

Simultaneous Wire Permutation, Inversion, and Spacing with Genetic Algorithm for Energy-Efficient Bus Design

Shanq-Jang Ruan*

National Taiwan University of Science and Technology
Dept. of Electronic Engineering
43, Keelung Rd., Sec. 4, 106 Taipei, Taiwan
sjruan@et.ntust.edu.tw

Edwin Naroska and Uwe Schwiegelshohn

University of Dortmund
Computer Engineering Institute
Otto-Hahn-Str. 4, 44229 Dortmund, Germany
{Edwin.Naroska|Uwe.Schwiegelshohn}@udo.edu

Abstract

With decreasing feature size on silicon, the coupling capacitances of buses grow rapidly causing a significant impact on the power consumption of the whole chip. Thus, buses should be designed and optimized to dissipate less power without sacrificing performance. In this paper, we address this problem by simultaneously optimizing wire permutation, inversion and spacing (space between consecutive wires) using a combination of optimal as well as genetic algorithms. Unlike previous studies, our approach is applicable to not only address buses (behave more regularly), but also instruction buses of microprocessors. For the spacing problem, an algorithm is presented which determines the optimal solution instead of applying time consuming heuristic algorithms as presented in [10].

For our experiments, we used instruction bus traces obtained from 12 SPEC2000 benchmark programs. We simulate different combinations among permutation, spacing, and inversion. Integrated all optimization techniques together, our approach can save energy up to 68% for the best case and 58% on average while only increasing the total wire space by about 50% (compared to a bus with minimal spacing between adjacent wires for a particular technology).

1. Introduction

Nowadays, on-chip interconnect (especially in a SoC) contributes about 15% to 30% total power dissipation in modern microprocessors [6] (up to 50% [12]). As technology scales down to nanometer, coupling power between bus wires is becoming the dominant power dissipation source

on buses. For example, in standard 0.13 μ m technologies with minimum distance between bus wires, the ratio of coupling and wire capacitances (also known as line- or self-capacitances) is greater than eight, and will continue to grow [19]. Therefore, the conventional approaches to optimize switching activity on individual wires might be inappropriate because switching activity does not necessarily reflect the power consumed by the bus [13, 20, 2, 16, 4, 1].

Based on this observation, recent researchers have focused on various encoding schemes that minimize the self, as well as coupling transitions for power reduction [18, 5]. However, these methods are only applicable to address buses. In [8], a coupling-driven bus-invert scheme was proposed. If the coupling effect of the inverted signals is less than that of the original signals, the signals are inverted. [21] separately handled the odd and even bus wires to reduce coupling. However, it did not take self transition into account. In these approaches, the encoder and decoder (codec) always introduced extra power and delay, hence it often offset power savings.

Instead of bus encoding, shuffling bus lines to minimize the number of coupling transition is another choice [17, 11, 9]. These approaches are based on profiling the bus wire activity and are especially well suited for embedded systems, which typically execute a fixed and known set of programs. The advantage of the shuffling approach is that, compared to bus encoding, it causes a relatively small delay and area overhead.

It is worth noticing that all the aforementioned approaches are optimized for equal wire spacing. However, in practical design the bus channel width is adjustable. I.e., depending on the layout of the chip additional space may be used for the bus. As a result, we can exploit the extra width to reduce power consumption.

Some research exploited the extra space for power saving. In [15, 14] an intelligent spacing approach to minimize the crosstalk effect has been proposed. The basic idea is that in dual rail code the duplicate bit definitely runs the

* This work is partially supported by the National Science Council of Taiwan under Grant NSC92-2218-E011-006.

same signal as original one. As a result, we can place original bit and duplicate one as close as possible, and leave more space for other signal wires, thus reducing crosstalk effects. However, the authors did not exploit the statistical properties of the bus traffic. Further, the space between the dual rail wire groups has been equally distributed. [10] added non-uniform spacing between wires to achieve low power. Unfortunately, the approach is only suitable for address buses, and the power saving is limited as the order of the wires were not modified. In addition, time consuming heuristic approaches were used to determine wire spacing.

In this paper, we propose a bus architecture that uses wire permutation, inversion, and spacing to improve energy-efficiency. All architecture options (permutation, inversion and spacing) are static and determined during chip synthesis. Note that our approach is applicable to both address and instruction/data buses.

We present the power model considering coupling capacitances, and formalize the permutation and spacing problems. Then we use genetic algorithms to determine the optimal bus permutation and introduce optimal algorithms for the spacing and wire inversion problem. To our knowledge this is the first time that an *optimal* polynomial algorithm for the spacing problem is presented. Since there is no encoding-decoding circuitry needed, our proposed approach can save power at nearly no cost in performance or design complexity compared to other codec methods. In contrast to [10], we can significantly improve power saving by reordering the wires. In addition, we also discuss different optimization algorithm options and parameters that may be used to determine our bus architecture.

The rest of the paper is structured as follows. In Section 2, we illustrate the bus framework and describe the coupling effect model used throughout the paper. In Section 3, we formulate the problem and illustrate algorithms to find the *optimal solution* for wire inversion and spacing. In Section 4, a genetic algorithm is described that we use to find the nearly optimal solution for wire permutation problem as well as to find a solution for the combined permutation, inversion and spacing problem. Section 5 shows experimental results and compare them to an bus architecture that uses only equal spacing (that is, no permutation and individual spacing). Finally we draw the conclusions in Section 6.

2. Preliminaries

2.1. Bus Model

For nanometer buses, the capacitive coupling between nonadjacent wires is very weak compared to that between adjacent wires [19]. Therefore, we adopt the simplified model shown in Figure 1, which only takes directly neigh-

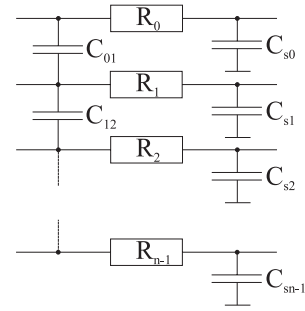


Figure 1. Bus model

		new values			
		00	01	10	11
old values	00	0	1	1	0
	01	1	0	4	1
	10	1	4	0	1
	11	0	1	1	0

Table 1. Normalized coupling effect

boring wires into consideration. In deep sub-micron era, the coupling capacitances C_{ij} strongly dominate the wire capacitances C_{si} . For example, in $0.13 \mu m$ technologies, C_{ij} can be greater than eight times of C_{si} . As we focus on coupling effects throughout the paper, R_i and C_{si} are neglected. Finally, we assume that the bus drivers are clocked and hence switch simultaneously.

For a simple bus layout, the coupling capacitance between two neighboring wires can be estimated by

$$C = \epsilon \frac{A}{d}, \quad (1)$$

where A depends on the height and length of the wires, d is the space between the wires, and ϵ is a technology constant. In the following, we assume ϵ and A to be the same for all wires. It means that the wire geometry is fixed and only d is modifiable.

2.2. Bus Energy Characteristic

To rate the coupling effect for CMOS circuits, we use Table 1 from [21]. The table does not represent the energy *drawn* (energy consumption) from the energy source. Instead, it shows the energy that is *dissipated* due to the corresponding transition pattern. Note that all power that is drawn from the energy source is finally also dissipated. Hence, using power dissipation is actually equivalent to using power consumption if an appropriate long sequence of

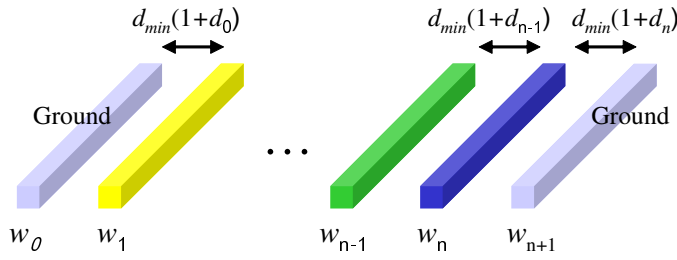


Figure 2. Bus configuration

pattern is considered. However, contrary to the power consumption table that is typically used, Table 1 is symmetric. This is especially advantageous when it comes to implementing optimization algorithms.

For our analysis, we assume a bus with n signal wires and two additional shields at both sides as shown in Figure 2. The entire bus layout consists of $n + 2$ wires, and the first wire w_0 and the last wire w_{n+1} are shields that are tied to ground.

In order to easily calculate the crosstalk effects caused by two neighboring wires i and j , a *crosstalk table* $xtalk(i, i_p, j, j_p)$ is determined, where $i_p, j_p \in \{0, 1\}$ denote the inversion level of wire i and j . If the inversion level of a wire is 1, then the values are transferred in negative (inverted) logic over that wire. Otherwise, positive logic is used. The crosstalk table is determined according to

$$\begin{aligned}
 xtalk(i, 0, j, 0) = & p_{00,01}(i, j) + p_{10,11}(i, j) + \\
 & p_{01,00}(i, j) + p_{11,10}(i, j) + \\
 & p_{00,10}(i, j) + p_{01,11}(i, j) + \\
 & p_{10,00}(i, j) + p_{11,01}(i, j) + \\
 & 4 \cdot p_{01,10}(i, j) + 4 \cdot p_{10,01}(i, j)
 \end{aligned} \quad (2)$$

$$\begin{aligned}
 xtalk(i, 0, j, 1) = & p_{00,01}(i, j) + p_{10,11}(i, j) + \\
 & p_{01,00}(i, j) + p_{11,10}(i, j) + \\
 & p_{00,10}(i, j) + p_{01,11}(i, j) + \\
 & p_{10,00}(i, j) + p_{11,01}(i, j) + \\
 & 4 \cdot p_{11,00}(i, j) + 4 \cdot p_{00,11}(i, j)
 \end{aligned} \quad (3)$$

and

$$\begin{aligned}
 xtalk(i, 1, j, 1) &= xtalk(i, 0, j, 0) \\
 xtalk(i, 1, j, 0) &= xtalk(i, 0, j, 1)
 \end{aligned} \quad (4)$$

A term $p_{ab,xy}(i, j)$ denotes the probability value that wire pair (w_i, w_j) is ab before and xy after a clock cycle ($ab, xy \in \{00, 10, 10, 11\}$).

Using the crosstalk table, the overall coupling effect of a bus of $n + 2$ wires can be determined as follows:

$$xtalk_{eff} = \sum_{i=0}^N c_i, \quad (5)$$

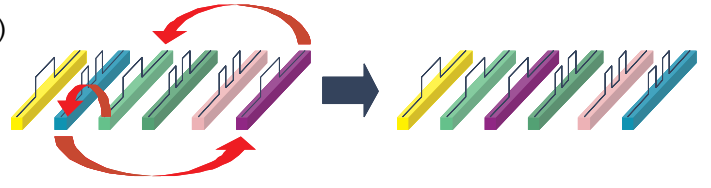


Figure 3. Bus permutation example

with

$$c_i = \epsilon \frac{A}{d_{min}} \cdot xtalk(w_i, inv_i, w_{i+1}, inv_{i+1}). \quad (6)$$

Where w_0 and w_{n+1} denote the left and right shield wires, respectively. The inv_i is 1 if data on wire w_i is transmitted in inverted logic, 0 otherwise. Note that $inv_0 = inv_{n+1} = 0$ and we assume minimal spacing between their neighboring wires. Finally, d_{min} is the minimal allowed distance (space) between two wires.

3. Problem Formulation

In the following three sub-sections, the basic optimization problems are defined and optimal solutions are given for two of them. Thereafter, a combined optimization problem is described.

3.1. Optimal Permutation Problem

In order to efficiently reduce the coupling power on a bus, the wires are statically reordered so that bus lines with a similar behavior are adjacent (see Figure 3 for example). As a result, the coupling effect $xtalk_{eff}$ will change accordingly. The optimal permutation problem is to reorder the wires so that the total coupling effect $xtalk_{eff}$ is minimal.

In detail, a permutation is defined by a mapping π that associates each wire i on the permuted bus with its position j on the original bus: $\pi(i) = j$. Note that the shields are not permuted. That is $\pi(0) = 0$ and $\pi(n + 1) = n + 1$.

As shown in [7], the bus ordering problem is NP-hard. Hence, an appropriate heuristic algorithm is needed to calculate a good solution in reasonable time.

3.2. Optimal Inversion Problem

In contrast to previous studies [16, 21], our optimal inversion problem is to *statically* assign each wire a logic level. That is, the data over a wire are either transmitted using positive or negative logic. The optimal inversion for a given bus layout can be determined by the algorithm shown in Figure 4. In detail, `get_optimal_inversion` is called two

```

PROCEDURE get_optimal_inversion (bus, init1) IS
  Input: inversion value of wire 1
  Input: bus permutation vector bus = (w0, w1, ... wn+1)
  Output: optimal inversion vector inv = (v0, v1 ... vn+1)
BEGIN
  vlevel = init1;
  v0 = 0; vn+1 = 0;
  FOR i IN 1 TO n LOOP
    vi = vlevel;
    IF xtalk(wi, vi, wi+1, vlevel) >
      xtalk(wi, vi, wi+1, vlevel ⊕ 1) THEN
        vlevel = vlevel ⊕ 1;
    END IF;
  END LOOP;
  RETURN (v0, v1 ... vn+1);
END PROCEDURE;

```

Figure 4. Algorithm to optimize bus wire inversion for a given wire permutation (⊕ denotes the logical exclusive or operator)

times. First, parameter *init*₁ is set to 0, then it is set to 1. The configuration that gives a lower *xtalk*_{*eff*} is the optimal solution.

The algorithm exploits the symmetry property of the crosstalk table:

$$xtalk(i, v_i, j, v_j) = xtalk(i, v_i \oplus 1, j, v_j \oplus 1)$$

where ⊕ denotes the logical exclusive-or operator (see also Equation 4). As a result, for a given bus permutation *w* = (*w*₀, *w*₁, ... *w*_{*n*-1}) and two inversion vectors *v* = (*v*₀, *v*₁, ... *v*_{*n*-1}) and $\bar{v} = (v_0, v_1, \dots, v_m, v_{m+1} \oplus 1, v_{m+2} \oplus 1, \dots, v_{n-1} \oplus 1)$ the following equation holds for all $0 \leq m < n - 1$:

$$\begin{aligned}
 xtalk(w, v) - xtalk(w, \bar{v}) = \\
 xtalk(w_m, v_m, w_{m+1}, v_{m+1}) - \\
 xtalk(w_m, v_m, w_{m+1}, v_{m+1} \oplus 1).
 \end{aligned}$$

Accordingly, if for a given inversion vector, the inversion levels are toggled for all wires with a index *i*, *i* > *m*, then coupling reduction/increment for the *entire* bus is determined by the change of the crosstalk value between (neighboring) wires *m* and *m* + 1.

Note that the inversion level can be easily incorporated into the bus architecture, as only the bus driver must be modified accordingly at the sender. At receiver side, the corresponding inverter can be typically merged with the following combinational logic. Hence, wire inversion nearly comes for free.

3.3. Optimal Spacing Problem

The optimal spacing problem is to adjust the spacing between wires so that the total coupling factor between adjacent wires is minimized. However, the accumulative additional wire space that can be exploited is equal to a given value Δ.

The coupling factor between two wires *w*_{*i*} and *w*_{*i*+1} is

$$Q_i(d_i) = c_i \cdot \frac{1}{1+d_i}. \quad (7)$$

As shown in Figure 2, *d*_{*i*} is the additional space between the wires divided by *d*_{*min*}. I.e., wire spacing is equal to *d*_{*min*}(1 + *d*_{*i*}), with *d*_{*i*} ≥ 0. If *d*_{*i*} = 0, then the wire spacing is *d*_{*min*}. Hence, the term *c*_{*i*} is the coupling value for minimal space (see Equation 6). Note that doubling the space will decrease the coupling value by a factor of 2.

As a result, the optimization goal is to minimize

$$xtalk_{s,eff} = \sum_{i=0}^n Q_i(d_i) \quad (8)$$

with respect to $\sum_{i=0}^n d_{min} \cdot d_i = \Delta$.

The wire spaces *d*_{*i*} that give a *minimal* *xtalk*_{*s,eff*} can be calculated as follows:

- First, calculate *Q*'_{*i*}(0) for all *i* and sort the results in increasing order. Without loss of generality, we assume *Q*'₀(0) ≥ *Q*'₁(0) ≥ ... *Q*'_{*N*}(0). *Q*'_{*i*} denotes the derivate of *Q*_{*i*} with respect to *d*_{*i*}. Note that *Q*'_{*i*}(*d*) < 0 for *d* ≥ 0.
- For all *x* ∈ 0, 1, 2, ... *n* calculate a constant *K*_{*x*} to *K*_{*x*} = *Q*'_{*x*}(0) and determine

$$\Delta_x = \sum_{i=0}^N d_{min} \cdot d_i(K_x),$$

where *d*_{*i*}(*K*) is defined as follows:

$$d_i(K) = \begin{cases} \sqrt{-\frac{c_i}{K}} - 1 & : K \geq -c_i \\ 0 & : K < -c_i \end{cases} \quad (9)$$

Let \hat{x} be the largest *x* for which Δ_{*x*} > Δ.

- Calculate the optimal *K*_{*opt*} according to

$$K_{opt} = - \left(\frac{\sum_{i=\hat{x}+1}^n \sqrt{c_i}}{\Delta/d_{min} + n - \hat{x}} \right)^2. \quad (10)$$

- From *K*_{*opt*} all *d*_{*i*} can be calculated using Equation 9.

Note that the proof of the spacing solution given above is optimal is out of the scope in this paper.

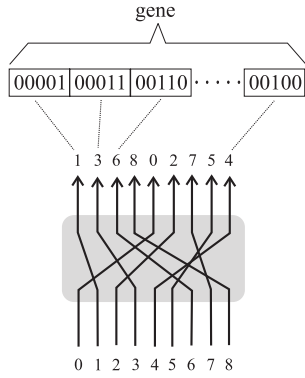


Figure 5. Structure of a gene

3.4. Combined Permutation, Inversion and Spacing Problems

Permutation, inversion, and spacing can now be combined to build a bus architecture that produces a minimal effective coupling value $xtalk_{eff}$. In detail, a bus configuration consists of

- a wire permutation π ,
- an inversion vector $inv = (v_0, v_1, \dots, v_{n+1})$ and
- a wire spacing $D = (d_0, d_1, \dots, d_n)$

so that the effective coupling value

$$xtalk_{tot} = \frac{\epsilon \cdot A}{d_{min}} \sum_{i=0}^N \frac{xtalk(w_{\pi(i)}, v_{\pi(i)}, w_{\pi(i+1)}, v_{\pi(i+1)})}{1 + d_i} \quad (11)$$

is minimal.

As finding the optimal solution for the permutation problem is already NP-hard, we used a heuristic approach to find a suitable architecture for this combined problem.

4. Optimization Algorithms

In order to synthesize a bus architecture that is tailored to a specific bus activity, we developed a combined approach to simultaneously optimize permutation, inversion, and spacing as described in Section 3.4.

Similar to our previous work in [11], our optimization technique adopted here is also based on genetic algorithms (GA).

During GA optimization, a set of individual genes is created where each single individual is a solution to the problem. The structure of a gene is shown in Figure 5. It consists of a string of N numbers, where the leftmost number determines the number of the wire that is routed to the leftmost position. New genes are generated by merging two existing (parent) genes from the population. Selection of parent

PROCEDURE simultaneous_GA_opt **IS**

BEGIN

WHILE (continue) **LOOP**

WHILE ($|Population| < upper_limit$) **LOOP**

$(father, mother) = random_select (Population);$

$g = new_gene (father, mother);$

$g.inversion = calculate_best_inversion (g);$

$g.spacing = calculate_best_spacing (g);$

$Population = Population \cup g;$

END LOOP;

$Population = select_genes (Population);$

END LOOP;

RETURN $get_best_gene (Population);$

END PROCEDURE;

Figure 6. Simplified pseudo-code for simultaneous optimization

genes is randomly done. However, genes with a better *fitness value* are preferred. Each gene is associated with a fitness value that describes the quality of the solution represented by the gene. We use Equation 11 to rate the genes.

Here, only the permutation is determined by the GA. As shown in Section 3.2 and 3.3, the inversion vector as well as wire spacing can be determined optimally for a given permutation without much effort. Hence, for each gene generated by the GA, an optimal inversion vector and optimal spacing were calculated (see Figure 6).

Further, we also developed an approach that first optimizes permutation and inversion using a fixed, equal wire spacing (see Figure 7). After a good solution has been determined, optimal spacing is determined for this configuration using the algorithm described in Section 3.3. I.e., instead of generating an optimal spacing for each gene, permutation and inversion is done by assuming constant spacing (all spaces were set to d_{min}). This approach is called *sequential* optimization in the following while the previous is named *simultaneous* optimization.

5. Experimental Results

In order to analyze the effectiveness of our approach, we implemented the GA in C++ on a LINUX system. The transition pattern were obtained from SimpleScalar processor simulator [3]. We extracted traces for the instruction bus (bus width $N = 40$) between processor core and instruction cache of the PISA processor architecture, which is derived from the MIPS-IV ISA. In detail, we gathered trace information from complete runs of 12 SPEC 2000 integer and floating point benchmark programs. In each case the opti-

```

PROCEDURE sequential_GA_opt IS
BEGIN

```

```

  WHILE (continue) LOOP

```

```

    WHILE ( $|Population| < upper\_limit$ ) LOOP
      ( $father.mother$ ) = random_select ( $Population$ );
       $g$  = new gene ( $father, mother$ );
       $g.inversion$  = calculate_best_inversion ( $g$ );
       $g.spacing$  = equal_spacing;
       $Population = Population \cup g$ ;

```

```

    END LOOP;

```

```

       $Population = select\_genes (Population)$ ;

```

```

  END LOOP;

```

```

   $winner = get\_best\_gene (Population)$ ;

```

```

   $winner.spacing = calculate\_best\_spacing (winner)$ ;

```

```

  RETURN  $winner$ ;

```

```

END PROCEDURE;

```

Figure 7. Simplified pseudo-code for sequential optimization

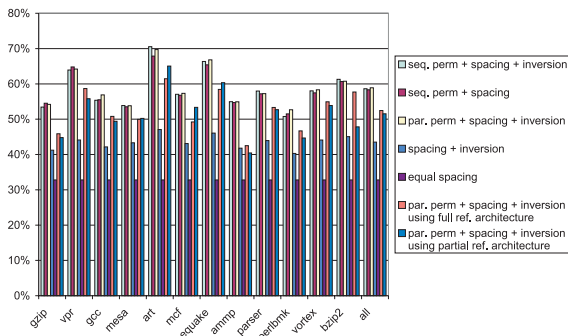


Figure 8. Energy reduction for various bus schemes ($\Delta = 20 \times d_{min}$)

mization programs were executed for one hour on a Pentium 4 2.4 GHz with 1G Byte of main memory. That is, for a fair comparison all programs had the same amount of run-time to perform optimization.

For our experiments, we set the total additional wire space Δ to $10 \cdot d_{min}$, $20 \cdot d_{min}$, $40 \cdot d_{min}$ and $80 \cdot d_{min}$. All coupling reduction numbers are calculated as follows:

$$100 \cdot \frac{\text{minimal_spacing_power} - \text{optimized_power}}{\text{minimal_spacing_power}}$$

Figure 8 compares seven different bus schemes :

- Sequentially wire permutation, spacing, and inversion optimization (seq. perm + spacing + inversion).

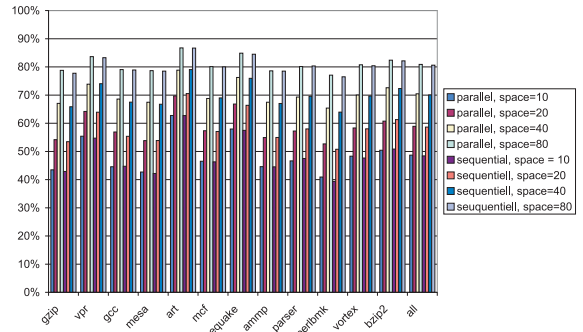


Figure 9. Energy reduction for various additional wire space

- Sequentially wire permutation and spacing optimization (seq. perm + spacing).
- Simultaneously (parallel) wire permutation, inversion and spacing optimization (par. perm + spacing + inversion).
- An architecture that uses spacing and inversion only (spacing + inversion).
- A configuration that amortizes the additional width to each wire space (equal spacing).

For the previous five schemes a *separate* architecture has been generated for each individual benchmark. The following two schemes are each based on a *single reference* architecture. That is, permutation, inversion and spacing were determined from a “artificial” benchmark that was generated by merging several of the original benchmarks. We tested two configurations:

- An *artificial* benchmark were generated by merging all 12 benchmarks (par. perm + spacing + inversion using full ref. architecture).
- An *artificial* benchmark were generated by merging benchmarks gzip, vpr, gcc, mesa, art and mcf (the first 6 shown in the figure; par. perm + spacing + inversion using partial ref. architecture).

We used simultaneous wire permutation, inversion and spacing to do optimization. The power saving number were then measured by applying the various benchmarks to this particular architecture.

As shown in Figure 8, sequential and simultaneous permutation, spacing and inversion schemes produce almost the same energy reduction rate, and each has special skills on different benchmarks. Besides, the energy reduction of first three schemes are significantly better than the gain obtained with “spacing + inversion” and “equal spacing”. For

some cases, using permutation and optimal spacing doubles the power reduction results. On average nearly 60% power reduction can be achieved (compared to equal spacing). This is a very promising result as our architecture adds only marginal delay overhead to the bus. The results also show that the gain achieved by inversion is not significant. Further, in some cases the architectures with inversion performed even worse than architectures without inversion. Hence, permutation and spacing are sufficient to achieve good results. Compared to an approach that uses only optimal spacing and inversion (see “spacing + inversion”), approximately 15% on average and in some cases even more than 20% *additional* power reduction could be achieved by applying permutation. Hence, permutation and optimal spacing are key technologies for power reduction.

The results obtained from the *artificial* benchmarks also show a good performance. Note that the “partial reference architecture” were obtained by applying the first 6 benchmarks. Nevertheless, the power saving numbers for the remaining benchmarks that were *not* used to synthesize the architecture are significantly better than using only spacing and inversion in most cases. Hence, we expect that an architecture generated from a carefully selected set of benchmarks can significantly save power for most typical programs.

Figure 9 shows the sequentially and simultaneously optimized result for different additional widths Δ . It is reasonable that we can get more reduction rate by providing more additional space. However, note that the gain is not the same among the benchmarks. For example, the power reduction of benchmark “mesa” increases from approximate 40% to 80% while the gain for “art” only rises from approximate 60% to 85%. As a result, the gain that can be achieved strongly depends on the additional spaces as well as on the statistical properties of the transition pattern that are traveling over the bus. Note that for high Δ values, the gain becomes nearly the same for all benchmarks. For example, when $\Delta = 10 \cdot d_{min}$ the gain is varying by up to 20%, the results for $\Delta = 80 \cdot d_{min}$ are nearly the same among all benchmarks.

Overall, any additional space allocated to the bus channel on the die can be efficiently used to combat power dissipation. For example, if area permits increasing the bus width by a factor of three, 80% power can be saved on average (compared to a bus with minimal spacing).

6. Conclusion

With decreasing feature size on silicon, the coupling capacitances of buses grow rapidly causing a significant impact on the power consumption of whole chip. Thus, buses should be designed and optimized to dissipate less power without sacrificing performance. In this paper, we devel-

oped a new approach that synthesizes a static bus architecture from a given bus transition profile for embedded systems. The architecture exploits wire permutation and inversion. Further, it increases the space between adjacent bus wires so that the total coupling effect is minimized. In contrast to other work, we are the first to show an algorithm that finds the *optimal* solution for the spacing problem in polynomial time. Previous techniques used time consuming heuristics to achieve nearly optimal solutions.

In contrast to other approaches we applied all techniques (permutation, inversion and spacing) at the same time to significantly decrease coupling power. As the permutation problem is NP hard, we used a genetic algorithm (GA) to determine a bus architecture. However, the GA only determined the wire order (permutation), while inversion and spacing were calculated using optimal algorithms.

Note that our architecture does not use any encoding/decoding circuitry. Hence, no delay is added (typically, the wiring delay caused by permutation can be neglected). As a result, the architecture is especially advantageous if timing budget is tight. However, the additional spacing produces some area overhead. Fortunately, power reduction can be easily balanced with area overhead.

For our experiments, we used instruction bus traces obtained from 12 SPEC2000 benchmark programs. We simulated different combinations among permutation, spacing, and inversion. Integrated all optimization techniques together, our approach can save energy up to 68% for the best case and 58% on average while increasing the total wire space by approx. 50% (compared to a bus with minimal wire space; $\Delta = 20d_{min}$). Even if using only 25% additional space is used, still about 50% power could be saved on average. Our experiments also show that a common single instruction bus architecture can be synthesized that significantly saves power for typical programs.

References

- [1] L. Benini, A. Macii, E. Macii, M. Poncino, and R. Scarsi. Synthesis of Low-Overhead Interfaces for Power-Efficient Communication over Wide Buses. In *in Proceedings DAC-1999*, 1999.
- [2] L. Benini, G. D. Micheli, E. Macii, D. Sciuto, and C. Silvano. Address bus encoding techniques for system-level power optimization. In *in Proceedings EuroDAC-97*, pages 861–867, Nov. 1997.
- [3] D. Burger and T. M. Austin. The simplescalar tool set, version 2.0. <http://www.simplescalar.com>, 1997.
- [4] W. Fornaciari, M. Polentarutti, D. Sciuto, and C. Silvano. Power optimization of system-level address buses based on software profiling. In *International Workshop on Hardware/Software Codesign (CODES)*, pages 29–33, May 2000.

- [5] J. Henkel and H. Lekatsas. A2BC: Adaptive Address Bus Coding for Low Power Deep Sub-Micro Designs. In *in Proceeding of Design Automation Conference (DAC)*, pages 744–749, Jun. 2001.
- [6] M. J. Irwin. Tutorial: Low power design for soc. International Conference on ASICSoC, month =.
- [7] I. H.-R. Jiang, Y.-W. Chang, and J.-Y. Jou. Crosstalk-Driven Interconnect optimization by Simultaneous Gate and Wire Sizing. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 19(9):999–1010, Sep. 2000.
- [8] K.-W. Kim, K.-H. Baek, N. R. Shanbhag, C. L. Liu, and S. Kang. Coupling-driven signal encoding scheme for low-power interface-design. In *in Proceedings ICCAD-2000*, pages 318–321, Nov. 2000.
- [9] C.-G. Lyuh, T. Kim, and K.-W. Kim. Coupling -aware high-level interconnect synthesis. *IEEE Trans. Computer-Aided Design of ICS*, 23(1):157–164, Jan. 2004.
- [10] L. Macchiarulo, E. Macii, and M. Poncino. Wire Placement for Crosstalk Energy Minimization in Address Bus. In *in Proceedings DATE-2002*, pages 158–162, 2002.
- [11] E. Naroska, S. Ruan, F. Lai, U. Schwiegelshohn, and L. Liu. On optimizing power and crosstalk for bus coupling capacitance using genetic algorithms. In *in Proceedings of the 2003 IEEE International Symposium on Circuits and Systems (ISCAS 2003)*, volume 5, pages V277–V280, May 2003.
- [12] V. Raghunathan, M. B. Srivastava, and R. K. Gupta. A survey of techniques for energy efficient on-chip communication. In *in Proceedings DAC-2003*, pages 900–905, 2003.
- [13] S. Ramprasad, N. R. Shanbhag, and I. N. Hajj. A Coding Framework for Low-Power Address and Data Busses. *IEEE Trans. on VLSI Systems*, June 1998.
- [14] D. Rossi, S. Cavallotti, and C. Metra. Error Correcting Codes for Crosstalk Effect Minimization. In *in Proceedings of IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT'03)*, pages 257–264, Nov. 2003.
- [15] D. Rossi, V. van Dijk, R. Kleihorst, and C. Metra. Coding Scheme for Low Energy Consumption Fault-Tolerant Bus. In *in Proceedings of IEEE On-Line Testing Workshop (IOLTW'02)*, pages 8–12, Jul. 2002.
- [16] Y. Shin, S. Chae, and K. Choi. Partil Bus-Invert Coding for Power Optimization of Application-Specific Systems. *IEEE Trans. on VLSI Systems*, 9(2):377–383, Apr. 2001.
- [17] Y. Shin and T. Sakurai. Coupling-driven bus design for low-power application-specific systems. In *in Proceeding of Design Automation Conference (DAC)*, pages 18–22, Jun. 2001.
- [18] P. P. Sotiriadis and A. Chandrakasan. Low Power Bus Coding Techniques Considering Inter-wire Capacitances. In *in Proceeding of IEEE Conf. on Custom Integrated Circuits (CICC '00)*, pages 507–510, 2000.
- [19] P. P. Sotiriadis and A. P. Chandrakasan. Bus Energy Reduction by Transition Pattern Coding Using a Detailed Deep Submicrometer Bus Model. *IEEE Trans. on Circuits Syst. I*, 50(10):1280–1295, Oct. 2003.
- [20] D. Sylvester and C. Hu. Analytical Modeling and Characterization of Deep-Submicrometer Interconnect. *Proceedings of the IEEE*, 89(5):634–664, May 2001.
- [21] Y. Zhang, J. Lach, K. Skadron, and M. R. Stan. Oddeven bus invert with two-phase transfer for buses with coupling. In *in Proceeding of ISLPED '02*, pages 80–83, Aug. 2002.