A Coding Framework for Low-Power Address and Data Busses

Sumant Ramprasad, Naresh R. Shanbhag, *Member, IEEE*, and Ibrahim N. Hajj, *Fellow, IEEE*

***Abstract*—This paper presents a source-coding framework for the design of coding schemes to reduce transition activity. These schemes are suited for high-capacitance busses where the extra power dissipation due to the encoder and decoder circuitry is offset by the power savings at the bus. In this framework, a data source (characterized in a probabilistic manner) is first passed through a decorrelating function f_1. Next, a variant of entropy coding function f_2 is employed, which reduces the transition activity. The framework is then employed to derive novel encoding schemes whereby practical forms for f_1 and f_2 are proposed. Simulation results with an encoding scheme for data busses indicate an average reduction in transition activity of 36%. This translates into a reduction in total power dissipation for bus capacitances greater than 14 pF/b in 1.2-μm CMOS technology. For a typical value for bus capacitance of 50 pF/b, there is a 36% reduction in power dissipation and eight times more power savings compared to existing schemes. Simulation results with an encoding scheme for instruction address busses indicate an average reduction in transition activity by a factor of 1.5 times over known coding schemes.**

***Index Terms*—CMOS VLSI, coding, high-capacitance busses, low-power design, switching activity.**

I. Introduction

POWER dissipation has become a critical design concern in recent years driven by the emergence of mobile applications. Reliability concerns and packaging costs have made power optimization relevant even for tethered applications. As system designers strive to integrate multiple systems on-chip, power dissipation has become an equally important parameter that needs to be optimized along with area and speed. Therefore, extensive research into various aspects of low-power system design is presently being conducted [3], [4], [10]. Power-reduction techniques have been proposed at all levels of the design hierarchy beginning with algorithms [3], architectures [14], and ending with circuits [1], [13], and technological innovations [6].

The power dissipated at the input/output (I/O) pads of an integrated circuit (IC) ranges from 10% to 80% of the total power dissipation with a typical value of 50% for circuits optimized for low power [18]. The high-power dissipation at the I/O pads is because off-chip busses have switching capacitances that are orders of magnitude greater than those internal to a chip. Power dissipation on these busses mainly occur during signal transitions and reducing them will reduce total power dissipation. Therefore, various techniques have been proposed in literature [2], [8], [12], [17]–[21] to encode data on a bus to reduce the average and peak number of transitions.

In this paper, we present a source-coding framework for describing low-power encoding schemes and then employ the framework to develop new encoding schemes. This paper is a continuation of our effort in developing an information-theoretic view of very large scale integration (VLSI) computation [9], [16], whereby equivalence between computation and communication is being established. This equivalence has provided lower bounds on power dissipation for digital VLSI systems [9], [16] and has, for the first time, provided a common thread linking various levels of the VLSI design hierarchy. In the framework proposed here, a data source (characterized in a probabilistic manner) is first processed by a decorrelating function f_1. Next, a variant of entropy coding function f_2 is employed, which reduces the transition activity. All past work in this area [2], [8], [17]–[20] can be shown to be special cases of this framework.

Data transfers on microprocessor address busses are often sequential (i.e., current data value equals the previous data value plus a constant increment) due to fetches of instructions and array elements. This property can be exploited to reduce transitions by the use of Gray codes, as proposed in [20]. The Gray code reduces the number of transitions for sequential accesses because consecutive code words differ in only one bit position. This code does not, however, significantly reduce transitions for data busses because consecutive data values are typically not sequential. Another encoding scheme for address busses (denoted as the $T0$ scheme) is presented in [2], in which, if the next address is greater than the current address by an increment of one, then the bus lines are not altered and an extra "increment" bit is set to one. As mentioned before, this encoding scheme is not effective for data busses.

In [8], a scheme, termed bus-invert coding in [18], is presented to reduce the number of transitions on a bus. This scheme determines the number of bus lines that normally change state when the next output word is clocked onto the bus. When the number of transitions exceeds half the bus width, the output word is inverted before being clocked onto the bus. An extra output line is then employed to signal the inversion. The bus-invert coding scheme performs well when the data is uncorrelated; i.e., it assumes a decorrelating function precedes the bus-invert step. For uniformly distributed,

Manuscript received May 13, 1997; revised January 23, 1998. This work was supported by the Defense Advanced Research Projects Agency under Contract DABT63-97-C-0025, by the National Science Foundation under Career Award MIP-9623737, and by the Joint Services Electronics Program under Contract 00014-96-1-0129.

The authors are with the Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA.

Publisher Item Identifier S 1063-8210(99)02612-8.

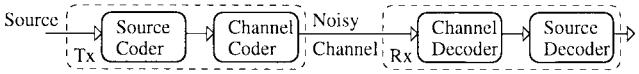


Fig. 1. Generic communication system.

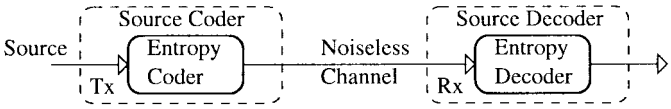


Fig. 2. Generic communication system for a noiseless channel.

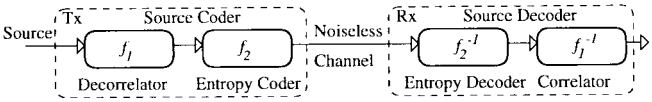


Fig. 3. Practical communication system for a noiseless channel.

uncorrelated data and a bus width of 8 b, the reduction in transitions is approximately 18%. Bus-invert coding also halves the peak number of transitions. This scheme does not, however, perform well for correlated data and for larger bus widths. It was proposed in [19] that larger busses be split into smaller busses and that correlated data be compressed in a lossless manner to achieve decorrelation. The power dissipation in compression and uncompression steps, however, can be considerable, thereby offsetting any savings in power on the bus. Employing the source-coding framework in this paper, we show that a two-step process involving functions f_1 and f_2, in most cases, is better (in terms of coding hardware overhead and reduction of transition activity) than compression followed by bus-invert coding.

One of the choices for the function f_2 in our framework is similar to the scheme in [7], where signal samples having higher probability of occurrence are assigned code words with fewer "ON" bits. This scheme is suited for optical networks where the power dissipation depends on the number of ON bits. In VLSI systems, however, power dissipation depends on the number of transitions rather than the number of ON bits.

Simulation results are presented employing the proposed encoding schemes, which indicate an average reduction in transition activity of 36% for one such encoding scheme for data streams. In addition, it is shown that a reduction in power dissipation occurs if the bus capacitance is greater than 14 pF/b in 1.2-μm CMOS technology. Simulation results with an encoding scheme for instruction address busses indicate an average reduction in transition activity by a factor of three and 1.5 times over the Gray and $T0$ [2] coding schemes, respectively. We also present an adaptive scheme that monitors the statistics of the input and adapts its encoding scheme over time.

The remainder of this paper is organized as follows. In Section II, a framework for describing encoding schemes is presented. In Section III, we develop the proposed encoding schemes and examine the effect of adding redundancy to the encoding schemes. In Section IV, simulation results are presented to compare the performance of the proposed and existing schemes.

II. Source-Coding Framework

In this section, we describe the components of a generic communications system and then develop the proposed framework.

A. Generic Communications System

A generic communication system (see Fig. 1) consists of a *source coder*, *channel coder*, *noisy channel*, *channel decoder*, and *source decoder*. The source coder (decoder) compresses (decompresses) the input data so that the number of bits required in the representation of the source is minimized. While the source coder removes redundancy, the channel coder adds just enough of it to combat errors that may arise due to the noise in the physical channel. This view of a communication system has been the basis for the enormous growth in the distinct areas of source coding, channel coding, and error correcting codes. In the present context, we consider the bus between two chips as the physical channel and the transmitter and receiver blocks to be a part of the pad circuitry, driving (in case of the transmitting chip) or detecting (in case of the receiving chip) the data signals. Furthermore, unlike in [16], we will assume here that the signal levels are sufficiently high so that the channel can be considered as being noiseless. While this *noiseless channel* assumption is true for most systems today, this will not be the case for future systems where the signal swings will be reduced to reduce power. The noiseless channel assumption allows us to eliminate the channel coder, resulting in the system shown in Fig. 2. Here, we have an *entropy coder* at the transmitter, which compresses the source into a minimal representation i.e., a representation requiring the minimum number of bits. This number is given by the *entropy* $H(X)$ [5] of the source X. In order to define the entropy, consider the source in Fig. 2 to be a discrete source generating symbols from the set $S_X = \{X_0, X_1, \cdots, X_{L-1}\}$ according to a probability distribution $\Pr(X)$. A measure of the information content of this source is its *entropy* $H(X)$, given by (1) where $P_i \triangleq \Pr(X = X_i)$ for $i = 0, \cdots, L-1$ and where $H(X)$ is in bits as follows:

$$H(X) = -\sum_{i=0}^{L-1} P_i \log_2(P_i) \text{ bits.} \tag{1}$$

In practice, due to data dependencies and the constraint of having integer code-word lengths, it becomes necessary to take data blocks $\{x(n), x(n-1), \cdots, x(n-M+1)\}$ to create a "supersymbol" and then apply entropy coding. Doing so, however, increases the coder hardware complexity exponentially with the block length M. To address this problem, low-complexity schemes (see Fig. 3) involving a decorrelating function f_1 followed by a scalar entropy coder f_2 have been proposed. This is a suboptimal structure if the function f_1 is unable to remove all the data dependencies. If the function f_1 is a linear predictor, then the structure shown in Fig. 4 is obtained, where $\boldsymbol{F}$ represents the vector of the impulse response of a linear filter. It can be seen that the linear predictor f_1 removes the

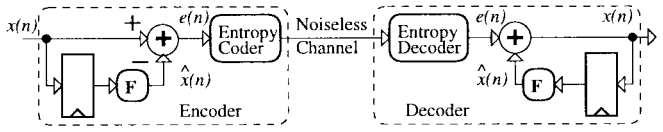


Fig. 4. Linear prediction configuration.

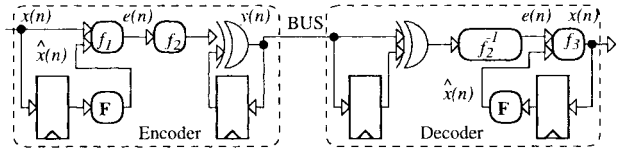


Fig. 5. Framework for low-power encoder and decoder.

linear dependencies in $x(n)$, and the entropy coder f_2 then compresses the output of f_1 in a lossless manner.

B. The Source-Coding Framework

The proposed framework in Fig. 5 is based upon the low-complexity source-coder architecture (see Fig. 4). As discussed before, the function f_1 decorrelates the input $x(n)$. Therefore, the prediction error $e(n)$ is a general function of the current value of $x(n)$ and the prediction, $\hat{x}(n)$ as follows:

$$e(n) = f_1(x(n), \hat{x}(n)) \tag{2}$$

where f_1 could be a linear or a nonlinear function of its arguments. The prediction $\hat{x}(n)$ is a function of the past values of $x(n)$ as follows:

$$\hat{x}(n) = \boldsymbol{F}(x(n-1), x(n-2), \cdots, x(n-M+1)). \tag{3}$$

For complexity reasons, we restrict ourselves to a value of $M = 2$.

The function f_2 employs a variant of entropy coding whereby, instead of minimizing the average number of bits at the output, it reduces the average number of transitions. The function f_2 employs the error $e(n)$ to generate an output $y(n)$, which has a "one" to indicate a transition and a "zero" to indicate no transition. This code word is then passed through an XOR gate to generate the corresponding signal waveforms on the bus. Hence, the output $y(n)$ of the encoder is given by

$$y(n) = f_2(f_1(x(n), \hat{x}(n))) \oplus y(n-1). \tag{4}$$

In summary, the function f_1 decorrelates the input and, in the process, skews the input probability distribution so that f_2 can reduce the transition activity by a memoryless mapping of $e(n)$. The output $x(n)$ of the decoder is given by

$$x(n) = f_3(f_2^{-1}(y(n-1) \oplus y(n)), \hat{x}(n)) \tag{5}$$

where f_3 is determined by the choice of f_1. The function f_2^{-1} assigns a level to the input based on an exclusive-or of the previous input $y(n-1)$ and the current input $y(n)$. The function f_3 calculates $x(n)$ employing the error $e(n)$ and the prediction $\hat{x}(n)$. A chip or a pad, which both sends and receives data to and from a bus, will need an encoder and a decoder. In order to implement the encoder and the decoder for a B-b input, at most $4B$ delays and $2B$ exclusive-or gates are needed, in addition to the hardware required to implement the functions $\boldsymbol{F}$, f_1, f_2, f_2^{-1}, and f_3. It is possible to reduce the hardware depending on the actual choices for f_1 and f_2 by optimizing the encoder and decoder each as a whole and also by sharing logic between the encoder and decoder in a bidirectional pad.

In Section II-C, we propose practical choices for $\boldsymbol{F}$, f_1, and f_2 and then evaluate the performance of encoding schemes employing different combinations of $\boldsymbol{F}$, f_1, and f_2. Since the aim of the encoding is to reduce power dissipation, for a choice of $\boldsymbol{F}$, f_1, and f_2 to be practical, the power dissipation at the encoder and decoder should be less than the savings achieved at the bus. This, in turn, implies that the amount of hardware in the encoder and decoder should be as small as possible. In addition, in systems where the latency for bus access has to be small, the delay through the encoder and decoder has to be low.

C. Alternatives for $\boldsymbol{F}$

1) Identity: The output of the *identity* function is equal to the input

$$\text{Identity}(a) = a. \tag{6}$$

Hence, if *identity* is employed for $\boldsymbol{F}$, then $\hat{x}(n) = x(n-1)$. The *identity* function requires no hardware to implement and is useful if the data source has significant correlation.

2) Increment The output of the *increment* function is equal to the input plus one as follows:

$$\text{Increment}(a) = a + 1. \tag{7}$$

Hence, if *increment* is employed for $\boldsymbol{F}$, then $\hat{x}(n) = x(n-1)+1$. The *increment* function is useful if $x(n)$ is the data on the address bus of a microprocessor because, due to fetches of instructions and array elements, the next address on the address bus usually equals the current address plus unity (or some power of two). This function requires an incrementer each at the encoder and decoder.

D. Alternatives for f_1

1) Exclusive-Or (XOR): The *exclusive-or* function xor is given by a bit-wise exclusive-or of the current input and the prediction

$$\text{xor}(x(n), \hat{x}(n)) = x(n) \oplus \hat{x}(n). \tag{8}$$

An example of the xor function for a 3-b input word is shown in Table I. If the input $x(n)$ is B-b wide, then the xor function will require B exclusive-or gates at the encoder. It can be shown that if f_1 is an xor function (at the encoder), then f_3 is also an xor function (at the decoder). Hence, B exclusive-or gates are required to implement f_3. If $\boldsymbol{F}$ is *identity*, then the only difference between the encoder and decoder is that the encoder employs f_2, whereas the decoder employs f_2^{-1}. Therefore, the hardware between the encoder and decoder in a bidirectional pad can be shared, as shown in Fig. 6, by employing an additional control line into a multiplexer that

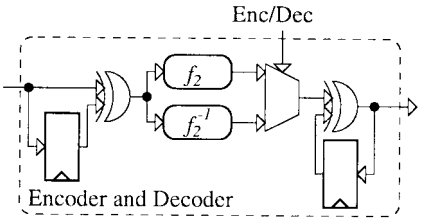


Fig. 6. Encoder and decoder can share hardware when f_1 is xor and $\boldsymbol{F}$ is *identity*.

```
if (x(n) ≥ x̂(n) && 2x̂(n) ≥ x(n))
   dbm = 2x(n) − 2x̂(n);
else if (x(n) < x̂(n) && 2x̂(n) − x(n) < 2^B)
   dbm = 2x̂(n) − 2x(n) − 1;
else if (x̂(n) < 2^(B−1))
   dbm = x(n);
else
   dbm = 2^B − 1 − x(n);
```

Fig. 7. dbm function.

TABLE I
EXAMPLE OF XOR AND DBM

$\hat{x}(n)$	$x(n)$	$xor(x(n), \hat{x}(n))$	$dbm(x(n), \hat{x}(n))$
010	000	010	011
010	001	011	001
010	010	000	000
010	011	001	010
010	100	110	100
010	101	111	101
010	110	100	110
010	111	101	111

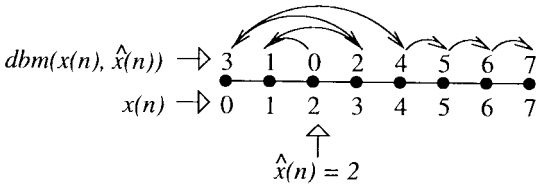


Fig. 8. Example of (dbm).

selects either the output of f_2 or that of f_2^{-1} depending on whether the input is to be encoded or decoded, respectively.

2) Difference-Based Mapping (dbm): The *difference-based mapping* dbm is described in Fig. 7, where the dbm function returns the difference between $x(n)$ and $\hat{x}(n)$ properly adjusted so that the output fits in the available B b. If the input $x(n)$ is B–b wide, then the dbm function will require three B-b subtracters, B inverters, and B 4-to-1 multiplexers each at the encoder and decoder. An example of dbm is shown in Table I, where we see that the dbm output is zero when the current and previous inputs are equal, and it increases as the distance (absolute difference) between the current input and the prediction increases. This is also shown graphically in Fig. 8.

In this paper, we will use the audio, video, random, and ASCII data sets in Table II to provide simulation results.

TABLE II
DESCRIPTION OF DATA SETS

Data	Description
A3	2.88MB of 16 bit PCM audio data (pop music)
A4	2.88MB of 16 bit PCM audio data (pop music)
A7	2.88MB of 16 bit PCM audio data (classical music)
CO	0.80MB of 16 bit communications channel data
V1	3.80MB of 8 bit video data (miss america)
V2	22.7MB of 8 bit video data (football)
V3	9.70MB (380 QCIF frames) of 8 bit video data (car phones)
R1	0.10MB of white, uniformly distributed data
PS	0.10MB Postscript file

TABLE III
CORRELATION AND KULLBACK–LEIBLER DISTANCE BEFORE AND AFTER XOR AND DBM

Data	Correlation			Kullback Leibler distance (in bits)		
	Orig.	*xor*	*dbm*	D(Orig.‖unif.)	D(*xor*‖unif.)	D(*dbm*‖unif.)
A3	0.9628	**0.1750**	0.4752	1.17	2.76	**3.25**
A4	0.9712	**0.1463**	0.5484	2.05	4.03	**4.52**
A7	0.9922	**0.1116**	0.8079	3.20	5.28	**5.80**
CO	0.2952	**-0.2350**	-0.3430	**1.64**	1.28	1.38
V1	0.9672	**0.2238**	0.4537	2.25	3.80	**4.18**
V2	0.8747	**0.2550**	0.4470	1.05	2.20	**2.56**
V3	0.9199	**0.2657**	0.4007	0.64	2.54	**2.93**
R1	**0.0012**	-0.0013	-0.0023	**0.00**	**0.00**	**0.00**
PS	**0.2367**	0.2791	0.4954	**3.75**	2.72	2.44

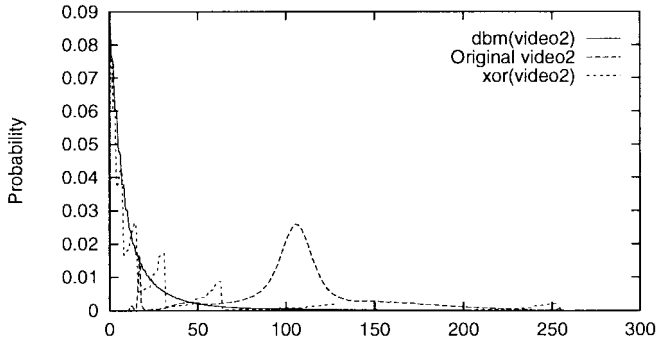


Fig. 9. Probability distribution for $V2$ data before and after applying xor and dbm.

In Table III, we compare signal correlation before and after applying xor and dbm to the data sets in Table II assuming $\boldsymbol{F}$ is the *identity* function, i.e., $\hat{x}(n) = x(n-1)$. The signal correlation ρ is given by

$$\rho = \frac{E[(x(n)-\mu)(x(n-1)-\mu)]}{E[(x(n)-\mu)^2]} \tag{9}$$

where "μ" is the mean of the signal $x(n)$ and $E[]$ is the expectation operator. We see that there is a reduction in correlation after applying xor and dbm. The probability distribution at the output of xor and dbm for $V2$ data in Table II is shown in Fig. 9 along with the original probability distribution. Both xor and dbm skew the original distribution for most of the data sets and, hence, enable f_2 to reduce the number of transitions even more. The skew in the probability distributions at the output of f_1 can be measured in terms of the Kullback–Leibler distance [5] between the given probability distribution p and the uniform distribution q, and is given by (10). A higher Kullback–Leibler distance from the uniform

```
if (n1(e(n)) > B/2)
  inv = invert bits in e(n) and set invert bit to 1
else
  inv = do not invert bits in e(n) and set invert bit to 0
```

Fig. 10. The function $\mathrm{inv}(e(n))$.

TABLE IV
EXAMPLE OF inv, pbm, AND vbm FUNCTIONS

$a = x(n) \bigoplus y(n-1)$	$\Pr(a)$	$inv(a)$	$pbm(a)$	$vbm(a)$
000	0.2864	0000	000	000
001	0.1766	0001	010	001
010	0.2201	0010	001	010
011	0.1712	1100	100	100
100	0.0573	0100	011	011
101	0.0344	1010	110	110
110	0.0259	1001	101	101
111	0.0280	1000	111	111

distribution indicates a more effective f_1 because it indicates a more skewed distribution, which in turn, would enable f_2 to reduce the number of transitions even more as follows:

$$D(p\|q) = \sum p(x) \log_2 \left(\frac{p(x)}{q(x)} \right) \text{ bits.} \tag{10}$$

E. Alternatives for f_2

1) Invert (inv): The inv function is given in Fig. 10, where $n1(\cdot)$ returns the number of ones in the binary representation of its argument. If the number of ones in $e(n)$ [or the number of transitions in $x(n)$] exceeds half the number of bus lines, then the input is inverted and the inversion is signaled using an extra bit. The function inv has been employed in bus-invert coding [18]. An example of the inv function is shown in Table IV. The hardware required to implement the inv function is described in [8] and [18].

2) Probability-Based Mapping (pbm)*:* In the pbm function, the number of ones in the input is reduced by assigning, as in [7], code words with fewer ones to the more frequently occurring code words. We then map a "one" to a transition waveform and a "zero" to a transitionless waveform using an exclusive-or gate. Thus, pbm satisfies (11) as follows:

$$\forall(a,b), \qquad \Pr(a) > \Pr(b) \Rightarrow n1(\mathrm{pbm}(a)) \leq n1(\mathrm{pbm}(b)). \tag{11}$$

The probabilities in (11) can be computed using a representative data sequence. The function pbm reduces the number of ones. Thus, if the most probable value of $e(n)$ is a, then $\mathrm{pbm}(a) = 0$. The next B most probable values of $e(n)$ are mapped to 2^i $(i = 0 \cdots B-1)$ by pbm. The next $\binom{B}{2}$ most probable values are mapped to all values with exactly two ones, and so on. An example of pbm is shown in Table II. The hardware required to implement pbm will depend on the input probability distribution. An 8-b input typically requires approximately 800 gates each to implement pbm and pbm^{-1}. Since the hardware requirement of pbm can grow exponentially with the input bit-width, we can split wide busses into multiple narrow busses and apply pbm independently on each of the smaller busses.

```
Determine i, the number of 1's in vbm(a), using (2.13);
val = a − Σ_{j=0}^{i−1} C(k,j) + 1;
for (j = k; j > 0; j − −)
  if (val ≤ C(j−1, i))
    set bit j of vbm to 0;
  else {
    set bit j of vbm to 1;
    val = val − C(j−1, i);
    i − −;
  }
```

Fig. 11. The algorithm to implement $\mathrm{vbm}(a)$.

3) Value-Based Mapping (vbm): In Fig. 9, we see that after xor and dbm are applied, smaller values are generally more probable than larger values. This is especially true if f_1 is dbm. We make use of this feature in vbm, in which code words with fewer ones are assigned to smaller values and then map a "one" to a transition waveform and a "zero" to a transitionless waveform using an exclusive-or gate. The function vbm is such that it satisfies (12) as follows:

$$\forall(a,b), \qquad a < b \Rightarrow n1(\mathrm{vbm}(a)) \leq n1(\mathrm{vbm}(b)). \tag{12}$$

The function vbm makes the assumption that smaller values are more probable than larger values. An example of vbm is shown in Table IV. The advantage of vbm over pbm is that a representative data sequence is not needed. The reduction in transitions with vbm, however, is usually lower than pbm. Note that the function vbm can be generated algorithmically by realizing that if there are i ones in $\mathrm{vbm}(a)$ then bit $k-1$ of $\mathrm{vbm}(a)$ is one if $a - \Sigma_{j=0}^{i-1} \binom{k}{j} > \binom{k-1}{i}$. Once bit $k-1$ of $\mathrm{vbm}(a)$ is determined, the lower order bits can be similarly determined in an iterative fashion. The number of ones, i, in $\mathrm{vbm}(a)$ can be determined by finding i such that (13) is satisfied as follows:

$$\sum_{j=0}^{i} \binom{k}{j} \leq a+1 < \sum_{j=0}^{i+1} \binom{k}{j}. \tag{13}$$

The algorithm to implement vbm is shown in Fig. 11. This algorithm can also be employed to generate a j-limited-weight code [19] for an arbitrary j. A similar algorithm is employed at the decoder. The algorithm in Fig. 11 requires k clocks, where k is the number of bits in the output $y(n)$. The binomial coefficients $\binom{j}{i}$ can be precomputed; $(k-2)(k-4)/4$ binomial coefficients need to be precomputed. The function vbm can be implemented either algorithmically, employing the algorithm in Fig. 11, or by employing combinational logic. An 8-b input typically requires 700 gates each to implement vbm and vbm^{-1}.

III. ENCODING SCHEMES

In this section, we present reduced transition activity encoding schemes based upon alternatives for the functions $\boldsymbol{F}$, f_1, and f_2, defined in the previous section. The proposed encoding schemes are summarized in Table V, where we have seven encoding schemes using different combinations of $\boldsymbol{F}$, f_1, and f_2.

TABLE V
ENCODING SCHEMES

Encoding scheme	F	f_1	f_2
xor-pbm	*Identity*	*xor*	*pbm*
xor-vbm	*Identity*	*xor*	*vbm*
dbm-pbm	*Identity*	*dbm*	*pbm*
dbm-vbm	*Identity*	*dbm*	*vbm*
xor-inv (Bus-Invert)	*Identity*	*xor*	*inv*
dbm-inv	*Identity*	*dbm*	*inv*
inc-xor	*Increment*	*xor*	*Identity*

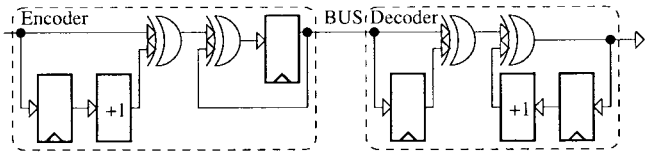


Fig. 12. Encoder and decoder for inc–xor.

The xor–pbm scheme reduces the number of transitions by assigning fewer transitions to the more frequently occurring set of transitions in the original signal. The closest approach in literature to xor–pbm is [7], where signal samples having higher probability of occurrence are assigned code words with fewer ON bits. In VLSI circuits, power dissipation depends on the number of transitions occurring at the capacitive nodes of the circuit. The xor–pbm scheme differs from [7] in two respects. It reduces the power dissipation at the capacitive nodes by reducing the number of transitions by assigning fewer transitions to the more frequently occurring set of transitions. The xor–pbm scheme also achieves a greater reduction in transitions by skewing the input probability distribution by employing xor for f_1. The xor–vbm scheme has the advantage over xor–pbm of being an input independent mapping and requiring lesser hardware to implement at the cost of typically lesser reduction in transitions. The scheme dbm–pbm requires more hardware than xor–pbm, but also reduces transitions more because the function dbm skews the input probability distribution more than xor.

The framework in Fig. 5 can be employed to derive and improve existing coding schemes. For example, the Gray coding scheme can be derived by assigning $f_1(x(n), \hat{x}(n)) = x(n)$, f_2 being the Gray mapping, and removing the exclusive-or at the output of the encoder. The bus-invert [18] coding scheme can be obtained by assigning $f_1 = \text{xor}$ and $f_2 = \text{inv}$. A variant of the bus-invert coding scheme can be obtained by employing dbm instead of xor for f_1, resulting in the dbm–inv scheme. The scheme in [7] can be derived by assigning $f_1(x(n), \hat{x}(n)) = x(n)$, $f_2 = \text{pbm}$, and removing the exclusive-or at the output of the encoder. An improved version of the $T0$ scheme in [2] can be derived using our framework by assigning $\boldsymbol{F}$ = *increment* (with overflow being ignored), $f_1 = \text{xor}$, and $f_2 =$ *identity*. This improved scheme, called inc–xor, is shown in Fig. 12. Unlike the $T0$ scheme, the inc–xor scheme does not require an extra bit and, as shown in Section IV, has a shorter critical path and provides the same or more reduction in transitions.

The reduction in transitions due to a coding scheme can be increased by introducing, either spatial redundancy in the form of extra lines on the bus, or temporal redundancy in the form of extra clocks to transfer the data, or both [17]. For instance, if an extra bit is added to the bus, f_2 can use the extra bit to further reduce transitions. For instance, in the function pbm, the most probable value is assigned zero. Assuming the bus is $B+1$ bits wide, the next $B+1$ most probable values are assigned 2^i $(i = 0 \cdots B)$, and so on. The maximum number of transitions on the bus is less than or equal to $B/2$, resulting in a perfect $B/2$ limited weight code [18].

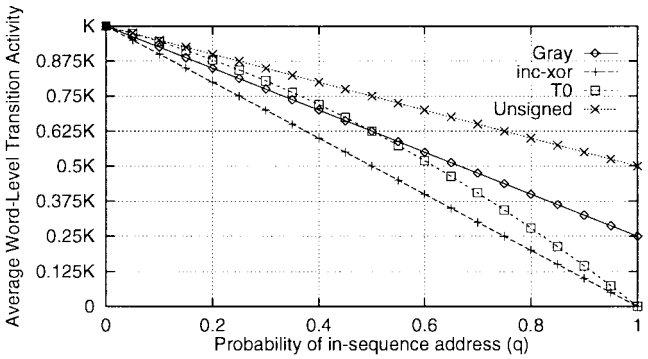


Fig. 13. Analytical estimate of transition activity for unsigned, Gray, $T0$, and inc–xor coding schemes.

We can employ the probabilistic model in [2] to estimate the transition activity of an address stream after encoding. This model employs a parameter q, the probability of having two consecutive addresses on the bus in two successive clock cycles. In order to simplify analysis, this model assumes that K lines make a transition on average when two nonconsecutive addresses are issued on the bus. In general, the value of q will depend on the source generating the data on the bus, while the value of K will depend both on the source and encoding scheme employed. With this model, the average number of transitions produced by the unsigned, Gray, $T0$, and inc–xor codes can be calculated as

$$T_{\text{Unsigned}} = (1-q)K_{\text{Unsigned}} + 2q \tag{14}$$

$$T_{\text{Gray}} = (1-q)K_{\text{Gray}} + q \tag{15}$$

$$T_{\text{inc–xor}} = (1-q)K_{\text{inc–xor}} \tag{16}$$

$$T_{T0} = (1-q)K_{T0} + 2q(1-q). \tag{17}$$

In Fig. 13, we plot T_{Unsigned}, T_{Gray}, T_{T0}, and $T_{\text{inc–xor}}$ as a function of the probability q assuming $K_{\text{Unsigned}} = K_{\text{Gray}} = K_{\text{inc–xor}} = K_{T0} = K$. We see that inc–xor always has the least transition activity, Unsigned always has the highest transition activity, and $T0$ is better than Gray only for $q > (1/2)$.

IV. SIMULATION RESULTS

In this section, we present simulation results for the reduction in transition activity and power dissipation using the encoding schemes and compare them with results obtained by existing schemes. In order to compare coding schemes, we classify them into two groups, based on whether they are more suited for data busses or address busses. For data busses, we compared the reduction in transition activity due to

TABLE VI
PERCENTAGE REDUCTION IN TRANSITION ACTIVITY

Data	*xor-pbm*	*xor-vbm*	*dbm-pbm*	*dbm-vbm*	*xor-pbm* with *pbm* opt. for	Redn.	Adapt. scheme
A3	33	25	**36**	29	A3	33	25
A4	38	29	**41**	31	A3	37	29
A7	42	33	**45**	25	A3	39	33
CO	27	4	**28**	**28**	A3	20	4
V1	40	34	**45**	45	V2	39	40
V2	33	26	**39**	39	V2	33	33
V3	34	26	**40**	40	V2	33	33
R1	**2**	0	2	0	V2	0	0
PS	**43**	23	38	25	PS	**43**	43

TABLE VII
PERCENTAGE REDUCTION IN TRANSITIION ACTIVITY WITH 1-b REDUNDANCY

Data	*xor-pbm*	*xor-vbm*	*dbm-pbm*	*dbm-vbm*	Bus-Inv.	*dbm-inv*	Bus-Inv. & gzip	*xor-pbm* with *pbm* opt. for	Redn.	Adapt. scheme
A3	36	32	**39**	39	8	12	2	A3	36	32
A4	40	36	**43**	43	7	13	5	A3	40	36
A7	43	36	**47**	47	7	15	10	A3	41	36
CO	32	22	**33**	32	15	16	17	A3	27	22
V1	42	39	**47**	46	7	32	4	V2	41	42
V2	38	34	**43**	43	10	22	0	V2	38	38
V3	38	34	**44**	43	11	27	6	V2	38	38
R1	**19**	18	19	18	18	18	18	V2	18	18
PS	47	33	42	36	11	14	**63**	PS	47	46

xor–pbm, xor–vbm, dbm–pbm, dbm–vbm, and bus-invert, and reduction in power dissipation due to xor–pbm and bus-invert. For address busses, we compared the reduction in transition activity and power dissipation due to $T0$, Gray, and inc–xor.

A. Reduction in Transition Activity

The reduction in transition activity when the schemes in Table V are applied to the data sets in Table II is shown in Table VI. The xor–vbm scheme, as expected, results in a slightly lesser reduction in transitions than xor–pbm. We can achieve an average reduction in transitions of 36% for audio data employing xor–pbm with pbm optimized for $A3$ data and an average reduction of 35% for video data employing pbm optimized for $V2$ data. Hence, pbm optimized for one video/audio sequence performs well for other video/audio sequences, thus indicating the robustness of these schemes to variations in signal statistics. There is little change in transitions for uniformly distributed uncorrelated data ($R1$ data). This occurs since $x(n)$ and $\hat{x}(n)$ are uncorrelated and, hence, all values of $f_1(x(n), \hat{x}(n))$ are equally probable. Therefore f_2 does not reduce the total number of transitions for $R1$ data. The dbm–pbm scheme reduces the transition activity around five percentage points more than xor–pbm because dbm skews the input probability distribution more than xor.

The reduction in transition activity with 1 b of spatial redundancy is shown in Table VII. For audio and video data, assuming 1 b of redundancy, xor–pbm performs better than the bus-invert scheme in [18] or the bus-invert with compression (gzip) scheme in [17]. For $R1$ data, xor–pbm does approximately the same as bus-invert, which is optimal for the given redundancy for $R1$ data. Compression followed by bus-invert is better for ASCII files because gzip is very effective in compressing ASCII files. It is possible to combine gzip with xor–pbm to obtain a 63% reduction in transitions for the ASCII data. The performance of xor–vbm with redundancy is only slightly worse than xor–pbm with redundancy. In addition, dbm–inv does better than bus-invert for all the data sets, except for $R1$ data, for which the performance is approximately equal. The advantage of the dbm–inv scheme over bus-invert increases as the data to be encoded is more correlated.

In Table VIII, we report the average word-level transition activity when unsigned, Gray, $T0$, and inc–xor encoding schemes are used to encode the addresses generated when different benchmark programs are executed on the SGI Power Challenge with an MIPS R10000 processor. As in [2], we consider the following three cases:

1) transitions on the instruction address bus (I);
2) transitions on the data address bus (D);
3) transitions on a multiplexed address bus (M).

Since the addresses of successive instruction and data elements on the MIPS processor differ by four, the encoding schemes encode only the most significant 30 bits. We ignore the transitions on the least significant two bits since they are usually zero. The greatest reduction in transition activity employing inc–xor is observed for instruction address streams because the probability of addresses being sequential is highest for such streams. We also see that inc–xor is better than $T0$ for 20 of the 24 streams. This is because inc–xor does not use an extra "increment" bit and, hence, saves on transitions on that bit while providing a similar reduction in transitions on the other bits. Of the four encoding schemes, we see that inc–xor provides the greatest reduction in transition activity for all the I-bus streams and six of the eight streams on the M-bus. The Gray encoding scheme provides the greatest reduction in transition activity for all the D-bus streams and two of the eight streams on the M-bus. The analytical estimate in Fig. 13 matches well with measured data in Table VIII in which we see that for the I-bus, which has a high value of q, the transition activity is in the descending order, unsigned, Gray, $T0$, inc–xor. For the D-bus, which has a low value of q, the Gray code is better than inc–xor because K_{Gray} is typically less than $K_{\text{inc–xor}}$. The values of K_{Unsigned}, K_{Gray}, $K_{\text{inc–xor}}$, K_{T0}, and q for benchmark programs are shown in Table VIII.

B. Reduction in Power Dissipation

The original uncoded power dissipation at the bus is given by (18) where T_x is the word-level transition activity of the input and C_L is the bus capacitance per bit as follows:

$$P_{D,\text{uncoded}} = T_x C_L V_{dd}^2 f. \tag{18}$$

For a given encoding and decoding scheme, we employed (19) to calculate the power dissipation as the sum of the power dissipation at the encoder, bus, and decoder

$$P_{D,\text{coded}} = P_{D,\text{enc}} + P_{D,\text{bus}} + P_{D,\text{dec}}. \tag{19}$$

In (19), power dissipation of $P_{D,\text{enc}}$ occurs in the transmitting chip and $P_{D,\text{dec}}$ occurs in the receiving chip. Thus, the power dissipation in the transmitting and receiving chips is increased in order to reduce total power dissipation. If T_y is the reduced

TABLE VIII
AVERAGE WORD-LEVEL TRANSITION ACTIVITY FOR REAL ADDRESSES

Address Type	Program	Stream Length	Transition Activity				q	K			
			Unsigned	Gray	T0	*inc-xor*		Unsigned	Gray	T0	*inc-xor*
I	gzip	11722992	2.09	1.32	0.65	**0.38**	0.89	3.86	4.05	4.08	**3.59**
	MPEG decoder	11481528	2.15	1.18	0.41	**0.29**	0.93	4.20	**3.69**	4.01	4.23
	espresso	8455329	2.18	1.24	0.50	**0.35**	0.92	4.74	**4.08**	4.55	4.62
	gunzip	3148893	2.22	1.23	0.51	**0.36**	0.92	4.65	**3.92**	4.52	4.70
	postgres	2378798	2.28	1.39	1.02	**0.54**	0.84	3.66	3.37	4.22	**3.32**
	ghostview	10884925	2.32	1.42	0.93	**0.60**	0.86	4.40	**3.99**	4.58	4.23
	gcc	413896	2.34	1.44	1.01	**0.70**	0.81	3.86	**3.39**	3.45	3.80
	gnuplot	1506882	2.44	1.46	0.86	**0.66**	0.89	5.90	**5.07**	5.58	5.83
D	gcc	175684	4.34	**3.46**	4.33	3.94	0.40	6.50	**5.19**	6.13	6.56
	espresso	6544671	5.60	**3.91**	5.65	5.13	0.38	8.65	**5.90**	8.05	8.31
	postgres	784467	6.42	**5.56**	6.41	6.40	0.08	7.18	**6.30**	6.86	6.96
	gunzip	818875	6.46	**6.02**	6.43	6.56	0.05	6.92	**6.50**	6.68	6.89
	MPEG decoder	3518525	6.92	**6.46**	6.91	7.06	0.08	7.68	**7.27**	7.37	7.65
	gzip	3277008	6.97	**6.48**	6.96	6.91	0.06	7.47	**7.03**	7.34	7.38
	ghostview	4115073	7.13	**5.70**	7.09	7.14	0.12	8.34	**6.81**	7.91	8.13
	gnuplot	662513	7.82	**6.27**	7.79	7.81	0.11	9.18	**7.38**	8.59	8.79
M	gzip	15000000	5.34	5.38	4.88	**4.37**	0.50	8.74	9.70	8.91	**8.69**
	gcc	589578	5.61	4.89	5.34	**4.73**	0.49	9.35	**8.66**	9.34	9.32
	gunzip	3967768	5.70	4.98	4.89	**4.58**	0.55	10.12	**9.82**	10.10	10.14
	MPEG decoder	15000000	6.68	**5.45**	5.91	5.64	0.50	11.30	**9.88**	11.23	11.27
	postgres	3163265	7.55	6.92	6.97	**6.61**	0.45	11.99	**11.76**	11.98	12.00
	gnuplot	2169395	7.55	7.03	7.14	**6.82**	0.38	11.03	**10.74**	10.97	11.01
	ghostview	15000000	8.06	7.35	7.60	**7.22**	0.40	12.07	**11.58**	12.08	12.04
	espresso	15000000	8.09	**5.38**	8.11	7.49	0.40	12.33	**8.24**	12.37	12.38

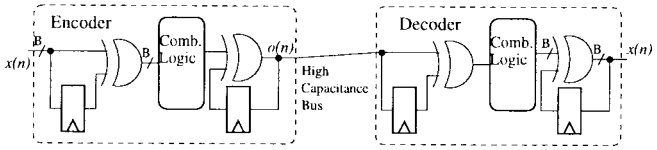


Fig. 14. Encoder and decoder for xor–pbm.

TABLE IX
AREA-DELAY POWER FOR BUS-INVERT AND xor–pbm

Measure	Bus-Invert	*xor-pbm*
$P_{D,enc} + P_{D,dec}$ (mW)	0.301	3.38
Critical Path (ns)	23	40
No. of Trans. in Enc. & Dec.	406	6482
Min. Bus Capacitance for Redn. in Total P.D. (pF/bit)	11.52	13.86

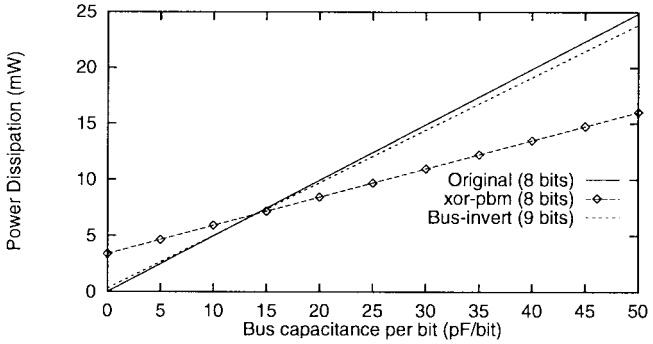


Fig. 15. Power dissipation for bus-invert and xor–pbm.

word-level transition activity at the bus after encoding, then the total power dissipation is given by (20) as follows:

$$P_{D,\text{coded}} = P_{D,\text{enc}} + T_y C_L V_{dd}^2 f + P_{D,\text{dec}}. \qquad (20)$$

We employed SIS [15] to generate the net-lists for the encoder and the decoder for xor–pbm and bus-invert and PSPICE[1] to estimate the power dissipation in the encoder $P_{D,\text{enc}}$ and the decoder $P_{D,\text{dec}}$. The encoder and decoder for xor–pbm are shown in Fig. 14. We employed 1.2-μm CMOS technology with 3.3-V supply voltage and 20-MHz frequency. The area-delay-power information is presented in Table IX. The clock frequency of 20 MHz was chosen so as to accommodate the slowest critical path of 40 ns in the xor–pbm coder. In Fig. 15, we plot the power dissipation for different bus capacitances C_L. The power dissipation was estimated using 30 samples of $V2$ input. Since the input is highly correlated, we see that for small bus capacitances (<11 pF/b), it is best to not encode the data at all. For capacitances above 11 pF/b, bus-invert provides a small reduction in power dissipation and for capacitances above 14 pF/b, xor–pbm has the lowest total power dissipation. For a bus capacitance of 50 pF/b, xor–pbm provides a power savings of 8.82 mW (or 36%), which is eight times the power savings of 1.01 mW provided by bus-invert. The slope of the graph of power dissipation versus bus capacitance is determined by T_y, the reduced word-level transition activity, and the y-intercept is determined by the total power dissipation in the encoder $(P_{D,\text{enc}})$ and decoder $(P_{D,\text{dec}})$. In our experiments, we synthesized for minimum-delay. Therefore, the power dissipation of the encoder and decoder can be further reduced by employing low-power synthesis techniques [11]. This would reduce the y-intercept in Fig. 15 and, in turn, reduce the crossover point at which xor–pbm would have the lowest power dissipation. The slope of the curve is determined only by the reduced word-level transition activity T_y and is independent of the

[1] *MicroSim PSPICE A/D Reference Manual*, MicroSim Corporation, Irvine CA, 1996.

TABLE X
AREA-DELAY POWER FOR GRAY, $T0$, AND inc–xor

Measure	Gray	T0	*inc-xor*
$P_{D,enc} + P_{D,dec}$ (mW)	0.2043	0.6358	0.6763
Critical Path (ns)	59.6	72.3	63.6
No. of Trans. in Enc. & Dec.	928	3320	4148

power dissipation in the encoder and decoder. The slope of the curve can be reduced further by employing a different coding scheme, which could result in a greater reduction in transitions.

Table X shows the area-delay-power information for the encoder and decoder for the Gray, $T0$, and inc–xor encoding schemes employing 1.2-μm CMOS technology, 3.3-V supply voltage, and 10-MHz clock frequency. The clock frequency of 10 MHz was chosen so as to accommodate the slowest critical path of 72.3 ns in the $T0$ coder. The power dissipation was estimated using 93 samples of an I address stream. All the encoders and decoders were optimized for minimum area. Hence, a ripple–carry structure was employed for the incrementers. The Gray encoder–decoder has the lowest area, delay, and power. It, however, does not always provide the most reduction in transition activity. The inc–xor scheme consumes slightly more area than $T0$, but has a shorter critical path, slightly lower power dissipation, and provides a higher reduction in transition activity than $T0$ and does not require an extra bit on the bus. The inc–xor scheme is better than the Gray coding scheme for all the I address streams and most of the M address streams in Table VIII because it provides a higher reduction in transition activity which, from (20), translates into a higher reduction in power dissipation for sufficiently high bus capacitance.

C. Extensions: Adaptive Encoding Scheme

In the previous subsections, we presented schemes which implemented a fixed I/O mapping that does not vary with time. Since the input data statistics can vary over time, there is a reason to examine the performance of adaptive schemes that adapt the mapping f_2 to changes in the input data statistics.

We simulated one such adaptive scheme whereby the encoder and decoder both keep track of the probabilities of the output of $f_1(x(n), \hat{x}(n))$. The function f_2 is implemented using a random-access memory (RAM) since its contents will change with time. After processing each sample, both the encoder and decoder update the RAM based on the probability distribution of $f_1(x(n), \hat{x}(n))$. The reduction in transitions for the adaptive scheme is given in Table VI when f_1 is xor. It can be seen that for video, random, and ASCII data which are 8-b wide, the adaptive scheme does nearly as well as xor–pbm without requiring that the input probability distribution be known *a priori*. The performance for the audio data, which are 16-b wide, is slightly less than xor–pbm because the data sequences available were not long enough for the adaptive scheme to converge to xor–pbm. Similar results were obtained for the adaptive scheme with 1 b of spatial redundancy, as shown in Table VII. One disadvantage of the adaptive scheme is the extra hardware required and, hence, this scheme is to be employed only when the bus capacitance is high, an input independent mapping is required, and vbm (also an input independent mapping) does not result in significant reduction in transitions.

V. CONCLUSIONS

We have presented a source-coding framework to describe encoding schemes to reduce transition activity and employed this framework to develop novel encoding schemes. The encoding schemes are suited for high-capacitance busses where the extra capacitance due to the encoder and decoder circuitry is offset by the savings in power dissipation at the bus. Simulation results show that two of the encoding schemes derived from the proposed framework (xor–pbm for data busses and inc–xor for address busses) perform better than existing ones. In summary, the proposed framework allows one to develop novel low-power encoding schemes and characterize existing schemes.

In the future, we will examine employing low-power design techniques to reduce the power dissipation in the encoder and decoder so that the encoding schemes can be employed for on-chip busses. We will also extend this framework by allowing the channel to be noisy and by including a channel coder.

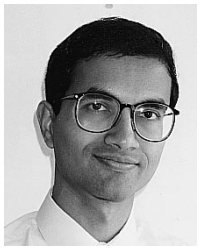
REFERENCES

[1] W. C. Athas, L. J. Svensson, J. G. Koller, N. Tzartzanis, and E. Y.-C. Chou, "Low-power digital systems based on adiabatic switching principles," *IEEE J. Solid-State Circuits*, vol. 2, pp. 398–407, Dec. 1994.
[2] L. Benini, G. De Micheli, E. Macii, D. Sciuto, and C. Silvano, "Asymptotic zero-transition activity encoding for address busses in low-power microprocessor-based systems," in *Great Lakes VLSI Symp. Dig.*, Urbana, IL, Mar. 13–15, 1997, pp. 77–82.
[3] A. P. Chandrakasan and R. W. Brodersen, "Minimizing power consumption in digital CMOS circuits," *Proc. IEEE*, vol. 83, pp. 498–523, Apr. 1995.
[4] A. P. Chandrakasan, M. Potkonjak, R. Mehra, J. Rabaey, and R. W. Brodersen, "Optimizing power using transformations," *IEEE Trans. CAD*, vol. 14, pp. 12–31, Jan. 1995.
[5] T. Cover and J. A. Thomas, *Elements of Information Theory*. New York: Wiley, 1991.
[6] B. Davari, R. H. Dennard, and G. G. Shahidi, "CMOS scaling for high-performance and low-power—The next ten years," *Proc. IEEE*, vol. 83, pp. 595–606, Apr. 1995.
[7] D. W. Faulkner, "PCM signal coding," U.S. Patent 5 062 152, Oct. 1991.
[8] R. J. Fletcher, "Integrated circuit having outputs configured for reduced state changes," U.S. Patent 4 667 337, May 1987.
[9] R. Hegde and N. R. Shanbhag, "Energy-efficiency in presence of deep submicron noise," in *Int. Conf. Computer-Aided Design*, San Jose CA, Nov. 8–12, 1998.
[10] M. Horowitz, T. Indermaur, and R. Gonzalez, "Low-power digital design," in *Proc. IEEE Symp. Low Power Electron.*, San Diego, CA, Oct. 1994, pp. 8–11.
[11] S. Iman and M. Pedram, "An approach for multilevel logic optimization targeting low power," *IEEE Trans. Computer-Aided Design*, vol. 15, pp. 889–901, Aug. 1996.
[12] E. Musoll, T. Lang, and J. Cortadella, "Exploiting the locality of memory references to reduce the address bus energy," in *Proc. Int. Symp. Low Power Electron. Design*, Monterey, CA, Aug. 18–20, 1997, pp. 202–207.
[13] Y. Nakagome, K. Itoh, M. Isoda, K. Takeuchi, and M. Aoki, "Sub-1-V swing internal bus architecture for future low-power ULSI's," *IEEE J. Solid-State Circuits*, vol. 28, pp. 414–419, Apr. 1993.
[14] K. K. Parhi and D. G. Messerschmitt, "Pipeline interleaving and parallelism in recursive digital filters—Part I: Pipelining using scattered look-ahead and decomposition," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 37, pp. 1099–1117, July 1989.
[15] E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton, and A. Sangiovanni-Vincentelli, "SIS: A system for sequential circuit

synthesis," Univ. California at Berkeley, Berkeley, Memo. UCB/ERL M92/41, May 1992.
[16] N. R. Shanbhag, "A mathematical basis for power-reduction in digital VLSI systems," *IEEE Trans. Circuits Syst. II*, vol. 44, pp. 935–951, Nov. 1997.
[17] M. R. Stan and W. P. Burleson, "Low-power encodings for global communication in CMOS VLSI," *IEEE Trans. VLSI Syst.*, vol. 5, pp. 444–455, Dec. 1997.
[18] ———, "Bus-invert coding for low-power I/O," *IEEE Trans. VLSI Syst.*, vol. 3, pp. 49–58, Mar. 1995.
[19] ———, "Limited-weight codes for low-power I/O," in *Proc. Int. Workshop Low Power Design*, Napa, CA, Apr. 1994, pp. 209–214.
[20] 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, Winter 1994.
[21] J. Tabor, "Noise reduction using low-weight and constant weight coding techniques," M.S. thesis, MIT, Cambridge, MA, May 1990.

Sumant Ramprasad received the B.Tech. degree in computer science and engineering from the Indian Institute of Technology, Bombay, India, in 1988, the M.S. degree in computer and information sciences from the Ohio State University, Columbus, in 1990, and is currently working toward the Ph.D. degree at the University of Illinois at Urbana-Champaign.

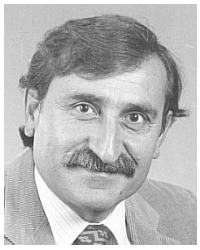
From 1991 to 1996 he worked at the Chicago Corporate Research Laboratories, Motorola Inc. He is currently working on high-level estimation, synthesis, and methodologies for low-power design.



Naresh R. Shanbhag (S'87–M'88) received the B.Tech degree from Indian Institute of Technology, New Delhi, India, in 1988, the M.S. degree from Wrigth State University, Dayton, OH, and the Ph.D. degree from University of Minnesota, Minneapolis, in 1993, all in electrical engineering.

From July 1993 to August 1995, he was with AT&T Bell Laboratories, Murray Hill, NJ, where he was a member of the Wide-Area Networks Group, responsible for development of VLSI algorithms, architectures, and implementation for high-speed data communications applications. In particular, he was the Lead Chip Architect for AT&T's 51.84-Mb/s transceiver chips over twisted-pair wiring for asynchronous transfer mode (ATM)–local area network (LAN) and broadband access applications. Since August 1995, he has been a Research Assistant Professor with the Coordinated Science Laboratory, VLSI Circuits Group, and an Assistant Professor with the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign. His research interests are in exploring the limits of computation in an integrated-circuit media in terms of power dissipation, reliability and throughput, and developing VLSI algorithms, architectures, and IC's for signal processing and communication systems. He has published 70 papers on these subjects and holds three U.S. patients.

Dr. Shanbhag received the IEEE Leon K. Kirchmayer Prize Paper Award (1999), the Xerox Faculty Award (1999), the Distinguished Lecturership Award of IEEE Circuit and Systems Society (1997–1999), the National Science Foundation Career Award (1996), and the Darlington Best Paper Award (1994) from the IEEE Circuits and Systems Society. Since July 1997, he has served as an associate editor for IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS PART II: ANALOG AND DIGITAL SIGNAL PROCESSING.



Ibrahim N. Hajj (S'64–M'70–SM'82–F'90) received the B.E. degree (with distinction) from the American University of Beirut, Beirut, Lebanon, the M.S. degree from the University of New Mexico, Albuquerque, and the Ph.D. degree from the University of California at Berkeley, all in electrical engineering.

He is currently a Professor of electrical and computer engineering and a Research Professor at the Coordinated Science Laboratory, University of Illinois at Urbana-Champaign. Prior to joining the University of Illinois at Urbana-Champaign, he was with the Department of Electrical Engineering University of Waterloo, Waterloo, Ont., Canada. His current research interests include computer-aided design of VLSI circuits, design for reliability and low-power synthesis, physical design, and testing. He has published over 160 journal and conference papers and book chapters on these subjects. He has co-authored *Switch-Level Timing Simulation of MOS VLSI Circuits* (Norwell, MA: Kluwer, 1989).

Dr. Hajj is a member of the Computer-Aided Network Design (CANDE), ACM, and Sigma Xi. He currently serves on the Board of Governors of the IEEE Circuits and Systems Society. He has served as an associate editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS PART II: ANALOG AND DIGITAL SIGNAL PROCESSING, and an associate editor of the *IEEE Circuits and Systems Magazine*. In 1992, he was corecipient of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN Best Paper Award.