Combining Wire Swapping and Spacing for Low-Power Deep-Submicron Buses [*]

Enrico Macii
Politecnico di Torino
Torino, ITALY 10129
enrico.macii@polito.it

Massimo Poncino
Università di Verona
Verona, ITALY 37134
massimo.poncino@univr.it

Sabino Salerno
Politecnico di Torino
Torino, ITALY 10129
sabino.salerno@polito.it

ABSTRACT

We propose an approach for reducing the energy consumption of address buses that targets both the switching and the crosstalk components of power dissipation.

The method is based on the combined application of two techniques. First, selective wire swapping is applied in such a way that bus wires with high coupling activity are kept far away from each other. Then, the slack available in the floorplanning for the routing of the bus wire is exploited to realize a bus with non-uniform inter-wire spacing. Both swapping and placement are driven by the switching data obtained from the analysis of typical address bus traces, and can be successfully applied to any address bus.

Results on a set of profiled address streams show the effectiveness of the proposed approach.

Categories and Subject Descriptors

J.6 [**Computer Applications**]: Computer-Aided Engineering; B.4 [**Hardware**]: Input/Output and Data Communication; C.3 [**Computer Systems Organization**]: Special-Purpose and Application-Based Systems

Keywords

Physical Design, Bus Encoding, Low-Power Design, Crosstalk.

General Terms

Algorithms, Design

1. INTRODUCTION

As technology scales, interconnects have an increasingly larger impact on the overall delay and power consumption of a design; this is mainly due to (i) the relative scaling of cell capacitances, and (ii) the increase of inter-wire, or *coupling* capacitances.

The latter effect, in particular, becomes relevant in in technologies below $0.25\mu m$, in which coupling capacitances between adjacent wires are significantly larger than the capacitances between a wire and the substrate (the *self* capacitance). Such coupling effect manifests itself mostly as *crosstalk* delay, in which transitions to opposite values on adjacent wires will exhibit longer delays than other types of transitions. Increased coupling capacitances are indeed critical for power consumption as well, because "critical" transitions will cause these capacitances to switch.

Coupling effects are particularly critical in long, cross-chip buses, because of the large capacitances due to their length, and also because conventional routing algorithms tend to keep bus wires close together, thus increasing the number of adjacent wires.

While the impact of crosstalk on delay has been mainly regarded as a technological problem, its impact on power has drawn the attention of the low-power community, that has leveraged the vast literature on low-power bus encoding to devise new techniques that incorporate coupling-driven metrics into the bus power optimization problem. Several recent contributions have provided promising results in this direction. All these approaches tackle crosstalk bus power by minimizing the number of simultaneous transitions on adjacent bus lines, either by modifying the data transmitted on the bus [1, 2, 3, 4] through explicit hardware encoders, or by swapping some of the bus lines during the layout step [5, 6]. The latter solution has the significant advantage of requiring a marginal amount of logic for encoding/decoding with respect to solutions based on encoding of the data.

However, all these schemes do not actually try to solve the real problem: they reduce crosstalk *power*, but do not consider crosstalk *by itself.* The main reason why crosstalk makes deep-submicron design hard is because it affects signal integrity. Thus, even a sensible reduction of crosstalk power is marginal to a designer, if it does not guarantee the proper functionality of the design.

In other terms, in the context of buses, the minimization of crosstalk power should be achieved as a side effect of the minimization of crosstalk capacitances.

Two are the options to reduce coupling capacitances between any two wires. The most intuitive one consists of increasing the space between them, which tends to reduce crosstalk at the source. Another approach consists of *shielding* the wires through the insertion of "neutral lines" (V_{dd} or ground). One such solution is presented in [7]. The lat-

[*]This work was supported, in part, by the European Commission under projects IST-2000-30125 (POET) and IST-2000-30093 (EASY).

ter solution requires a considerable area overhead, since the layout grid imposes that wires must be kept to a minimum distance. Shielding a N-lines bus implies adding $N-1$ interleaved wires, which roughly doubles the layout area.

In this work, we present a novel low-power bus design technique that considers both switching and coupling power. In particular our technique combines bus spacing as a way to reduce *coupling capacitance*, and wire swapping as a low-overhead way to further reduce the *coupling activity*. By concurrently taking into account both factors that determine power (capacitance and activity) our technique yields significant savings. The reduction of coupling capacitance, in particular, intrinsically reduces crosstalk effects on delay and signal integrity, which are the main objective.

The proposed bus design technique is based on an initial wire selection algorithm that determines the optimal wire ordering, corresponding to a permutation of the initial bus lines. The actual placement of the bus lines is then carried out by non-uniformly spacing the bus lines, according to a given available, user-defined width slack, and to bus activity information, obtained from the profiling of a set of typical traces, as done in [8].

We focus on address buses because they allow some predictability of the bus access patterns, without sacrificing the generality of the method, as shown in [5].

The proposed technique leverages accurate capacitive models that considers all electrical effects in order to express the dependency of both the coupling and the self capacitance with respect to the inter-wire distance. Results show energy savings of 48.6% on average (53.2% maximum) with marginal performance overhead or complexity of the design.

2. CROSSTALK EFFECTS AND POWER

Crosstalk is mainly a capacitive effect, and can be summarized as a high *coupling capacitance* between wires (Figure 1. In deep-submicron technologies, such coupling capacitance (C_X) exceeds the capacitance between the individual wire and ground (*self capacitance*, C_L). Electrical-level simulations for $0.18\mu m$ technologies have shown that C_X can even be three times larger than C_L [1].

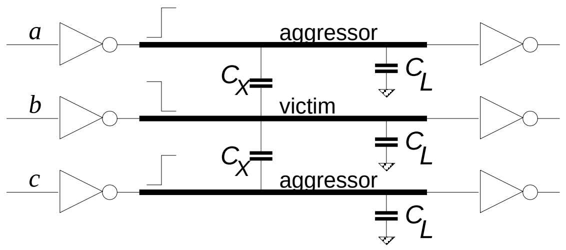

Figure 1: Crosstalk on a 3-Line Bus.

As an example, consider the situation of a three-line bus depicted in Figure 1, where the two outer wires (a and c) exhibit a rise transition, and wire b exhibits a falling transition. All transitions are simultaneous, because we assume that all the bus drivers are clocked. Wires a and c (the *aggressors*) will cause the transition on wire b (the *victim*) to become non-ideal. The effect of the large coupling capacitance between a and b, and c and b will be that of (i) delaying the transition on b, and (ii) causing an initial negative spike at b.

When considering power consumption, we notice that the three transitions on the bus lines will cause a total capacitance of $3 \cdot C_L + 2 \cdot C_X$ to switch. Assuming, for instance, that $C_X = 3C_L$, the total switched capacitance ($9 \cdot C_L$) is precisely three times the capacitance that would switch if we neglect coupling.

Our technique targets the reduction of bus power by using the slack made available by the designer to reduce crosstalk to an acceptable value. Before going into the details of the design methodology, we need to (i) devise an energy model, and (ii) characterize the dependency of coupling capacitance on the distance between a wire pair. These issues are discussed in the two subsections.

2.1 Energy Model

The energy model per cycle of a bus must take into account the effects due to both coupling and self capacitance. For a N-line bus, the total energy is given by:

$$E_{bus} = N \cdot (\alpha_L C_L + \alpha_X C_X) V_{dd}^2, \qquad (1)$$

where α_L and α_X denote the rates at which each capacitance is switched. While α_L represents the conventional switching activity of the lines, α_X is related to the simultaneous switching of two adjacent lines. When the transitions on two adjacent lines a and b are aligned in time, there are only two transition pairs between a and b that cause C_X to switch: (i) When both a and b switch to different final values, and (ii) when one of the two lines switches, while the other does not, and their final values are different.

2.2 Capacitance Model

The energy minimization paradigm we adopt is based on spacing the wires, therefore the energy model should be parameterized with respect to the distance of adjacent wires. We therefore decided to set up a model which considers the effect of spacing on capacitance values and, at the same time, has an analytical expression which is simple enough to be employed inside an optimization engine. In the following, we refer to a specific technology, but a similar approach can be extended to other technologies as well. We first obtained accurate values of capacitances through the use of a well-known 3D capacitance extractor (FastCap from MIT [9]), on a 0.25 micron technology. The details of the extraction are discussed in [8]. The values for the coupling capacitance of the middle wire (the plus marks in Figure 2) exhibit the well-known inverse proportionality relation. However, due to the non-uniformity of fringe effects, and we resorted to a different functional dependency obtained by interpolation of the data (solid curve).

More interesting is the behavior of the self-capacitance (the crosses in Figure 2). We notice that spacing the wires increases self-capacitance, and the effect is far from being negligible: capacitance more than doubles in the distance range considered. The physical explanation for this behavior lies in that by increasing the distance between wires, a larger part of the wire is electrically closer to the ground plane than to the adjacent wires, thus increasing the self-capacitance. Another interesting consideration concerns the effect of the wire distance on total capacitance. The third curve in the figure represents the total capacitance, obtained by summing the self-capacitance with twice the coupling capacitance. We notice that monotonically decreasing behavior of coupling dominates.

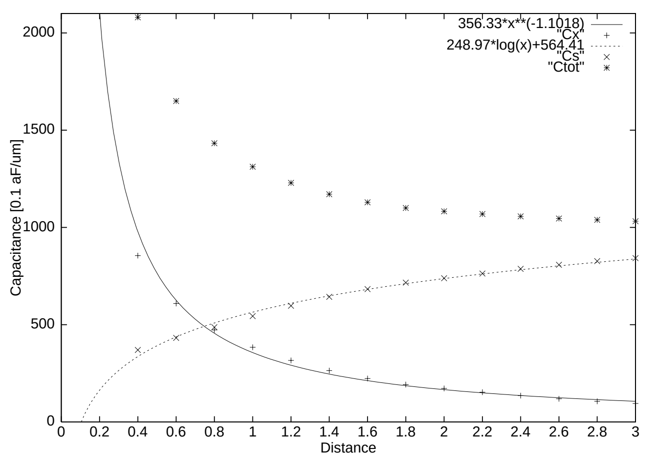

Figure 2: Crosstalk and Self Capacitance vs. Wire Distance.

In the case of asymmetrically place wires (i.e., whose distance from right and left adjacent wire are different), self capacitances for a given line can be calculated by averaging the contributions due to the left and the right wire. This greatly simplifies the modeling of asymmetrical buses, because it allows to express self-capacitance as a function of only one parameter.

3. BUS DESIGN METHODOLOGY

The proposed methodology consists of two main phases.

In the first phase, a characterization is carried out to extract the relevant statistics of the bus. To ensure generality, the bus access patterns are made of several address traces relative to various applications profiled on a set of distinct ISAs. In this way, the extracted values are not bound to a specific application or architecture, and represent switching characteristics of general validity.

In the second phase, the bus placement is carried out using our combined wire swapping and spacing algorithm. It is worth emphasizing that the algorithm does not simply consists of the cascaded application of wire swapping and wire spacing. Rather, our algorithm tries to get on optimal placement by concurrently determining the position of a wire and its distance from its neighbors.

3.1 Concurrent Wire Swapping and Spacing Algorithm

Before discussing the algorithm, we first introduce some notation for the relevant quantities used in the sequel. Let W denote the width of the die area that is available for the placement of the bus.

Since the proposed technique is based on non-uniformly spaced bus lines, we need a representation for how the available width W is allocated. One suitable representation is an array of distances $\mathbf{D} = [d_0, \ldots, d_N]$; the elements $d_i, 1 \leq i \leq N-1$ identify the distance between wire i and $i+1$. The elements d_0 and d_N denote the distance between the boundary wires of the bus (line 1 and N), and the neighbor wires. Obviously, $\sum_{i=0}^{N} d_i \equiv W$. All the d_i's are integer multiples of a technology-dependent quantity s, that represents the step of the layout grid. In the following, without loss of generality, we will assume $s = 1$. Values of d_i range

```
1 SwapAndSpace ($\mathbf{C}$,$\mathbf{T}$,$\mathbf{D}$,$\mathbf{S}$, $\sigma$) {
2 $\mathbf{D}$' = AssignSlack($\mathbf{D}$,$\sigma$);
3 $MinCost$ = Cost($\mathbf{D}$',$\mathbf{C}$,$\mathbf{T}$, $\mathbf{S}$);
4 for ($i = 0$ to $N$) {
5    Place line $i$ and update $\mathbf{S}$;
6    $l = r = i$;
7    while (there are wires still to be placed) {
8       j = PickMinCouplingWire($\mathbf{C}$,$l$);
9       k = PickMinCouplingWire($\mathbf{C}$,$r$);
10      if ($\mathbf{C}[l,j] \leq \mathbf{C}[r,k]$) {
11         $\mathbf{D}$ = setInterwireDistance($\mathbf{D}$,$l$,$j$,$thres$)
12         Update $\mathbf{S}$;
13         $l = j$;
14      } else {
15         $\mathbf{D}$ = setInterwireDistance($\mathbf{D}$,$r$,$k$,$thres$)
16         Update $\mathbf{S}$;
17         $r = k$;
18      }
19   }
20   if (Cost($\mathbf{D}$,$\mathbf{C}$,$\mathbf{T}$, $\mathbf{S}$) < $MinCost$){
21      $MinCost$ = Cost($\mathbf{D}$,$\mathbf{C}$,$\mathbf{T}$, $\mathbf{S}$);
22      $\mathbf{D}_{opt} = \mathbf{D}$;
23      $\mathbf{S}_{opt} = \mathbf{S}$;
24   }
25 }
}

27   setInterwireDistance($\mathbf{D}$,$a$,$b$,$thres$) {
28   Set initial distance $d_{ab}$ between $a$ and $b$ to $d_{min}$;
29   do {
30      $d_{ab}++$;
31      Update $\mathbf{D}$;
32      } while ($d_{ab} < d_{max}$ and LocalCost$(a,b)$ >$thres$ )
33   return ($\mathbf{D}$);
}
```

Figure 3: Pseudocode of the Combined Swapping and Spacing Algorithm.

between a lower and an upper bound. The lower bound is determined by the amount of tolerated crosstalk. Let this quantity be d_{min}. The upper bound is given by the case in which all the wires have minimum inter-distance but one. This value is $d_{max} = (W - N \cdot d_{min})$. The value σ (*slack*) is used to denote the available space to be used for placing wires, and is given by $W - (N+1) \cdot d_{min}$.

Coupling and of the switching activity are represented by a matrix $\mathbf{C} = [c_{00}, \ldots, c_{NN}]$ (the coupling activity between each wire pair, including the neighbor wires), and by an array $\mathbf{T} = [t_0, \ldots, t_{N-1}]$ (the switching activity of each wire), respectively. All the switching values are normalized values between 0 and 1.

The pseudo-code of the SwapAndSpace algorithm is shown in Figure 3. It takes as inputs the switching information $\mathbf{C}$ and $\mathbf{T}$, the distance vector $\mathbf{D}$ the slack σ, and the array $\mathbf{S}$ representing the ordering of the bus lines, used to store the results of the swapping. Initially, all elements of $\mathbf{D}$ are set to d_{min}, and $\mathbf{S} = [1, 2, \cdot, N]$.

First, a distance array $\mathbf{D}$' is computed, corresponding to the distribution allocation of the available slack σ uniformly over all bus lines. $\mathbf{D}$' is used to compute an energy *MinCost* used as an initial value of the cost function (Lines 2-3).

The algorithm is based on a main loop over the bus wires (Lines 4–24), in which a given wire i is used to start a new bus configuration. The various iterations of this loop correspond to starting the bus placement from a different line.

The inner loop of Lines 7–19 performs the basic move of the algorithm. As the algorithm proceeds, the bus lines are split into two sets: those already placed and those that are not. The first set defines the the current configuration of the bus. At each iteration, the outer lines of currently placed bus l (left) and r (right) are stored. Then, two lines are picked from the set of "free" lines: j, the one having minimum coupling with l, and k, the one having minimum coupling with r (Lines 8–9). The line that has the smallest coupling between j or k is chosen for placement (Lines 11-13 or Lines 15–17). The choice on how much distance will be used between this newly placed line and the bus is carried out by the function `setInterwireDistance`, that is described later on.

Once all lines have been placed, the cost of this new bus configuration is evaluated and compared to the currently minimum (Lines 20-23). The algorithm returns the bus distance configuration **D**, the energy cost, and the bus configuration **S**, that is, the new position of the wires resulting from the placement.

The main **for** loop implements a greedy choice of the *nearest neighbor* heuristic used to solve instances of the traveling salesman problem (TSP). This is consistent with the formulation of the swapping problem as a TSP instance, as first observed in [5]. The configuration built by this main loop incorporates the assignment of the wire distances.
Procedure `setInterwireDistance` (Lines 27–33) shows the basic idea behind distance assignment. The wire placement is done in non-decreasing order of coupling, thus we will initially try to place non-critical wires. Therefore, the distance d_{ab} between the newly placed wire a and one of the bus outer wires b is initially set to d_{min} (Line 28). d_{ab} is then progressively increased, if the condition in the **while** applies (Line 32). The condition includes a trivial check on the upper bound of d_{ab}, and specifies a tunable convergence criterion based on the comparison of the energy cost between a and b (`LocalCost`) and a threshold *thresh*. The latter is determined from a reference bus configuration, namely, the equally-spaced one. In practice, this condition ranks the degree of "criticality" of a and b when they become adjacent.

As the number of placed bus lines increases, in fact, the coupling between the not yet placed lines and the placed one will typically increase. Therefore, in order to use all the available slack, d_{ab} will also increase accordingly. The procedure returns the bus distance configuration **D**, in which a and b are adjacent, and their distance is updated with the value d_{ab} just computed.

4. EXPERIMENTAL RESULTS

To assess the effectiveness of the proposed bus design technique, we have considered address traces of some typical embedded applications, collected using two different instruction set simulators (namely, `Armulator` for an ARM platform, and `pixie` for a MIPS platform). In addition, we have considered two traces that represent typical corner cases of address traces (i.e., highly sequential streams): `Counter` represents a perfect counter that starts from a random address, whereas `CountSkip` is a counter sequence intermixed with random jumps to new locations. The characterization procedure to extract the switching information used to drive the algorithms has been described in Section 3.

Tables 1 and 2 shows the results obtained for values of the slack of 20% and 30%, respectively. We have used a value of $d_{min} = 0.4\mu m$, corresponding to relatively high tolerated level of crosstalk. The tables compare our results (Column *Swap w. Spacing*) with a straightforward uniform spacing (Column *Uniform*), and the method of [8] (Column *Non-Uniform*). All the energy savings are relative to the case of a bus with all wires placed at d_{min} (Column *Min. Dist*).

The results show significant savings with respect to the other methods. In particular, the proposed algorithm reduces energy of a 49.5% average for the case with $\sigma = 20\%$, while the non-uniform scheme only achieves a 32.4% saving.

Notice that, for an increased slack, the difference between the two algorithms narrows, since a larger slack emphasizes the spacing operation with respect to swapping.

4.1 Encoding Overhead

The only overhead required by our technique is relative to the swapping of the bus wires. In principle, the implementation of a set of wire swaps does not require any logic. In practice, however, it will have some impact at the physical level.

A detailed discussion on the implementation of the permutation network is described in [5]. The results of their analysis still apply to our scheme, in spite of the fact that our permutation is done between wires with variable interdistance. As a matter of fact, they show that the overhead caused by wire swapping is mainly "vertical" (i.e., the number of layers crossed and the number of vias used), and does not really depend on what wires are swapped (and thus, on their distance).

To summarize the results of [5], it was shown that the delay of the permutation network is about $5ps$, less than 5% of the delay of a 1mm-long bus. Power overhead is even smaller ($< 1\%$).

It is also important to emphasize that the routing overhead caused by the permutation network (occupation of other levels of metal and vias) appears to be confined to relatively small areas close to the components, and has thus small impact to the global routing.

5. CONCLUSIONS

We have presented a novel design technique for address buses that reduces the power consumption due to both coupling and self capacitances. Unlike previous approaches, in our method the reduction of crosstalk effects on delay and on signal integrity is the primary objective, consistently to common physical design practice.

Power reduction is obtained thanks to two transformations: First, as a by-product of our crosstalk-aware bus wire placement technique. Second, by a further encoding of the bus lines based on wire swapping, which provides significant saving with the least possible overhead.

This two-fold optimization is achieved with an algorithm that concurrently determines the placement of the bus lines, and their relative spacing, based on pre-characterized switching information.

Results show that the proposed technique provides higher power savings than previous works, with a marginal routing overhead, and by reducing the amount of crosstalk to a desired level.

Trace	*Min. Dist*	*Uniform*	Δ	*Non-Uniform*	Δ	*Swap w. Spacing*	Δ
AdaptFilt	1119.4	978.4	18.3	851.4	28.9	661.8	44.7
Bfly	1059.6	864.9	18.4	754.5	28.8	570.9	46.1
Counter	802.1	610.9	23.8	498.1	37.9	367.8	54.1
CountSkip	98.0	74.7	23.8	60.9	37.9	44.9	54.1
DashBoard	754.3	563.2	25.3	510.5	32.3	375.8	50.2
DCT	51.7	37.1	28.3	33.9	34.5	23.7	54.2
FFT	107.6	78.5	27.1	71.0	34.0	52.2	51.5
IIR	715.6	586.2	18.1	508.7	28.9	396.4	44.6
Integr	505.0	414.1	18	364.7	27.8	290.9	42.4
MM	113.5	83.5	26.4	75.7	33.2	53.1	53.2
Average			22.7		32.4		49.5

Table 1: Energy Results Comparison ($\sigma = 20\%, d_{min} = 0.4$).

Trace	*Min. Dist*	*Uniform*	Δ	*Non-Uniform*	Δ	*Swap w. Spacing*	Δ
AdaptFilt	1119.4	921.2	23.1	819.6	31.5	660.2	41.0
Bfly	1059.6	813.5	23.2	705.4	33.4	568.3	46.4
Counter	802.1	583.1	27.3	449.0	44.0	392.6	51.1
CountSkip	98.1	71.3	27.3	57.8	41.0	48.0	51.1
DashBoard	754.3	542.7	28.1	496.3	34.2	388.5	48.5
DCT	51.7	36.5	29.4	32.9	36.4	25.2	51.3
FFT	107.6	76.6	28.8	68.5	36.4	54.9	49.0
IIR	715.6	552.0	22.9	509.2	28.8	394.7	44.9
Integr	505.0	389.4	22.9	347.1	31.3	284.3	43.7
MM	113.5	81.2	28.5	70.6	37.8	55.8	50.8
Average			26.1		35.5		47.8

Table 2: Energy Results Comparison ($\sigma = 30\%, d_{min} = 0.4$).

6. REFERENCES

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

[2] J. Henkel, H. Lekatsas, "A^{2}BC: Adaptive Address Bus Coding for Low-Power Deep Sub-Micron Designs," *38th DAC*, pp. 744-749, June 2001.

[3] 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.

[4] B. Victor, K. Keutzer, "Bus Encoding to Prevent Crosstalk Delay", *ICCD'01*, pp. 57–63, Nov. 2001.

[5] L. Macchiarulo, E. Macii, M. Poncino, "Low-Energy Encoding for Deep-Submicron Address Buses," *ISLPED'01*, pp. 176–181, August 2001.

[6] Y. Shin, T. Sakurai, "Coupling-Driven Bus Design for Low-Power Application-Specific Systems", *38th DAC*, pp. 750–753, June 2001.

[7] S.P. Khatri, A. Mehrotra, R.K. Brayton, R.H.J.M. Otten, A. Sangiovanni-Vincentelli "A novel VLSI layout fabric for deep sub-micron applications" *36th DAC*, pp. 491-496, June 1999.

[8] L. Macchiarulo, E. Macii, M. Poncino, "Wire Placement for Crosstalk Energy Minimization in Address Buses," *DATE'02*, pp. 158–162, Mar. 2002.

[9] K. Nahors, J. White, "FastCap: a Multi-pole Accelerated 3-D Capacitance Extraction Program", *IEEE Transactions on CAD*, 10(11), pp. 1447-1459, Nov. 1991.