Coupling-Driven Bus Design for Low-Power Application-Specific Systems

Youngsoo Shin and Takayasu Sakurai
Center for Collaborative Research and Institute of Industrial Science
University of Tokyo, Tokyo 153-8505, Japan
{ysshin, tsakurai}@iis.u-tokyo.ac.jp

ABSTRACT

In modern embedded systems including communication and multimedia applications, large fraction of power is consumed during memory access and data transfer. Thus, buses should be designed and optimized to consume reasonable power while delivering sufficient performance. In this paper, we address a bus ordering problem for low-power application-specific systems. A heuristic algorithm is proposed to determine the order in a way that effective lateral component of capacitance is reduced, thereby reducing the power consumed by buses. Experimental results for various examples indicate that the average power saving from 30% to 46.7% depending on capacitance components can be obtained without any circuit overhead.

1. INTRODUCTION

As the scale of process technologies steadily shrinks and the size of designs increases, interconnects have increasing impact on the area, delay, and power consumption of circuits [1]. Specifically, reduction in scale causes the lateral component of capacitance to dominate the total capacitance of interconnects. This is because wire-to-wire spacing is shrinking for higher densities and the aspect ratios of interconnects have to be increased to compensate for increasing interconnect resistance, which in turn is due to shrinking wire widths. For example of metal 3 layer in typical 0.35 μm CMOS process, the lateral component of capacitance reaches 5 times the sum of fringing and vertical components when the substrate serves as a bottom plane.

In the domain of embedded systems, an increasing fraction of implementations make use of core processors as basic computational units. In these systems, especially in communication and multimedia applications, large fraction of power is consumed during memory access and data transfer. Thus, *system bus*, which is an essential system component to interconnect subsystems for data transfer, should be designed and optimized to consume reasonable power while providing sufficient performance. Although there has been significant work devoted to reduce power consumption of off-chip buses with coding techniques [2], [3], [4], [5], [6], [7], the overhead of coding logic in terms of delay, area, and power cannot be tolerated if the same techniques are to be used for on-chip buses [8]. Furthermore, the effects from lateral component of capacitance should be taken into account when on-chip buses in deep submicron technologies are of concern.

In this paper, we propose a low-power on-chip bus design technique for embedded application-specific systems, which takes the lateral as well as vertical and fringing components of capacitance into consideration. Specifically, we are given a processor core and the embedded application that runs on it. We assume that capacitance components of buses are available from the layout of the processor core and the address streams from typical runs of the application code. The sequence of the address patterns can be available a priori after the algorithm of an application is specified such as in signal and image processing applications. Based on the capacitance components and the address streams, we determine the optimal order of the bus in such a way that power consumption of the bus is minimized. The rationale for ordering is that the effective lateral capacitance is reduced if bus lines, with high probability of switching in the same direction, are located adjacent to each other. The obtained order can be used to modify the processor layout without any circuit overhead.

We present the power model, which incorporates all the capacitance components, and define a bus ordering problem. Then, we propose a heuristic algorithm to determine the bus order. Note that the optimal order can be obtained only if the bus is narrow and the length of address streams is small, which are not the case for most of embedded applications. We also evaluate the proposed heuristic algorithm by comparing it with the simulated annealing algorithm.

The remainder of the paper is organized as follows. In the next section, we present a power model and the definition of a problem followed by a heuristic algorithm. In Section 3, we present results of experiments for several examples, and in Section 4 we draw conclusions.

2. COUPLING-DRIVEN BUS DESIGN

2.1 Power Model and Problem Definition

We are given a bus $B=(b_0, b_1, \ldots, b_{n-1})$, which transfers a sequence of patterns $B_i = (b_0^i, b_1^i, \ldots, b_{n-1}^i)$, where i is the time index, n is the bus width, and b_j^i is the value of a bus line b_j at time i. We shuffle bus lines in B such that effective load capacitances seen from driving ends are minimized. Note that if load capacitances are constant, which is the case when there are only vertical capacitance components, the shuffling has no effects with respect to dynamic power consumption, which is a dominant source of power dissipation in a digital CMOS circuit. However, because of lateral capacitance components, the load capacitances are not constant but depend on signal transitions of neighboring wires.

DAC2001, June 18-22, 2001, Las Vegas, Nevada, USA.

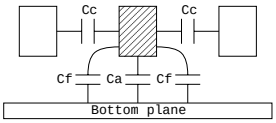

Figure 1: Parasitic capacitances with adjacent lines.

Figure 1 shows parasitic capacitances involved with adjacent bus lines. C_c denotes a lateral component between electromagnetically coupled lines. C_a and C_f denote the vertical and fringing components, respectively, between a metal line and a bottom plane. We denote the sum of C_a and $2C_f$ as C_l. The ratio between C_c and C_l is denoted by

$$\eta = \frac{C_c}{C_l}. \tag{1}$$

The effect of lateral capacitance (C_c) is that total load capacitance seen by a gate is no longer a constant value, but depends on signal activities of neighboring lines due to the Miller effect [9]. Assume that the *physical* lateral capacitance between two neighboring lines, b_i and b_{i+1}, is C_c. The Miller effect states that if two lines switch in opposite directions, the *effective* lateral capacitance between them is $2C_c$ because the effective voltage swing between them is doubled. On the contrary, the effective lateral capacitance becomes 0 if both lines switch in the same direction.

In a digital CMOS circuit, the dynamic power is proportional to load capacitance and switching activity. Thus, if load capacitance is constant, power consumption is proportional to the number of transitions. In other words, if we have two sets of patterns with the same width and length to be transferred on the same bus, we can compare the power consumption of the bus from each set by comparing the number of transitions. In order to take a similar approach when we incorporate the effect of C_c as well as C_l, we first define the *switching encoding* for j-th bus line as

$$s_j^i = \begin{cases} 1, & \text{if } b_j^{i-1} = 0 \text{ and } b_j^i = 1 \\ -1, & \text{if } b_j^{i-1} = 1 \text{ and } b_j^i = 0 \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

If $\eta = 0$, we can readily obtain the total number of bus transitions by summing $|s_j^i|$ over j and i. However, if $\eta \neq 0$, we should take switching polarities of adjacent lines into account to include the effect of C_c. For this purpose, we define the *switching similarity* between adjacent lines (j-th and k-th) at time i, denoted by $\zeta_i(j,k)$, as the amount of effects from C_c seen from j-th line (thus, $\zeta_i(k,j)$ is different from $\zeta_i(j,k)$). For example, if two lines make transitions in opposite directions at time i, $\zeta_i(j,k) = 2$. Then, it can be readily shown that $\zeta_i(j,k)$ is given by

$$\zeta_i(j,k) = |s_j^i|(2 - |s_j^i + s_k^i|). \tag{3}$$

Now, we define the *effective bus transition* of b_j at time i, denoted by α_j^i, as the measure of effective transitions induced both from C_l and C_c, which is normalized to transitions considering only C_l. It can be expressed by

$$\alpha_j^i = \begin{cases} |s_j^i|(1+\eta\,\zeta_i(j,j+1)), & \text{if } j = 0 \\ |s_j^i|(1+\eta\,\zeta_i(j,j-1)), & \text{if } j = n-1 \\ |s_j^i|(1+\eta\,(\zeta_i(j,j-1)+\zeta_i(j,j+1))), & \text{otherwise.} \end{cases} \tag{4}$$

Based on the effective bus transitions, our problem of bus ordering can be defined as follows:

Ordering bus lines

Compute transition probability (p_j) of each line;
Compute switching correlation (ρ_{jk}) of each pair of lines;
Find a set of shielding lines $S \leftarrow \{b_j | p_j < \xi\}$;
$R \leftarrow \{b_0, b_1, \ldots, b_{n-1}\} - S$;
$\Psi \leftarrow$ build-clusters(R, p_j, ρ_{jk});
$\{\,\} \leftarrow$ arrange-clusters(Ψ, S);

Figure 2: Heuristic algorithm for bus ordering.

- Given η and a bus B=$(b_0, b_1, \ldots, b_{n-1})$, which transfers a sequence of patterns $B_i = (b_0^i, b_1^i, \ldots, b_{n-1}^i)$,
- Find the shuffled bus $\tilde{B}$ that minimizes the sum of α_j^i over j and i.

2.2 Coupling-Driven Bus Ordering Algorithm

Because of the exponential number of alternatives for $\tilde{B}$ (n! alternatives for n-b wide bus), the optimal one can be obtained only when the width of bus is narrow and the number of patterns is small, which are not the case for most of embedded applications. In this subsection, we propose a heuristic algorithm based on both switching correlation and transition probability. The *switching correlation coefficient* or simply *switching correlation* for two bus lines (j-th and k-th) is defined by

$$\rho_{jk} = \frac{K_{jk}}{\sigma_j \sigma_k}, \tag{5}$$

where σ_j is the standard deviation of s_j, defined in (2), over time i. K_{jk} is the covariance of s_j and s_k and defined by

$$K_{jk} = E\{s_j s_k\} - m_j m_k, \tag{6}$$

where $E\{x\}$ is the expected value of x and m_j is the mean of s_j. Now, if the switching correlation between two lines is close to 1, that means that they have high possibility to have transitions in the same direction. However, switching correlation does not give all the information we need. For example, if the switching activities of two lines are very low, it does not have much effect to make them adjacent even though their switching correlation is close to 1. Thus, we consider transition probability of each line together with switching correlation.

The overall algorithm is outlined in Figure 2. Initially, we group bus lines with relatively low transition probability (below some threshold[1], ξ). These lines serve as *shielding lines* between clusters. The remaining bus lines are subdivided into a group of ordered sets, called *clusters*. The clusters and a set of shielding lines are ordered to result in the final order. Note that orders of lines in each cluster is fixed once each cluster is built, except for the possibility of conditional reverse.

The heuristic algorithm to group bus lines into a set of clusters is shown in Figure 3. For each cluster, we first select the line with the highest transition probability among lines not selected and then build a new cluster. At each iteration of inner **while** loop, we select a line (b_k) that maximizes the switching correlation between the line and the first or the last element of the cluster (recall that cluster is an ordered set) under consideration. This continues until there are no candidate lines having positive switching correlation with the first or the last element of the cluster. In this way, each cluster

[1]The threshold can be selected in various ways. The selection is obvious if there is a jump in the distribution of transition probabilities. If a boundary is not clear, we can iterate the alogirthm outlined in Figure 2 while we vary the threshold, and then select the best case.

build-clusters(R, p_j, ρ_{jk})
while R not empty **do**
Select $b_j \in R$ s.t. p_j is maximum, and $R \leftarrow R - \{b_j\}$;
Form a new cluster $\Psi_i \leftarrow \{b_j\}$;
while true **do**
Find $b_k \in R$ maximizing $\rho_{kl} > 0$, where b_l is the first or the last element of Ψ_i;
If b_k is not found **then** exit loop; **end if**
If b_l is the first element of Ψ_i **then** $\Psi_i \leftarrow \{b_k\} \cup \Psi_i$;
else $\Psi_i \leftarrow \Psi_i \cup \{b_k\}$; **end if**
$R \leftarrow R - \{b_k\}$;
end do
$\Psi \leftarrow \Psi \cup \Psi_i$;
end do
return Ψ;

Figure 3: Heuristic algorithm for clustering.

is formed in such a way that lines with high transition probabilities and high switching correlations are more likely to be grouped together, thereby reducing the effective lateral capacitances. Furthermore, lines located at both ends of each cluster have relatively low transition probabilities that also contributes toward reducing the effective lateral capacitances, which is to be clarified in Figure 4.

arrange-clusters(Ψ, S)
Select $\Psi_l \in \Psi$ s.t. $p(\Psi_l)$ is maximum, and $\Psi \leftarrow \Psi - \Psi_l$;
Select $\Psi_r \in \Psi$ s.t. $p(\Psi_r)$ is maximum, and $\Psi \leftarrow \Psi - \Psi_r$;
if the first element of Ψ_l has $p(\Psi_l)$ **then** $F \leftarrow \Psi_l$;
else $F \leftarrow reverse(\Psi_l)$; **end if**
foreach $\Psi_i \in \Psi$ **do**
$\Psi \leftarrow \Psi - \Psi_i$, and $F \leftarrow F \cup \Psi_i$;
Select $b_i \in S$, $S \leftarrow S - \{b_i\}$, and $F \leftarrow F \cup \{b_i\}$;
end do
if S is not empty **then** $F \leftarrow F \cup S$; **end if**
if the last element of Ψ_r has $p(\Psi_r)$ **then** $F \leftarrow F \cup \Psi_r$;
else $F \leftarrow F \cup reverse(\Psi_r)$; **end if**
return F;

Figure 4: Heuristic algorithm to find the final order. $p(\Psi_i)$ is the maximum of transition probabilities of the first and the last element of Ψ_i. $reverse(\Psi_i)$ reverses the partial order of elements of Ψ_i.

From a set of clusters and a set of shielding lines, the final order of bus lines is determined by the heuristic algorithm as shown in Figure 4. First, we select two clusters (Ψ_l and Ψ_r) to be located at both ends of the final order. Because the lines to be located at both ends of the final order will have only one lateral capacitances, they should be lines with high transition probabilities. Then, the remaining clusters are located sequentially with shielding lines in-between clusters, if there are enough shielding lines to be located. Because clusters are built in a way that any combination of two clusters results in negative switching correlation between lines located in the boundaries of clusters (see Figure 3), locating a shielding line in-between clusters decreases the effective lateral capacitances. The above process is illustrated in the following example.

Example 1 Consider the following example of 8-b patterns, where the left-most column corresponds to a bus line b_7 and the right-most one to b_0.

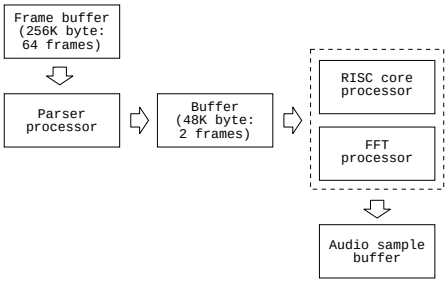

Figure 5: Block diagram of audio decoder.

```
00011100
01110011
00110010
01001101
10000101
```

Assume that b_7 and b_4 are selected as shielding lines. Among the remaining lines, we first select b_6 because it has most transitions. Then, we select b_0 that has the highest positive switching correlation with b_6 (0.9). b_3 is selected and become adjacent to b_6 because $\rho_{36} = 0.3 > \rho_{30} = 0.1$. Now, the partial order for the first cluster is $\{b_0, b_6, b_3\}$ or the reverse of it. b_2 is selected and become adjacent to b_3 because $\rho_{23} = 0.9 > \rho_{20} = 0.0$. Since there are no more lines having positive switching correlation with the first or the last element of the current cluster (b_0 and b_2), the first cluster becomes $\{b_0, b_6, b_3, b_2\}$ or the reverse of it. The second cluster consists of the remaining two lines (b_1 and b_5).

In the first cluster, b_0 has more transitions than b_2 meaning that the partial order ($\{b_0, b_6, b_3, b_2\}$) is maintained if the cluster is located at the left-side of the final order or it is reversed if the cluster is located at the right-side. Suppose that it is located at the left-side, while the second cluster is located at the right-side. Then, the final order from the algorithms is $\{b_0, b_6, b_3, b_2, b_4, b_7, b_1, b_5\}$. □

3. EXPERIMENTAL RESULTS

To evaluate the efficiency of the proposed algorithm, we perform experiments for the following set of sample patterns:

- `wavelet`, `linear`, `laplace`, `compress`, and `lowpass`: data address patterns in benchmark examples collected from typical image or signal processing algorithms [10]. We assume 16-b wide data address buses for all the programs. Patterns are extracted with the help of Shade [11].

- `fft`: 7-b wide data address patterns between 128-point complex *fft processor* of an audio decoder [12], shown in Figure 5, and memory. Patterns are extracted through VHDL simulation.

- `ac3`: 16-b wide data address patterns between memory and *parser processor* of the audio decoder, which reads input data stored in a frame memory. Patterns are extracted through VHDL simulation.

The threshold for shielding lines (ξ) is set to 0.01. We assume that C_c and C_l are constants over all bus lines, and perform experiments with $\eta = 1, 2, 3, 4, 5, \infty$. The resulting percentage saving in power with the proposed heuristic algorithm is shown in Figure 6. Figure 7 corresponds to the result after ordering is done using simu-

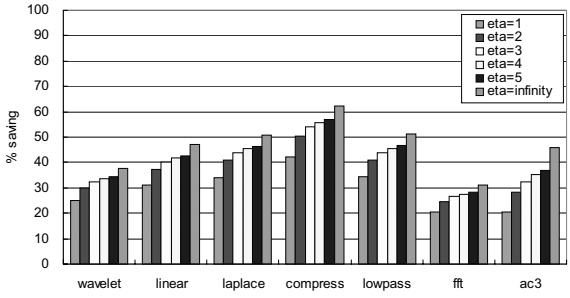

Figure 6: Percentage saving with heuristic algorithm.

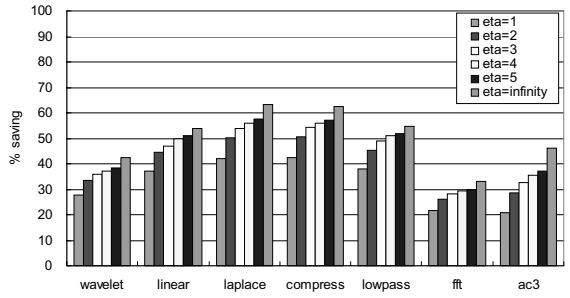

Figure 7: Percentage saving with simulated annealing algorithm.

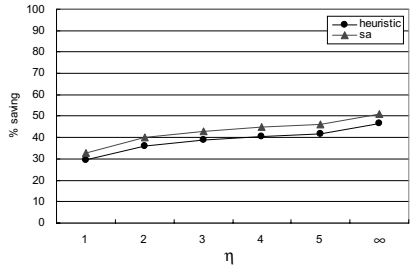

Figure 8: Comparison of heuristic algorithm and simulated annealing.

lated annealing (SA)[2] [13] instead of the heuristic algorithm, which gives an idea of how good the solutions obtained by the proposed heuristic algorithm are. However, SA itself can be used instead of the proposed heuristic algorithm when the bus width and length is of reasonable size[3]. We also obtain the optimal order for `fft`, which is 7-b wide and consists of 782 patterns. Interestingly, the result is the same as that obtained with SA.

The results of the average percentage saving with the heuristic algorithm are compared to those of SA in Figure 8. The heuristic algorithm gives 30% on the average when $\eta = 1$ up to 46.7% when η is infinity. The difference between heuristic algorithm and SA ranges from 1.9% to 4.4%.

[2]Two kinds of moves are used. One is one-to-one exchange between randomly selected two bus lines. Another is group-to-group exchange between randomly selected two groups of bus lines. The move itself is chosen randomly.

[3]Only part of the original patterns are used in the experiments to obtain results with SA in a reasonable time. For example of `linear`, the first 2000 out of 485503 samples are used. Even with that, it takes about 14 m on Ultra 1 with SA, while it takes less than 1 s with the proposed heuristic algorithm.

4. CONCLUSION

In this paper, we address on-chip bus design technique targeting low-power application-specific systems. In the proposed scheme, we shuffle bus lines in order to minimize the number of effective bus transitions, which includes effects from both lateral and vertical capacitance components, thereby minimizing the power consumed by on-chip buses. We present a heuristic algorithm of shuffling bus lines. The proposed scheme is particularly suitable for address buses in memory-intensive application-specific systems. Experimental results show that savings are substantial for benchmark examples and a large example such as an audio decoder. The performance of the proposed heuristic algorithm is compared to that of simulated annealing.

References

[1] M. T. Bohr, "Interconnect scaling – the real limiter to high performance ULSI," in *Proc. IEEE Int'l Electron Devices Meeting*, pp. 241–244, Dec. 1995.

[2] M. R. Stan and W. P. Burleson, "Bus-invert coding for low-power I/O," *IEEE Trans. on VLSI Systems*, vol. 3, pp. 49–58, Mar. 1995.

[3] L. Benini, G. D. Micheli, E. Macii, D. Sciuto, and C. Silvano, "Asymptotic zero-transition activity encoding for address busses in low-power microprocessor-based systems," in *Proc. Great Lakes Symposium on VLSI*, pp. 77–82, Mar. 1997.

[4] L. Benini, G. D. Micheli, E. Macii, M. Poncino, and S. Quer, "System-level power optimization of special purpose applications: The Beach Solution," in *Proc. Int'l Symposium on Low Power Electronics and Design*, pp. 24–29, Aug. 1997.

[5] E. Musoll, T. Lang, and J. Cortadella, "Exploiting the locality of memory references to reduce the address bus energy," in *Proc. Int'l Symposium on Low Power Electronics and Design*, pp. 202–207, Aug. 1997.

[6] S. Ramprasad, N. R. Shanbhag, and I. N. Hajj, "A coding framework for low-power address and data busses," *IEEE Trans. on VLSI Systems*, vol. 7, pp. 212–221, June 1999.

[7] Y. Shin and K. Choi, "Narrow bus encoding for low power systems," in *Proc. Asia South Pacific Design Automat. Conf.*, pp. 217–220, Jan. 2000.

[8] P. Sotiriadis and A. Chandrakasan, "Low power bus coding techniques considering inter-wire capacitances," in *Proc. IEEE Custom Integrated Circuits Conf.*, pp. 507–510, May 2000.

[9] H. B. Bakoglu, *Circuits, Interconnections and Packaging for VLSI*. Addison-Wesley, 1990.

[10] P. Panda and N. Dutt, "1995 high level synthesis design repository," in *Proc. Int'l Symposium on System Synthesis*, 1995.

[11] R. Cmelik and D. Keppel, "Shade: A fast instruction-set simulator for execution profiling," Tech. Rep. TR-93-12, Sun Microsystems Laboratories, 1993.

[12] S. Lee and W. Sung, "A parser processor for MPEG-2 audio and AC-3 decoding," in *Proc. Int'l Symposium on Circuits and Systems*, pp. 2621–2624, June 1997.

[13] S. Kirkpatrick, J. C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 671–680, May 1983.