Low-Power Encodings for Global Communication in CMOS VLSI

Mircea R. Stan, *Member, IEEE,* and Wayne P. Burleson, *Member, IEEE*

***Abstract*— Technology trends and especially portable applications are adding a third dimension (power) to the previously two-dimensional (speed, area) VLSI design space [30]. A large portion of power dissipation in high performance CMOS VLSI is due to the inherent difficulties in global communication at high rates and we propose several approaches to address the problem. These techniques can be generalized at different levels in the design process. Global communication typically involves driving large capacitive loads which inherently require significant power. However, by carefully choosing the data representation, or encoding, of these signals, the average and peak power dissipation can be minimized. Redundancy can be added in *space* (number of bus lines), *time* (number of cycles) and *voltage* (number of distinct amplitude levels). The proposed codes can be used on a class of terminated off-chip board-level buses with level signaling, or on tristate on-chip buses with level or transition signaling.**

***Index Terms*— Global communication in VLSI, low-power encoding, low-power I/O, space encoding, time encoding, two-dimensional (2-D) codes.**

I. Introduction

THE low-power community has been generally concerned with *average* power consumption and ways to minimize it. Average power is directly related to battery life in case of portable applications, as well as with costly package and heatsink requirements for high-end devices [24]. More recently the interest has also shifted to minimizing the *instantaneous*, or peak, power dissipation. There are cases when the instantaneous power can be much higher than the average power, and this leads to an undesired increase in simultaneous switching noise [14], metal electromigration problems [6], and local physical deformations due to nonuniform temperatures on the die.

This paper focuses on low-power techniques for global communication in CMOS VLSI using *data encoding* methods. Such encodings can decrease the power consumed for transmitting information over heavily loaded communication paths (buses) by reducing the switching activity without affecting the I/O information entropy [17]. Some of the techniques presented here are particularly effective in reducing the peak power consumption.

Global communication is typically achieved with *buses* [8], [7]. Such buses can be *on-chip* (between different functional blocks), *off-chip* (between different IC's on a PCB) or at the *system* level (as a back-plane bus connecting several boards together). The main motivation behind this work was the fact that buses have a relatively small number of electrical nodes (the number of bus lines), but they can still consume a disproportionately large amount of power because of the large capacitive loads.

1) Example: The dynamic power dissipation is proportional with the capacitive load [40]. A bus line is likely to have a load 2–3 orders of magnitude larger than an internal node [1], hence one transition on a bus line will dissipate as much as 100–1000 internal transitions.

A. Random Data Bus Model

Buses can behave differently depending on the type of *information* carried. The bus model considered here is a data bus on which the activity can be characterized as a random uniformly distributed sequence of values. Of course this is just an approximation, since in general the data is not random and the correlation can be exploited by lossless compression techniques [43]. Data compression can be an efficient method to decrease power dissipation and, by removing the extra redundancy, the data can be more accurately approximated as a random sequence.

1) Example: Fig. 1 shows the effects of compression and encoding on the number of transitions generated when transferring several typical Unix files over an 8-bit bus. The effect of low-power encoding alone (Bus-Invert is the encoding used here, see Section II-A) can be seen on the columns labeled *inv*, the effect of compression alone on the columns labeled *gz*, and the combined effect of compression and encoding on the columns labeled *gzinv*. Depending on the data type, compression can have a very high impact on switching activity required at the I/O. Of course, compression/decompression is desirable only for buses where the extra latency can be either tolerated or masked (e.g., main-memory or system buses) and if the overhead power is less than the savings on the bus which can be the case if all the operations can be done on-chip without external memory accesses.

The assumption of independent uniformly distributed inputs is also conveniently made by many statistical power estimation methods [28]. With this assumption, for any given cycle, the data on a K-bit wide bus can be any of 2^K possible values with equal probability. The average number of transitions per cycle will be $K/2$, hence there will be an average of 1/2 transitions per bus line.

Manuscript received September 8, 1996; revised May 15, 1997. This work was supported in part by a CISE-MIP 9703440 NSF CAREER Award.

M. R. Stan is with the Electrical Engineering Department, University of Virginia, Charlottesville, VA 22903 USA.

W. P. Burleson is with the Electrical Computer Engineering Department, University of Massachusetts, Amherst, MA 01003 USA.

Publisher Item Identifier S 1063-8210(97)09201-9.

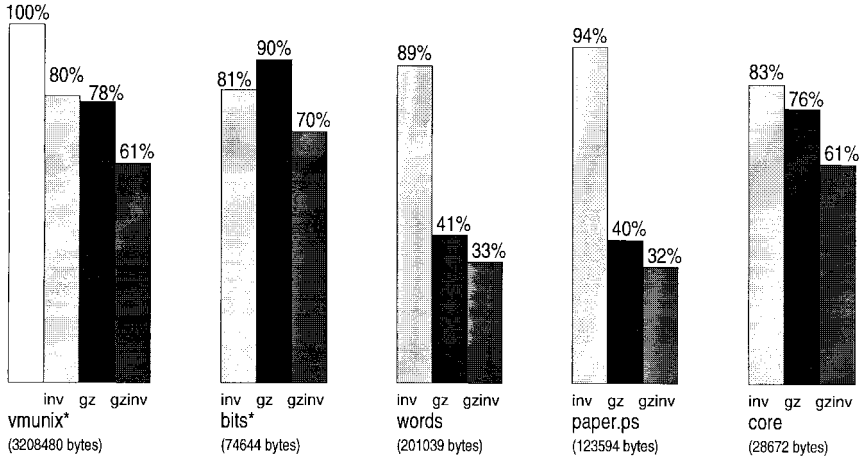

Fig. 1. Normalized number of transitions for typical Unix files and the effect of compression and encoding.

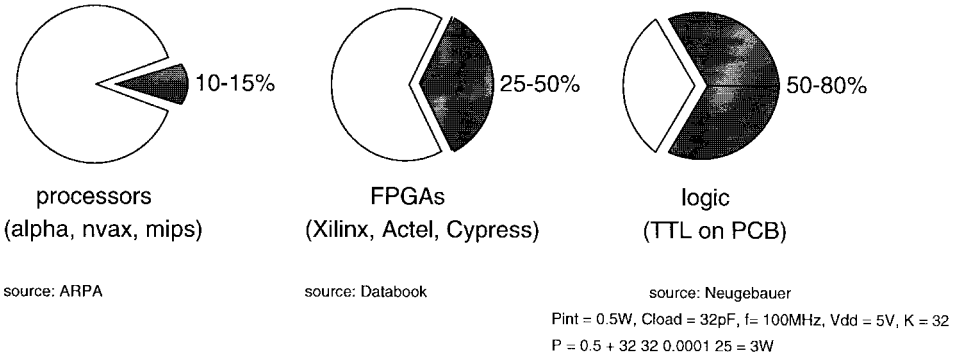

Fig. 2. Typical external I/O power for different applications (a) microprocessors, (b) FPGA's, and (c) TTL.

B. Level Signaling and Transition Signaling

Many system buses use active-low level signaling for the physical layer. For an active-low bus with level signaling, a logical "1" is represented by a LO voltage level and a logical "0" by a HI level, where the actual HI and LO voltage levels are determined by the technology used (e.g., TTL or ECL) and by whether the bus uses a full or reduced swing signaling. In the case of an active-high bus, a logical "1" is represented by a HI voltage and a logical "0" by a LO voltage. *Transitions* (from HI to LO or from LO to HI) determine the dynamic power consumption [4] on a tri-state bus with level signaling. Level signaling is applicable to both transaction oriented and packet oriented buses.

As we will see in Section II-A, low-power encodings with level signaling need a complex one-to-many *context-dependent* correspondence between codewords and information symbols. An alternative is to use transition signaling for packet-oriented buses. With transition signaling a logical 1 is represented by a *transition* (from HI to LO or LO to HI) while a 0 is represented by the lack of such a transition. Transition signaling will not reduce switching activity by itself but, as we show in Section II-B, in this case low-power encoding can be done in a simpler one-to-one *context-independent* manner. If we use transition signaling and at the same time we reduce the number of 1's in the codewords we can directly reduce the switching activity on the bus.

Transition signaling has a simple algorithmic description which corresponds to a straightforward implementation. Denoting by $v(t)$ the symbol to be transmitted, the modulated symbol with transition signaling, $b(t)$, is obtained by the simple expression ($\oplus$ is the bitwise XOR of the symbol bits)

$$b(t) = v(t) \oplus b(t-1).$$

Demodulation is similarly simple

$$v'(t) = b(t) \oplus b(t-1).$$

In a very different context, transition signaling was also found convenient by the asynchronous design community when they adopted Signal Transition Graphs (STG) over state diagrams for describing asynchronous behavior [13]. Transition signaling with capacitive coupling was also proposed for solving the "known-good die" problem for multi-chip modules (MCM) [27].

C. I/O Versus Internal Power

The amount of I/O and internal power depends on the application and type of data transmitted over the bus, as can be seen in Fig. 2. The power dissipated for the I/O can be as low as 10% [9] and as high as 80% [20] of the total power.

D. Previous Work on Low-Power Encodings

There are many sources of noise in a digital circuit, including the switching of large currents. Consequently, noise and dynamic power are both directly related to switching activity at the I/O, and the work by Park and Maeder [23] and Tabor [38] for minimizing transient noise due to I/O activity has many similarities with our work. Park and Maeder propose encoding tri-state I/O buses with transition signaling and limit the number of transitions in order to reduce switching noise. Tabor considers both tri-state and ECL terminated buses and proposes "starvation codes" and "ration codes" for minimizing switching transients. Even closer to the Bus-Invert method presented in [33] is the patent on a similar technique issued to Fletcher [10].

Even as some of the methods from [23], [38], [10] are related to our work, the published results in [32]–[36], [31], on which this paper is founded, represent the first time a comprehensive methodology for low-power I/O at several levels of abstraction is proposed, of a larger scale than any of the isolated results previously published.

Encoding address buses for low power was studied by Su *et al.* [37], Wuytack *et al.* [41] and Stan and Burleson [33]. Generally, an address bus tends to have a sequential behavior and a Gray code is optimal in such a case. With only one transition per cycle, or $1/K$ (0.125 for an 8-bit bus) transitions per bus line, it represents a big improvement (45% for an 8-bit bus) over the unencoded sequential case.

It turns out that for purely sequential behavior even the Gray code is too complex. This can be seen by looking at the information *entropy* of an address bus, which is zero for purely sequential accesses. In other words there is no need to actually send a new sequential address over the bus, since this address can be locally generated. This is the principle behind *burst* transfers on a synchronous bus, where only the first address is actually transferred, while the next addresses in the burst are locally generated.

Finite-state machines (FSM) have been the "workhorse" of the logic synthesis community for a long time. Mustang, Nova, and Diet are just a few CAD tools that attempt to optimize FSM's for area, performance and testability. It is natural to extend the research done for optimally encoding FSM's to yet another objective function: power. Hachtel *et al.* [12] have looked at reencoding a previously encoded FSM for low-power without changing the number of bits in the encoding, while Olson and Kang [22], and Tsui *et al.* [39] have looked at the more general problem of optimally encoding a state-machine for low power.

Due to the range and resolution of data represented as 2's complement numbers in the data-path of a DSP processor, the most significant bits on such a numerical bus are typically correlated, as opposed to the least significant bits which are essentially random, or uncorrelated. Mehra *et al.* [19], [18] have studied algorithmic and architectural level methodologies for modeling such correlated behavior at a high level and for accurately estimating power dissipation for DSP applications. Chandrakasan *et al.* [3] have proposed the use of sign-magnitude representations which can take advantage of the correlated behavior of the most-significant bits in order to reduce switching activity.

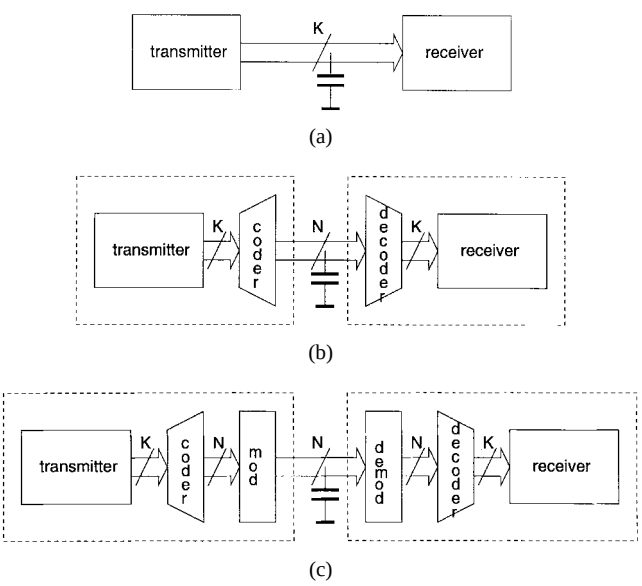

Fig. 3. Parallel bus: (a) K-bit wide unencoded, (b) K-bit to N-bit encoded, and (c) signaling extra step.

The one-hot residue logic design proposed by Chren [5] for arithmetic circuits also tries to minimize power by the way data is represented. Of great interest is the work on information theoretic measures for low power by Marculescu *et al.* [16] which relates power consumption to information theory concepts like spatial and temporal correlations and redundancy.

II. One-Dimensional Codes for Low-Power

In this section we consider encoding the data in *space* by adding redundancy in the form of extra bus lines. In order to be effective, the method requires that the power dissipated in the encoder and decoder be small.

1) Definition: A *code* is a mapping $C : U \rightarrow V$ where the elements of V are represented using N bits, and U has dimension 2^K, hence the elements of U are represented using K bits, with $N \geq K$.

The encoding process, as shown in Fig. 3, is done in *parallel* in the spirit of the codes for computer systems discussed by Fujiwara and Pradhan [11] and Rao and Fujiwara [26].

With low-power encodings it is possible to add redundancy in a *controlled* manner such that the correlation in time between successive data values reduces the switching activity.

In order to give an intuitive explanation for the effect of low-power encodings on switching activity, let us consider a 2-bit active-high bus with four possible symbols:

$$U = \{u0 = 00, u1 = 01, u2 = 11, u3 = 10\}.$$

The four codewords can be arranged as the vertices of a square-graph as in Fig. 4(a). A transmitted sequence over the bus will be equivalent to traversing a path in the square-graph and the number of transitions between two consecutive bus transfers will be—0 with probability 1/4, 1 with probability 1/2, and 2 with probability 1/4.

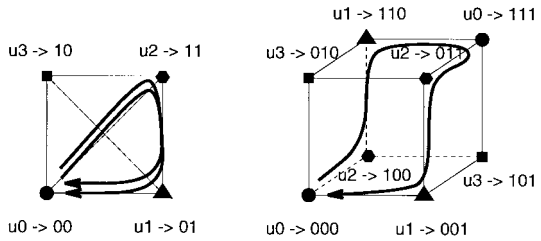

Fig. 4. Codes represented by graphs: (a) four symbols minimally encoded and (b) four symbols encoded with 3 bits.

TABLE I
1-TO-2 ENCODING FOR LOW ACTIVITY BUS

symbol	minimal encoding	low-power encoding
$u0$	00	000 or 111
$u1$	01	001 or 110
$u2$	11	011 or 100
$u3$	10	010 or 101

2) Example: The sequence: $u0$, $u2$, $u1$, $u0$, $u2$, $u1$, $u0$ (transmitted as 00, 11, 01, 00, 11, 01, 00) will generate eight transitions by traversing the diagonal twice [see Fig. 4(a)].

The number of transitions can be reduced with a one-to many (1-to-2) mapping ($N = 3$ code) as in Table I, by using a 3-bit wide bus. The eight codewords can now be arranged as the vertices of a cube-graph as in Fig. 4(b). By properly choosing at each step one of the two representations of a symbol, it is possible to traverse the new graph without using diagonals, thus generating at most one transition per cycle.

3) Example: The same sequence: $u0, u2, u1, u0, u2, u1, u0$, will generate only six transitions by using the following encoding: 000, 100, 110, 111, 011, 001, 000.

A. Bus-Invert Encoding

A coding scheme for low-power I/O that extends the previous intuitive example to lengths $K > 2$ is the Bus-Invert code [10], [33]. The idea is to use one extra bus line, called *invert*. The method computes the *Hamming distance* [2] between the present *encoded* bus value and the next data value. If the distance is $\frac{\geq K}{2}$ then the transmitter *inverts* the next value and signals the inversion with *invert* $= 1$. If the distance is $\leq \frac{K}{2}$ the next value is left un-inverted and *invert* $= 0$. The value of *invert* must be transmitted over the bus in order to recover the correct information at the receiver, hence the method increases the number of bus lines from K to $N = K + 1$.

The Bus-Invert method generates a code that has the property that the *maximum* number of transitions per time slot is reduced from K to $\frac{K}{2}$ and thus the peak power dissipation for the I/O is reduced by half. From the coding theory point of view, the Bus-Invert code is a time-dependent Markovian code. Since the encoding process has *memory*, the Bus-Invert method can also be considered a $(K, K+1, 1)$ convolutional code [15]. A Markov chain of the Bus-Invert encoding process is shown in Fig. 5.

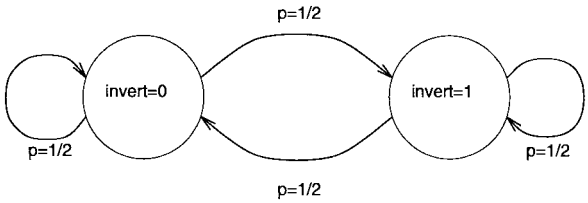

Fig. 5. Markov chain for the Bus Invert method.

Bus-Invert is a context dependent one-to-many encoding which uses level signaling. Two codewords exist for each information symbol (1 bit of redundancy), from which the codeword leading to the lowest activity factor is selected during encoding. This idea could in principle be extended to more than 1 bit of redundancy (and a broader context, i.e., more than two codewords for each information symbol) but the coding complexity would quickly become unmanageable. A solution is to use limited-weight codes (LWC) [32] which are one-to-one (context-independent) encodings, hence are simpler to implement.

B. Limited-Weight Codes

Limited-weight codes *require* transition signaling in order to reduce the switching activity since with transition signaling only 1's generate transitions. Transition signaling is convenient for low-power as it offers a direct way of controlling the bus activity factor simply by reducing the number of logical 1's transmitted over the bus [32]. This generally works only with packet-oriented buses, but it should be noted that there is a clear trend for modern buses to be packet-oriented [25].

Limited-weight codes can be defined as having codewords with weight$[v(t)] \leq M$ (weight$[v(t)]$ denotes the Hamming weight, or total number of 1's in $v(t)$). For an M-limited-weight code $1 \leq M \leq K$, the smaller M is, the lower the resulting bus switching activity will be (the worst-case number of transitions per cycle is M). The two extremes for the value of M are as follows:

- $M = K$ and the encoding is actually nonredundant with $p(0) = p(1) = 1/2$ and
- $M = 1$ (1-limited weight code) and $p(1) \approx 1/N$ and $p(0) \approx (N-1)/N$.

Because the source entropy [2] must remain unchanged (we want to transmit the same amount of information) it follows that in order to have a sufficient number of codewords, the following inequality must be satisfied [32]:

$$\binom{N}{0} + \binom{N}{1} + \binom{N}{2} + \cdots + \binom{N}{M} \geq 2^K. \qquad (1)$$

Here, the left-hand side represents the total number of possible codewords with weight $\leq M$, and the right-hand side represents the number of words for the unencoded source.

1) Definitions:

- A *perfect* M-LWC satisfies (1) with equality (uses *all* the patterns of length N with weight $\leq M$ as codewords);
- a *semiperfect* M-LWC consists of all possible codewords with weight $\leq M-1$ and only some with weight $= M$, and satisfies (1) as a strict inequality.

TABLE II
M VERSUS N FOR GIVEN K

information bits (K)	2	4	4	8	8	8	8	16
maximum nr. of $1's$ (M)	1	1	2	1	2	3	4	8
codeword length (N)	3	15	5	255	23	11	9	17

TABLE III
2-LWC AND 1-LWC FOR A 16-SYMBOL SOURCE

symbol	minimally encoded	2-LWC	1-LWC
0	0000	00000	000000000000000
1	0001	00001	000000000000001
2	0010	00010	000000000000010
3	0011	00011	000000000000100
4	0100	00100	000000000001000
5	0101	00101	000000000010000
6	0110	00110	000000000100000
7	0111	11000	000000001000000
8	1000	01000	000000010000000
9	1001	01001	000000100000000
10	1010	01010	000001000000000
11	1011	10100	000010000000000
12	1100	01100	000100000000000
13	1101	10010	001000000000000
14	1110	10001	010000000000000
15	1111	10000	100000000000000

Perfect and semiperfect limited weight codes are optimal in the sense that any other code with the same length N cannot have better statistical properties for low power.

The inequality (1) shows that there is a clear tradeoff between the limiting weight M and the length of the code N: the smaller M gets (and thus power dissipation is decreased), the larger N (and thus the extra required redundancy) must be. In general the number of necessary extra code bits grows exponentially and the method becomes impractical when very small switching activities are needed, as can be seen in Table II.

Two different points of view are possible in order to design good LW codes for a given K as follows:

- with a given level of redundancy $N - K$, build a code that has a minimum M and
- with a desired M, build a code with the smallest extra redundancy.

2) Example: A 2-LWC and a 1-LWC with $K = 4$ are shown in Table III.

C. Encoding Terminated Buses for Low Power

Until now we have only considered tri-state buses on which the power dissipated is mainly dynamic (charging and discharging of bus line capacitances) and the codes must minimize the switching activity (number of transitions).

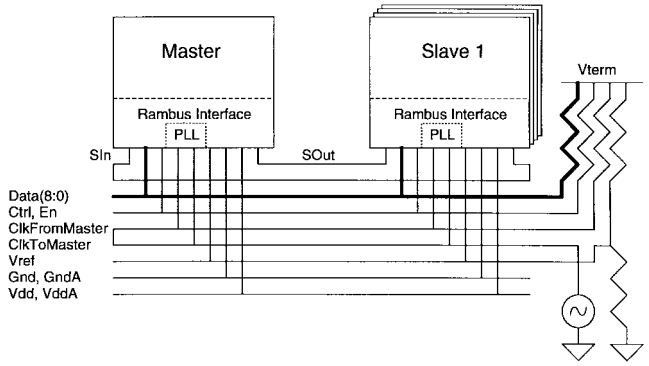

Fig. 6. Typical modern terminated bus (Rambus).

A type of modern parallel terminated buses with only pull-up resistors dissipates power only when transmitting logical 1's. In such a case limited-weight codes can be directly used for reducing the number of 1's in order to reduce power, without the need for transition signaling [34].

1) Example: For a standard bus like the Rambus (see Fig. 6) the main constraint is to code the data at the *logical* level without affecting the *physical* specification. The Rambus has $8 + 1$ data bits, the ninth bit's use being left to the system designer. We can take advantage of this ninth bit and use it as the *invert* line. The derived code will have codewords of length 9. With 9 bits, there are $2^9 = 512$ possible patterns out of which only $2^8 = 256$ are needed. It can be observed that

$$\binom{9}{0} + \binom{9}{1} + \binom{9}{2} + \binom{9}{3} + \binom{9}{4} = 2^8 = 256.$$

It follows that a perfect 4-limited weight code that uses all 9-bit patterns with at most four 1's is optimal. The data can be either decoded at the receiver and stored unencoded as 8-bit values, or can be stored in encoded form in a 9-bit wide Rambus DRAM (RDRAM). Storing the encoded data has the advantage of using only off-the-shelf RDRAM's, with modifications needed only on the Rambus ASIC (RASIC) side. Because the resulting codewords have at most four 1's, the worst case power dissipation on the data lines is decreased by 50% (from 168 to 84 mW). The decrease in average power dissipation depends on the statistics of the bus transfers and is generally smaller. If the data is random uniformly distributed the average I/O power dissipation is reduced by approximately 18% (from 84 to 68 mW).

D. The LWC–ECC Duality

There is an unexpected formal relationship between Limited-Weight codes and general error-correcting codes. It is unexpected because LWC's do not exhibit any of the nice algebraic properties of ECC's and because they were formulated in a completely different setting. The fundamental observation that links LWC's to ECC's is the following: the *codewords* of a Limited Weight code when viewed as patterns of 1's and 0's are the same as the *error patterns* corrected by a standard block ECC. Generally, an ECC is *designed* to correct up to m errors, which means that it will correct all

TABLE IV
CORRESPONDENCE BETWEEN ECC'S AND LWC'S

ECC	LWC
syndrome	unencoded symbol
coset leader	LW codeword
syndrome length $n-k$	unencoded length K
codeword length n	codeword length N
information bits k	extra redundancy $N-K$
perfect ECC	perfect LWC
semi-perfect ECC	semi-perfect LWC
repetition code	$\frac{m}{2}$-LWC
Hamming ECC	1-LWC

patterns with at most m 1's. This is "dual" to the definition of a M-LWC which *contains* codewords with at most M 1's.

1) Example: A (15, 11) Hamming ECC has 11 *information* bits and four *parity* bits [2]. Any two of the 2^{11} codewords are at a Hamming distance ≥ 3 and thus the (15, 11) Hamming ECC can correct any single bit in error. An error pattern is represented by a 15-bit word which has a "1" in the position of the erroneous bit. If $v(t)$ is the transmitted codeword, $v'(t)$ the received codeword and $e(t)$ the error pattern, the effect of the error on the codeword can be written as:

$$v'(t) = v(t) \oplus e(t).$$

It is clear that there are exactly 15 1-bit error patterns and together with the all-zero word they are the same patterns as the words of a 1-limited weight code (1-hot encoding with "no-hot" encoding for "0").

Decoding an ECC requires finding the error pattern and, since the error pattern is similar to a LW codeword, this is the same as *encoding* a LWC. It must be noted though that the efficient decoding of EC codes is a hard problem except for particular cases, and this duality between ECC decoding and LWC encoding shows that LWC generation is also intrinsically hard in general.

The Hamming bound [2] for an ECC gives a relation between the length n, number of check bits k and the number of correctable bits m

$$\binom{n}{0} + \binom{n}{1} + \binom{n}{2} + \cdots + \binom{n}{m} \leq 2^{n-k}. \qquad (2)$$

The bound simply expresses the fact that the total number of syndromes (2^{n-k}) must be greater than the number of correctable error patterns because the syndromes must be unique for all errors (coset leaders).

The inequality (1) for LWC's is very similar to (2), with one difference: the sign of the innequality. For a LWC and the dual ECC, the (1) inequality has one more term than (2), which is true when both inequalities are *strict*. In the case of perfect codes, when (1) and (2) are both equations, the number of terms is the same.

The various analogies between LW and EC codes are summarized in Table IV. Further work is needed in order to use this theoretical result for the practical generation of novel LW codes.

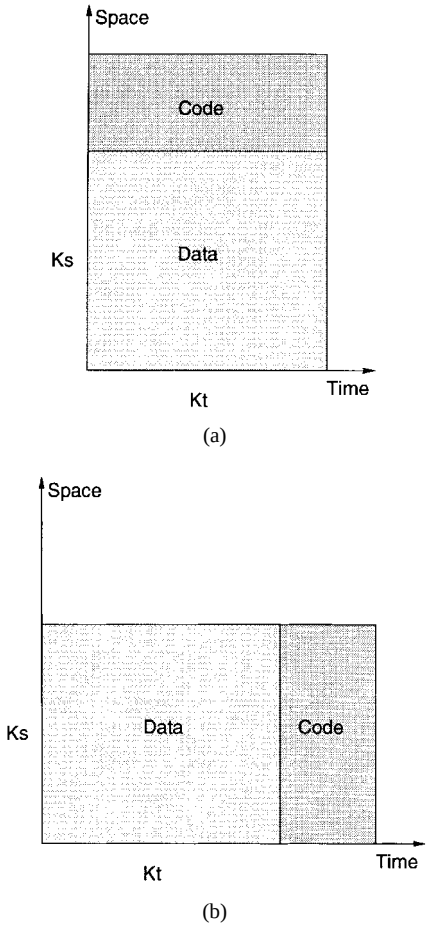

Fig. 7. A packet of data: (a) with redundancy in space and (b) with redundancy in time.

III. TWO-DIMENSIONAL ENCODINGS FOR LOW POWER

The codes proposed in Section II use redundancy in *space* (number of bus lines) for reducing the bus transition activity. 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 techniques impractical. An alternative is to keep the number of bus lines constant and inject redundancy in *time* by using extra transfer cycles. The same assumptions as in Section II are made here about the randomness of the data bus and the use of transition signaling.

Time encoding requires that the data be transmitted in packets, which is typical for global buses where there is an advantage in transmitting bursts of data for improved throughput [25]. Fig. 7(a) shows a data-packet with redundancy in space, while 7(b) 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-bits wide, the same coding techniques that in Section II used redundancy in space can now be applied in time. Limited-weight codes with redundancy in time and transition signaling can use extra transfer cycles for encoding the K_t bits that are successively transmitted over each bus line in order to minimize the number of 1's and hence the number of transitions.

1) Example: For $K_t = 4$ a low-power *time* encoding will first count the number of 1's that are to be transmitted over

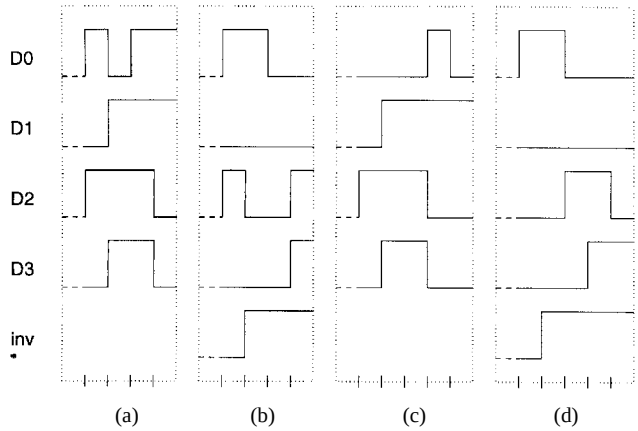

Fig. 8. Packet of four 4-bit words: (a) with transition signaling, (b) encoded in space, (c) encoded in time, and (d) encoded in both space and time.

TABLE V
FOUR-WORD PACKET OVER A 4-BIT BUS

	T0	T1	T2	T3
D0	1	1	1	0
D1	0	1	0	0
D2	1	0	0	1
D3	0	1	0	1

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 signaled by a "1" in an extra fifth transfer cycle. Otherwise, the $K_t = 4$ bits will be transmitted as they are and the extra fifth 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.

2) Example: Transmitting the four-word packet in Table V takes four cycles and generates eight transitions over a four-line bus with transition signaling. Fig. 8(a) shows the corresponding waveforms, where it is assumed that the 4-bit words are arranged in columns and are transferred from left to right. With redundancy in space or time (Bus-Invert with transition signaling) the number of 1's (hence transitions) is reduced to seven at the expense of an extra bus line or an extra transfer cycle, as shown in Table VI. The waveforms for the encoding in space are shown in Fig. 8(b), and for encoding in time in Fig. 8(c). Transition signaling is used in both cases.

Unfortunately, when encoding in time, it is still possible to have all bus lines switching simultaneously, hence such encodings do not improve peak power and simultaneous switching noise and this is a great disadvantage of time encodings compared to space encodings (for which even the simplest encoding, Bus-Invert, reduces the peak power by 50%). 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.

TABLE VI
FOUR-WORD PACKET WITH ENCODING IN SPACE (LEFT) OR IN TIME (RIGHT)

	T0	T1	T2	T3
D0	1	0	1	0
D1	0	0	0	0
D2	1	1	0	1
D3	0	0	0	1
inv	0	1	0	0

	T0	T1	T2	T3	T4
D0	0	0	0	1	1
D1	0	1	0	0	0
D2	1	0	0	1	0
D3	0	1	0	1	0

TABLE VII
FOUR-WORD PACKET WITH ENCODING IN BOTH SPACE AND TIME

	T0	T1	T2	T3	T4
D0	1	0	1	0	0
D1	0	0	0	0	0
D2	0	0	1	0	1
D3	0	0	0	1	0
inv	0	1	0	0	0

	T0	T1	T2	T3	T4
D0	0	0	0	0	1
D1	0	1	0	1	0
D2	1	0	0	0	0
D3	0	1	0	0	0
inv	0	0	0	1	0

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.

A. Coding in Both Space and Time

If power needs to be further reduced, redundancy in both space and time can be used. Encoding in two-dimensions 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, with the same average power reduction being obtained in both cases.

1) Example: For the previous example the number of 1's can be reduced to six with two-dimensional coding as can be seen in Table VII. Column-wise encoding (in space) is done first in the table on the left, row-wise encoding (in time) is done first in the table on the right. The waveforms corresponding to the case where encoding is done column-wise first can be seen in Fig. 8(d). Transition signaling is also used in this case.

Although the average power reduction is the same whether the two-dimensional encoding is done first in space or in time, the same is not true about the peak power and simultaneous switching noise. In the previous section we saw that encoding in time (as opposed to encoding in space) can still lead to transitions on all bus lines, hence, in order to reduce the peak power and simultaneous switching noise, the encoding should be done first in time and then in space (in this way we can make sure that the maximum number of transitions will be determined by the encoding in space).

Table VIII shows all the codewords of the smallest possible two-dimensional low-power code, with column-wise encoding followed by row-wise encoding. A code with the same parameters, but with row-wise encoding followed by column-wise encoding, is shown in Fig. IX. There are 16 such codewords, one for each of the 2×2 possible patterns of 1's and 0's.

TABLE VIII
TWO-DIMENSIONAL CODE WITH FOUR INFORMATION BITS AND FOUR CODE BITS

		0 0				0 0				0 0				0 0	
0 0	⇒	0 0 0		1 0	⇒	1 0 0		0 1	⇒	0 1 0		1 1	⇒	0 0 1	
0 0		0 0 0		0 0		0 0 0		0 0		0 0 0		0 0		0 0 0	
		0 0				1 0				0 0				1 0	
0 0	⇒	0 0 0		1 0	⇒	0 0 0		0 1	⇒	0 1 0		1 1	⇒	0 1 0	
1 0		1 0 0		1 0		0 0 0		1 0		1 0 0		1 0		0 0 0	
		0 0				0 0				0 1				0 1	
0 0	⇒	0 0 0		1 0	⇒	1 0 0		0 1	⇒	0 0 0		1 1	⇒	1 0 0	
0 1		0 1 0		0 1		0 1 0		0 1		0 0 0		0 1		0 0 0	
		0 0				1 0				0 1				1 1	
0 0	⇒	0 0 0		1 0	⇒	0 0 0		0 1	⇒	0 0 0		1 1	⇒	0 0 0	
1 1		0 0 1		1 1		0 1 0		1 1		1 0 0		1 1		0 0 0	

TABLE IX
ANOTHER TWO-DIMENSIONAL CODE WITH FOUR INFORMATION BITS AND FOUR CODE BITS

		0 0				0 0				0 0				0 0	
0 0	⇒	0 0 0		1 0	⇒	1 0 0		0 1	⇒	0 1 0		1 1	⇒	0 0 1	
0 0		0 0 0		0 0		0 0 0		0 0		0 0 0		0 0		0 0 0	
		0 0				1 0				0 0				0 0	
0 0	⇒	0 0 0		1 0	⇒	0 0 0		0 1	⇒	0 1 0		1 1	⇒	0 0 1	
1 0		1 0 0		1 0		0 0 0		1 0		1 0 0		1 0		1 0 0	
		0 0				0 0				0 1				0 0	
0 0	⇒	0 0 0		1 0	⇒	1 0 0		0 1	⇒	0 0 0		1 1	⇒	0 0 1	
0 1		0 1 0		0 1		0 1 0		0 1		0 0 0		0 1		0 1 0	
		0 0				0 0				0 0				0 0	
0 0	⇒	0 0 0		1 0	⇒	1 0 0		0 1	⇒	0 1 0		1 1	⇒	0 0 1	
1 1		0 0 1		1 1		0 0 1		1 1		0 0 1		1 1		0 0 1	

TABLE X
TWO-DIMENSIONAL CODE WITH FOUR INFORMATION BITS AND FIVE CODE BITS

		0 0 0				0 0 0				0 0 0				0 0 0	
0 0	⇒	0 0 0		1 0	⇒	1 0 0		0 1	⇒	0 1 0		1 1	⇒	0 0 1	
0 0		0 0 0		0 0		0 0 0		0 0		0 0 0		0 0		0 0 0	
		0 0 0				1 0 0				0 0 0				1 0 0	
0 0	⇒	0 0 0		1 0	⇒	0 0 0		0 1	⇒	0 1 0		1 1	⇒	0 1 0	
1 0		1 0 0		1 0		0 0 0		1 0		1 0 0		1 0		0 0 0	
		0 0 0				0 0 0				0 1 0				0 1 0	
0 0	⇒	0 0 0		1 0	⇒	1 0 0		0 1	⇒	0 0 0		1 1	⇒	1 0 0	
0 1		0 1 0		0 1		0 1 0		0 1		0 0 0		0 1		0 0 0	
		0 0 0				1 0 0				0 1 0				0 0 1	
0 0	⇒	0 0 0		1 0	⇒	0 0 0		0 1	⇒	0 0 0		1 1	⇒	0 0 0	
1 1		0 0 1		1 1		0 1 0		1 1		1 0 0		1 1		0 0 0	

TABLE XI
ANOTHER TWO-DIMENSIONAL CODE WITH FOUR INFORMATION BITS AND FIVE CODE BITS

		0 0 0				0 0 0				0 0 0				0 0 0	
0 0	⇒	0 0 0		1 0	⇒	1 0 0		0 1	⇒	0 1 0		1 1	⇒	0 0 1	
0 0		0 0 0		0 0		0 0 0		0 0		0 0 0		0 0		0 0 0	
		0 0 0				1 0 0				0 0 0				0 0 0	
0 0	⇒	0 0 0		1 0	⇒	0 0 0		0 1	⇒	0 1 0		1 1	⇒	0 0 1	
1 0		1 0 0		1 0		0 0 0		1 0		1 0 0		1 0		1 0 0	
		0 0 0				0 0 0				0 1 0				0 0 0	
0 0	⇒	0 0 0		1 0	⇒	1 0 0		0 1	⇒	0 0 0		1 1	⇒	0 0 1	
0 1		0 1 0		0 1		0 1 0		0 1		0 0 0		0 1		0 1 0	
		0 0 0				0 0 0				0 0 0				0 0 1	
0 0	⇒	0 0 0		1 0	⇒	1 0 0		0 1	⇒	0 1 0		1 1	⇒	0 0 0	
1 1		0 0 1		1 1		0 0 1		1 1		0 0 1		1 1		0 0 0	

Two bits of redundancy are used in space and two in time. The average switching activity is reduced by 31%, which is better than the $\leq$25% for one-dimensional (1-D) Bus-Invert.

Table X shows another 2-D code which is an extension of the code in Table VIII. There is an extra ninth bit which encodes in time the space codebits. A similar extension of the code in Table IX is shown in Table XI, this time the extra ninth bit encodes in space the time codebits. The average power dissipation for these 9-bit 2-D codes is reduced by 34%, slightly better compared to the smallest two-dimensional codes.

A useful application of such two-dimensional encodings is the generation of new *one-dimensional* codes by projecting (unrolling) the 2-D code in 1-D (projecting a 2-D array $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ horizontally will lead to the one-dimensional array $[a\ b\ c\ d]$). For example, by unrolling the 2-D codes in Tables VIII or IX, we can obtain 1-D *semiperfect* 2-limited-weight codes of length 8. Similarly, by unrolling the codes in Tables X or XI, the codes obtained are semiperfect 2-LW codes of length 9. This is an important result for several reasons:

- the algorithmic generation of codes for low-power is intrinsically hard in the general case and
- such unrolled 2-D codes provide a compromise between the two extremes of Bus-Invert (minimum redundancy, one extra line) and one-hot encoding (minimum transition activity, one transition per cycle), and offer another practical design alternative.

B. 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 [33]. 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.

1) Example: The block diagram of an encoder for the two-dimensional code in Table XI (time followed by space encoding) is shown in Fig. 9. Since there are only two information bits the majority voter in this case is only an AND gate. For this example it was chosen to transmit the redundant bit before the information bits. Encoding in time (row-wise) is followed by encoding in space (column-wise) and then by transition signaling. Shift registers with parallel D and T inputs are used for time encoding and T registers are also used for transition signaling. There will be at most two transitions for each four bits of information (nine codebits transmitted). It can be seen that, although conceptually similar, time encoding

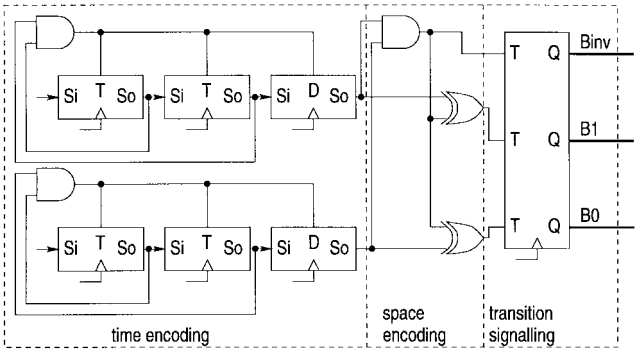

Fig. 9. Two-dimensional encoding in both space and time.

is more expensive in this case than space encoding because it needs to access the entire data packet at once.

C. Modulation in Time

Until now, when we addressed redundancy in time we implicitly assumed that the time domain has exactly the same "integer" restrictions as the space domain but this need 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.

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 by intersymbol interference. Another bound is given by the minimum spacing ΔT with which the exact position in time of a transition can be determined. This resolution depends on the amount of noise and jitter on the bus and is very much implementation dependent. Until now it was implicitly assumed that the two bounds are the same and equal to the "bus cycle", but in most cases the resolution ΔT can be made much smaller than the minimum pulse width $T_{\min}$ (see Fig. 10). This means that we can use the *position* in time of a transition for encoding several bits per transition [21]. By considering a "virtual" cycle equal to ΔT and $T_{\min}$ as a multiple of ΔT, then in terms of transition signaling the lower bounds translate into the necessity of having a certain number of 0's between any two consecutive 1's (the 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 for improving the code efficiency when $\Delta T < T_{\min}$. A RLL(d,k) code has at least d and at most k 0's between any two 1's [29].

1) Example: For $T_{\min} = 3 \cdot \Delta T$ the very popular variable-length RLL(2,7) code, given in Table XII, can be used. 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, or action, is also reduced by 50%).

More impressive results in low power can be obtained by observing that for a TTL type interface the typical values are $T_{\min} \approx 5 - 10$ ns and $\Delta T \approx 0.3 - 1.0$ ns [21]. This theoretically enables the use of RLL codes with large values for d and k (e.g., $d = 16$) which would generate much smaller number of transitions than the RLL(2,7), unfortunately implementing such codes becomes very complex.

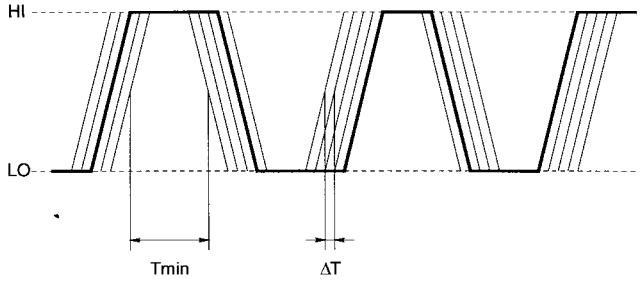

Fig. 10. Phase modulation on a bus line.

TABLE XII
RLL(2,7) CODE

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

Phase modulation [21], as shown in Fig. 10, is a straightforward nonoptimal RLL code, with $d = (T_{\min}/\Delta T) - 1$ and $k = d + 2 \cdot (p - 1)$, which encodes several bits of data in the position of a transition. Although inefficient from the information theory point of view, phase modulation is relatively easy to implement for large values of d and k, hence it can be very useful as a low-power encoding. In such a scheme one cycle equals $(d+p)\Delta T$ and there is exactly one transition in the p part of the cycle such that the minimum pulse width is $T_{\min}$. Since each transition can have one of p different positions we can transmit $\log_2 p$ bits per transition, and, if p is large, there is a potential for important power reductions.

From the low-power coding point of view (see Section II-B), phase modulation can be viewed as one-hot encoding in time, with transition signaling, with the constraint on $T_{\min}$ which requires d extra 0's in-between any two one-hot codewords.

2) Example: As shown in Fig. 10, if $T_{\min} = 10 \cdot \Delta T$ there will be a string of $d = 9$ extra 0's between each pair of codewords. In this example each transition can have any one of five possible positions, hence $\log_2 5$ bits can be transmitted per transition.

For every $\log_2 p$ transmitted bits we have to transmit $(d+p)$ ΔT periods, hence the phase modulation *code rate* [29] will be

$$\text{rate} = \frac{\log_2 p}{d+p}. \tag{3}$$

Generally $T_{\min}$ and ΔT are given, 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 (3)

$$p_{\text{opt}} \cdot (\ln 2 \log_2 p_{\text{opt}} - 1) = d. \tag{4}$$

TABLE XIII
SAVINGS IN TRANSITION ACTIVITY FOR PHASE MODULATION AT $p_{\rm opt}$ FOR OPTIMAL CODE RATE

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
rate	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
sav.	0%	0%	14%	23%	23%	29%	29%	33%	33%	37%	37%	40%	40%	40%	42%	42%

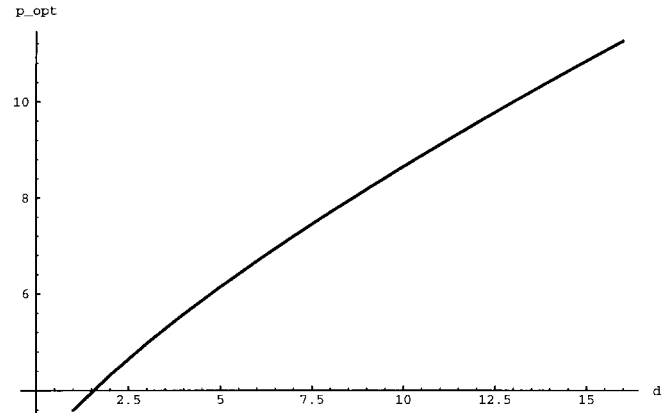

Fig. 11. Growth of $p_{\rm opt}$ with d for phase modulation.

TABLE XIV
SAVINGS IN TRANSITION ACTIVITY FOR PHASE MODULATION AT $p_{\rm opt}$ AS d INCREASES

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

Table XIII shows the values of $p_{\rm opt}$ (rounded to nearest integer) for different values of d, as well as the rate (number of bits per transition) and the average savings in the number of I/O transitions. Fig. 11 shows the growth of $p_{\rm opt}$ with d. Although very large values of d are not practical anyhow, it is interesting to note that the growth of $p_{\rm opt}$ with d is less than linear, hence the power savings are diminishing as d increases to large numbers, which can be also seen in Table XIV.

Extra power savings can be obtained by realizing that for low power we may use a somewhat larger value than $p_{\rm opt}$. In (4), $p_{\rm opt}$ was computed for optimal data rate (or minimum transfer time for a given packet) but for low power we are more interested in minimizing the number of transitions than in optimizing the transfer rate. Another argument is given by Fig. 12 which shows the code rate as a function of d and p. As can be seen the code rate has a very shallow peak at $p_{\rm opt}$, hence we can safely choose a larger value for p without much penalty in code rate.

We can quantify this choice by using as a measure of low-power efficiency the *energy-delay product* of the number of transitions in a packet $\sim 1/\log_2 p$, and the transfer time for the packet $\sim 1/\text{rate}$. The optimal $p_{\rm low}$ for low power will minimize [compare to (3)]

$$\text{energy} \cdot \text{delay} \sim \frac{d+p}{(\log_2 p)^2}. \tag{5}$$

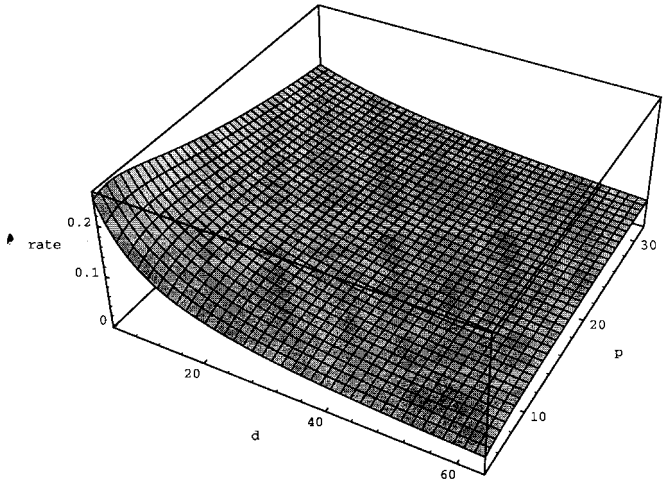

Fig. 12. Variation of transmission rate with d and p for phase modulation.

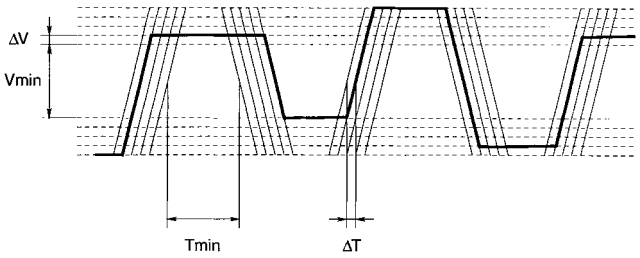

Fig. 13. Modulation in both voltage and time.

In this way the optimal $p_{\rm low}$ values are larger and there can be better savings in transition activity (up to 58% savings at $d = 16$ for $p_{\rm low} = 26$) as can be seen in Table XV. The equation which determines $p_{\rm low}$ for minimum energy-delay is

$$p_{\rm low} \cdot (\ln 2 \log_2 p_{\rm low} - 2) = 2 \cdot d. \tag{6}$$

The tradeoff is that the implementation of real circuits for large values of p will be more difficult.

D. Modulation in Both Amplitude and Time

As in the case of *space* and *time*, we can also define 2-D codes that use modulation in *amplitude* and *time*. By modulating the signal amplitude with $2 \cdot p_{\rm volt}$ levels distanced ΔV apart, in each cycle we can transmit $\log_2 p_{\rm volt}$ bits in addition to the $\log_2 p_{\rm time}$ bits transmitted in time with each transition.

1) Example: Fig. 13 shows such a two-dimensional encoding with $p_{\rm time} = 5$ and $p_{\rm volt} = 5$ that can transmit 5 bits in amplitude and 5 bits in time for a total of 10 bits per transition and savings of 80% in switching activity.

TABLE XV
SAVINGS IN TRANSITION ACTIVITY AT p_{low} FOR OPTIMAL ENERGY-DELAY PRODUCT

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
rate	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
sav.	37%	42%	44%	46%	49%	50%	51%	52%	53%	54%	54%	55%	56%	56%	57%	58%

There are many possible implementations for a phase modulation scheme and the one in [21] 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 further improve coding efficiency and reduce implementation complexity have been also proposed [42]. We believe that phase modulation is practical, improves coding efficiency and is suitable for low-power applications.

Unfortunately, amplitude modulation does not seem to be a power efficient option because very low-power analog-to-digital (A/D) and digital-to-analog (D/A) converters are not feasible yet, hence 2-D codes in amplitude and time will probably remain just a theoretical possibility for the near future.

IV. CONCLUSION

In this paper, we have attempted to present a unified framework for low-power communication on global buses. Such buses can be at the module, subsystem, die, package, or multichip module (MCM) level, the only common characteristic being that the bus capacitances are much larger than the capacitances of the internal circuit.

Low-power techniques at the architectural level reduce the I/O transition activity by *coding* the data on the bus. Besides being applicable in practice, the topics presented here shed light on a number of interesting theoretical issues:

- for low-power operation redundancy should be generally avoided by compressing the data, but *controlled* redundancy through low-power encodings can improve the signal temporal correlation for reduced transition activity;
- time redundancy can be as effective as redundancy in space for reducing the average switching activity;
- low-power operation can be achieved by reducing the activity on highly-loaded nodes, even if both the *total* number of transitions and the *total* capacitance are increased;
- phase modulation can be explained in terms of low-power encodings;
- Limited-weight codes are duals of error-correcting codes with Hamming distance in the same sense as quantization is the dual of coding with Euclidean distance.

There are several directions of research that can extend the results of this paper:

- finding new practical limited-weight codes based on the ECC duality,
- extending low-power coding methods to other bus models (e.g., with cross-coupled capacitances),
- extending low-power coding methods to different data models (e.g., Gaussian distribution),
- implementing low-power codes in real applications, and
- including low-power coding methods in the specifications of standard buses.

The most important aspect in the future will be to apply the low-power methods described here to more practical circuits, hopefully with the cooperation of interested chip designers and manufacturers.

ACKNOWLEDGMENT

The authors would like to thank Profs. I. Koren, P. Menon, and R. Moll from the University of Massachusetts, Amherst, for useful feedback on this work. They also thank the anonymous reviewers for their comments which improved the final version of this paper.

REFERENCES

[1] H. Bakoglu, *Circuits, Interconnections, and Packaging for VLSI*. Menlo Park, CA: 1990.
[2] R. E. Blahut, *Theory and Practice of Error Control Codes*. Reading, MA: Addison Wesley, 1983.
[3] A. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low-power techniques for portable real-time DSP application," in *VLSI Design*, India, 1992.
[4] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low-power CMOS digital design," *Proc. IEEE*, vol. 83, pp. 498–523, Apr. 1995.
[5] W. A. Chren, "Low delay-power product CMOS design using one-hot residue coding," in *Proc. Int. Symp. Low Power Design*, Dana Point, CA, Apr. 1995, pp. 145–150.
[6] A. Dasgupta and R. Karri, "Electromigration reliability enhancement via bus activity distribution," in *Proc. Design Automat. Conf.*, Las Vegas, NV, June 1996, pp. 353–356.
[7] D. del Corso, H. Kirrmann, and J. D. Nicoud, *Microcomputer Buses and Links*. New York: Academic, 1986.
[8] J. di Giacomo, *Digital Bus Handbook*. New York: McGraw-Hill, 1990.
[9] D. Dobberpuhl *et al.*, "A 200 mhz, 64 b, dual issue CMOS microprocessor," in *Proc. Int. Solid-State Circuits Conf.*, 1992, pp. 106–107.
[10] R. J. Fletcher, "Integrated circuit having outputs configured for reduced state changes," U.S. Patent no. 4 667 337, May 1987.
[11] E. Fujiwara and D. K. Pradhan, "Error-control coding in computers," *Comput.*, pp. 63–72, July 1990.
[12] G. D. Hachtel, M. Hermida, A. Pardo, M. Poncino, and F. Somenzi, "Re-encoding sequential circuits to reduce power dissipation," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 1994, pp. 70–73.
[13] S. Hauck, "Asynchronous design methodologies: An overview," in *Proc. IEEE*, Jan. 1995, pp. 69–92.
[14] S. Kodical, "Simultaneous switching noise," in *IDT High-Speed CMOS Logic Design Guide*, Santa Clara, CA, pp. 41–47, 1993.
[15] S. Lin, J. Daniel, and J. Costello, *Error Control Coding: Fundamentals and Applications*, Prentice-Hall, Englewood Cliffs, NJ, 1983.
[16] D. Marculescu, R. Marculescu, and M. Pedram, "Information theoretic measures for energy consumption at register transfer level," in *Proc. Int. Symp. Low Power Design*, Dana Point, CA, Apr. 1995, pp. 81–86.
[17] R. Marculescu, D. Marculescu, and M. Pedram, "Efficient power estimation for highly correlated input streams," in *Design Automat. Conf.*, June 1995, pp. 628–634.
[18] R. Mehra, D. B. Lidsky, A. Abnous, P. E. Landman, and J. M. Rabaey, "Algorithm and architectural level methodologies for low power," in

Low Power Design Methodologies, J. M. Rabaey and M. Pedram, Eds. Boston, MA: Kluwer Academic, 1996, pp. 335–362.
[19] R. Mehra and J. Rabaey, "Behavioral level power estimation and exploration," in *Proc. Int. Workshop Low Power Design*, Napa, CA, Apr. 1994, pp. 197–202.
[20] C. A. Neugebauer and R. O. Carlson, "Comparison of wafer scale integration with VLSI packaging approaches," *IEEE Trans. Comp., Hybrids, and Manufact. Technol.*, June 1987, pp. 184–189.
[21] K. Nogami and A. El Gamal, "A CMOS 160 Mb/s phase modulation I/O interface circuit," in *Proc. Int. Solid-State Circuits Conf.*, San Francisco, CA, Feb. 1994, pp. 108–109.
[22] E. Olson and S. Kang, "Low-power state assignment for finite state machines," in *Proc. Int. Workshop Low Power Design*, Napa, CA, Apr. 1994, pp. 63–68.
[23] A. Park and R. Maeder, "Codes to reduce switching transients across VLSI I/O pins," *Comput. Arch. News*, pp. 17–21, Sept. 1992.
[24] M. Pedram, "Power minimization in IC design," *ACM Trans. Design Automat. Electron. Syst.*, vol. 1, no. 1, Jan. 1996.
[25] Rambus, Inc., Mountain View, CA, *Rambus Architectural Overview*, 1993.
[26] T. R. N. Rao and E. Fujiwara, *Error Control Coding for Computer Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1989.
[27] D. Salzmann, T. Knight, and P. Franzon, "Application of capacitive coupling to switch fabrics," in *Proc. Multi-Chip Modules Conf.*, Santa Cruz, CA, Jan. 1995, pp. 195–199.
[28] A. A. Shen, A. Ghosh, S. Devadas, and K. Keutzer, "On average power dissipation and random pattern testability of CMOS combinational logic networks," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 1992, pp. 402–407.
[29] P. H. Siegel, "Recording codes for digital magnetic storage," *IEEE Trans. Magnet.*, pp. 1344–1349, Sept. 1985.
[30] D. Singh, J. M. Rabaey, M. Pedram, F. Catthoor, S. Rajgopal, N. Sehgal, and T. J. Mozdzen, "Power conscious cad tools and methodologies: A perspective," *Proc. IEEE*, vol. 83, pp. 570–594, Apr. 1995.
[31] M. R. Stan, "Low-power techniques in CMOS VLSI," Ph.D. dissertation, Univ. Massachusetts, Dep. Elec. Comput. Eng., Sept. 1996.
[32] M. R. Stan and W. P. Burleson, "Limited-weight codes for low-power I/O," in *Proc. Int. Workshop Low Power Design*, Napa, CA, Apr. 1994, pp. 209–214.
[33] M. R. Stan and W. P. Burleson, "Bus-invert coding for low power I/O," *IEEE Trans. VLSI Syst.*, vol. 3, pp. 49–58, Mar. 1995.
[34] M. R. Stan and W. P. Burleson, "Coding a terminated bus for low power," in *Great Lakes Symp. VLSI*, Buffalo, NY, Mar. 1995, pp. 70–73.
[35] M. R. Stan and W. P. Burleson, "Low-power CMOS clock drivers," in *Proc. TAU Int. Workshop on Timing Issues*, Seattle, WA, Nov. 1995, pp. 149–156.
[36] M. R. Stan and W. P. Burleson, "Two-dimensional codes for low power," in *Proc. Int. Symp. Low Power Electron. Design*, Monterey, CA, Aug. 1996.
[37] C.-L. Su, C.-Y. Tsui, and A. M. Despain, "Saving power in the control path of embedded processors," *IEEE Design Test Comput.*, vol. 11, pp. 24–30, 1994.
[38] J. F. Tabor, "Noise reduction using low weight and constant weight coding techniques," Master's Thesis, Massachusetts Inst. Technol., Cambridge, May 1990.
[39] C.-Y. Tsui, M. Pedram, C.-H. Chen, and A. M. Despain, "Low power state assignment targeting two- and multi-level logic implementations," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 1994, pp. 82–87.
[40] N. Weste and K. Eshraghian, Eds., *Principles of CMOS VLSI Design*. Reading, MA: Addison Wesley, 1993.
[41] S. Wuytack, F. Catthoor, F. Franssen, L. Nachtergaele, and H. D. Man, "Global communication and memory optimizing transformations for low power systems," in *Proc. Int. Workshop Low Power Design*, Napa, CA, Apr. 1994, pp. 203–208.
[42] T. Yamauchi, Y. Morooka, and H. Ozaki, "A low power and high speed data transfer scheme with asynchronous compressed pulse width modulation for AS-memory," in *Proc. Symp. VLSI Circuits*, Kyoto, Japan, June 1995, pp. 27–28.
[43] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Trans. Inform. Theory*, vol. 23, pp. 337–343, 1977.

Mircea R. Stan (M'94) received the Diploma in electronics from the Polytechnic Institute of Bucharest, Romania, in 1984, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Massachusetts, Amherst, in 1994 and 1996, respectively.

Since 1996, he has been an Assistant Professor in the Electrical Engineering Department at the University of Virginia, Charlottesville. His research interests include low-power VLSI, mixed-mode analog and digital circuits, high-performance design, embedded systems and hardware/software codesign, FPGA's, and reconfigurable computing. He has accumulated more than seven years of industrial experience as an R&D Engineer in Bucharest, Romania, Tokyo, Japan, and Atlanta, GA.

Dr. Stan received an NSF Career Award for investigating low-power design techniques in 1997. He is a member of the ACM, Usenix, and also a Phi Kappa Phi and Sigma Xi.

Wayne P. Burleson (M'95) received the B.S.E.E. and M.S.E.E. degrees from Massachusetts Institute of Technology, Cambridge, in 1983, and the Ph.D. degree from the University of Colorado, Boulder, in 1989.

He has been working in the area of VLSI Signal Processing for 14 years. His work has included research, development, teaching, and industrial work at a variety of levels including development of algorithms, architectures, circuits, and CAD tools. He is currently an Associate Professor of Electrical and Computer Engineering at the University of Massachusetts, Amherst, where he has been since 1990. He worked as a custom DSP chip designer for four years for VLSI Technology, Inc., and Fairchild Semiconductor. He was a Visiting Professor at the Ecole Nationale Superieure des Telecommunications, Paris, France, from September 1996 to August 1997. He had edited two special issues of journals in this area.

Dr. Burleson is a member of the ACM and Sigma Xi and served on various program committees in VLSI Signal Processing.