Low-Energy Encoding for Deep-Submicron Address Buses

Luca Macchiarulo Enrico Macii Massimo Poncino

Politecnico di Torino
Torino, ITALY 10129

ABSTRACT

In this paper, we introduce a new encoding scheme that explicitly targets the minimization of the bus energy due to the crosstalk capacitances between adjacent bus lines. The key transformation operated by the code consists of a permutation of the bus lines, implemented directly during physical design; as a desirable consequence, no additional encoding/decoding logic is required at the bus boundaries, thus implying that no latency penalty is introduced on the processor-memory path. An additional feature of the permutation-based code is that the encoding function can be determined without any knowledge of the binary stream being transmitted. Therefore, the code can be effectively exploited in general-purpose computing systems. The proposed code works best on address buses; savings obtained for different address traces generated by two different processors are in the order of 26% with respect to the unencoded streams.

1 INTRODUCTION

Several bus encoding schemes for low power have been proposed in the literature in the last few years. Most of these approaches target the minimization the transition activity on the bus, with the objective of reducing the switching of the capacitances of the bus lines. Some codes, e.g., the Bus-Invert [1], its variants [2, 3] and the Adaptive [4], have general applicability, i.e., they do not require any knowledge about the streams being transmitted. Others achieve more aggressive transition activity minimization by taking advantage of some characteristics of the pattern sequence traveling on the bus. In particular, a whole class of encoding methods is targeted towards address bus power minimization, whose strong sequentiality is exploited by codes such as Gray [5], T0 [6] and their modifications [7, 8]. Unfortunately, sizable transition activity reductions are accompanied by high-overhead codecs. In particular, while for Gray the penalty is in performance, T0 encoders and decoders are more energy demanding. Thus, the T0 solution is only suitable to off-chip buses, where high wiring capacitances allow to amortize the codec overhead.

Further optimizations can be obtained when the encoding function is derived from the analysis of a (set of) specific stream(s). In this case, the statistical properties of the input sequence are assumed to be known and can be used to automatically generate the encoding/decoding functions and their corresponding interface circuitry.

Codes of this kind are well suited for embedded systems, where cores and microcontrollers tend to repeatedly execute a given application. Beach [9] and Working Zone [10] are examples of application-specific codes thought explicitly for address buses, while the information-theoretic code of [4], a practical generalization of the work of [11, 12], finds application also in data bus encoding.

All the encoding solutions mentioned above have one point in common: They try to minimize the number of transitions on the bus lines, because this translates to a reduction of the capacitance that is switched during the communication. With technology scaling, however, the importance of inter-wire, or crosstalk, capacitances is becoming predominant.

It has been estimated that the simultaneous transition to opposite values (i.e., $0 \rightarrow 1$ and $1 \rightarrow 0$) of two adjacent bus lines dissipates about four times more energy than without considering coupling effects, for technologies below $0.25\mu m$ [13]. As a consequence, accounting for crosstalk capacitances between pairs of adjacent wires is becoming key for the applicability of low-power bus encoding in modern designs.

Recent papers [13, 14] provide some pioneering research on this subject. The solutions proposed in those works share however a major limitation with any other bus encoding technique available in the literature: They require some interface circuitry to implement the transformation defined by the encoding. In several cases, e.g., performance-constrained designs, the addition of the extra logic cannot be afforded because it lies on the critical path of memory-to-processor transfers, even if some encoding schemes reduce this hardware overhead to the minimum.

In this work, stemming from the observation that the minimization of the coupling capacitances can be achieved with a permutation of the bus lines, we propose a new address bus encoding scheme that does not require explicit encoding/decoding circuitry. The permutation of the bus lines is accomplished at the layout level; this is a distinctive feature of our approach, because it allows the introduction of the encoding during physical design, where bus capacitance information can be accurately estimated. Conversely, all existing techniques need to be applied at the architectural level, since the presence of the encoding/decoding logic must be taken into account during RTL-to-layout synthesis in order to possibly enable a recovery of the extra latency imposed by the codec.

We formulate the problem of finding an effective permutation of the bus lines as a graph problem (a variant of the well-known TSP [15]), for which very efficient heuristics do exist. We provide experimental evidence that permutation-based encoding is general enough not to require a pre-processing phase of the address stream of binary patterns being transmitted over the bus; therefore, the method can be categorized among the general-purpose address bus encoding schemes, such as the Gray code.

The effectiveness of the code has been assessed for a large variety of address streams corresponding to various programs run on two core processors (namely, the MIPS and the ARM). When used in its general-purpose form (i.e., no preliminary program profiling is executed), the permutation-based encoding yields an average bus energy savings of 26% with respect to the case of no encoding.

The performance of the proposed code is comparable to those of codes like Gray, with the advantage that it no encoding is required. Furthermore, the proposed code should not be considered as an alternative to conventional encodings; rather, it could be combined with this encoding to further reduce bus energy with no latency penalty, as proved by our results.

Usage of the permutation-based encoding in the context of data buses has also been tried for a set of different data streams of different nature. As expected, the achieved energy savings are limited (average of 5%) if the code is not customized to each specific trace. On the other hand, savings increase to an average of 14% if preliminary analysis of each data sequence is performed and the codec functions determined accordingly.

2 DEEP SUBMICRON BUS MODEL

The physical capacitances of a bus line can be modeled as shown in Figure 1. Besides the usual capacitance between the line and ground, C_L (the *self* capacitance), also the coupling capacitance between the line and its adjacent lines, C_I, must be modeled. The ratio $\lambda = \frac{C_I}{C_L}$ represents the relative weight of the two capacitive effects. The case $\lambda = 0$ represents the conventional bus energy model based on the switching of the bus self capacitances.

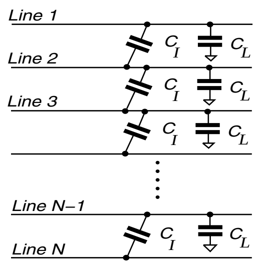

Figure 1: Model of a Sub-Micron Bus.

In deep-submicron technologies, λ tends to become larger than 1, expressing the increasingly dominant effect of coupling capacitances w.r.t. to self capacitances. The energy model per cycle for a bus line will thus include the two capacitive effects:

$$E_{bus} = (\alpha_L C_L + \alpha_I C_I) V_{dd}^2, \tag{1}$$

where α_L and α_I denote the rates at which each capacitance is switched. While α_L represents the conventional switching activity of the lines, α_I is related to the simultaneous switching of two adjacent lines. There are four types of transition pairs between two adjacent lines a and b:

1. a and b switch to different final values;
2. One of the two lines switches, while the other does not;
3. a and b switch to the same final value;
4. None of the lines switches.

Of these four types, only the first two cause C_I to switch, yet by a different amount. The first type of transitions will cause C_I to switch twice in a cycle, while transitions of type 2 imply a single switch per clock cycle. In the case of type 2 transitions, however, C_I will switch only if the final values on the two bus lines are different.

We assume that the transitions on a and b are perfectly aligned in time. This assumption is reasonable because the drivers typically latch the data sent on the bus, thus eliminating possible misalignments between two simultaneous transitions. If this assumption does not apply, we should also consider possible switchings for transitions of type 3.

Table 1 shows the normalized energy consumption for a two-line bus, when all capacitive effects are considered [13]. In the table, only transitions from 0 to 1 are counted as power dissipating transitions on C_L. This is because the $0 \rightarrow 1$ transition is the one that actually charges the capacitance, drawing energy from the power supply. In practice, this distinction replaces the $\frac{1}{2}$ factor in the conventional power dissipation model.

		(b^t, b^{t+1})			
		$0 \rightarrow 0$	$0 \rightarrow 1$	$1 \rightarrow 0$	$1 \rightarrow 1$
(a^t, a^{t+1})	$0 \rightarrow 0$	0	$1+\lambda$	0	0
	$0 \rightarrow 1$	$1+\lambda$	2	$1+2\lambda$	1
	$1 \rightarrow 0$	0	$1+2\lambda$	0	λ
	$1 \rightarrow 1$	0	1	λ	0

Table 1: Normalized Energy on Two Adjacent Bus Lines.

The table clearly shows that increasingly larger values of λ, as a result of smaller technologies, will tend to emphasize the importance of the energy consumption due to switching of the coupling capacitances, as opposed to that of the self capacitances. Conventional bus encoding schemes that target the minimization of the energy component due to C_L will thus target an increasingly smaller fraction of the overall energy consumption.

3 WIRE PERMUTATION ALGORITHM

In the existing literature [13, 14], the problem of minimizing the energy consumption due to coupling capacitances has been tackled by properly encoding the information transmitted on the bus so that the crosstalk energy is minimized. In other terms, redundancy, i.e., additional logic, is used to minimize the cost function.

The solution we propose in this paper is based on a different idea. A careful analysis of the problem shows that the minimization of the energy due to the coupling capacitances can be achieved by properly permuting (some of) the bus lines. Such a solution has the advantage of not requiring an explicit codec, because the transformation of the data is equivalent to a crossbar switch.

Problem Formulation

The problem of finding the best permutation of the bus lines can be formulated as a graph problem. We build a completely connected undirected graph $G(V, E, W)$, where $V = \{v_i\}$ is the set of vertices, $E = \{e_{i,j}\}$ is the set of edges, and $W = \{w_{i,j}\}$ is the set of edge weights. Vertices correspond to bus lines, and weights denote the conditional switching probabilities between pairs of bus lines. More precisely, given the asymmetric nature of the weights (not all transition pairs have the same importance, as shown in Table 1), the actual weights are the product of conditional switching probabilities and the coefficients that multiply parameter λ in Table 1. From the statistical point of view, the computation of the weights requires the knowledge of both transition and signal probabilities of each bus line.

The graph G is completely connected because it represents all the potential permutations of the nodes. The *cost* of a permutation, that is, of a path in the graph, is simply the sum of the costs of the edges that belong to the path.

The problem of finding the permutation with the minimum overall cost is therefore equivalent to finding the Hamiltonian *path* (i.e., a path that visits each vertex exactly once) with minimum cost, starting from a given vertex v_i, and ending with another vertex v_f. The minimum-cost Hamiltonian path in the graph represents the permutation of the bus lines that minimizes the given cost function. This problem is a variant of the well-known traveling salesman problem (TSP), that solves the more specific problem of finding the minimum cost Hamiltonian *cycle*, that is, a path with the same initial and final vertices. A further difference in our case is that the initial and final vertices are not specified in advance, because all possible permutations must be explored. Nevertheless, minimal modifications to an existing TSP solver are sufficient to make it applicable to our case.

Although the size of the TSP that must be solved is relatively small (the number of nodes in the graph is equal to the number of bus lines), it is large enough to prevent the use of an exact TSP algorithm. We must then resort to a heuristic solution, to allow reasonable execution time. In general, we can sacrifice some performance for a better quality of the solution, since the TSP will be executed once and for all for a given system.

Among the vast choice of heuristic TSP solvers in the literature, we have implemented a local search algorithm inspired by the work of Lin [16]. Local searches are based on the exploration of the neighborhood of an initial solution to the problem. To guarantee a reasonably large exploration of the search space, several initial solutions should be tried as starting points of the search. Customizing the general paradigm of local search to a specific problem requires the definition of: (i) The neighborhood and its size; (ii) The set of starting points and its cardinality.

In the case of TSP, the definition of neighborhood is based on the representation of a solution as a permutation $\Pi = \{\pi_1, \ldots, \pi_N\}$ of the vertices. Solving the TSP thus consists of finding the minimum-cost permutation of a set of elements, as opposed to a path, according to the conventional definition of TSP.

The neighborhood of a solution for the TSP can then be defined as the set of permutations that can be obtained by swapping a subset of the elements of the initial permutation. The size of the neighborhood depends on the number k of elements in this subset; we thus speak of the k-neighborhood of a solution.

In general, if N is the number of vertices, this amounts to evaluating $(k!-1) \cdot \begin{pmatrix} N \\ k \end{pmatrix}$ points. The term -1 accounts for the fact that one out of $k!$ permutations of k elements coincides with the initial solution.

It has been proved that the best tradeoff between the size of the neighborhood and the quality of the solution is achieved with $k = 3$; in our case, this amounts to evaluating $(3!-1) \cdot \begin{pmatrix} 32 \\ 3 \end{pmatrix} =$ 24800 neighbor points of the initial solution. Furthermore, in our implementation the search using the 3-neighborhood scheme has been repeated for $R = 100$ different random starting points. This value has shown to yield the optimum length of the TSP with a probability of 0.99 [16]. The average runtime of the algorithm with $R = 100, k = 3$ is of a few seconds on an Alpha Personal Workstation 433 with 128MB of main memory.

4 PHYSICAL DESIGN ISSUES

In principle, the encoding/decoding functions can be realized without any area, performance and power overhead, because it simply consists of a permutation of some of the bus lines. In practice, however, their implementation may have introduce some overhead at the physical level. As a general observation, we note that the routing overhead (occupation of other levels of metal and vias) is confined to relatively small areas close to the components, and therefore has small impact to the global routing. In this section, we analyze and quantify the impact of the encoding/decoding logic on conventional design metrics.

4.1 Design of the Permutation Network

In order to evaluate the impact on area, performance and power of the codec on the final design, we must perform an accurate analysis at the physical level. The first issue is how to realize an effective layout of the permutation network.

Since our objective is that of employing a unique permutation for all applications, we can think of inserting a sort of "hard-wired" crossbar switch into the layout of the design. However, a plain implementation of a sorting network requires as many vias as the number of bus lines, in the worst case.

For these reasons, we decided to implement a different strategy, that guarantees the use of at most two vias per wire. The idea is to route a signal directly to its final position (wire i goes to π_i). Care must be taken, though, to prevent shorts between distinct wires. We implemented a technique that is suitable for any permutation: We first consider exchanges such that $\pi_i \geq i$ and route them toward the final position π_i using a second metal layer: This allows the routing of 2 different signals on the same "bus slot" (see, for example, the two overlapping layers at the bottom of the layout of Figure 2).

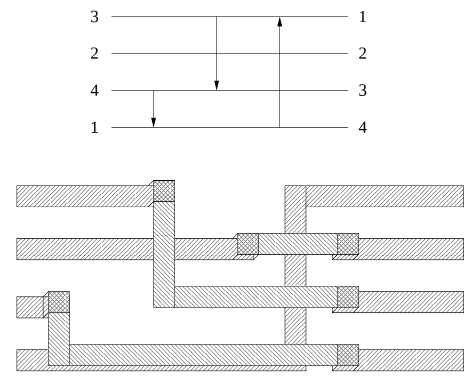

Figure 2: Via-Constrained Routing Strategy.

In this solution, the overlap area has to be strictly controlled to avoid high mutual capacitances. It is possible to route all $\pi_i \geq i$ signals without causing any short by using a slight modification of the left-edge channel routing algorithm. After all $\pi_i \geq i$ wires are routed (wires 4, 3 and 2 in the example of Figure 2), the remaining wires are treated similarly, without introducing vias, in the same metal layer as the rest of the bus. It might be necessary, in some cases, to build a sort of "bridge" to cross over a previously routed signal. This shows how it is possible to reduce to 2 the maximum number of vias needed, and try to compact the network.

Figure 2 shows that the complexity of the encoder is somehow related to the number of wires to be permuted. This is intuitive, because, if m wires have to be permuted, at most m vertical tracks are needed. In the following, we refer to the number of displaced wires as the *weight* D of a permutation $\Pi = \{\pi_1, \ldots, \pi_N\}$, defined as $D = \sum_{i=1}^{i=N} (\pi_i \neq i)$. This quantity can be used to control the complexity of the permutation network, so that the various cost metrics can be parameterized with respect to D.

For an acceptable estimation of performance and power overhead introduced by the codec it is necessary to accurately evaluate all parasitics of the interconnection network. To this purpose, we generated the actual layout, using the algorithm discussed above, for 32 different permutations encompassing all possible values of D (from 0 to 32, excluding 1), and we extracted resistance and capacitance values of a distributed net describing the electrical behavior of the encoder-bus-decoder cascade. The target technology is a 0.25 μm with 6 levels of metal. We decided to route the bus on metal3, and route the codec nets on metal3/metal4. Minimum pitch has been used throughout the designs. The detailed Standard Parasitics Format obtained from an extraction with Cadence Affirma Hyperextract (a state-of-the-art extractor based on 2.5D extraction that proved to give well-matched results with 3D extractors), has been included in a SPICE netlist and then simulated. The overall distance from the transmitter end to the receiver end of the bus (including the presence of the encoder-decoder pair) was chosen to be 1-mm long. The bus loads at the receiver end were chosen according to actual memory blocks input characteristics, and the driver capability according to its loads. The netlists were simulated with HSPICE with inputs switching at the maximal activity. The reason was that we were interested in the comparison between the unencoded bus and the coded version, rather than absolute values. For each net, all maximum timings were considered in performance reports.
Figure 3 synthetically reports the results of the characterization. The plot depicts area, delay, and power curves relative to the encoder with respect to D. To allow the visualization of the three quantities in the diagram, values have been normalized with respect to the values without the permutation network (assumed to be 1).

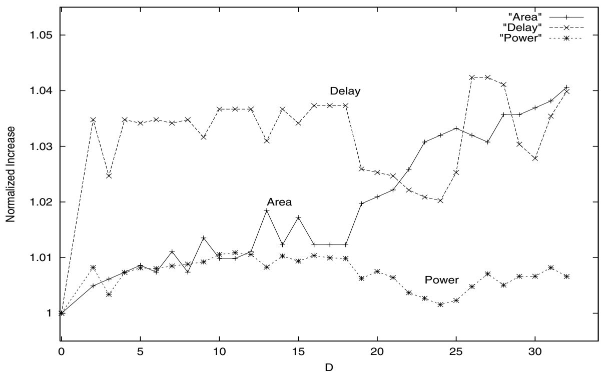

Figure 3: Normalized Area, Delay, and Power vs. D.

The area overhead is roughly proportional to the weight of the permutation. In fact, while the height of the encoder almost stays constant, its width depends on the value of D. The reference value without encoding is the total area of the bus (i.e, 26,000 μm^2). We observe that, since the permutation network does not use active area (i.e., cells), we are using the same chip area that would be occupied by the bus itself. The real area overhead is given only by the second level of metal which is necessary to exchange the wires. The delay overhead never exceeds 6.3 ps, a value which is well below the delay of a single library cell. This amounts to less than 5% of the delay of the entire bus (158 ps). We observe that the delay is almost insensitive to D, while it depends on the resistive effects of the vias (that are never more than 2). Power figures are also insensitive to variations of D, and the overhead is always below 1%, which is within the noise margin of the SPICE simulation.

4.2 Effect of Boundary Capacitances

The previous analysis considered the bus as it was isolated or, equivalently, far enough from other sources of crosstalk (e.g., ground lines). Obviously, this is an ideal assumption and should be removed when the effectiveness of the encoding must assessed with accuracy.
The presence of neighboring wires adversely affects the potential energy optimization achievable by our method because the designer typically has no control on the information traveling on these wires. Therefore, the switching of the coupling capacitances between the two boundary bus lines (i.e., Lines 1 and 32) and their respective neighboring lines will cause additional energy consumption.
Taking these effects into account in the algorithm of Section 3 requires the analysis of the information that is carried by neighboring wires. In particular, we should evaluate the possible correlation between these wires and the boundary bus lines.
The typical interaction between the bus and the neighbor wires is depicted in Figure 4. In general, there will be several wire segments of different lengths (typically much shorter than the length of the bus), at different distances from the bus (and thus with different coupling capacitances) and running in parallel with it.

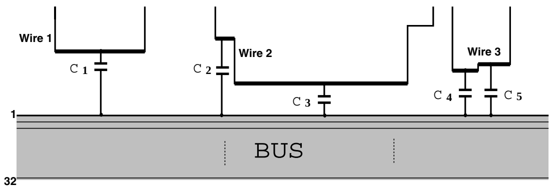

Figure 4: Effect of Boundary Capacitances.

In practice, however, the neighbor wires will not be driven by the same drivers as the bus. From the point of view of energy balance, lines belonging to different drivers can be considered as independent. Therefore, the energy due to the boundary coupling capacitances depends only on the statistics of the bus boundary lines, and not on their relation with local boundary wires. This implies that transition probabilities (of the bus boundary lines) can be used instead of conditional transition probabilities (as for internal bus lines).
The energy model per cycle can thus be modified as follows:

$$E_{bus} = (\alpha_L C_L + \alpha_I C_I + (\alpha_B^1 + \alpha_B^N) C_B) V_{dd}^2 \qquad (2)$$

where α_B^1 and α_B^N denote the transition probabilities of the two boundary bus lines, $C_B = \frac{C_I}{\beta}$ the boundary capacitance seen by the two wires, parameterized with respect to the value of the distance β of the neighbor wires. The minimum value of β is 1, indicating that neighbor wires are at the same distance as the bus lines (typically, the minimum pitch).

5 EXPERIMENTAL RESULTS

5.1 PB Encoding

We have applied the permutation-based (PB) encoding to several address traces obtained from two different code profilers (namely, `pixie` for a MIPS processor, and `Armulator` for an ARM processor). Two artificial traces that cover a corner case typical of address traces (i.e., highly sequential streams) have also been considered: `Counter` represents a perfect counter that starts from a random address; `CountSkip` is a counter sequence intermixed with random jumps to new locations.

To determine the encoding and decoding functions, we have first built the cost matrices (corresponding to the weights of the graph edges) for all the chosen benchmarks. Then, all the matrices have been averaged to a single correlation matrix. Finally, this matrix has been used to compute the best permutation by applying the TSP solver on the graph corresponding to the single correlation matrix.

Energy data are shown in Table 2. They are determined assuming a supply voltage of $2.5V$. Capacitance values are $C_I = 1.5pF$ and $C_L = 0.5pF$, that is, $\lambda = 3$. This ratio corresponds a 15-mm bus, whose lines are spaced by 1 μm (more than than the minimum pitch – 0.8 μm). Considering a relatively long bus increases absolute capacitance values (but not the value of β), thus allowing a fair comparison with other encoding techniques (which require relatively high-capacitance buses to amortize the cost of the codecs). The non-minimum-pitch spacing between lines allows to mitigate the effect of crosstalk noise.

Columns *Orig* and *PB* report the energy dissipated by a 32-bit bus when transmitting the unencoded and the PB encoded address traces, respectively. For comparison reasons, the table also reports energy consumption obtained with Gray; values include the cost of the codec, synthesized with Synopsys DesignCompiler on the usual $0.25\mu m$ technology and estimated with the original streams using Synopsys DesignPower.

Stream	*Orig*	*PB*		*Gray*	
	E [nJ]	E [nJ]	Δ [%]	E [nJ]	Δ [%]
`AdaptFilter`	3515	2704	23.0	2545	27.6
`Butterfly`	3107	2373	23.6	2206	29.0
`DCT`	127	93	26.6	79	37.8
`DashBoard`	1964	1472	25.0	1377	29.9
`FFT`	271	204	24.5	187	31.0
`IirDemo`	2108	1631	22.6	1596	24.3
`Integrator`	1489	1165	21.7	1100	26.1
`MatMult`	290	205	29.1	208	28.2
`Count`	2146	1456	32.1	1155	46.2
`CountSkip`	262	178	32.1	142	45.8
Average			26.0		32.5

Table 2: Energy Results for PB and Gray.

PB encoding achieves an average energy decrease of 26% with respect to the unencoded stream, while Gray reduces energy, on average, by 32.5%. Although PB savings are slightly worse, we emphasize that they are only due to minimization of crosstalk energy, while Gray savings come mainly from a reduction of the number of bus transitions (which, as a by-product, also reduces crosstalk energy). This indicates that the usage of PB does not exclude a preliminary stage of Gray encoding; in the next subsection, experimental data will confirm that PB can be combined with other codecs to sinergically minimize the total bus energy.

We observe also that Gray adds a significant amount of logic on the critical path. The sum of the delay introduced by the Gray codec is $6.2ns$, confirming the fact that the usage of codes that require complex interface circuitry is constrained to low-performance systems, where some latency penalty on bus transfers can be tolerated.

5.2 Combined Gray+PB Encoding

As mentioned in the previous subsection, the PB code attempts to minimize a cost function (i.e., the number of simultaneous pairwise transitions) which is different from that targeted by codes such as Gray (i.e., total number of transitions). It may thus be wise, whenever the design constraints (e.g., available timing slack or latency on the bus) allow it, applying PB after a preliminary step of conventional (i.e., Gray) encoding.

While this obviously reduces the absolute effectiveness of the PB code (streams encoded with Gray exhibit less transitions than the original traces), it helps in achieving a more sizable bus energy optimization.

The results of Table 3 confirm our claim. In fact, cascading PB to Gray yields, on average, a total energy savings w.r.t. the unencoded stream of 46% (14% more than using Gray alone and 20% more than using PB alone).

Stream	*Orig* E [nJ]	*Gray+PB* E [nJ]	Δ [%]
`AdaptFilter`	3515	2167	38.3
`Butterfly`	3107	1920	38.2
`DCT`	127	60	52.8
`DashBoard`	1964	1097	44.1
`FFT`	271	143	47.2
`IirDemo`	2108	1342	36.3
`Integrator`	1489	941	36.8
`MatMult`	290	162	44.1
`Count`	2146	810	62.3
`CountSkip`	262	99	62.2
Average			46.2

Table 3: Energy Results for Combined Gray+PB.

5.3 General-Purpose vs. Custom PB Encoding

It was mentioned in Section 5.1 that the permutation used by PB for the experiments referred to the correlation matrix calculated as the average of all the matrices corresponding to the various benchmark programs.

If, for each program, a specific permutation is computed starting from the corresponding correlation matrix, we obtain encoding and decoding functions which are optimal with respect to the considered program, and thus produce a more relevant minimization of the crosstalk energy. Draw-back of this approach is that code profiling of each application is required, and a custom codec must be synthesized, thus making PB an application-specific code.

We have run an experiment to evaluate how much potential energy savings is lost when the general-purpose permutation network (computed as explained in Section 5.1) is used instead of a trace-specific encoder for each individual stream; Table 4 shows the results.

Stream	*General PB* E [nJ]	*Custom PB* E [nJ]	Δ [%]
`AdaptFilter`	2704	2663	1.5
`Butterfly`	2373	2255	5.0
`DCT`	93	88	5.4
`DashBoard`	1472	1415	3.9
`FFT`	204	194	4.9
`IirDemo`	1631	1601	1.8
`Integrator`	1165	1145	1.7
`MatMult`	205	186	9.3
`Count`	1456	1455	0.1
`CountSkip`	178	177	0.6
Average			3.4

Table 4: Energy Results with General and Custom PB.

We note that address streams have a significant degree of uniformity in their statistics: The energy savings using the encoder for the "average" case (column *General PB*) cause only a marginal degradation (3.4% on average) of the savings with respect to the custom case (column *Custom PB*), where each stream would have a specific encoder. This results confirm the claim that PB can be considered a general-purpose code, whose applicability does not require the knowledge of the address streams being transfered.

5.4 Boundary Capacitance Effect

In this section, we quantify the impact of the neighbor wires, as discussed in Section 4.2, on the reduction of the energy savings versus the value of β in Equation 2.

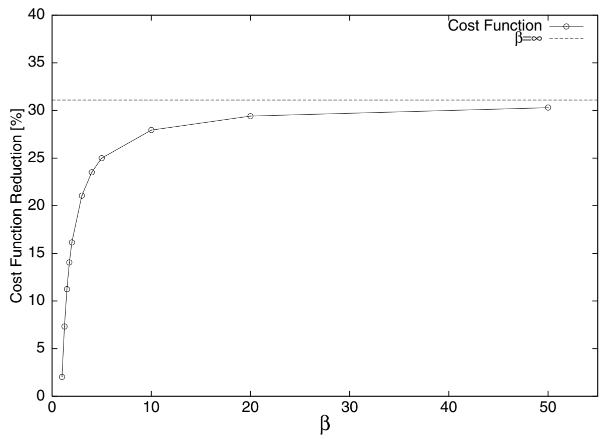

Figure 5: Effect of Boundary Capacitances.

Figure 5 reports the crosstalk energy savings averaged over all streams. On the x-axis, the value of β is reported. The horizontal line corresponds to the case of $\beta = \infty$ (i.e., the results of Table 4). From the curve, we note that for realistic values of β (i.e., between 5 and 10) the reduction in achievable savings is relatively limited, and it is still above 20% for the case of $\beta = 3$.

5.5 Data Streams

We have applied the permutation-based encoding to a set of data traces representing various types of information, with the objective of analyzing the applicability of the code to data buses. The traces we used include ASCII files, binary files and multimedia data (music, images, etc.), and the results are summarized in Table 5 for the cases of both general and custom PB.

Stream	*Orig* E [nJ]	*General PB* E [nJ]	Δ [%]	*Custom PB* E [nJ]	Δ [%]
`Ascii`	5055	4290	15.1	3851	31.3
`Bison`	4996	5020	-0.5	4705	6.2
`Flex`	6305	6210	1.5	5979	5.5
`gcc`	1752	1695	3.3	1541	13.7
`HTML`	3078	2860	7.1	2726	12.9
`M31`	10561	10513	0.5	9880	6.9
`MP3`	4401	4341	1.4	4187	5.1
`PS`	4221	3693	12.5	3493	20.8
`Screen`	2459	2226	9.5	1648	49.2
`Sound`	174410	174280	0.1	169760	2.7
Average			5.0		15.4

Table 5: Energy Results for Data Streams.

From the results in column *General PB* we observe that the general-purpose version of PB is effective only for streams that exhibit explicit non-randomness (e.g., text files, binaries, and bitmaps), while it provides marginal savings for streams containing information in compressed format (e.g., images, sound files). The intuitive explanation for this is that compression tends to flatten the statistical distribution of the bits towards the random case.

Due to this non-uniform statistical characteristics of the data streams, the custom version of the PB code is much more effective than the general purpose one, as shown in column *Custom PB* of the table.

6 CONCLUSIONS

We have presented a permutation-based (PB) encoding scheme that explicitly targets the minimization of energy dissipated due to the switching of crosstalk capacitances. Code calculation does not require the knowledge of the binary stream being transmitted on the bus, and its implementation does not require explicit encoding/decoding circuitry.

The method is particularly effective for address buses: Savings obtained for different address traces generated by two different processors are in the order of 26% with respect to the unencoded streams. Our scheme is orthogonal with respect to conventional encodings that target the energy dissipated due to the switching of self capacitances. Therefore, it can be combined with such encodings (e.g., Gray), yielding average energy savings around 46%.

Applicability to data buses is more dependent on the type of data that must be transferred; in this case, using an application-specific variant of the code is more appropriate.

References

[1] M. R. Stan, W. P. Burleson, "Bus-Invert Coding for Low-Power I/O," *IEEE Trans. on VLSI*, Vol. 3, No. 1, pp. 49-58, Mar. 1995.

[2] M. R. Stan, W. P. Burleson, "Low-Power Encodings for Global Communication in CMOS VLSI," *IEEE Trans. on VLSI*, Vol. 5 No. 4, pp. 444-455, Dec. 1997.

[3] Y. Shin, S.-I. Chae, K. Choi, "Partial Bus-Invert Coding for Power Optimization of System Level Bus," *ISLPED-98*, pp. 127-129, Aug. 1998.

[4] L. Benini, A. Macii, E. Macii, M. Poncino, R. Scarsi, "Architectures and Synthesis Algorithms for Power-Efficient Bus Interfaces," *IEEE Trans. on CAD*, Vol. 19 No. 9, pp. 969-980, Sept. 2000.

[5] C. L. Su, C. Y. Tsui, A. M. Despain, "Saving Power in the Control Path of Embedded Processors," *IEEE Design and Test of Computers*, Vol. 11, No. 4, pp. 24-30, Winter 1994.

[6] L. Benini, G. De Micheli, E. Macii, D. Sciuto, C. Silvano, "Asymptotic Zero-Transition Activity Encoding for Address Buses in Low-Power Microprocessor-Based Systems," *GLS-VLSI-97*, pp. 77-82, Mar. 1997.

[7] H. Mehta, R. M. Owens, M. J. Irwin, "Some Issues in Gray Code Addressing," *GLS-VLSI-96*, pp. 178-180, Mar. 1996.

[8] L. Benini, G. De Micheli, E. Macii, D. Sciuto, C. Silvano, "Address Bus Encoding Techniques for System-Level Power Optimization," *DATE-98*, pp. 861-866, Feb. 1998.

[9] L. Benini G. De Micheli, E. Macii, M. Poncino, S. Quer, "Reducing Power Consumption of Core-Based Systems By Address Bus Encoding," *IEEE Trans. on VLSI*, Vol. 6, No. 4, pp. 554-562, Dec. 1998.

[10] E. Musoll, T. Lang, J. Cortadella, "Working-Zone Encoding for Reducing the Energy in Microprocessor Address Buses," *IEEE Trans. on VLSI*, Vol. 6, No. 4, pp. 568-572, Dec. 1998.

[11] S. Ramprasad, N. R. Shanbhag, I. N. Hajj, "A Coding Framework for Low Power Address and Data Busses," *IEEE Trans. on VLSI*, Vol. 7, No. 2, pp. 212-221, Jun. 1999.

[12] S. Ramprasad, N. R. Shanbhag, I. N. Hajj, "Signal Coding for Low Power: Fundamental Limits and Practical Realizations," *IEEE Trans. on CAS II*, Vol. 46, No. 2, pp. 923-929, Jul. 1999.

[13] P.P. Sotiriadis, A. Chandrakasan, "Bus Energy Minimization by Transition Pattern Coding (TPC) in Deep SubMicron Technologies," *ICCAD'00*, pp. 322-327, Nov. 2000.

[14] K.-W. Kim, K.-H. Baek, N. Shanbag, C.L. Liu, S.-M. Kang, "Coupling-Driven Signal Encoding Scheme for Low-Power Interface Design", *ICCAD'00*, pp. 317-321, Nov. 2000.

[15] C. H. Papadimitriou, *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, 1982.

[16] S. Lin, "Computer Solution of the Traveling Salesman Problem," *Bell Systems Technical Journal*, Vol. 44, No. 10, pp. 2245-2269, Dec. 1965.