-dimensional codes with redundancy in both time and space can then be developed for extra power reduction. Interestingly, these two-dimensional codes can be unrolled in either space or time in order to obtain new one-dimensional codes in the other dimension. More powerful codes using Run-Length Limited (RLL), phase-modulation and amplitude-modulation techniques are finally proposed.**

Keywords| **Low-power interconnect, coding.**

I. INTRODUCTION

Coding was previously proposed for reducing power consumption in CMOS. Gray codes were used for address buses [16], [12], [14], sign-magnitude representations for DSP applications [2], [3], Bus-Invert [12] and Limited-Weight Codes [11] for data buses. All these codes use redundancy in *space* (number of bus lines) for reducing the bus transition activity (and consequently dynamic power and simultaneous switching noise). Similar techniques were also proposed for reducing power on a class of terminated buses [13]. A remarkable and non-intuitive fact about such encodings is the reduction in overall power in spite of increases in both the *total* area and the *total* number of transitions. Some of the power estimation inaccuracies based on information entropy [6], [5] can be better understood in light of such encodings for which the source entropy remains unchanged (no information loss) but the power varies. Several assumptions (see also [11], [12]) are made in this paper:

- The data on the bus is random uniformly distributed. This assumption approximates fairly well the conditions on a data bus and is only necessary for the analysis of the low-power effectiveness of the proposed encodings. If the assumption does not hold and the data is correlated, a good low-power solution is to *compress* (lossless) the data before transmitting it over the bus. Compressing data will restore the randomness assumption and will also reduce power. Figure 1 shows the effects of compression and Bus-Invert encoding [12] on the number of transitions generated when transferring several typical Unix files over an 8-bit bus. It should be clear that compression/decompression is desirable only for buses where the extra latency can be tolerated or masked (e.g. main-memory or system buses).
- The bus can be on- or off-chip and it typically involves sending *packets* of data over highly-capacitive bus lines. The load capacitance on such a bus line is 2-3 orders of magnitude larger than on-chip loads [1]. This enables us to ignore (within limits) the extra power consumed by the minimum-size on-chip coder and decoder if they have reasonable complexity [11], [12]. For example if a bus load is 2 orders of magnitude larger than normal on-chip loads and the encoding requires 10 extra transitions in the encoder/decoder for each transition saved on the bus, then the extra power consumed by encoding will still be 10 times lower than the amount of saved power.
- *Transition signalling* is used instead of level signalling (transitions denote logical 1's and lack of transitions denote logical 0's). This was a key element in the initial development of Limited-Weight Codes [11] and we believe it is an ideal way of controlling low-power coding efficiency by the number of 1's in the encoding. In a very different context, transition signalling was also found convenient by the asynchronous design community when they adopted Signal Transition Graphs (STG's) over state diagrams for describing asynchronous behavior [4]. Transition signalling with ca-

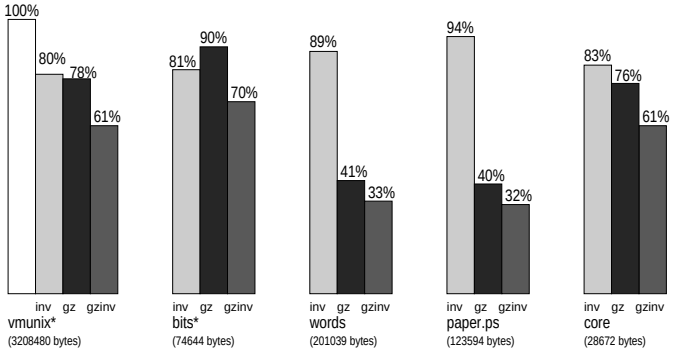

Fig. 1. Effects of compression and encoding on the normalized number of transitions for five typical Unix files. The white bar represents the initial number of transitions (normalized to 100%), the light grey bars represent the number of transitions after Bus-Invert, the dark grey bars represent the number of transitions after compression (gzip) and finally the medium grey bars represent the number of transitions after both compression and Bus-Invert.

This work was performed while the first author was a PhD student in the ECE Department at the U. of Massachusetts

ISLPED 1996 Monterey CA USA
0-7803-3571-8/96/$5.00 © 1996

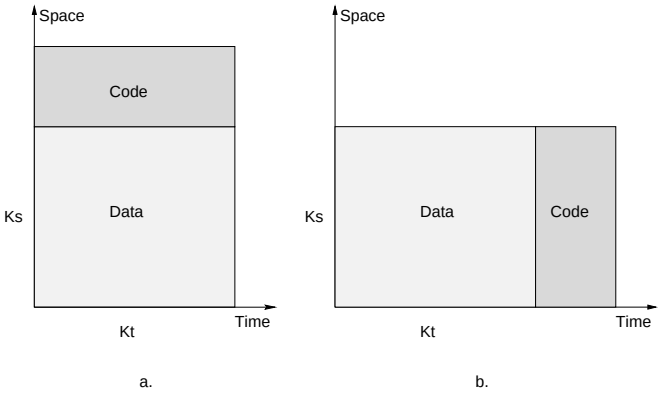

Fig. 2. A typical packet of data is transmitted over a K_s-bit wide bus in K_t clock periods. a.) Extra redundancy in space can be added for reducing the bus activity while transmitting the data over more bus lines. b.) If the increase in number of bus lines is not desired the extra redundancy can be added in time.

pacitive coupling was also proposed for solving the "known-good die" problem for MCM's [9].

The paper is organized as follows. Section II first proposes straightforward coding extensions in which transition signalling and redundancy in *time* are used for minimizing transition activity. Then *two-dimensional* codes with redundancy in both time and space are proposed for extra power reduction. These two-dimensional codes can be unrolled in either space or time in order to obtain new one-dimensional codes in the other dimension. Section III looks at Run-Length Limited (RLL) and phase-modulation techniques which use the extra freedom in the time domain for obtaining better codes for low power. Redundancy in *amplitude* can then be combined with time redundancy for obtaining other two-dimensional codes for low-power.

II. CODING IN SPACE AND TIME FOR LOW POWER

Initial low-power encodings were defined using redundancy in *space* in the form of extra bus lines. One problem with such encodings is that the required number of extra bus lines increases exponentially as the transition activity is reduced, and this can make the technique non-practical [11]. An alternative is to keep the number of bus lines constant and inject redundancy in *time* by using extra transfer cycles.

A. Coding in time for low-power

Time encoding requires that the data be transmitted in packets but this is not a big constraint on global buses where there is now a clear trend of transmitting bursts of data for improved throughput [8]. Figure 2a. shows a data-packet with redundancy in space, while 2b. shows the same packet with redundancy added in time. With the transmitted data arranged into K_t-word packets where each word is initially K_s-bit wide, the same coding techniques that in [11], [12] used redundancy in space can now be applied in time. In particular Bus-Invert [12] and Limited-Weight codes [11] with redundancy in time and transition signalling, can use extra transfer cycles for encoding the K_t bits that

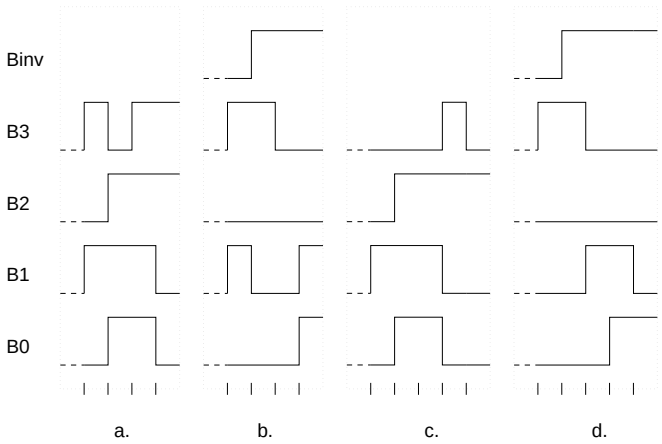

Fig. 3. a.) This packet of four 4-bit words transmitted over a 4-line bus (B0-B3) generates 8 transitions (with transition signalling). b.) The same packet encoded in space with Bus-Invert (extra bus line Binv) generates only 7 transitions. c.) With redundancy in time (extra clock cycle) the packet also generates 7 transitions. d.) With redundancy in both space and time only 6 transitions are generated.

are successively transmitted over each bus line in order to minimize the number of 1's transmitted and hence the number of transitions. For example, if $K_t = 4$ the equivalent Bus-Invert time encoding would look at the number of 1's that are to be transmitted over each bus-line. If the number of 1's for a bus-line is greater than $K_t/2 = 2$, then the $K_t = 4$ bits on that line will be inverted and this inversion will be signalled by a 1 in an extra (5^{th}) transfer cycle. Otherwise, the $K_t = 4$ bits are transmitted as they are and the extra (5^{th}) bit will be 0. The computation of the redundant bit needs to be done for each of the K_s bus-lines, in series or in parallel.

By simple probabilistic reasoning it can be shown that with the same amount of redundancy, time encodings will have the same average power savings as the equivalent space encodings. As an example, transmitting the following 4-word packet takes 4 cycles and generates 8 transitions over a 4-line bus (with transition signalling, see figure 3a.). It is assumed that the 4-bit words are arranged in columns and are transferred from left to right:

$$\begin{matrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{matrix}$$

With redundancy in space or time (Bus-Invert with transition signalling) the number of 1's (hence transitions) is reduced to 7 at the expense of an extra bus line (the first row on left) or an extra transfer cycle (the last column on right, see also figures 3b. and 3c.):

$$\begin{array}{cccc} 0 & 1 & 0 & 0 \\ \hline 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{array} \qquad \begin{array}{cccc|c} 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \end{array}$$

The average I/O power dissipation for packets of four 4-bit words with coding in one dimension (space or time) is

TABLE I

Two-dimensional codes with 4 information bits and 4 code bits. Encoding is done column-wise followed by row-wise, or vice-versa (in parentheses).

$\begin{matrix} & & 0\ 0 \\ 0\ 0 & \Rightarrow & 0\ 0\ 0 \\ 0\ 0 & & 0\ 0\ 0 \end{matrix}$ $\left(\begin{matrix} 0\ 0 \\ 0\ 0\ 0 \\ 0\ 0\ 0 \end{matrix}\right)$	$\begin{matrix} & & 0\ 0 \\ 1\ 0 & \Rightarrow & 1\ 0\ 0 \\ 0\ 0 & & 0\ 0\ 0 \end{matrix}$ $\left(\begin{matrix} 0\ 0 \\ 1\ 0\ 0 \\ 0\ 0\ 0 \end{matrix}\right)$	$\begin{matrix} & & 0\ 0 \\ 0\ 1 & \Rightarrow & 0\ 1\ 0 \\ 0\ 0 & & 0\ 0\ 0 \end{matrix}$ $\left(\begin{matrix} 0\ 0 \\ 0\ 1\ 0 \\ 0\ 0\ 0 \end{matrix}\right)$	$\begin{matrix} & & 0\ 0 \\ 1\ 1 & \Rightarrow & 0\ 0\ 1 \\ 0\ 0 & & 0\ 0\ 0 \end{matrix}$ $\left(\begin{matrix} 0\ 0 \\ 0\ 0\ 1 \\ 0\ 0\ 0 \end{matrix}\right)$
$\begin{matrix} & & 0\ 0 \\ 0\ 0 & \Rightarrow & 0\ 0\ 0 \\ 1\ 0 & & 1\ 0\ 0 \end{matrix}$ $\left(\begin{matrix} 0\ 0 \\ 0\ 0\ 0 \\ 1\ 0\ 0 \end{matrix}\right)$	$\begin{matrix} & & 1\ 0 \\ 1\ 0 & \Rightarrow & 0\ 0\ 0 \\ 1\ 0 & & 0\ 0\ 0 \end{matrix}$ $\left(\begin{matrix} 1\ 0 \\ 0\ 0\ 0 \\ 0\ 0\ 0 \end{matrix}\right)$	$\begin{matrix} & & 0\ 0 \\ 0\ 1 & \Rightarrow & 0\ 1\ 0 \\ 1\ 0 & & 1\ 0\ 0 \end{matrix}$ $\left(\begin{matrix} 0\ 0 \\ 0\ 1\ 0 \\ 1\ 0\ 0 \end{matrix}\right)$	$\begin{matrix} & & 1\ 0 \\ 1\ 1 & \Rightarrow & 0\ 1\ 0 \\ 1\ 0 & & 0\ 0\ 0 \end{matrix}$ $\left(\begin{matrix} 0\ 0 \\ 0\ 0\ 1 \\ 1\ 0\ 0 \end{matrix}\right)$
$\begin{matrix} & & 0\ 0 \\ 0\ 0 & \Rightarrow & 0\ 0\ 0 \\ 0\ 1 & & 0\ 1\ 0 \end{matrix}$ $\left(\begin{matrix} 0\ 0 \\ 0\ 0\ 0 \\ 0\ 1\ 0 \end{matrix}\right)$	$\begin{matrix} & & 0\ 0 \\ 1\ 0 & \Rightarrow & 1\ 0\ 0 \\ 0\ 1 & & 0\ 1\ 0 \end{matrix}$ $\left(\begin{matrix} 0\ 0 \\ 1\ 0\ 0 \\ 0\ 1\ 0 \end{matrix}\right)$	$\begin{matrix} & & 0\ 1 \\ 0\ 1 & \Rightarrow & 0\ 0\ 0 \\ 0\ 1 & & 0\ 0\ 0 \end{matrix}$ $\left(\begin{matrix} 0\ 1 \\ 0\ 0\ 0 \\ 0\ 0\ 0 \end{matrix}\right)$	$\begin{matrix} & & 0\ 1 \\ 1\ 1 & \Rightarrow & 1\ 0\ 0 \\ 0\ 1 & & 0\ 0\ 0 \end{matrix}$ $\left(\begin{matrix} 0\ 0 \\ 0\ 0\ 1 \\ 0\ 1\ 0 \end{matrix}\right)$
$\begin{matrix} & & 0\ 0 \\ 0\ 0 & \Rightarrow & 0\ 0\ 0 \\ 1\ 1 & & 0\ 0\ 1 \end{matrix}$ $\left(\begin{matrix} 0\ 0 \\ 0\ 0\ 0 \\ 0\ 0\ 1 \end{matrix}\right)$	$\begin{matrix} & & 1\ 0 \\ 1\ 0 & \Rightarrow & 0\ 0\ 0 \\ 1\ 1 & & 0\ 1\ 0 \end{matrix}$ $\left(\begin{matrix} 0\ 0 \\ 1\ 0\ 0 \\ 0\ 0\ 1 \end{matrix}\right)$	$\begin{matrix} & & 0\ 1 \\ 0\ 1 & \Rightarrow & 0\ 0\ 0 \\ 1\ 1 & & 1\ 0\ 0 \end{matrix}$ $\left(\begin{matrix} 0\ 0 \\ 0\ 1\ 0 \\ 0\ 0\ 1 \end{matrix}\right)$	$\begin{matrix} & & 1\ 1 \\ 1\ 1 & \Rightarrow & 0\ 0\ 0 \\ 1\ 1 & & 0\ 0\ 0 \end{matrix}$ $\left(\begin{matrix} 0\ 0 \\ 0\ 0\ 1 \\ 0\ 0\ 1 \end{matrix}\right)$

TABLE II

Two-dimensional codes with 4 information bits and 5 code bits. Encoding is done column-wise followed be row-wise, or vice-versa (in parentheses).

$\begin{matrix} & & 0\ 0\ 0 \\ 0\ 0 & \Rightarrow & 0\ 0\ 0 \\ 0\ 0 & & 0\ 0\ 0 \end{matrix}$ $\left(\begin{matrix} 0\ 0\ 0 \\ 0\ 0\ 0 \\ 0\ 0\ 0 \end{matrix}\right)$	$\begin{matrix} & & 0\ 0\ 0 \\ 1\ 0 & \Rightarrow & 1\ 0\ 0 \\ 0\ 0 & & 0\ 0\ 0 \end{matrix}$ $\left(\begin{matrix} 0\ 0\ 0 \\ 1\ 0\ 0 \\ 0\ 0\ 0 \end{matrix}\right)$	$\begin{matrix} & & 0\ 0\ 0 \\ 0\ 1 & \Rightarrow & 0\ 1\ 0 \\ 0\ 0 & & 0\ 0\ 0 \end{matrix}$ $\left(\begin{matrix} 0\ 0\ 0 \\ 0\ 1\ 0 \\ 0\ 0\ 0 \end{matrix}\right)$	$\begin{matrix} & & 0\ 0\ 0 \\ 1\ 1 & \Rightarrow & 0\ 0\ 1 \\ 0\ 0 & & 0\ 0\ 0 \end{matrix}$ $\left(\begin{matrix} 0\ 0\ 0 \\ 0\ 0\ 1 \\ 0\ 0\ 0 \end{matrix}\right)$
$\begin{matrix} & & 0\ 0\ 0 \\ 0\ 0 & \Rightarrow & 0\ 0\ 0 \\ 1\ 0 & & 1\ 0\ 0 \end{matrix}$ $\left(\begin{matrix} 0\ 0\ 0 \\ 0\ 0\ 0 \\ 1\ 0\ 0 \end{matrix}\right)$	$\begin{matrix} & & 1\ 0\ 0 \\ 1\ 0 & \Rightarrow & 0\ 0\ 0 \\ 1\ 0 & & 0\ 0\ 0 \end{matrix}$ $\left(\begin{matrix} 1\ 0\ 0 \\ 0\ 0\ 0 \\ 0\ 0\ 0 \end{matrix}\right)$	$\begin{matrix} & & 0\ 0\ 0 \\ 0\ 1 & \Rightarrow & 0\ 1\ 0 \\ 1\ 0 & & 1\ 0\ 0 \end{matrix}$ $\left(\begin{matrix} 0\ 0\ 0 \\ 0\ 1\ 0 \\ 1\ 0\ 0 \end{matrix}\right)$	$\begin{matrix} & & 1\ 0\ 0 \\ 1\ 1 & \Rightarrow & 0\ 1\ 0 \\ 1\ 0 & & 0\ 0\ 0 \end{matrix}$ $\left(\begin{matrix} 0\ 0\ 0 \\ 0\ 0\ 1 \\ 1\ 0\ 0 \end{matrix}\right)$
$\begin{matrix} & & 0\ 0\ 0 \\ 0\ 0 & \Rightarrow & 0\ 0\ 0 \\ 0\ 1 & & 0\ 1\ 0 \end{matrix}$ $\left(\begin{matrix} 0\ 0\ 0 \\ 0\ 0\ 0 \\ 0\ 1\ 0 \end{matrix}\right)$	$\begin{matrix} & & 0\ 0\ 0 \\ 1\ 0 & \Rightarrow & 1\ 0\ 0 \\ 0\ 1 & & 0\ 1\ 0 \end{matrix}$ $\left(\begin{matrix} 0\ 0\ 0 \\ 1\ 0\ 0 \\ 0\ 1\ 0 \end{matrix}\right)$	$\begin{matrix} & & 0\ 1\ 0 \\ 0\ 1 & \Rightarrow & 0\ 0\ 0 \\ 0\ 1 & & 0\ 0\ 0 \end{matrix}$ $\left(\begin{matrix} 0\ 1\ 0 \\ 0\ 0\ 0 \\ 0\ 0\ 0 \end{matrix}\right)$	$\begin{matrix} & & 0\ 1\ 0 \\ 1\ 1 & \Rightarrow & 1\ 0\ 0 \\ 0\ 1 & & 0\ 0\ 0 \end{matrix}$ $\left(\begin{matrix} 0\ 0\ 0 \\ 0\ 0\ 1 \\ 0\ 1\ 0 \end{matrix}\right)$
$\begin{matrix} & & 0\ 0\ 0 \\ 0\ 0 & \Rightarrow & 0\ 0\ 0 \\ 1\ 1 & & 0\ 0\ 1 \end{matrix}$ $\left(\begin{matrix} 0\ 0\ 0 \\ 0\ 0\ 0 \\ 0\ 0\ 1 \end{matrix}\right)$	$\begin{matrix} & & 1\ 0\ 0 \\ 1\ 0 & \Rightarrow & 0\ 0\ 0 \\ 1\ 1 & & 0\ 1\ 0 \end{matrix}$ $\left(\begin{matrix} 1\ 0\ 0 \\ 0\ 0\ 0 \\ 0\ 0\ 1 \end{matrix}\right)$	$\begin{matrix} & & 0\ 1\ 0 \\ 0\ 1 & \Rightarrow & 0\ 0\ 0 \\ 1\ 1 & & 1\ 0\ 0 \end{matrix}$ $\left(\begin{matrix} 0\ 0\ 0 \\ 0\ 1\ 0 \\ 0\ 0\ 1 \end{matrix}\right)$	$\begin{matrix} & & 0\ 0\ 1 \\ 1\ 1 & \Rightarrow & 0\ 0\ 0 \\ 1\ 1 & & 0\ 0\ 0 \end{matrix}$ $\left(\begin{matrix} 0\ 0\ 1 \\ 0\ 0\ 0 \\ 0\ 0\ 0 \end{matrix}\right)$

reduced by approximately 21.8% [12]. Unfortunately, the peak power and simultaneous switching noise, which with space encoding were also reduced, are not affected by time encoding as it is still possible to have all bus lines switching simultaneously. Ultimately, choosing the use of space or time encoding lies with the designer since the techniques are similar but the trade-offs involve either extra bus lines or extra transfer cycles. Because of an exponential increase in the required number of redundant bits, one-hot encodings for large K are probably practical only with time redundancy and only if the extra transfer time is acceptable.

B. Coding in both space and time

If power needs to be further reduced, redundancy in both space and time can be used. Two-dimensional coding is a two-step process and there is a choice whether to apply redundancy first column-wise (in space) and then row-wise (in time), or vice-versa. The same average power reduction is obtained in both cases but lower peak simultaneous switching noise can be obtained by encoding first in time and then in space. For the previous example the number of 1's can be reduced to 6 with two-dimensional coding (see also figure 3d.), column-wise encoding is done first on the left, row-wise encoding is done first on the right:

0	1	0	0	0
1	0	1	0	0
0	0	0	0	0
0	0	1	0	0
0	0	0	1	0

0	0	0	1	0
0	0	0	0	1
0	1	0	1	0
1	0	0	0	0
0	1	0	0	0

Table I shows the codewords of the smallest two-dimensional low-power codes, with column-wise encoding followed by row-wise encoding (or vice-versa, in parentheses). There are 16 such codewords, one for each of the 2×2 possible patterns of 1's and 0's. Two extra codebits are used in space and two in time. The average power dissipation is reduced by 31% (compare with less than 25% for one-dimensional Bus-Invert [12]).

Table II shows two other two-dimensional codes. There is an extra 9^{th} bit which encodes in time the space codebits (or vice-versa, in parentheses). The average power dissipation is reduced by 34%, slightly better compared to the previous codes. Again there is a choice whether to encode column-wise followed by row-wise, or vice-versa (in parentheses).

A useful application of such two-dimensional encodings is the generation of new one-dimensional codes by unrolling the two-dimensional code about one dimension. For example

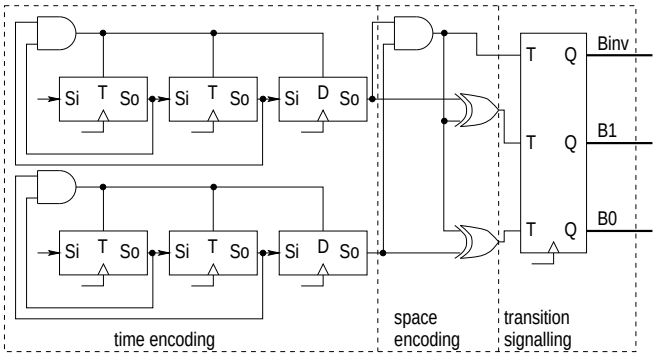

Fig. 4. Two-dimensional encoding in both space and time. Encoding in time (row-wise) is followed by encoding in space (column-wise) and then by transition signalling. Shift registers with parallel D and T inputs are used for time encoding and T registers are also used for transition signalling. There will be at most 2 transitions for each 4 bits of information (9 codebits transmitted).

by unrolling the two-dimensional code in table I, the one-dimensional obtained code is a *semiperfect* 2-Limited-Weight code [11] of length 8. Similarly by unrolling the code in table II the obtained code is a semiperfect 2-LWC of length 9 (by definition a semiperfect M-LWC of length N comprises the all-zeros pattern, *all* the N-bit patterns with 1, 2, ... $M-1$ 1's, *some* N-bit patterns with M 1's and no other patterns [11]). This is an important result for several reasons:

- The algorithmic generation of codes for low-power is intrinsically hard in the general case [11].
- Such unrolled two-dimensional codes provide a compromise between the two extremes of Bus-Invert (minimum redundancy, one extra line [12]) and one-hot encoding (minimum transition activity, one transition per cycle [11]), and offers another practical design alternative.

C. Implementation of coding in space and time

The techniques used for computing the code bits in space and time are similar, which means the circuits can also be similar. A key element is the efficient implementation of a majority voter, and this can be done in a digital or analog fashion [11], [12]. For time redundancy there are also issues related to accessing the entire data packet while encoding and decoding. For encoding, the entire packet must be stored and accessed, but decoding can be done on the fly if the extra code bits are transmitted before the data bits. The block diagram of an encoder for the two-dimensional code in table II (time followed by space encoding) is shown in figure 4. The majority voter in this case is an AND gate and it can be seen that although they are conceptually similar, time encoding is more expensive than space encoding because it needs to access the entire data packet at once.

III. CODING IN AMPLITUDE AND TIME

In section II when we discussed redundancy in time we implicitly assumed that the time domain has exactly the same "integer" restrictions as the space domain but this needs not be the case. While the number of bus lines must always be an integer, time is continuous and we can use the extra freedom for building more efficient low-power codes.

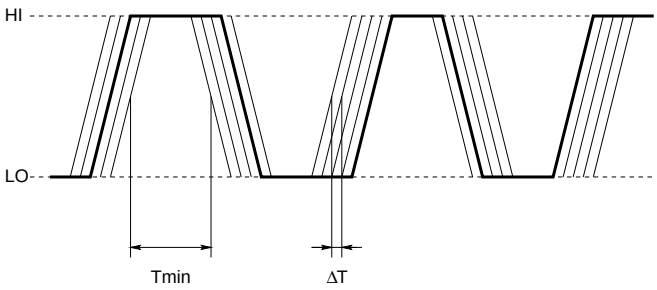

Fig. 5. Phase modulation on a bus line. T_{min} is the minimum pulse width, or the minimum distance between two consecutive transitions. $\triangle T$ is the resolution with which the position of a given transition can be determined.

A. Modulation in time

To understand how this can be done we must analyze the lower bounds on timing values on a bus line. A first bound has to do with the minimum possible width T_{min} for a pulse on the bus. This minimum width is determined by minimum rise and fall times and intersymbol interference. Another bound is given by the maximum resolution $\triangle T$ with which the exact position in time of a transition can be determined. This resolution depends on the amount of noise on the bus and is very much implementation dependent. In section II we implicitly assumed that the two bounds are the same and equal to the "bus cycle". In most cases the resolution $\triangle T$ can be much smaller than the minimum pulse width T_{min} (see figure 5), hence we can use the *position* in time of a transition for encoding several bits per transition [7]. By considering a "virtual" cycle equal to $\triangle T$, and T_{min} as a multiple of $\triangle T$, then in terms of transition signalling the lower bounds translate into the necessity of having a certain number of 0's between any two consecutive 1's (the minimum number of 0's will determine T_{min}). Similar constraints appear naturally in magnetic recording devices and the coding community has developed the class of Run-Length Limited (RLL) codes [10] for improving code efficiency when $\triangle T < T_{min}$. For example, if $T_{min} = 3 \cdot \triangle T$ the very popular variable-length RLL(2,7) code can be used (a RLL(d, k) code has at least d and at most k 0's between any two 1's [10]):

Data	Code
10	0100
11	1000
000	000100
010	100100
011	001000
0010	00100100
0011	00001000

With the RLL(2,7) the average number of transitions is only slightly reduced over the unencoded case, but for a given T_{min} the transfer time is reduced by 50% (hence the energy-delay product is also reduced by 50%). More impressive

TABLE III

p_{opt} for optimal code rate at different values of d with the corresponding average savings in transition activity.

d	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
p_{opt}	4	4	5	6	6	7	7	8	8	9	9	10	10	10	11	11
bits/tr.	2	2	2.3	2.6	2.6	2.8	2.8	3	3	3.2	3.2	3.3	3.3	3.3	3.5	3.5
savings	0%	0%	14%	23%	23%	29%	29%	33%	33%	37%	37%	40%	40%	40%	42%	42%

TABLE IV

p_{low} for optimal energy-delay product at different values of d with the corresponding average savings in transition activity.

d	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
p_{low}	9	11	12	13	15	16	17	18	19	20	21	22	23	24	25	26
bits/tr.	3.2	3.5	3.6	3.7	3.9	4	4.1	4.2	4.2	4.3	4.4	4.5	4.5	4.6	4.6	4.7
savings	37%	42%	44%	46%	49%	50%	51%	52%	53%	54%	54%	55%	56%	56%	57%	58%

TABLE V

Savings in transition activity at p_{opt} as d increases.

d	4	8	16	32	64	128	256
p_{opt}	6	8	11	17	28	45	77
bits/tr.	2.6	3	3.5	4.1	4.8	5.5	6.3
savings	23%	33%	42%	51%	58%	64%	68%

results in low power can be obtained by observing that for a TTL type interface the typical values are $T_{min} \approx 5 \div 10$ ns and $\triangle T \approx 0.3 \div 1.0$ ns [7]. This theoretically enables the use of RLL codes with large values for d and k (e.g. $d = 16$), but for such values RLL codes are impractical to implement.

Phase modulation [7] (see figure 5) is an inefficient (from the information theory point of view) RLL(d, k) code with $d = (T_{min} / \triangle T) - 1$ and $k = d + 2 \cdot (p - 1)$ which encodes several bits of data in the position of a transition. In such a scheme each transition can have one of p different positions (e.g. $p = 5$) and the minimum pulse width is T_{min}. With p positions we can transmit $\log_2 p$ bits per transition, and if p is large there is a potential for important power reductions (in the unencoded case the average rate is 2 bits per transition).

From the low-power coding point of view, phase modulation can be viewed as one-hot encoding with transition signalling with the extra constraint on T_{min} which translates into a necessary string of d 0's in-between any two one-hot codewords. For example, if $T_{min} = 10 \cdot \triangle T$ there will be a string of $d = 9$ 0's between each pair of one-hot codewords.

Let's analyze the coding efficiency of the phase modulation scheme. For every $\log_2 p$ transmitted bits we have to transmit $(d + p) \triangle T$ periods. The code rate [10] will then be:

$$\text{rate} = \frac{\log_2 p}{d + p} \tag{1}$$

Generally T_{min} and $\triangle T$ are given and hence d is also given which means that the only variable is p. The rate is maximized at a value p_{opt} obtained by differentiating (1):

$$p_{opt} \cdot (\ln 2 \log_2 p_{opt} - 1) = d \tag{2}$$

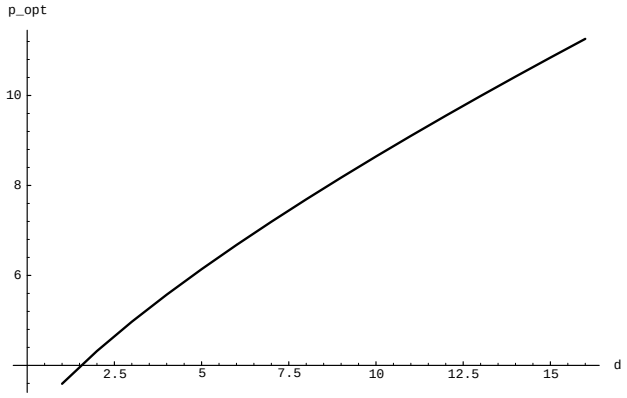

Fig. 6. p_{opt} for optimal code rate levels-off as d increases.

Table III shows the values of p_{opt} (rounded to nearest integer) for different values of d, as well as the number of bits per transition and the average savings in the number of I/O transitions. Figure 6 shows the growth of p_{opt} with d. Although very large values of d are not practical anyhow, it is interesting to note that the growth of p_{opt} with d is less than linear, hence the power savings are not impressive as d increases (see also table V).

Extra power savings can be obtained by realizing that for low power we may not want to use p_{opt} but a somewhat larger value. In (2) p_{opt} was computed for optimal data rate (or minimum transfer time for a given packet) but for low power we are interested more in minimizing the number of transitions than in transfer rate. Another argument is given by figure 7 which shows the code rate as a function of d and p. As can be seen, the code rate has a shallow peak at p_{opt}, which means that we can safely choose a value larger than p_{opt} without much penalty in code rate.

We can quantify this choice by proposing as a measure of low-power code efficiency the *energy-delay product* of the number of transitions in a packet ($\sim 1/\log_2 p$) and the time necessary to transmit the packet ($\sim 1/\text{rate}$). An optimal low-power value for p will minimize (compare with (1)):

$$\text{energy} \cdot \text{delay} \sim \frac{d + p}{(\log_2 p)^2} \tag{3}$$

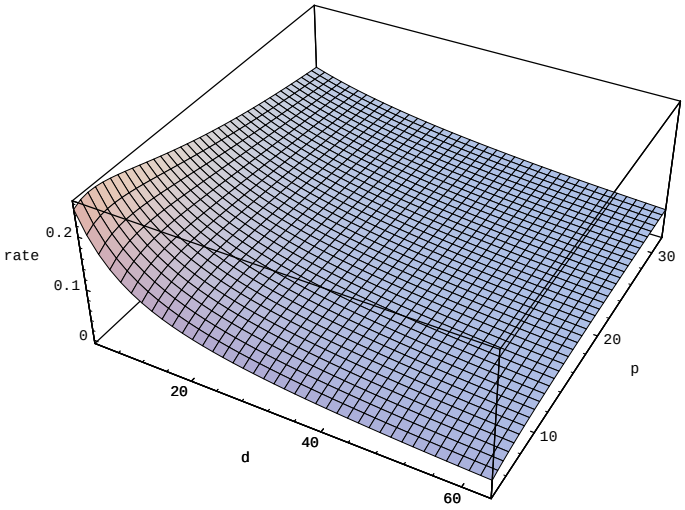

Fig. 7. Variation of coding rate with d and p. For a given d, p_{opt} will be the value of p for which the rate is maximum. As can be seen the rate has a very shallow (in p) peak at p_{opt}.

The equation which determines p_{low} for minimum energy-delay is (compare with (2)):

$$p_{low} \cdot (\ln 2 \log_2 p_{low} - 2) = 2 \cdot d \qquad (4)$$

As expected the optimal p_{low} values are larger and there are better savings in transition activity (up to 58% savings at $d = 16$ for $p_{low} = 26$) as can be seen in table IV.

B. Modulation in both amplitude and time

As in the case of space and time (see section II) we can also define two-dimensional low-power codes that use modulation in both amplitude and time. By modulating the signal amplitude with $2 \cdot p_{volt}$ levels distanced $\triangle V$ apart, in each cycle we can transmit $\log_2 p_{volt}$ bits in addition to the $\log_2 p_{time}$ bits transmitted in time for each transition. Figure 8 shows such a two-dimensional encoding with $p_{time} = 5$ and $p_{volt} = 5$ that can transmit 10 bits per transition for savings of 80% in transition activity.

C. Implementation of amplitude and time encoding

There are many possible implementations for a phase modulation scheme and the one in [7] is very convenient. For encoding and decoding it uses a PLL with a $(p+d)$-stage ring oscillator which can generate the p necessary phases and guarantees the minimum d zeros between two transitions. Other schemes which improve efficiency have been also proposed [15]. We believe that phase modulation is practical and it improves both coding efficiency and low power.

Unfortunately amplitude modulation doesn't seem to be a power efficient option (yet) because very low-power A/D and D/A converters are not known, so two-dimensional codes in amplitude and time will probably remain just a theoretical possibility for the near future.

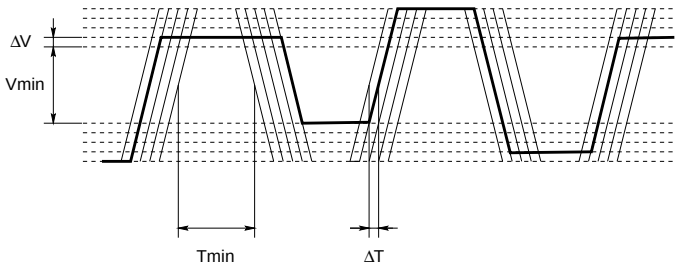

Fig. 8. Modulation in both amplitude and time.

CONCLUSIONS

Several new low-power encoding methods have been presented which can be readily applied in practical designs for reducing I/O power dissipation. The contributions of the paper are:

- Extension of low-power coding to the time domain.
- Two-dimensional space and time encodings for lower power.
- New semiperfect Limited-Weight Codes obtained by unrolling two-dimensional codes.
- Analysis of phase modulation techniques for optimal low power performance.

References

[1] H. B. Bakoglu, *Circuits, Interconnections and Packaging for VLSI*, Addison-Wesley, 1990.

[2] A. P. Chandrakasan, R. Allmon, A. Stratakos, R. W. Brodersen, "Design of Portable Systems", IEEE Custom Integrated Circuits Conference, pp. 259-266, 1994.

[3] A. P. Chandrakasan, R. W. Brodersen, "Minimizing Power Consumption in Digital CMOS Circuits", *Proceedings of the IEEE*, pp. 498-523, April 1995.

[4] S. Hauck, "Asynchronous Design Methodologies: An Overview", *Proceedings of the IEEE*, pp. 69-92, Jan. 1995.

[5] D. Marculescu, R. Marculescu, M. Pedram, "Information Theoretic Measures for Energy Consumption at Register Transfer Level", International Symposium on Low Power Design, pp. 81-86, April, 1995.

[6] F. Najm "Towards a High-Level Power Estimation Capability", International Symposium on Low Power Design, pp. 87-92, April, 1995.

[7] K. Nogami, A. El Gamal, "A CMOS 160MB/s Phase Modulation I/O Interface Circuit", ISSCC'94, pp. 108-109, Feb. 1994, San Francisco.

[8] *Rambus - Architectural Overview*, Rambus Inc., Mountain View, CA, 1993. Contact `ray@rambus.com`.

[9] D. Salzmann, T. Knight, P. Franzon, "Application of Capacitive Coupling to Switch Fabrics", IEEE MCM Conference, pp. 195-199, Jan. 1995, Santa Cruz, CA.

[10] P. H. Siegel, "Recording Codes for Digital Magnetic Storage", *IEEE Trans. on Magnetics*, pp. 1344-1349, Sept. 1985.

[11] M. R. Stan, W. P. Burleson, "Limited-weight codes for low power I/O", International Workshop on Low Power Design, , pp. 209-214, April 1994, Napa, CA.

[12] M. R. Stan, W. P. Burleson, "Bus-Invert coding for low power I/O", *IEEE Trans. on VLSI*, pp. 49-59, March 1995.

[13] M. R. Stan, W. P. Burleson, "Coding a Terminated Bus for Low-Power", Great Lakes Symp. on VLSI, pp. 70-73, Buffalo, NY, 1995.

[14] C.-L. Su, C.-Y. Tsui, A. M. Despain, "Saving Power in the Control Path of Embedded Processors", *IEEE Design and Test of Computers* , pp. 24-30, Winter 1994.

[15] T. Yamauchi, Y. Morooka, H. Ozaki, "A Low Power and High Speed Data Transfer Scheme with Asynchronous Compressed Pulse Width Modulation for AS-Memory", Symposium on VLSI Circuits, pp. 27-28, June 1995, Kyoto, Japan.

[16] S. Wuytack et al. "Global Communication and Memory Optimizing Transformations for Low Power Systems", International Workshop on Low Power Design, pp. 203-208, April 1994, Napa, CA.