Coding For Low-Power Address And Data Busses: A Source-Coding Framework and Applications*

Sumant Ramprasad, Naresh R. Shanbhag, Ibrahim N. Hajj
Coordinated Science Laboratory,
University of Illinois at Urbana-Champaign, Urbana IL 61801.

Abstract

Presented in this paper is 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 the decoder circuitry is offset by the power savings at the bus. A framework to characterize low-power encoding schemes is developed based upon the source-channel coding view. In this framework, a data source (characterized in a probabilistic manner) is passed through a decorrelating function f_1 first. 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 14pF/bit in 1.2μ CMOS technology and eight times more power savings compared to existing schemes with a typical value for bus capacitance of 50pF/bit. Simulation results with an encoding scheme for instruction address busses indicate an average reduction in transition activity by a factor of 3 times and 1.5 times over the Gray and T0 [1] coding schemes respectively.

1 INTRODUCTION

In recent years, power dissipation has become a critical design concern that needs to be optimized along with area and speed. At the system level, off-chip busses have switching capacitances that are orders of magnitude greater than those internal to a chip. Transitions on these busses result in considerable system power dissipation and reducing them will reduce total power dissipation. Therefore, various techniques have been proposed in literature [1, 3, 5, 9–13] to encode data on a bus to reduce the average and the 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 work is a continuation of our effort in developing an information-theoretic view of VLSI computation [8], whereby equivalence between computation and communication is being established. This equivalence has provided lower bounds on power dissipation for digital VLSI systems [8] 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. This framework is then employed to derive new encoding schemes and to analytically determine reduction in transition activity. All past work in this area [1, 3, 10–12] 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 employing Gray codes as proposed in [12]. The Gray code does not, however, significantly reduce transitions for data busses because consecutive data values may not be sequential. Another encoding scheme for address busses (denoted as the T0 scheme) is presented in [1] 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 [3], a scheme, termed Bus-Invert coding in [10], is presented to reduce the number of transitions on a bus. The Bus-Invert coding scheme performs well when the data is uncorrelated; i.e., it assumes a decorrelating function precedes the Bus-Invert step. This scheme does not, however, perform well for correlated data and for larger bus widths.

One of the choices for the function f_2 in our framework is similar to the scheme in [2], 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.

In this paper, 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 14pF/bit in 1.2μ CMOS technology. Simulation results with an encoding scheme for instruction address busses indicate an average reduction in transition activity by a factor of 3 times and 1.5 times over the Gray and T0 [1] coding schemes respectively. We also present an adaptive scheme that monitors the statistics of the input and adapts its encoding scheme over time. Finally, we present analyses to predict the reduction in transition activity of any coding scheme, which can be placed in the

*This work was supported by DARPA contract DABT63-97-C-0025, NSF CAREER award MIP-9623737, and Joint Services Electronics Program under contract #00014-96-1-0129.

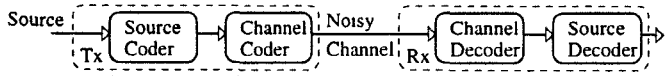

Figure 1: A Generic Communication System

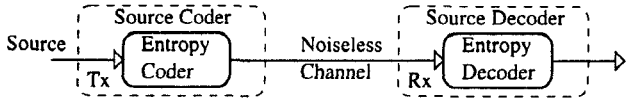

Figure 2: Commn. System for Noiseless Channel

proposed framework.

2 SOURCE-CODING FRAMEWORK

A generic communication system (see Figure 1) consists of a *source coder*, a *channel coder*, a *noisy channel*, a *channel decoder*, and a *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 [8], 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 lowered to reduce power. The noiseless channel assumption allows us to eliminate the channel coder resulting in the system shown in Figure 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. In practice, due to data dependencies and the constraint of having integer code-word lengths, it becomes necessary to take data blocks $\{x(n), x(n-1), \ldots, x(n-M+1)\}$ 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 Figure 3) involving a decorrelating function f_1 followed by a scalar entropy coder f_2 have been proposed. This is a sub-optimal structure if the function f_1 is unable to remove all the data dependencies. If the function $\mathbf{F}$ is a linear predictor then the structure shown in Figure 4 is obtained, where $\mathbf{F}$ represents the vector of the impulse response of a linear filter. The decorrelator f_1 removes all the linear dependencies in $x(n)$; and the entropy coder, f_2, then compresses the output of f_1 in a lossless manner.

2.1 The Source-Coding Framework

The proposed framework in Figure 5 is based upon the low-complexity source coder architecture (see Figure 4). The function f_1 decorrelates the input $x(n)$. Therefore, the

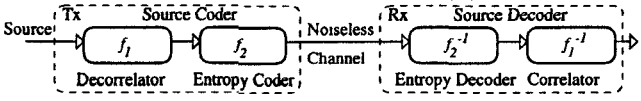

Figure 3: Practical Commn. System for Noiseless Ch.

prediction error, $e(n)$, is a function, f_1, of the current value of $x(n)$ and the prediction, $\widehat{x}(n)$. The function f_1 could be a linear or a non-linear function of its arguments. The prediction, $\widehat{x}(n)$, is a function, $\mathbf{F}(x(n-1), x(n-2), \ldots, x(n-M+1))$, of the past values of $x(n)$, 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 '1' to indicate a transition and a '0' 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 $f_2(f_1(x(n), \widehat{x}(n))) \bigoplus y(n-1)$, where $\bigoplus$ represents a bit-wise exclusive-or. 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, $f_3(f_2^{-1}(y(n-1) \bigoplus y(n)), \widehat{x}(n))$, 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, $\widehat{x}(n)$. 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-bit input, at most $4B$ delays and $2B$ exclusive-or gates are needed, in addition to the hardware required to implement the functions $\mathbf{F}$, 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 the decoder each as a whole and by also sharing logic between the encoder and the decoder in a bidirectional pad.

We now propose practical choices for $\mathbf{F}$, f_1, and f_2 and then evaluate the performance of encoding schemes employing different combinations of $\mathbf{F}$, f_1, and f_2 in section III. Since the aim of the encoding is to reduce power dissipation, for a choice of $\mathbf{F}$, 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, for a given reduction in signal transitions.

2.2 Alternatives for F, f_1, and f_2

In this paper, two alternatives for $\mathbf{F}$, referred to as *Identity* and *Increment*, are considered. The output of the *Identity* function is equal to its input whereas the output of the *Increment* function is equal to its input plus one. The *Identity* function requires no hardware to implement and is useful if the data source has significant correlation. 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 2). This function requires an incrementer each at the encoder and at the decoder.

Two alternatives for f_1, referred to as *Exclusive-Or* (*xor*) and *Difference-Based Mapping* (*dbm*), are considered in this paper. The *Exclusive-Or* function, *xor*, is given by a bit-wise exclusive-or of the current input and the prediction. If the input, $x(n)$, is B bits 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 **F** is *Identity*, then the only difference between the encoder and the decoder is that the encoder employs f_2, where as the decoder employs f_2^{-1}. Therefore, the hardware between the encoder and the decoder in a bidirectional pad can be shared by employing an additional control line into a multiplexer that 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. The *Difference-Based Mapping*, *dbm*, is described in Figure 6, 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 bits. If the input, $x(n)$, is B bits wide, then the *dbm* function will require 3 B-bit subtracters, B inverters, and B 4-to-1 multiplexers each at the encoder and at the decoder. The *dbm* output is 0 when the current and previous inputs are equal and it increases as the absolute difference between the current input and the prediction increases. This is also shown graphically in Figure 7. Both *xor* and *dbm* skew the original distribution for most data and hence enable f_2 to reduce the number of transitions even more.

Three possible choices for f_2 are considered in this paper, namely, *Invert* (*inv*), *Probability-Based Mapping* (*pbm*), and *Value-Based Mapping* (*vbm*). In the *inv* function, if the number of 1's in $e(n)$ exceeds half the number of bus lines, then the input is inverted and the inversion is signaled by setting an extra bit to '1', else the input is not inverted and the extra bit is set to '0'. The function *inv* has been employed in Bus-Invert coding [10]. The hardware required to implement the *inv* function is described in [3,10]. In the *pbm* function, the number of 1's in the input is reduced by assigning, as in [2], code-words with fewer 1's to the more frequently occuring code-words. We then map a '1' to a transition waveform and a '0' to a transitionless waveform employing an exclusive-or gate. The probabilities can be computed employing a representative data sequence. Thus, the most probable value of $e(n)$ is mapped to 0, the next B most probable values of $e(n)$ are mapped to 2^i (i=0 ... $B-1$), and the next $\binom{B}{2}$ most probable values are mapped to all values with exactly two 1's, and so on. The hardware required to implement *pbm* will depend on the input probability distribution. An 8 bit 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.

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 employ this feature in *vbm*, in which code-words with fewer 1's are assigned to smaller values and then map a '1' to a transition waveform and a '0' to a transitionless waveform employing an exclusive-or gate. The function *vbm* assumes that smaller values are more probable than larger values. 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 $vbm(a)$, then bit $k-1$ of $vbm(a)$ is 1 if $a - \sum_{j=0}^{i-1} \binom{k}{j} > \binom{k-1}{i}$. Once bit $k-1$ of $vbm(a)$ is determined, the lower order bits can be similarly determined in an iterative fashion. The number of 1's, i, in $vbm(a)$ can be determined by finding i such that, $\sum_{j=0}^{i} \binom{k}{j} \leq a+1 < \sum_{j=0}^{i+1} \binom{k}{j}$. The algorithm requires k clocks, where k is the number of bits in the output $y(n)$. The function *vbm* can be implemented either, algorithmically or by employing combinational logic. An 8 bit input typically requires 700 gates each to implement *vbm* and vbm^{-1}.

3 ENCODING SCHEMES

The proposed encoding schemes are summarized in Table 1, where we have 7 encoding schemes employing different combinations of **F**, f_1, and f_2. The *xor-pbm* scheme reduces the number of transitions by assigning fewer transitions to the more frequently occuring set of transitions in the original signal. The closest approach in literature to *xor-pbm* is [2] 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 occuring at the capacitive nodes of the circuit. The *xor-pbm* scheme differs from [2] in two respects. It reduces the power dissipation by reducing the number of transitions by assigning fewer transitions to the more frequently occuring 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 Figure 5 can be employed to derive and improve existing coding schemes. For example, the Gray coding scheme can be derived by letting $f_1(x(n), \hat{x}(n)) = x(n)$, f_2 be the Gray coding scheme, and removing the exclusive-or at the output of the encoder. The Bus-Invert [10] coding scheme can be obtained by letting f_1 be *xor* and f_2 be *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 [2] can be derived by letting $f_1(x(n), \hat{x}(n)) = x(n)$, f_2 be *pbm*, and removing the exclusive-or at the output of the encoder. An improved version of the T0 scheme in [1] can be derived from our framework by employing the *Increment* function for the predictor **F** (with overflow being ignored), *xor* for f_1, and *Identity* function for f_2. This improved scheme, called *inc-xor*, is shown in Figure 8. Unlike the T0 scheme, the *inc-xor* scheme does not require an extra bit, and as simulation results will show, 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 [11]. For instance, if an extra bit is added to the bus, f_2 can employ the extra bit to further reduce transitions. For instance, in the function *pbm*, the most probable value is assigned 0. Assuming the bus is $B+1$ bits wide, the next $B+1$ most probable values are assigned 2^i ($i = 0 \ldots B$) and so on. The maximum number of transitions on the bus is less than or equal to $\frac{B}{2}$.

4 ANALYSES AND SIMULATIONS

We now present analytical methods to estimate the transition activity, T_y, on the bus in Figure 5. We treat the case when the input, $x(n)$, is an address stream separately from the case when it is a data stream because the characteristics of address and data streams are different. We employed the probabilistic model in [1] to calculate the transition activity of an address stream after encoding. This model employs two parameters, q, the probability of having two consecutive addresses on the bus in two successive clock cycles, and K, the number of transitions on average when two non-consecutive addresses are issued on the bus. In general, q will depend on the source, while K will depend both on the source and the encoding scheme employed. With this model, the average number of transitions, $T_{Unsigned}$, produced by the Unsigned code is given by $(1-q)K_{Unsigned}+2q$, where $K_{Unsigned}$ is the number of transitions (on average) occuring when addresses are non-consecutive. It can be shown that for the Unsigned code and large bus widths, there are two transitions (on average) taking place when the addresses are consecutive. Similarly, the average number of transitions, produced by the Gray, *inc-xor*, and T0 codes is given by, $(1-q)K_{Gray}+q$, $(1-q)K_{inc-xor}$, and $(1-q)K_{T0}+2q(1-q)$ respectively. In Figure 9, we plot $T_{Unsigned}$, T_{Gray}, T_{T0}, and $T_{inc-xor}$ as a function of the probability q assuming $K_{Unsigned}=K_{Gray}=K_{inc-xor}=K_{T0}=4$. The *inc-xor* scheme always has the least transition activity, Unsigned always has the highest transition activity, and T0 is better than Gray only for $q>\frac{1}{2}$.

In Table 2 we report the average word-level transition activity when Unsigned, Gray, T0, and *inc-xor* encoding schemes are employed to encode the addresses generated when different benchmark programs are executed on the SGI Power Challenge with a MIPS R10000 processor. As in [1], we consider three cases, transitions on the instruction address bus (I), transitions on the data address bus (D), and transitions on a multiplexed address bus (M). We encode only the most significant 30 bits since the least significant 2 bits are usually zero because successive instruction and data elements on the MIPS processor differ by 4. The greatest reduction in transition activity employing *inc-xor* is observed for I-bus streams because the probability of addresses being sequential is highest for such streams. The *inc-xor* scheme 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, *inc-xor* provides the greatest reduction in transition activity for all the I-bus streams and 6 of the 8 M-bus streams. The Gray code provides the greatest reduction in transition activity for all the D-bus streams and 2 of the 8 M-bus streams. In the next section, we will present an analytical model to explain these results. The analytical estimate in Figure 9 matches well with measured data in Table 2 in which 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, Gray code is better than *inc-xor* because K_{Gray} is typically less than $K_{inc-xor}$. The values of $K_{Unsigned}$, K_{Gray}, $K_{inc-xor}$, K_{T0}, and q for benchmark programs are shown in Table 2.

The original, uncoded power dissipation at the bus is given by $T_x C_L V_{dd}^2 f$ where T_x is the word-level transition activity of the input and C_L is the bus capacitance per bit. For a given encoding and decoding scheme, we calculated the power dissipation, $P_{D,coded}$ as the sum of the power dissipation at the encoder, the bus, and the decoder. If T_y is the reduced word-level transition activity at the bus after encoding then, $P_{D,coded}=P_{D,enc}+T_y C_L V_{dd}^2 f+P_{D,dec}$. Table 7 shows the area-delay-power information for the encoder and decoder for the Gray, T0, and *inc-xor* encoding schemes employing 1.2μ CMOS technology, 3.3V supply voltage, and 10MHz clock frequency. The power dissipation was estimated employing 93 samples of an I address stream. All the encoders and decoders were optimized for minimum area. 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.

We estimated the transition activity, T_y, when the input, $x(n)$, is a data stream, by employing the model in [4] where a word-level signal is broken up into uncorrelated data bits, correlated data bits, and sign bits. The uncorrelated data bits extend from the least significant bit up to a certain break-point BP_0, which can be estimated from high-level statistics [4,6]. The lower bound on the number of output transitions, $T_{y,min}$, at the bus is given by, $\frac{\sum_{i=0}^{j-1} i\binom{B}{i}+j(2^{BP_0}-\sum_{i=0}^{j-1}\binom{B}{i})}{2^{BP_0}}$, where j is such that, $\sum_{i=0}^{j-1}\binom{B}{i} < 2^{BP_0} \leq \sum_{i=0}^{j}\binom{B}{i}$. We can also derive the following, empirical, estimate [7] for $T_{y,xor-pbm}$, $\frac{\sum_{i=0}^{j-1} i\binom{B}{i}+j(2^{2T_x}-\sum_{i=0}^{j-1}\binom{B}{i})}{2^{2T_x}}$, where j is such that, $\sum_{i=0}^{j-1}\binom{B}{i} < 2^{2T_x} \leq \sum_{i=0}^{j}\binom{B}{i}$. The derivations are omitted due to lack of space. The average error when the measured T_x is employed to estimate $T_{y,xor-pbm}$ is 6.4% and, excluding ASCII (PS) data, 14% when $T_{y,xor-pbm}$ is estimated employing high-level statistics [6]. The errors are higher for ASCII (PS) data because the model of [4] does not hold well for this type of data.

We now present results of simulations to measure the effectiveness of schemes to encode data streams. The reduction in transition activity when the first 6 schemes in Table 1 are applied to the data sets in Table 3 is shown in Table 4. 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). This occurs since $x(n)$ and $\widehat{x}(n)$ are uncorrelated, and hence all values of $f_1(x(n), \widehat{x}(n))$ 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 more than *xor-pbm* because *dbm* skews the input probability distribution more than *xor*.

The reduction in transition activity with 1 bit of spatial redundancy is shown in Table 5. For audio and video data, *xor-pbm* performs better than Bus-Invert or Bus-Invert

with compression (gzip) with the same redundancy. 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 (Postscript) 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 correlation of the data to be encoded increases.

We employed SIS to generate the net-lists for the encoder and the decoder for *xor-pbm* and Bus-Invert. We then employed PSpice to estimate the power dissipation in the encoder, $P_{D,enc}$, and the decoder, $P_{D,dec}$. We employed 1.2μ CMOS technology with 3.3V supply voltage and 20MHz frequency. The area-delay-power information is presented in Table 6. In Figure 10 we plot the power dissipation for different bus capacitances, C_L. The power dissipation was estimated employing 30 samples of V2 input. From Figure 10, we see that given a highly correlated input, for small bus capacitances (<10pF/bit), it is best to not encode the data at all. For capacitances above 10pF/bit, Bus-Invert provides a small reduction in power dissipation and for capacitances above 14pF/bit, *xor-pbm* has the lowest total power dissipation. For a bus capacitance of 50pF/bit, *xor-pbm* provides a power savings of 8.82 mW which is eight times the power savings of 1.03 mW provided by Bus-Invert. The slope of the graph of power dissipation versus bus capacitance is determined only by T_y, the reduced word-level transition activity and the y-intercept is determined by the total power dissipation in the encoder ($P_{D,enc}$) and decoder ($P_{D,dec}$). The slope of the curve can be reduced further by employing a different coding scheme which could result in a greater reduction in transitions. In our experiments, we synthesized for minimum-delay. Therefore, the power dissipation of the encoder and the decoder can be further reduced by employing low power synthesis techniques. This would reduce the y-intercept in Figure 10 and in turn reduce the cross-over point at which *xor-pbm* would have the lowest power dissipation.

We have so far examined encoding schemes which do not vary with time. Since the input data statistics can vary over time, we simulated one adaptive scheme whereby the encoder and the decoder both keep track of the probabilities of the output of $f_1(x(n), \hat{x}(n))$. The function f_2 is implemented employing a RAM since its contents will change with time. After processing each sample, both the encoder and the 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 4 when f_1 is *xor*. The RAMs were initialized to implement *vbm* as it is an input independent mapping. For video, random, and ASCII (PS) data which are 8 bits 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 bits 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 bit of spatial redundancy, as shown

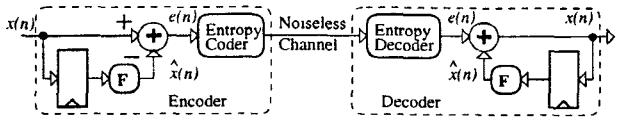

Figure 4: Linear Prediction Configuration

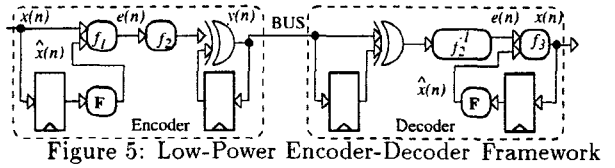

Figure 5: Low-Power Encoder-Decoder Framework

in Table 5. One disadvantage of the adaptive scheme is the extra hardware required, and hence this scheme is to be employed when an input independent mapping is required and *vbm* (also an input independent mapping) does not result in significant reduction in transitions.

References

[1] L. Benini et al, "Asymptotic Zero-Transition Activity Encoding for Address Busses in Low-Power Microprocessor-Based Systems," *GLVLSI Symp.*, pp. 77–82, March 1997.

[2] D. W. Faulkner, "PCM Signal Coding," *U.S. Patent no. 5,062,152*, October 1991.

[3] R. J. Fletcher, "Integrated Circuit Having Outputs Configured for Reduced State Changes," *U.S. Patent no. 4,667,337*, May 1987.

[4] P. E. Landman and J. M. Rabaey, "Architectural Power Analysis: The Dual Bit Type Method," *IEEE Trans. VLSI*, vol. 3, no. 2, pp. 173–187, June 1995.

[5] E. Musoll, T. Lang, J. Cortadella, "Exploiting the locality of memory references to reduce the address bus energy," *Int. Symp. Low Power Elect. Des. (ISLPED)*, pp. 202–207, 1997.

[6] S. Ramprasad, N. R. Shanbhag, and I. N. Hajj, "Analytical Estimation of Transition Activity from Word-Level Signal Statistics," *34th Design Automation Conf.*, 1997.

[7] S. Ramprasad, N. R. Shanbhag, and I. N. Hajj, "Coding Schemes to Reduce Transition Activity: An Information-Theoretic Framework and Applications," *submitted to IEEE TVLSI*, May 1997.

[8] N. R. Shanbhag, "Lower bounds on Power-Dissipation for DSP algorithms," *ISLPED*, pp. 43–48, August 1996.

[9] M. R. Stan and W. P. Burleson, "Limited-Weight Codes for Low-Power I/O," *Int. Workshop Low Power Design*, pp. 209–214, Napa CA, April 1994.

[10] M. R. Stan and W. P. Burleson, "Bus-Invert Coding for Low-Power I/O," *IEEE TVLSI*, pp. 49–58, March 1995.

[11] M. R. Stan and W. P. Burleson, "Two-dimensional Codes for Low-Power," *ISLPED*, pp. 335–340, August 1996.

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

[13] J. Tabor, "Noise reduction using low-weight and constant weight coding techniques," *M. S. Thesis*, MIT, May 1990.

Table 1: 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

```
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),
```

Figure 6: *dbm* function

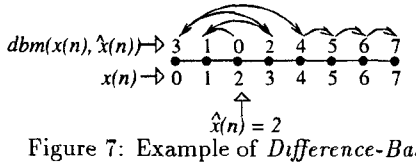

Figure 7: Example of *Difference-Based Mapping* (*dbm*)

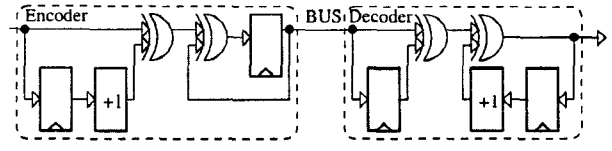

Figure 8: Encoder and Decoder for *inc-xor*

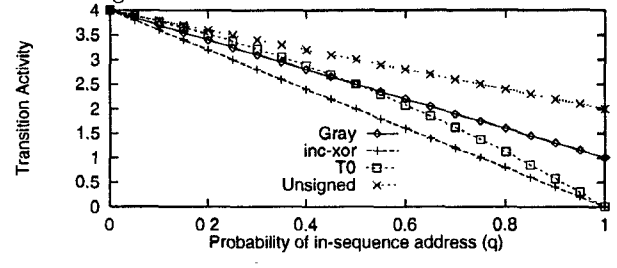

Figure 9: Analytical Estimate of Transition Activity

Table 2: Average Word-Level Transition Activity for Real Addresses

Addr. Type	Program	Stream Length	Transition Activity Uns.	Gray	T0	*inc-xor*	*q*	*K* Uns.	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 dec.	11481528	2.15	1.16	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	6.66	6.89
	MPEG dec.	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	6.70	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 dec.	15000000	6.68	5.45	5.91	5.64	0.50	11.30	9.66	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	8.38	8.11	7.49	0.40	12.33	8.24	12.37	12.38

Table 3: 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
R1	0.10MB of white, uniformly distributed data
PS	0.10MB Postscript file

Table 4: Percentage Reduction in Transition Activity

Data	*xor-pbm*	*xor-vbm*	*dbm-pbm*	*dbm-vbm*	*xor-pbm* with *pbm* opt. for	*xor-pbm* with *pbm* Redn.	Adaptive scheme
A3	32	24	36	29	A3	32	24
A4	38	29	41	30	A3	37	29
A7	41	32	45	25	A3	39	33
CO	26	4	28	28	A3	20	4
V1	39	34	44	44	V2	38	39
V2	33	26	39	39	V2	33	33
V3	33	25	40	39	V2	33	33
R1	1	0	1	0	V2	0	0
PS	43	22	38	25	PS	43	42

Table 5: % Redn. in Trans. Act. With 1 Extra Bit

Data	*xor-pbm*	*xor-vbm*	*dbm-pbm*	*dbm-vbm*	Bus-Invert	*dbm-inv*	Bus-Invert & gzip	*xor-pbm* with *pbm* opt. for	*xor-pbm* with *pbm* Redn.	Adaptive scheme
A3	35	32	39	38	8	12	1	A3	35	32
A4	40	36	43	43	6	12	-5	A3	39	36
A7	42	35	47	46	8	14	-10	A3	41	35
CO	31	21	32	31	15	16	16	A3	27	21
V1	42	38	46	46	6	31	-3	V2	40	42
V2	37	33	42	42	9	22	0	V2	37	37
V3	38	34	43	43	11	26	-6	V2	38	38
R1	19	18	19	18	18	18	18	V2	18	18
PS	46	33	42	35	10	14	62	PS	46	46

Table 6: Area-Delay-Power for Bus-Inv., *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 Transistors in Encoder & Decoder	406	6482
Minimum Bus Capacitance for Reduction in Total P.D. (pF/bit)	10.67	13.86

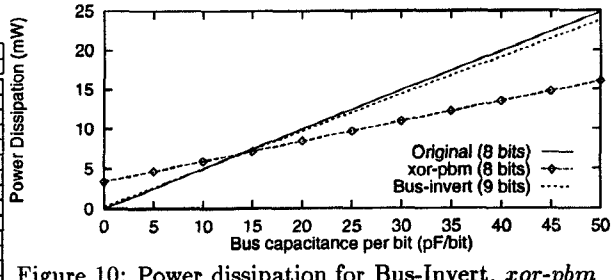

Figure 10: Power dissipation for Bus-Invert, *xor-pbm*

Table 7: Area-Delay-Power for Gray, T0, *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