Bus Encoding for Total Power Reduction Using a Leakage-Aware Buffer Configuration

Rajeev R. Rao, Harmander S. Deogun, David Blaauw, *Member, IEEE*, and Dennis Sylvester, *Senior Member, IEEE*

***Abstract*—Power consumption, particularly runtime leakage, in long on-chip buses has grown to be an unacceptable portion of the total power budget due to heavy buffer insertion used to combat *RC* delays. In this paper, we propose a new bus encoding algorithm and circuit scheme for on-chip buses that eliminates capacitive crosstalk while simultaneously reducing total power. We utilize a buffer design approach with a selective use of high-threshold voltage transistors and couple this buffer design with a novel bus encoding scheme. The proposed encoding scheme significantly reduces total power by 26% and runtime leakage power by 42% while also eliminating capacitive crosstalk. In addition, the proposed encoding is specifically optimized to reduce the complexity of the encoding logic, allowing for a significant reduction in overhead which has not been considered in previous bus encoding work.**

***Index Terms*—encoding, Buffer circuits, interconnect, low power.**

I. Introduction

CONTINUED scaling of process technologies has led to smaller device features, faster clock speeds, and rapidly shrinking interconnects. In order to maintain the performance gains associated with each technology generation, the threshold voltage ($V_{\rm th}$) of the MOSFET device is aggressively scaled as well. However, lowering $V_{\rm th}$ has resulted in an increase in the subthreshold current of the device at 3–5$\times$ per generation [1]. It is projected that, in the 90 nm node subthreshold leakage power will be as much as 40% of the total power for high-performance processors [2]. Buffers used to manage delay and signal integrity problems on long on-chip buses constitute a major component of this leakage power. In general, inverters or buffers contribute roughly 50% of the total device width on chips [3] and, due to the lack of stack effect, constitute a major fraction of the total leakage power. Further, it has been estimated [4] that, given the current trajectory of the design paradigm, 70% of the total cell count at the 32 nm node will be due to buffers and repeaters. Consequently, it is critical to develop approaches that aim to limit this component of total power.

Recently, a number of strategies have been proposed that utilize bus encoding to eliminate undesirable effects that would otherwise occur during transmission of the unencoded bits. Simple encoding schemes such as bus-invert coding [5] aim to reduce the number of transitions in bus lines but do not focus directly on crosstalk elimination or leakage reduction. The approaches in [6]–[9] seek to minimize the dynamic power and delay in buses through various encoding schemes. The work in [6] is well suited for delay reduction by elimination of crosstalk (through a "self-shield encoding") but it does not address power reduction. The authors in [7] extend the work in [6] and propose a method that eliminates crosstalk and reduces dynamic power. In [8], the authors shuffle the order of the bus lines to minimize opposite-phase transitions on adjacent bus lines to reduce power due to crosstalk. The results in [9] show a reduction in both static and dynamic power using their technique of low-voltage BiCMOS and termination networks.

While the above mentioned works describe methods of eliminating crosstalk and/or reducing dynamic power, most do not tackle the rising leakage power levels in such buses. These approaches also do not attempt to minimize the complexity of the encoder and decoder (codec) hardware and therefore may have high power and delay overheads. In this paper, we propose a new bus encoding method that minimizes total power while simultaneously eliminating crosstalk. Our approach builds on [6], [7] by using bus encoding for delay improvement through crosstalk elimination. The novelty in our bus encoding scheme is that it is leakage-aware and coupled with a dual-$V_{\rm th}$ buffer design. We demonstrate that by combining a leakage-aware encoding with a dual-$V_{\rm th}$ bus driver design, we can reduce average runtime leakage power by 42% and average total power by 26% while concurrently eliminating crosstalk. Our approach also minimizes the codec logic complexity resulting in significantly reduced power and delay overhead.

The remainder of this paper is organized as follows. Section II gives an overview of our approach for leakage-aware bus encoding. Section III details the algorithm that is used to derive the low leakage and crosstalk eliminating encoding. Section IV describes the experimental test setup and presents our results. Section V concludes the paper.

II. Overview of Encoding

In general, low-threshold voltage (LVT) buffers are used in on-chip memory buses to achieve high performance requirements. However, LVT devices are unsuitable from a power perspective due to their very high leakage power. A simple way to reduce subthreshold leakage current is by raising $V_{\rm th}$, which is accomplished by replacing the LVT buffers with high-threshold voltage (HVT) buffers. Using a HVT instead of a LVT device typically provides a leakage savings of 10$\times$ for the same size device. Note that there are other known techniques to reduce leakage during standby mode but in this paper we focus on runtime leakage reduction, which is a more difficult and pressing

Manuscript received April 1, 2005; revised August 1, 2005. This work was supported in part by the MARCO/DARPA GSRC, the National Science Foundation, and by equipment donations from Intel and Sun Microsystems.

The authors are with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109 USA (e-mail: rrrao@eecs.umich.edu; hdeogun@eecs.umich.edu; blaauw@eecs.umich.edu; dennis@eecs.umich.edu).

Digital Object Identifier 10.1109/TVLSI.2005.862718

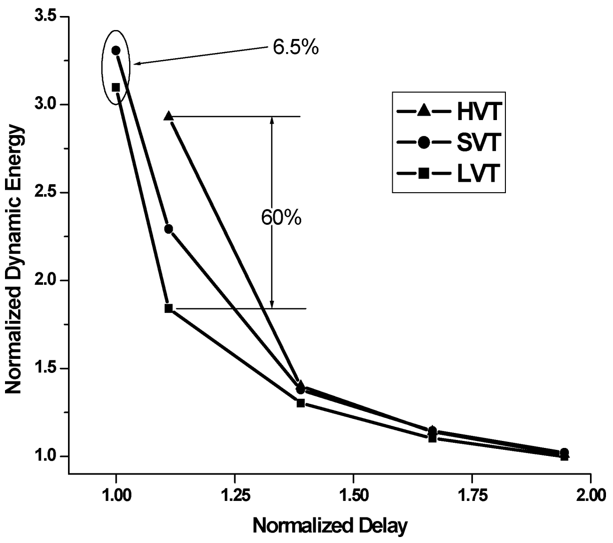


Fig. 1. Normalized worst case delay and dynamic energy for different bus line types.

problem. Currently dual-V_{th} is the only practical approach to achieving substantial runtime leakage reduction [10].

However, using HVT buffers leads to a large degradation in performance. Fig. 1 was generated using HSPICE simulations of a single bus line consisting of buffer configurations using different types of devices. The delay point specified by 1.00 on this plot corresponds to the minimum possible delay of a bus line using LVT buffers. Different delay targets were set and the buffers were sized optimally to meet these new targets. It can be seen that a bus using HVT buffers is not able to meet a stringent delay constraint even with device sizing as a variable. The HVT buffers are only able to meet a delay target that is about 13% slower than the LVT buffers while incurring a substantial penalty (60%) in dynamic energy due to the aggressive sizing requirements. Thus, HVT buffers can greatly reduce leakage current but only by incurring significant penalties in delay and dynamic energy. In many high-performance applications, a penalty in delay or dynamic energy cannot be tolerated—this leads to a difficult trade-off between meeting delay while trying to maintain power at manageable levels. Furthermore, it is known that delay becomes more sensitive to V_{th} in sub-1V technologies and that the corresponding delay penalty associated with using high-V_{th} devices in these processes will grow significantly [11].

To resolve this problem and address leakage issues, we propose the use of staggered threshold voltage (SVT) buffers. These buffers are constructed by combining LVT and HVT transistors in a staggered fashion, as shown in Fig. 2. In contrast to the method in [12] where the authors skew the sizes of the NMOS/PMOS transistors along a buffer line for delay improvement, we modify the threshold voltages of these buffers with the objective of static power minimization. We note that the idea of dual-V_{th} inverters has been presented previously [13]. In our work, we propose a novel construction method for bus lines using these SVT buffers.

SVT devices enable the design of high-performance buses that have a much reduced penalty in dynamic energy. In Fig. 1, we plot the energy-delay characteristics for the SVT buffers. Although the SVT devices cannot exactly achieve the minimum possible delay value, we observed that with sufficient sizing they can operate with a very small overhead of about 3–4 ps (about 2%). Further, the dynamic energy penalty has been reduced by nearly 10× to only 6.5% at the fastest achievable design point of SVT. This dynamic penalty is due to the slightly larger device sizes that must be used to ensure that the delay numbers of SVT and LVT are nearly identical.

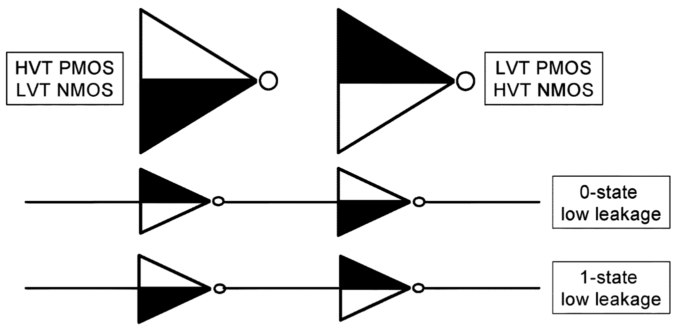


Fig. 2. SVT buffers.

In SVT buffers, if the active devices are LVT and the off-state devices are HVT then we achieve the optimal tradeoff between delay and power since the LVT devices ensure shorter propagation delays while the HVT devices result in lower leakage power. Thus, if the input values to the bus line are known, then the SVT buffers can be designed to achieve the optimal power-delay tradeoff. It has been pointed out recently that on-chip caches store primarily 0's which indicates that data buses connected to such caches may have a high probability of carrying 0's rather than 1's [14]. This type of application would benefit greatly from SVT buffers. However, assuming this sort of imbalance in input probabilities does not exist, we turn our attention to the use of bus encoding to enforce the input states that will result in the lowest leakage.

To ensure that the HVT device in a given inverter is usually off (to reduce subthreshold leakage), an encoding scheme is developed to skew the data bits of the bus to either the 0 or 1 state. The stagger configuration for the SVT bus is then chosen appropriately (see Fig. 2) such that each bus line can be designated as a 0-state or 1-state low leakage bus line. In this way, the bus line can spend, on average, the majority of the time in the designated low leakage state. This is the leakage-aware portion of the encoding.

The dynamic power expended by a set of bus lines can be reduced drastically by eliminating crosstalk effects between them. When a pair of adjacent wires transition in opposite directions, it results in worst case conditions for both delay and power [6]. The magnitude of coupling capacitance between adjacent wires is normally greater than the ground capacitance due to the increased aspect ratio of modern interconnects [16], therefore, the delay can be nearly twice as large as with just one wire switching next to a quiet wire. The crosstalk-aware portion of the encoding focuses on developing a coding mechanism that skews the states on the bus such that the possibility of the worst case transition between a pair of adjacent wires is eliminated. The self-shield encoding method presented in [6] uses encoder/decoder logic and some additional wires to implement such a mechanism. Since these codec stages are unavoidable, we require them to be as small a percentage of the total bus delay as possible, in order to minimize the overhead. The reduction in total power, along with the elimination of crosstalk, far outweighs this incremental delay penalty. Moreover, as devices become faster and areas shrink with each generation, the overhead on the bus line grows smaller. Thus, the tradeoff between extra logic and reduction in power and elimination of crosstalk is reasonable.

We now utilize the SVT buffer technique in our encoding algorithm that uses an enhanced self-shield mechanism to eliminate crosstalk while simultaneously minimizing the total power. Additionally, we minimize the delay overhead by optimizing the codec logic.

III. Proposed Encoding Algorithm

In the enhanced self-shield encoding scheme, the input bits are processed by the encoder architecture in order to produce a set of codewords. The mapping between the input bits and codewords is called a codebook. The Hamming distance (HD) between a pair of codewords is given by the number of 1's in the bitwise XOR between them. The HD term describes the number of bit differences between a pair of codewords.

As motivated in the previous section, we require an encoding scheme that has the following three features.

F1) Eliminate crosstalk between adjacent bus lines.
F2) Minimize leakage by skewing the probability of the bits.
F3) Minimize overhead due to the encoding and decoding logic.

We first address F3. When encoding b bits of input as e-bit codewords, it is essential to pick the smallest possible values for b and e to minimize the codec logic. We pick $(b, e) = (3, 4)$ since this encoding leads to fairly simple encoder/decoder circuits. We also note that there does not exist a codeword mapping for $(b, e) = (4, 5)$ because it is impossible to extract 16 codewords out of 32 possible choices such that all pairs of codewords do not contain a pair of adjacent bits with opposite transitions. For larger values of (b, e), the complexity of the codec logic increases considerably.

For a 32-bit bus, we split the full bus into sets of 3 bits each and then encode each set individually. Therefore, a 32-bit bus is encoded on a 43 line bus (10 groups of 3-bit inputs encoded using 4 bits and the remaining 2 bits encoded using three more bits). A shield (dedicated ground or V_{dd} wire) separates each set, increasing the bus line count to 53. Although wire spacing has been proven to be a better alternative for capacitively coupled interconnects [15], we note that shield insertion is a simple yet effective method used in high-speed buses to suppress inductive effects. Our use of one shield for every three or four wires is a typical method [17], [18]. Since insertion of shields is a common practice [6], [15], [16], we do not consider this an additional overhead in our specific design. A typical 32-bit bus would use 40 lines with shield insertion. Our encoded bus uses 53 lines, for a total overhead of 13 lines or a 33% increase. Thus, for each set of three input bits, we use 4-bit codewords and the size of the codebook being $2^3 = 8$.

To address F2, it is essential to pick an encoding method where the leakage incurred by the eight codewords is minimized. Since the SVT technique skews the buffers on each bus line such that there exists an ideal leakage state for each line, there exists only one codeword that corresponds to the minimum leakage state simultaneously for a set of four bus lines. We call such a 4-bit input combination the ideal codeword. From our codebook (of size eight), we need seven additional codewords that are as close as possible to the ideal leakage state. To accomplish this, we choose codewords that have the least HD to the ideal leakage state. For any 4-bit ideal codeword, there are $\binom{4}{1} = 4$ codewords within $\mathrm{HD} = 1$ and $\binom{4}{2} = 6$ codewords within $\mathrm{HD} = 2$.

Conceptually, any of the 16 4-bit codewords can be chosen as the ideal codeword since a bus line consisting of SVT buffers can be tailored toward having either 0 or 1 as its low leakage state. However, for our encoding scheme the selection of the ideal codeword is dictated by F1. We first prove the following lemma.

Lemma: The ideal codeword in a codebook that satisfies F1–F3 does not contain two adjacent bits that are the same.

Proof: Let $\mathrm{b_1 b_2 b_3 b_4}$ be the ideal 4-bit codeword. Suppose $\mathrm{b_2 = b_1}$. Since we need to pick all codes that are within $\mathrm{HD} = 1$ (to satisfy F2), $\mathrm{\overline{b}_1 b_1 b_3 b_4}$, $\mathrm{b_1 \overline{b}_1 b_3 b_4}$ need to both be part of the codebook. However, a transition between these two states violates the self-shield coding condition since two adjacent bits are switching in opposite directions. Hence, it is impossible to have an ideal codeword with two adjacent bits that are the same. $\square$

Using this lemma, we can identify the ideal 4-bit codewords as 0101 and 1010. Since these codewords are analogous, we only consider 0101. Among the $4 + 6$ $\mathrm{HD} = 1, 2$ codewords we eliminate three $\mathrm{HD} = 2$ codewords (0110, 0011, and 1001) since they violate the self-shield coding requirement. Thus, our codebook of size eight is given by one $\mathrm{HD} = 0$ codeword (0101), four $\mathrm{HD} = 1$ codewords (0100, 0111, 0001, and 1101), and three $\mathrm{HD} = 2$ codewords (0000, 1100, and 1111). With the codebook determined, we now assign the eight possible input states to the codewords; this assignment determines the overall performance of the encoding scheme and complexity of the codec logic.

For a given mapping from the input data bits to the codewords, we first define a power function $P = D + L$. Here, D represents the dynamic power and L represents the leakage power expended by the encoded data bits. The dynamic power is dependent on the transition characteristics of the input data bits while the leakage power is dependent on their state characteristics. Based on the simulation profile of a memory bus we generate both the state and transition probabilities for sets of 3-bit data bits. Our objective is to generate a mapping from the 3-bit input to the 4-bit codewords that will minimize this power function P in addition to satisfying F1–F3 and minimizing the logic complexity.

It is evident that, to minimize P, we need to assign the lowest probability values in both the state and transition probability tables to the codewords that consume the greatest amount of power. Since the codewords are known *a priori* and the codebook size of 8 is fairly small, we search the entire sample space of $8!$ mappings of symbols to codewords to determine the minimum value ($P_{\min}$) of the power function. However, the mapping corresponding to $P_{\min}$ may potentially require complicated codec logic that would result in unacceptably large overhead. To avoid this, we set a tolerance limit T on $P_{\min}$ and examine the logic complexity of all mappings that have $P \leq P_{\min}(1+T)$. Among these mappings, we choose the one with the smallest overhead (the overhead is quantified using Espresso [19] to determine the total number of gates required to construct the encoder and decoder for each mapping). Thus, we have obtained a mapping that consumes a sufficiently small amount of power while minimizing the logic overhead. As we will show in Section IV, a tolerance limit of approximately 5% typically captures the optimal/minimal number of gates. Note that $T = 0\%$ corresponds to selecting the power-optimal mapping regardless of encode/decode overhead, making this a special case of our approach. We give a summary of our proposed algorithm, called *BuffPower*, that has as inputs M, the memory trace of a program, and T, which is the tolerance limit set on $P_{\min}$.

A. Summary of the Proposed Algorithm

```
Algorithm BuffPower ($M, T$)
1. Construct state ($S$) and transition ($R$) probability tables from
   the memory trace $M$
2. $E = \{$ set of all 8! mappings from 3-bit I/P to 4- bit
      codes$\}$
3. for each (mapping $\epsilon$ $E$)
        Calculate $P$(mapping)
4. Sort mappings according to the $P$s
5. Set $E_{\text{trunc}} = \{$set of all mappings with $P \leq P_{\min}(1+T)\}$
6. for each (mapping $\epsilon$ $E_{\text{trunc}}$)
        Calculate delay(mapping)
7. Sort mappings according to the delays
8. return (mapping of min delay)
```

IV. Power and Performance Analysis

The encoding algorithm described previously was implemented using industrial 0.13 μm device models. The SVT buffers were characterized using SPICE simulations at a temperature of 105 °C. A bus line length of 8 mm was constructed with an inverting repeater inserted every 800 μm. There were ten inverters such that the total bus line remained non-inverted. We obtained a large number of traces of a 64-bit memory bus for nine different benchmarks (from the Spec CINT 2000 suite [20]) running on an Alpha architecture-based microprocessor. For each benchmark, we first constructed the state and transition probability tables. The static and dynamic power values were then scaled by the numbers in these tables such that the resultant normalized number was representative of the power consumed in a particular state or a transition between two states.

The various schemes using combinations of encoding and SVT/LVT buffers can be classified into four cases, as shown in Table I. We observe that we require 4 bits per block to use the self-shield encoding to achieve crosstalk elimination. Leakage reduction can be performed only by using SVT buffers. Here, Scheme1 corresponds to the typical method (baseline) where no encoding is used and LVT buffers are used uniformly for all buffers. Scheme2 corresponds to the method proposed in this paper where both crosstalk elimination and leakage reduction are possible. Scheme4 which adds an extra bus line but uses LVT is clearly suboptimal since crosstalk elimination by itself is insufficient to attain significant power gains. In Scheme3, the 3 bus bits are encoded using just 3 bits but the SVT buffers are used for leakage reduction.

TABLE I
Classification of Different Schemes for Encoding + Leakage Control

Scheme	Number of Bits per Block	Crosstalk Elimination? (Self-shield Encoding)	Buffer Type	Leakage Control?
1	3	No	LVT	No
2	4	Yes	SVT	Yes
3	3	No	SVT	Yes
4	4	Yes	LVT	No

We present a comparison in the normalized power values P_{static}, P_{dynamic}, and P_{total} for Schemes 1–3 in Table II for four benchmarks—the (generic) programs, gcc, gzip, mcf, and an artificially constructed program TEST_1 program that has high switching activity. First, we observe that, due to the addition of an extra bus line, the static power for Scheme2 is generally higher than Scheme3. However, the usage of the self-shield coding algorithm in Scheme2 helps to reduce the dynamic power significantly. ΔS_{23} represents the difference (in %) in dynamic power between Scheme2 and Scheme3. From this data, we clearly see that crosstalk elimination has decreased the dynamic power by about 30%–40%.

ΔS_{12} and ΔS_{13} represent the improvement (in %) in the total power value between the baseline (Scheme1) and Scheme2 and Scheme3. (Note that the ΔS_{12} will also be highlighted in Fig. 6 for the entire range of benchmarks). We clearly see that Scheme2 provides significantly better improvements compared to Scheme3. We observe an interesting phenomenon for the TEST_1 benchmark. With high switching activity, it can be expected that the use of larger sized SVT buffers will increase the dynamic power. However, in Scheme3 without crosstalk protection, the dynamic power increases to such a large extent that the total power of Scheme1 is lesser than the total power of Scheme3 (-18%). The usage of self-shield coding limits the dynamic power overhead considerably so that even in the total power number, Scheme2 is able to achieve a small positive gain (2%) compared to Scheme 1. In the reminder of this paper, we analyze the power improvements that can be achieved using Scheme2, which is the method proposed in this work.

The *BuffPower* algorithm described in Section IV requires a tolerance value T. The number of gates in the encoding and decoding logic versus T and the total power versus T are shown in Fig. 3 and the resulting intersection of the curves was chosen as the optimal tolerance value. In this plot, it is seen that the intersection occurs when the tolerance T is in the 5% range. This value of T yields a substantially reduced number of gates in the encoding and decoding logic while also limiting the total power consumed, thus providing an ideal tradeoff between codec overhead and power consumption. Therefore, we use $T = 5\%$ in

TABLE II
COMPARISON OF THE THREE SCHEMES (S1, S2, AND S3) FOR DIFFERENT BENCHMARKS

Prgm	P_{static}			$P_{dynamic}$			P_{total}			ΔS_{23}	ΔS_{12}	ΔS_{13}
	S1	S2	S3	S1	S2	S3	S1	S2	S3			
gcc	3.25	1.94	1.85	2.61	2.62	3.64	5.86	4.56	5.49	-38.9	22.2	6.3
gzip	3.27	1.79	1.71	2.15	1.96	2.74	5.41	3.75	4.44	-39.8	30.7	17.9
mcf	3.22	1.76	1.62	1.38	1.44	1.95	4.63	3.21	3.56	-35.4	30.9	23.1
test_1	3.25	2.23	2.21	4.66	5.51	7.13	7.91	7.74	9.34	-29.4	2.1	-18.1

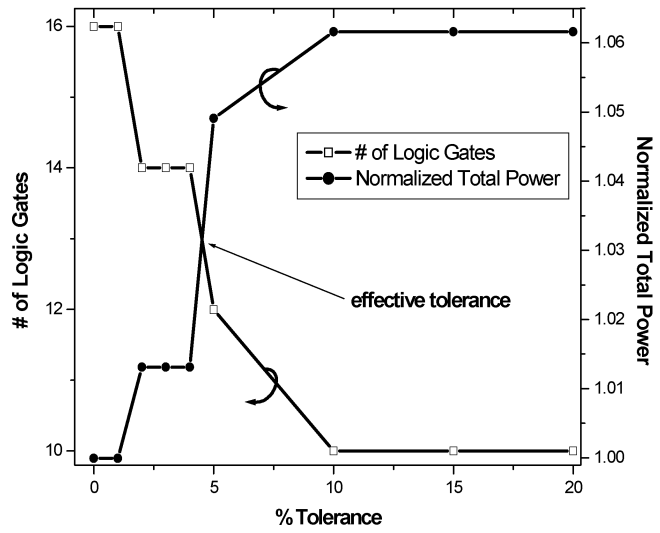


Fig. 3. Percent tolerance in the encoding/decoding logic with respect to total power and number of gates.

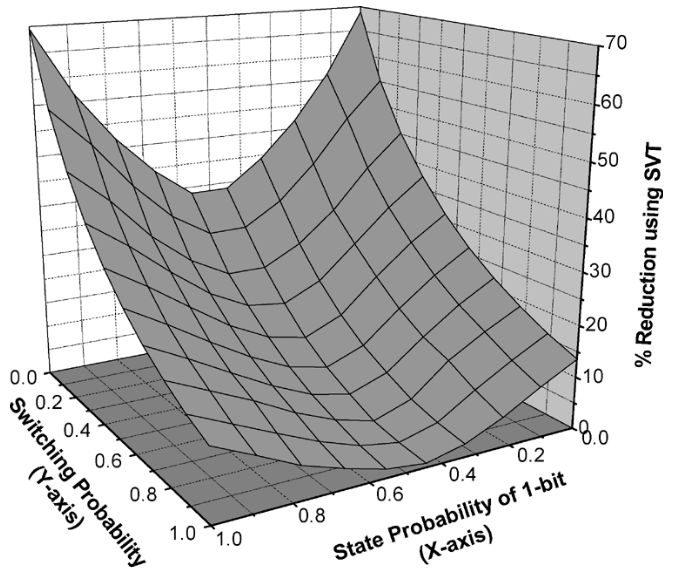


Fig. 5. Total power reduction for different probability profiles.

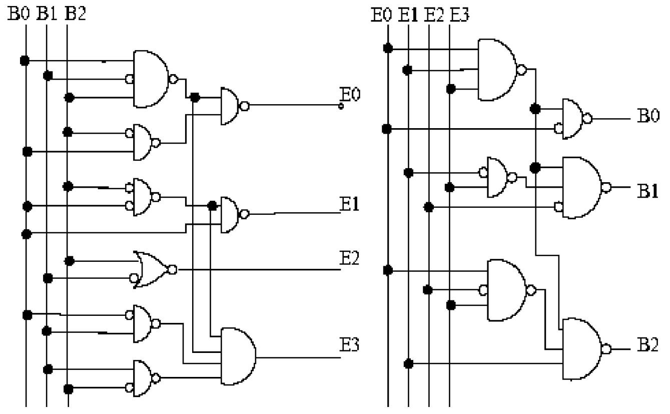


Fig. 4. Sample logic implementation for the benchmark gcc. Input bits are (B2-B0) and the encoded bits are (E3-E0).

the *BuffPower* algorithm to generate an ideal (power and overhead-aware) encoding. This encoding was used to determine the best logic implementation for the various memory bus traces. As an example, the encode and decode logic for the gcc case is shown in Fig. 4.

Before turning to the specific applications that we investigated, we first sought to identify the general probability profiles that provide maximum savings using the SVT buffer technique. To accomplish this, we conducted a theoretical survey where we constructed the complete range of state and transition probability tables and measured the savings obtained in total power using the encoded SVT buffer configuration. In Fig. 5, the X axis corresponds to the state probability that any one bit is a "0." For a 3-bit block, we assume that the state probabilities of all bits are independent and derive the state probability corresponding to the 3-bit block. For instance, if the X-axis value $= 0.7$, then the state probability of "001" will be equal to $(0.7^2 * 0.3)$. The Y axis represents the probability that a 3-bit block in a particular state switches to any of the seven other possible states. For the sake of simplicity in this exploratory analysis, we assume that all transition events are equally probable. Thus, if the Y-axis value $=$ 0.7, then, since there 56 possible transition events for the 3-bit block, the probability value for the transition of one state to a different state is given by (0.7/56) while the probability of a self-transition for each of the eight states is given by (0.3/8). We note that there is some amount of correlation between the values given on the X- and Y-axes. For instance, if the state probability is one then the transition probability value must be zero since all bits remain in the "0" state indefinitely. We identify such corner cases and enforce the X- and Y-axis values to be self-consistent accordingly. The Z-axis represents the reduction in total power using the encoded SVT buffer configuration.

From the plot, we first see that the power reductions are symmetric about the $X = 0.5$ line. This is to be expected since we can easily construct SVT buffers that can be optimized for either a dominant "0" or "1" bit. We also see that we obtain better

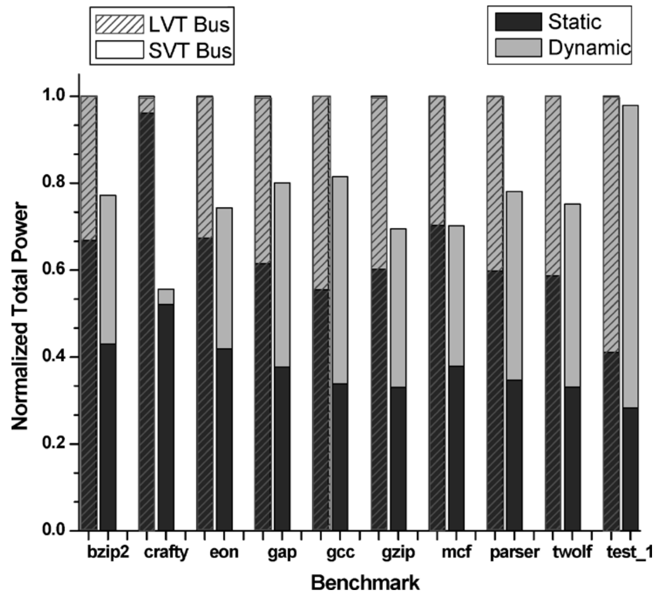


Fig. 6. Total power reduction on various benchmarks using SVT buffer scheme and enhanced self-shield encoding.

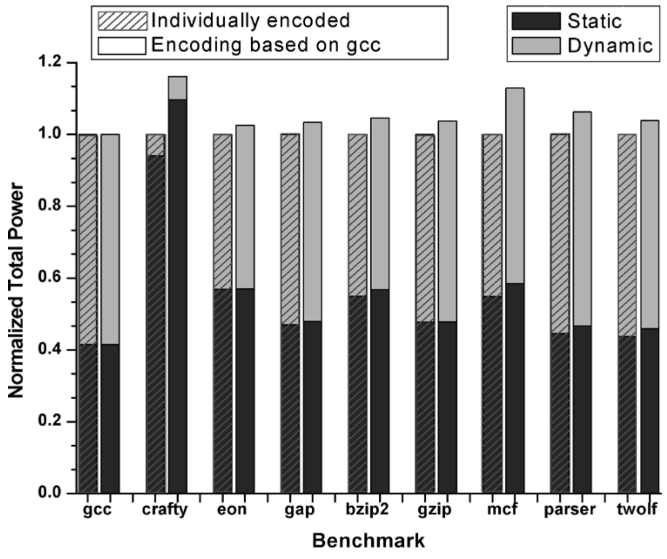


Fig. 7. Total power comparison when using a single application-independent encoding.

power reductions when the single-bit state probability (X-axis value) is either very high or very low. These endpoints correspond to the cases when either the 111 or the 000 state is the dominant one. In such situations, the *BuffPower* algorithm encodes the highly probable state to the one that expends the least amount of leakage, thus obtaining a significant power reduction. Finally, we see that, as we move along the Y-axis, the values for switching probability decrease and we can obtain greater total power reductions. In general, low switching activity will clearly enhance the contribution of leakage to total power, in which case the encoding algorithm achieves better power reductions.

In the theoretical analysis shown in Fig. 5, we assumed that the state probability of individual bits in a bus are independent and all transition events are equally probable. However, for a general purpose application, this is an unrealistic assumption as the probability tables are heavily dependent on the actual behavior of the bus lines. In the set of experiments that follow, we construct the probability tables for the given benchmarks using actual memory traces extracted by running the application on a microprocessor.

In Fig. 6, the base case (striped column) corresponds to an unencoded set of bus lines driven using only LVT buffers. This plot shows that, for every application, the total power is reduced in the SVT bus case. On average, our method provides a savings of 26% in total power and in the best case about 44%. There was an average leakage savings of 42% with a small increase ($<5\%$) in dynamic power, due to the additional bus line. For the TEST_1 application, although the dynamic power in the 4-bit encoded bus increases significantly, there was still enough savings in static power such that the total power was reduced slightly. This shows that our proposed encoding scheme is robust, even for cases where leakage power is a small portion of the total power.

In addition to finding the dynamic and static power reduction for each memory bus trace with respect to its own optimal encoding, we calculated the average state and transition probability table across all memory traces. It has been observed that on-the-fly encoding of the bus for each specific application would incur substantial overhead [21]. Instead, we used the memory traces of all applications and constructed the average state and transition probability tables. We observed that the tables obtained from such an averaging were almost identical to the tables corresponding to gcc. In Fig. 7, we use the encoding corresponding to gcc (tolerance $T = 5\%$) and calculate the power for each application. The increase in power when using a generic encoding compared to an application-specific encoding is about 10%–15% in two cases and is essentially zero for the remaining applications.

The encoding scheme requires the splitting of a wide bus into blocks of 3 bits each and the subsequent inclusion of encoder and decoder circuits for each such 3-bit block. In the experiments done previously, the codec circuits were constructed by creating average state and transition probability tables over all 3-bit blocks in the bus line. Thus, all 3-bit blocks had the same codec circuit. However, a more optimal configuration is one where each 3-bit block is considered separately and different codec circuits are constructed for each such block such that the switching behavior of each block is catered to individually. We explored the utility of such an approach by first creating a master memory trace consisting of the traces of all the nine benchmark programs. For this master trace, since the top 31 bits of the 64-bit bus line were all zeros, they did not require any encoding setup. The bottom 33 bits were split into eleven blocks of 3 bits each. In Table III, we summarize the difference between average-case encoding and block-specific encoding. For the sake of simplicity, we consider the case when tolerance $T = 0\%$ such that only the encoding corresponding to the minimum power is considered. From this table we see that block-specific encoding reduces the total power by about 11.0% while using 7.5% fewer number of gates. We note that such block-specific encoding comes at the expense of increased

TABLE III
ADVANTAGES OF BLOCK-SPECIFIC ENCODING VERSUS AVERAGE-CASE ENCODING

Block Number	Average-Case Encoding		Block-specific Encoding		% Reduction in Power
	# Gates	Total Power	# Gates	Total Power	
1	17	3.26	17	2.15	34.0
2	17	2.28	12	2.26	0.8
3	17	3.09	16	2.74	11.2
4	17	3.27	18	3.08	5.9
5	17	4.08	16	3.56	12.8
6	17	5.06	15	3.77	25.5
7	17	4.78	18	4.01	15.8
8	17	5.36	15	5.00	6.6
9	17	5.51	17	5.40	2.1
10	17	6.19	13	5.99	3.3
11	17	2.43	16	2.34	3.6
Total	187	45.28	173	40.30	11.0

logic synthesis effort since each codec block may require additional gates to be added to the cell libraries.

Finally, we mention that the self-shield encoding can reduce propagation delay in the bus line due to the elimination of crosstalk. Without encoding, in the transition 010 $\rightarrow$ 101, there is crosstalk between all three of the lines. This transition is encoded to 0001 $\rightarrow$ 1100 in the gcc application, for example, which has no crosstalk. The reduction in delay of the bus line (i.e., the improvement achieved due to the use of crosstalk-aware encoding) in this case is about 70 ps. This delay reduction helps compensate for the codec logic delay, which we also address and minimize in our approach for the first time. The codec logic overhead is roughly 140 ps, resulting in a net delay overhead of about 70 ps. Note that the codec overhead is common to all types of bus encoding including those that are not leakage- or crosstalk-aware. Thus, broadening the impact of encoding schemes to include more effects such as leakage (in this work) does not actually incur any additional delay overhead compared to less general dynamic power only or crosstalk-only encoding methods.

V. Conclusion

To address the growing issue of runtime leakage power consumed by on-chip buffers, we propose an enhanced self-shield encoding algorithm combined with a novel SVT buffering technique for crosstalk-aware low-power bus encoding. We present a configuration that uses HVT-NMOS/LVT-PMOS and *vice versa* placed along each bus line to create disparate high/low leakage states. The LVT devices enable better performance (and higher speeds) while the HVT devices ensure that the bus line remains in the low-leakage state as much as possible during runtime. We illustrate the uniqueness of the encoding by showing that there is exactly one type of codeword mapping that achieves both leakage minimization and crosstalk elimination. Also, we consider encode/decode logic and demonstrate that the logic overhead for this type of encoding contributes only a small amount to the overall delay of the bus line.

We achieve considerable reductions in total power consumption while simultaneously eliminating crosstalk-inducing transitions and reducing the encode/decode logic overhead. On average, we achieved a total power savings of about 26% with a best case savings of 44%. Compared to a crosstalk-only encoding method, our scheme achieves 54% power savings on average. We also show that when an application independent encoding scheme is used, there is no significant impact on the performance of our algorithm

References

[1] S. Borkar, "Design challenges of technology scaling," *IEEE Micro*, vol. 19, no. 4, pp. 23–29, Jul.-Aug. 1999.

[2] J. Kao, S. Narendra, and A. Chandrakasan, "Subthreshold leakage modeling and reduction techniques," in *Proc. IEEE/ACM Int. Conf. Computer Aided Design (ICCAD)*, Nov. 2002, pp. 141–148.

[3] K. Bernstein, C.-T. Chuang, R. Joshi, and R. Puri, "Design and CAD challenges in sub-90 nm CMOS technologies," in *Proc. IEEE/ACM Int. Conf. Computer Aided Design (ICCAD)*, Nov. 2003, pp. 129–136.

[4] P. Saxena, N. Menezes, P. Cocchini, and D. Kirkpatrick, "Repeater scaling and its impact on CAD," *IEEE Trans. Computer Aided Design Integr. Circuits Syst.*, vol. 23, no. 4, pp. 451–463, Apr. 2004.

[5] U. Narayanan, K.-S. Chung, and T. Kim, "Enhanced bus-invert encodings for low power," in *Proc. IEEE Int. Symp. Circuits Syst.*, vol. 5, May 2002, pp. 25–28.

[6] B. Victor and K. Keutzer, "Bus encoding to prevent crosstalk delay," in *Proc. IEEE/ACM Int. Conf. Computer Aided Design (ICCAD)*, Nov. 2001, pp. 57–63.

[7] C.-G. Lyuh and T. Kim, "Low power bus encoding with crosstalk delay elimination," in *Proc. IEEE Int. ASIC/SOC Conf.*, Sep. 2002, pp. 389–393.

[8] Y. Shin and T. Sakurai, "Coupling-driven bus design for low-power application-specific systems," in *Proc. IEEE/ACM Design Automation Conf.*, Jun. 2001, pp. 750–753.

[9] N. Chang, K. Kim, and J. Cho, "Bus encoding for low-power high-performance memory systems," in *Proc. IEEE/ACM Design Automation Conf.*, Jun. 2000, pp. 800–805.

[10] S. Tyagi *et al.*, "A 130 nm generation logic technology featuring 70 nm transistors, dual-V_t transistors and 6 layers of Cu interconnects," in *Proc. Int. Electron Devices Meeting*, Dec. 2000, pp. 567–570.

[11] D. Sylvester and H. Kaul, "Future performance challenges in nanometer design," in *Proc. IEEE/ACM Design Automation Conf.*, Jun. 2001, pp. 3–8.

[12] M. Khellah, J. Tschanz, Y. Ye, S. Narendra, and V. De, "Static pulsed bus for on-chip interconnects," in *Proc. VLSI Symp.*, 2002, pp. 78–79.

[13] Q. Wang and S. Vrudhula, "An investigation of power delay trade-offs for dual V_t CMOS circuits," in *Proc. Int. Conf. Computer Design*, Oct. 1999, pp. 556–562.

[14] N. Azizi, A. Moshovos, and F. Najm, "Low-leakage asymmetric-cell SRAM," in *Proc. Int. Symp. Low Power Electron. Design*, Aug. 2002, pp. 48–51.

[15] R. Arunachalam, E. Acar, and S. Nassif, "Optimal shielding/spacing metrics for low power design," in *Proc. IEEE Symp. VLSI*, Feb. 2003, pp. 167–172.

[16] C.-K. Cheng, J. Lillis, S. Lin, and N. Chang, *Interconnect Analysis and Synthesis*. New York: Wiley, 2000.

[17] S. Morton, "Inductance: Implications and solutions for high-speed digital circuits – On-chip signaling," in *Proc. IEEE Solid-State Circuits Conf.*, Feb. 2002, pp. 554–557.

[18] K. Lepak, I. Luwandi, and L. He, "Simultaneous shield insertion and net ordering under explicit RLC noise constraint," in *Proc. IEEE/ACM Design Automation Conf.*, Jun. 2001, pp. 199–202.

[19] O. Coudert and T. Sasao, "Two-level logic minimization," in *Logic Synthesis Verificat.*: Kluwer, 1987, pp. 1–27.

[20] Spec CINT 2000 Benchmarks [Online]. Available: http://www.specbench.org/osg/cpu2000/CINT2000/

[21] L. Li, N. Vijaykrishnan, M. Kandemir, and M. Irwin, "Adaptive error protection for energy efficiency," in *Proc. IEEE/ACM Int. Conf. Computer Aided Design (ICCAD)*, Nov. 2003, pp. 2–7.

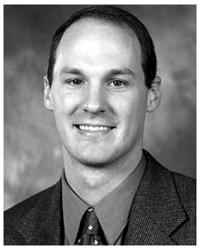
Rajeev R. Rao received the B.S. degree in electrical and computer engineering from Rutgers University, New Brunswick, NJ, in 2002, and the M.S.E. degree in computer science and engineering from the University of Michigan, Ann Arbor, in 2004, where he is currently working toward the Ph.D. degree.

In the summer of 2003, he was with IBM Austin Research Laboratories, Austin, TX, where he was a Research Co-op working on leakage power analysis. His research interests include modeling and analysis of robust, low-power very large scale integration (VLSI) designs and variability-aware circuit approaches.

Harmander S. Deogun received the B.S.E. degree in electrical and biomedical engineering, with distinction, from Duke University in 2001, and his M.S.E. degree in electrical engineering from University of Michigan – Ann Arbor in 2003. He is currently pursuing his Ph.D. degree in electrical engineering at the University of Michigan – Ann Arbor where his research interests include low power and process variation robust circuit design techniques. He has been an intern at IBM Research in the summers of 2003, 2004 and 2005.

David Blaauw (M'94) received the B.S. degree in physics and computer science from Duke University, Durham, NC, in 1986 and the M.S. and Ph.D. degrees in computer science from the University of Illinois, Urbana, in 1988 and 1991, respectively.

He was with IBM Corporation as a Development Staff Member until August 1993. From 1993 until August 2001, he was with Motorola Inc., Austin, TX, where he was the Manager of the High Performance Design Technology Group. Since August 2001, he has been on the faculty at the University of Michigan, Ann Arbor, as an Associate Professor. His work has focused on VLSI design and CAD with particular emphasis on circuit design and optimization for high-performance and low-power designs. He was the Technical Program Chair and General Chair for the International Symposium on Low Power Electronics and Design in 1999 and 2000, respectively, and was the Technical Program Co-Chair and member of the Executive Committee the ACM/IEEE Design Automation Conference in 2000 and 2001.



Dennis Sylvester (S'95–M'00–SM'04) received the B.S. degree (*summa cum laude*) from the University of Michigan, Ann Arbor, in 1995, and the M.S. and Ph.D. degrees from the University of California, Berkeley (UC-Berkeley), in 1997 and 1999, respectively, all in electrical engineering.

He is now an Associate Professor of electrical engineering with the University of Michigan. He previously held research staff positions with the Advanced Technology Group of Synopsys, Mountain View, CA, and with Hewlett-Packard Laboratories, Palo Alto, CA. He has published numerous papers along with one book and several book chapters in his field of research, which includes low-power circuit design and design automation techniques, design-for-manufacturability, and on-chip interconnect modeling. He also serves as a consultant and technical advisory board member for several electronic design automation firms in these areas.

Dr. Sylvester is a member of the Association for Computing Machinery (ACM), the American Society of Engineering Education, and Eta Kappa Nu. He was the recipient of a National Science Foundation CAREER Award, the 2000 Beatrice Winner Award at ISSCC, the 2004 IBM Faculty Award, and several Best Paper Awards and nominations. He was the recipient of the ACM SIGDA Outstanding New Faculty Award, the 1938E Award from the College of Engineering for teaching and mentoring, and the Henry Russel Award, which is the highest award given to faculty at the University of Michigan. His dissertation research was recognized with the 2000 David J. Sakrison Memorial Prize as the most outstanding research in the Electrical Engineering and Computer Science Department of UC-Berkeley. He has served on the technical program committees of numerous design automation and circuit design conferences and was General Chair of the 2003 ACM/IEEE System-Level Interconnect Prediction (SLIP) Workshop and the 2005 ACM/IEEE Workshop on Timing Issues in the Synthesis and Specification of Digital Systems (TAU). He is currently an Associate Editor for the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS. He also helped define the circuit and physical design roadmap of the International Technology Roadmap for Semicondcutors (ITRS) U.S. Design Technology Working Group from 2001 to 2003.