Transactions Briefs

Routability Checking for Three-Dimensional Architectures

William N. N. Hung, Xiaoyu Song, Timothy Kam, Lerong Cheng, and Guowu Yang

***Abstract*—We present a novel symbolic routability checking approach for three-dimensional interconnect layout. The model considered is a general architecture that can fit into different applications, such as ASIC, multichip modules, field-programmable gate arrays, and reconfigurable computing architectures. The method can incrementally incorporate additional constraints driven by timing, performance, and design. We used the latest satisfiability solver to validate the effectiveness of our approach. The experimental results demonstrate the encouraging performance on difficult routing benchmarks.**

***Index Terms*—Computer-aided design (CAD), layout, routability, satisfiability.**

I. Introduction

Routing plays a critical role in the high-performance deep submicron very large scale integration (VLSI) design process [15], [16]. Under the current technology, ASIC designs are frequently implemented with four to six layers of metals [16]. Most printed circuit boards (PCBs) and multichip modules (MCMs) use multiple layers where their logic cells occupy the bottom layers and their wires are routed in upper metal layers [9], [15]. Three-dimensional (3-D) layouts for VLSI were considered in [9]–[12]. Three-dimensional architectures have also been proposed for field-programmable gate arrays (FPGAs) [13] and reconfigurable computing [14].

In this paper, we present Boolean routability formulations in a 3-D routing model. Our model is a generalization of the 3-D multilayer routing architectures. Our Boolean formulations are used to determine routability. We used SAT solvers to solve our problem. But the same formulation can also be applied using binary decision diagrams (BDDs) [1717], [1818], multivalued decision diagrams (MDDs) [4], or other decision methods.

Devadas [1] was the first to use satisfiability to solve problems in physical design. Satisfiability based method has been proposed for FPGA board level routing [2] and for segmented channel routing [3]. MDD based routing solution limited to only two bends [4] was also studied for two-terminal planar model. In this paper, we propose a set of Boolean constraints to directly characterize exact routability (with arbitrary number of bends) for mutliterminal 3-D models.

II. Routing Model

Without loss of generality, we model a 3-D routing region by $R = \{0, 1, \ldots, W\} \times \{0, 1, \ldots, L\} \times \{0, 1, \ldots, H\}$ where W, L and H are nonnegative integers. Within the discrete routing space, the set of lines including the rectangle boundaries is called the *grid* (or the grid graph) and the lines are called *grid lines*. An intersection between grid lines is referred to as a *grid point*. We denote the set of all grid points by G. A subset of G is referred to as *terminal points*. A *terminal* t is denoted by a 3-tuple $(x(t), y(t), z(t))$ of the x, y, z coordinate values of t. An *edge* is a small interval connecting two adjacent grid points along one of the three dimensions. A *wire segment* is a small portion of a wire (net) on an edge. A *wire piece* is a wire connecting two grid points (not necessarily adjacent to each other) along one of the three dimensions. Thus, a wire piece may span several contiguous edges, therefore consists of several wire segments, along the same dimension. Fig. 1 illustrates the grid graph for 3-D routing model.

Consider each grid point b on the grid graph, there are at most six edges (two edges per dimension) connecting that point b with the rest of the grid graph. We denote $C(b)$ as the set of edges on the grid graph connected to grid point b.

The set of terminal points is partitioned into N sets, $n_0, n_1, \ldots, n_{N-1}$. For convenience, we denote $Q = \{0, 1, \ldots, N-1\}$ as the set of net indices. Each set n_i is called a *net*, and the set of nets is denoted by M, specifying the entire connectivity of the terminals. A net is a collection of terminals to be wired together. We use to $\hat{n}_i$ denote the number of terminals in net n_i. Nets containing exactly two terminals ($\hat{n}_i = 2$) are called *two-terminal* nets; otherwise, they are called *multiterminal* nets. In many cases, problems for two-terminal nets are easier to analyze than that for multiterminal nets.

The generalized 3-D routing problem is to find a solution for connecting all the terminals that belong to the same net within the given routing region R. Connection within each net is defined by contiguous wire pieces that touch all terminals of that net. All nets must be routed without crossing each other. Hence, no edge can be occupied by more than one net; and no grid point can be touched by more than one net. A two-layer Manhattan routing model is a special case of 3-D routing where all horizontal wiring (rows) are located in one layer and all vertical wiring (columns) are located in another layer. Our general routing model can be extended to any specific routing model with performance constraints.

III. Boolean Formulation

Given a set of N multiterminal nets to be routed: $n_0, n_1, \ldots, n_{N-1}$, we denote the number of terminals in each net by $\hat{n}_0, \hat{n}_1, \cdots, \hat{n}_{N-1}$, respectively. For each multiterminal net $n = \{(x_1, y_1, z_1), (x_2, y_2, z_2), \cdots, (x_{\hat{n}}, y_{\hat{n}}, z_{\hat{n}})\}$, we create $\hat{n} - 1$ two-terminal subnets

$$\begin{aligned} s(n,1) &= \{(x_1, y_1, z_1), (x_2, y_2, z_2)\} \\ s(n,2) &= \{(x_1, y_1, z_1), (x_3, y_3, z_3)\} \\ &\cdots\cdots \\ s(n, \hat{n}-1) &= \{(x_1, y_1, z_1), (x_{\hat{n}}, y_{\hat{n}}, z_{\hat{n}})\}. \end{aligned}$$

A net is different from a subnet. An edge (on the 3-D grid graph) can be shared by two or more subnets of the same net, but the edge can be occupied by at most one net only.

Theorem 1: Routability for a multiterminal net n is equivalent to routability of all its subnets $s(n,1), s(n,2), \ldots, s(n, \hat{n}-1)$.

Proof: If there is a routing for all the above subnets, then all the terminals in n are connected to the first terminal (x_1, y_1, z_1), so all the terminals are connected together. Hence, it is a routing for the multiterminal net n. Conversely, if there is a routing for the multiterminal net

Manuscript received September 20, 2003.

W. N. N. Hung was with Portland State University, OR 97207 USA. He is now with Synplicity Inc., Sunnyvale, CA 94086 USA (e-mail: whung@synplicity.com).

X. Song, L. Cheng, and G. Yang are with Portland State University, OR 97207 USA.

T. Kam is with Intel Corporation, Hillsboro, OR 97124 USA.

Digital Object Identifier 10.1109/TVLSI.2004.837999

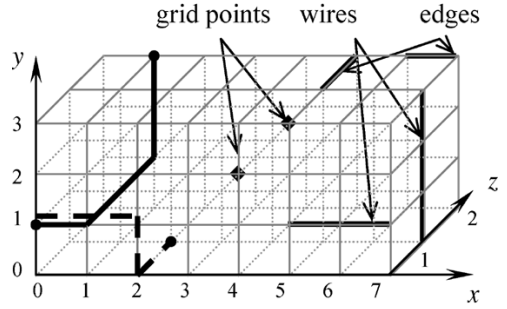

Fig. 1. Grid graph for 3-D routing.

n, then the routing will provide a connection between the first terminal (x_1, y_1, z_1) and all the other terminals in n, leading to a routing for all the above subnets. ■

To route multiterminal nets, an edge (wire segment) can be assigned with several different subnets simultaneously as long as all these two-terminal subnets belong to the same multiterminal net. However, subnets that belong to different nets cannot be assigned to the same edge (wire segment). For example, given a three-terminal net $n = \{(0,1,0),(2,3,2),(2,0,1)\}$. We create two, two-terminal subnets from this three-terminal net: $s(n,1) = \{(0,1,0),(2,3,2)\}$; $s(n,2) = \{(0,1,0),(2,0,1)\}$. An example of their routing is shown in Fig. 1. Notice that the wire piece between (0,1,0) and (2,1,0) are assigned to both subnets. This is a valid routing as both subnets are actually wires in the same multiterminal net.

For each edge e on the grid graph, we define a variable vector, $\vec{r}(e)$, to represent the identifier of the net crossing over edge e

$$\vec{r}(e) = r_1(e)r_2(e)\ldots r_w(e), \quad \text{where} \quad w = \lceil \log_2 N \rceil \quad \text{and} \quad r_i(c) \in \{0,1\}, \quad \text{for} \quad i = 1,2,\ldots,w.$$

The vector represents the Boolean numeric encoding of the net identifier. Let T be the maximum number of subnets for any net $T = \max\{\hat{n}_0, \hat{n}_1, \cdots \hat{n}_{N-1}\} - 1$. We define T valid bits, $v_i(e)$ where $i = 1,2,\ldots,T$, for every variable vector $\vec{r}(e)$.

Let $q(n) = \{1,2,\ldots,n-1\}$ be the set of subnet indices for each net n. The assignment of a subnet $s(n_m,i)$ of net n_m to any edge e can be represented by the symbolic function $p(m,i,e)$

$$p(m,i,e) = (\vec{r}(e) = m) \wedge v_i(e), \quad \text{where} \quad i \in q(n_m). \tag{1}$$

The assignment of a multiterminal net n_m to any edge e can be represented by the symbolic function $\varphi(m,e)$

$$\varphi(m,e) = (\vec{r}(e) = m) \wedge (\exists i \in q(n_m) \cdot v_i(e)). \tag{2}$$

Note that the existential quantifier over subnet indices is simply a disjunction over the subnet indices. This is not a quantified Boolean formula.

The valid bits, $v_i(e)$ where $i = 1,2,\ldots,T$, clearly indicate whether each subnet passes through edge e. For every edge e, we impose the following constraints.

- **Constraint** (3): If the variable vector $\vec{r}(e)$ is not equal to any net index (for cases where N is not a power of 2), all the valid bits associated with that vector are invalid

$$(\forall m \in Q \cdot (\vec{r}(e) \neq m)) \Rightarrow \forall i \in \{1,2,\ldots,T\} \cdot \neg v_i(e).$$

- **Constraint** (4): If the variable vector $\vec{r}(e)$ is equal to a net index, all the valid bits associated with subnet indices outside the subnet range of this net should be invalid

$$(\forall m \in Q \cdot (\vec{r}(e) = m)) \Rightarrow \forall i \notin q(n_m) \cdot \neg v_i(e).$$

For every grid point b, we impose the following constraints.

- **Constraint** (5): If b is a terminal of $s(n_m,i)$, the routing of subnet $s(n_m,i)$ must be connected to the grid point b through exactly one neighboring edge $e \in C(b)$, and none of the neighboring edge $w \in C(b)$ can be assigned to any subnet that does not belong to the net n_m. (Other subnets of the same net may or may not connect to this terminal, see the constraint below.)
- **Constraint** (6): If b is not a terminal of $s(n_m,i)$ and the routing of this subnet is connected to b, then $s(n_m,i)$ must be crossing exactly two of the grid point's adjacent edges, and none of the other nets can be connected to b.

Theorem 2: By satisfying constraints (3)–(6), a terminal cannot be connected to any set of connected wired segments (assigned to the same subnet) that lead to a loop.

Proof: We prove by contradiction. Suppose a terminal is indeed connected to some wire segments (assigned to the same subnet) that lead to a loop. That terminal is either inside the loop or outside the loop.

1) If the terminal is inside the loop, there must be at least two distinct edges from the terminal that are wired with the same subnet, thus violating constraint (5). The terminal at (1, 0, 4) is violating constraint (5).
2) If the terminal is outside the loop, the wiring from the terminal must eventually lead to a grid point that lies inside the loop. Since this grid point is inside the loop and is also connected to the terminal (which is outside the loop), it must have at least three edges that are wired with the same subnet. Thus violating constraint (6).

In both cases, we have shown that there is a contradiction with constraints (5) or (6). Therefore, the terminal cannot be connected to any wire segments (assigned to the same subnet) that lead to a loop. ■

Theorem 3: By satisfying constraints (3)–(6), both terminals of the same subnet are connected together through a set of connected wire segments assigned to the same subnet.

Proof: According to constraint (5), one of the (six) edges surrounding any terminal s must be assigned to the same subnet (which s belongs to). We have already proven in theorem 2, that s cannot be connected to any wired loops assigned to the same subnet. Starting from this terminal s, we can trace the grid points along the wire segments assigned to the same net. Every grid point must be a new grid point that we have not traced before, because a repetition of some earlier grid point would form a wired loop which violates theorem 2. Since there are only a finite number of grid points in the grid graph, our trace (without any loop) must eventually stop at some grid point t. There are two cases for this ending grid point t: a) t is a terminal of a subnet containing s. This satisfies our theorem. b) Otherwise, it leads to a contradiction: Since terminal s cannot be connected to any wired loops, there is only one path (along the trace) that we can reach t from s. Now that t is the end point, there is only one edge connected to t that is assigned to the subnet of s. Hence, it is violating constraint (6) at that grid point t. ■

Theorem 4: An assignment to all the Boolean variables, ($\vec{r}(e)$, $v_i(e)$, where $i = 1,2,\ldots,T$, for every edge e in the grid graph), such that constraints (3)–(6) are satisfied, implies that every subnet has a routing.

Proof: From theorem 3, both terminals of each subnet are connected together. Thus, every subnet has a routing. ■

Theorem 5: Subnet routability (every subnet has a routing) implies that there is an assignment to all the Boolean variables, ($\vec{r}(e)$, $v_i(e)$, where $i = 1,2,\ldots,T$, for every edge e in the grid graph), such that constraints (3)–(6) are satisfied.

Proof: There may be multiple routing paths that connect the two terminals of each subnet. For every subnet, we pick only one routing path that connects the two terminals of that subnet. We assign values to

all the Boolean variables according to the definition in (1). Since we are assigning variables according to the specified subnets (that belong to a given set of nets), $\overrightarrow{r}(e)$ and $v_i(e)$ will never be assigned to any value outside the set of net indices Q or undefined subnets, hence, satisfying (3) and (4). Since we have picked only one routing path for each two-terminal subnet, there is only one route edge connected to each terminal of any subnet, satisfying constraint (5). For any other grid point, it is either on a routing path (which means it is connected to exactly two route edges of a subnet) or not on any routing path (which means it is not connected to any route edges of any subnet). Thus, constraint (6) is also satisfied. ■

Theorem 6: The problem is routable if and only if there is an assignment to all the Boolean variables satisfying constraints (3)–(6).

Proof: According to theorems 4 and 5, we can obtain routability of all the subnets if and only if there is an assignment to all the Boolean variables satisfying constraints (3)–(6). Theorem 1 already established that routability of all nets is equivalent to routability of all subnets. Hence, all the (multiterminal) nets are routable if and only if there is an assignment to all the Boolean variables satisfying constraints (3)–(6). ■

Our formulation above provides a detailed routing solution if-and-only-if the problem is routable. All the constraints (3)–(6) are formulated with localized variables, i.e., the constraints for every edge and grid point only use variables in neighboring edges and require no fix-point computation or recursion.

We can prune down the original routing region given obstacles, prerouted regions (or irregular regions), because their associated edges are unroutable. Thus, all variables associated with these unroutable edges become FALSE. All edges prerouted by other algorithms (useful for large number of easily routable nets) can be treated as constants here. By propagating these constants, constraints (3)–(6) would be simplified. We can put in additional constraints depending on the fabrication process, technology or performance requirements. For example, one can introduce crosstalk constraints that stipulate minimum distance between parallel nets. Instead of a decision problem, we can also solve a corresponding constrained optimization problem. The objective cost function to be optimized would depend on the physical design requirements, and solved through a number of SAT runs (instead of just one run). We have also worked on bounded routability, which is the existence of a routing where each net has only a limited (bounded) number of turn points. A subset of this problem is bounded two-layer Manhattan routing. A special case here is via-bounded two-layer Manhattan routing as explained before. Our model can also deal with shielding and crosstalk by adding additional constraints.

IV. Experiments

We converted the Boolean constraints in Section III to conjunctive normal form (CNF), and used zChaff [8] as the SAT solver. All routing results had been verified. The experiments were conducted in Linux on Intel 850 MHz Pentium III with 1 GB memory.

Table I shows the results for 3-D routing. The W, L, and H columns show the dimension of the grid graph. The *edge* column indicates the total number of edges for the grid graph. We randomly inject objects and indicate the object percentage (pruned out) of the original routing region. The *Vars* and *Clause* are measurements for the CNF. The runtime (seconds) for CNF generation and SAT checking are reported in the *Gen* and *Chaff* columns respectively. The S column indicates the routability of the specification.

From the table, it is pretty clear that generation time is proportional to the size of the CNF (vars and clauses). The CNF is based on constraints in Section III, which is directly proportional (since we have no fix-point computation or recursion) to the number of edges (that are not prerouted to constant) and the number of nets (log2 for binary encoding). Runtime of the SAT solver, however, does not necessarily correlate to the CNF size.

TABLE I
General 3-D Routing

W	L	H	N	edge	Obj%	Vars	Clause	Gen	Chaff	S
11	11	1	14	220	0	5798	35020	15	22	Y
11	11	1	19	220	0	5905	38325	16	1	Y
16	16	6	15	4160	33	91154	553140	239	37733	Y
41	41	1	10	3280	0	103972	635716	265	7994	Y
41	41	1	2	3280	0	79056	373676	160	85	Y
35	35	5	25	16800	14	565184	3594723	926	67	Y
51	51	4	12	28203	79	197728	1159076	334	279	Y
51	51	1	35	5100	26	147167	1017478	448	1702	Y
51	51	1	10	5100	2	275508	1263912	601	36	Y
101	101	1	35	20200	0	1340540	6523360	3073	1554	N
101	101	1	6	20200	0	565680	3127446	1331	10778	Y
101	101	1	22	20200	54	338505	2148960	618	1142	Y

TABLE II
Switchbox Routing

Bench	R	C	N	Vars	Clause	Gen	Chaff
SFSR	7	6	5	24936	89028	39.6	1.3
LRS	9	9	5	28665	102227	44.5	0.3
MIR	11	6	8	56554	201953	89.7	9.1
Dense	16	18	19	474456	1704095	758.7	196.2
Burstein	23	15	24	721939	2594199	1152.7	34834.3
Burstein	22	16	24	709595	2547823	1124.5	13211.5
Burstein	23	16	24	724024	2600454	1168.9	29543.3
Terms	23	16	24	1048400	3765486	1656.7	15181.2

As a case study, we use the benchmarks of the switchbox routing where all terminals are located at the four borders (except the four corners) of the routing region. All nets can only change direction inside the switchbox and no via can be located at the border. We perform our bounded routability (Section III) experiments on benchmarks taken from [5]–[7]. The results are shown in Table II. The size of each switchbox is indicated by the number of rows (R) and columns (C). Column N shows the number of nets. The Burstein and terminal-intensive benchmarks [5] are known to be difficult routing problems. Our largest CNF size is the terminal-intensive benchmark, where every grid point on the four edges of the switchbox is a terminal of a net. Still this turns out to be an easier problem for the SAT solver than the Burstein benchmark, which results in slightly smaller CNF size but took longer time to solve.

V. Conclusion

In this paper, we presented a Boolean formulation for 3-D routability and proved that our formulation can guarantee exact routability. Our approach can also incorporate additional performance driven constraints. The encouraging experimental results obtained using zChaff demonstrated the effectiveness of the method.

References

[1] S. Devadas, "Optimal layout via Boolean satisfiability," *Int. J. Comput. Aided VLSI Design*, no. 2, pp. 251–262, 1990.

[2] X. Song, W. N. N. Hung, A. Mishchenko, M. Chrzanowska-Jeske, A. Kennings, and A. Coppola, "Board-level multiterminal net assignment for the partial crossbar architecture," *IEEE Trans. VLSI Syst.*, vol. 11, pp. 511–514, June 2003.

[3] W. N. N. Hung, X. Song, E. M. Aboulhamid, A. Kennings, and A. Coppola, "Segmented channel routability via satisfiability," *ACM Trans. Design Automation Electron. Syst.*, vol. 9, no. 4, pp. 517–528, Oct. 2004.

[4] A. Srinivasan, T. Kam, S. Malik, and R. Brayton, "Algorithms for discrete function manipulation," in *Proc. Int. Conf. Computer-Aided Design*, 1990, pp. 92–95.

[5] W. K. Luk, "A greedy switch-box router," *Integration: The VLSI J.*, vol. 3, pp. 129–149, 1985.
[6] L. R. Smith, T. Saxe, J. Newkirk, and R. Mathews, "A new area router, the LRS algorithm," in *Proc. IEEE Int. Conf. Circuits Computers*, 1982, pp. 256–259.
[7] K. J. Supowit, "A minimum-impact routing algorithm," in *Proc. Design Automation Conf.*, 1982, pp. 104–112.
[8] L. Zhang and S. Malik, "Conflict driven learning in a quantified Boolean satisfiability solver," in *Proc. Int. Conf. Computer-Aided Design*, San Jose, CA, Nov. 2002.
[9] A. Hanafusa, Y. Yamashita, and M. Yasuda, "Three-dimensional routing for multilayer ceramic printed circuit boards," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 1990, pp. 386–389.
[10] L. Cheng, W. N. N. Hung, G. Yang, and X. Song, "Congestion estimation for 3-D circuit architectures," *IEEE Trans. Circuits Syst. II*, vol. 12, pp. 655–659, Dec. 2004.
[11] F. Preparata, "Optimal three-dimensional VLSI layout," *Math. Syst. Theory*, vol. 16, pp. 1–8, 1983.
[12] C. C. Tong and C. Wu, "Routing in a three-dimensional chip," *IEEE Trans. Computers*, vol. 44, pp. 106–117, Jan. 1995.
[13] M. J. Alexander, J. P. Cohoon, J. L. Coleflesh, J. Karro, and G. Robins, "Three dimensional field-programmable gate arrays," in *Proc. IEEE Int. ASIC Conf.*, Austin, TX, Sept. 1995, pp. 253–256.
[14] K. Bazargan, R. Kastner, and M. Sarrafzadeh, "3-D floorplanning: simulated annealing and greedy placement methods for reconfigurable computing systems," *J. Design Automation Embedded Syst.*, no. 5, pp. 329–338, Aug. 2000.
[15] N. Sherwami, S. Bhingarde, and A. Panyam, *Routing in the Third Dimension: From VLSI Chips to MCMs*. Piscataway, NJ: IEEE Press, 1995.
[16] N. Sherwami, *Algorithms for VLSI Physical Design Automation*. Norwell, MA: Kluwer, 1992.
[17] R. E. Bryant, "Graph based algorithms for Boolian function manipulation," *IEEE Trans. Computers*, vol. C-35, pp. 677–691, Aug. 1986.
[18] W. N. N. Hung, X. Song, E. M. Aboulhamid, and M. A. Driscoll, "BDD minimization by scatter search," *IEEE Trans. Computer-Aided Design*, vol. 21, pp. 974–979, Aug. 2002.

Bus-Switch Coding for Reducing Power Dissipation in Off-Chip Buses

Mauro Olivieri, Francesco Pappalardo, and Giuseppe Visalli

***Abstract*—We present a novel coding scheme for reducing bus power dissipation. The presented approach is well suited to driving off-chip buses, where the line capacitance is a dominant factor. A distinctive feature of the technique is the dynamic reodering of bus line positions, in order to minimize the toggling activity on physical bus wires. The effectiveness of the approach is demonstrated through cycle-accurate simulation of industrial benchmarks in conjunction with post-layout evaluation of speed, power and area overhead.**

***Index Terms*—Bus Transfer, encoding, power demand, very large scale integraton (VLSI).**

I. Introduction and Background

Differently from technologic solutions, such as voltage swing reduction and charge recycling [8], [14], bus data encoding aims at lowering the average energy consumption by reducing the switching activity of bus lines. In bus drivers—especially off-chip bus drivers—the large load capacitance maintains dynamic (i.e., switching) power the target to be defeated, also in emerging technologies like systems-in-packages [12].

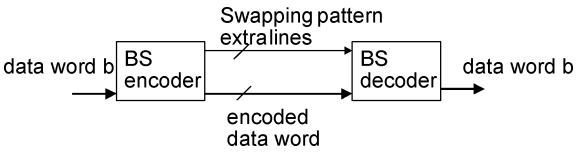

Fig. 1. Basic architecture scheme of BS coding/decoding system.

Bus encoding for low power started with bus-invert (BI) [6] and clustered bus-invert (CBI) coding. In [1] and [5], the idea of adaptive coding was introduced. In particular, adaptive partial BI (APBI) applies BI only to lines with high toggling probability. In [4], Ramprasad showed a theoretical lower bound in reducing the activity of a bus. Other more recent works address the problem of reducing power consumption caused by interwire coupling capacitance [7], [9], [13] which is a technologically growing factor in *on-chip* buses ("thin and tall" wires), while remains a second-order effect in off-chip buses.

This paper introduces a novel point of view in off-chip bus encoding, which is not embraced by Ramprasad's baseline scheme. The method, that we call bus-switch (BS), is based on dynamically reordering the bus lines, to obtain a physical bit vector as similar as possible to the previous one. The method has been functionally simulated in C language and implemented in synthesizable Verilog targeting 0.13-μm and 0.18-μm CMOS standard cell libraries [10], [11]. Here we report the technique, the switching activity analysis and the postayout power consumption analysis.

II. The Bus-Switch Coding Mechanism

In principle, the BS technique can be *logically* expressed as a four-step process.

1) The bus is organized in clusters of M lines each.
2) Each M-line cluster is tentatively encoded by swapping the input lines using a particular *sorting pattern*.
3) An encoded data is obtained by applying a fixed *coding function* to the swapped M lines.
4) The process is repeated $M!$ times from step 2 until the optimal sorting pattern is found, that minimizes the output switching activity in the encoded data of the whole bus.

In the following, a formal definition of the process is given. Let $b(t)$ be input data word to the bus encoder and $B_{\rm opt}(t)$ the encoded data word on the bus, at clock cycle t. The single bits of any M-bit data word $x(t)$ will be indicated as $x(t)(0), x(t)(1), \ldots, x(t)(M-1)$.

Definition 1: A sorting pattern $p(t)$ is an ordered set of M indices $i_0, \ldots, i_{M-1}$, associated with clock cycle t. Given a data word $x(t)$, the swap operator S with sorting pattern p is a combinational logic function producing a swapped data word $y(t) = S(x(t), p(t))$, such that

$$y(t)(0) = x(t)(i_0)$$
$$y(t)(1) = x(t)(i_1)$$
$$\ldots$$
$$y(t)(M-1) = x(t)(i_{M-1}).$$

As an example, if $p(t) =$ "$1, 2, 3, 0$" and $x(t) =$ "0100" then $S(x(t), p(t)) =$ "1000" (index 0 is rightmost digit). Note that each sorting pattern $p(t)$ has a unique inverse $p^{-1}(t)$, such that. $x(t) = S(S(x(t), p(t)), p^{-1}(t))$ For instance, if $p(t) =$ "$1, 2, 3, 0,$" then $p^{-1}(t) =$ "3 0 1 2."

Manuscript received August 29, 2003; revised March 31, 2004.

The authors are with the Department of Electronic Engineering, University "La Sapienza," 00184 Rome, Italy, and also with STMicroelectronics, 95121 Catania, Italy (e–mail: olivieri@die.uniroma1.it; francesco.pappalardo@st.com; giuseppe-ast.visalli@st.com).

Digital Object Identifier 10.1109/TVLSI.2004.837998