Interconnect Design Methods for Memory Design*

Chanseok Hwang
Department of Electrical Engineering-Systems
Univ. of Southern California, Los Angeles, CA 90089, USA
Tel : 1-213-740-4472 Fax : 1-213-740-9803
chanseoh@usc.edu

Massoud Pedram
Department of Electrical Engineering-Systems
Univ. of Southern California, Los Angeles, CA 90089, USA
Tel : 1-213-740-4458 Fax : 1-213-740-9803
pedram@usc.edu

Abstract - This paper presents a solution to the problem of designing interconnects for memory devices. More precisely, it solves the automatic routing problem of memory peripheral circuits as an over-the-cell channel routing problem under pre-specified routing topologies and performance constraints. The proposed routing method, named TANAR, consists of two steps: a performance-driven net partitioning step, which constructs a routing topology for each net according to performance constraints, and a performance-driven track assignment step, which reduces the crosstalk noise. Experimental results demonstrate that TANAR significantly reduces both crosstalk for noise sensitive nets, and delay for timing critical nets while minimizing channel height.

1. INTRODUCTION

Memory devices are perhaps the most common components on integrated circuits. They are generally classified into DRAM, SRAM, Mask ROM and Flash Memory according to how they store data. Memory devices comprise of a memory arrays and peripheral circuits. The memory arrays can be designed manually because of its highly regular layout structure. In contrast, the peripheral circuits pose a significant challenge to memory designers. Figure 1 shows the typical architecture of a 1GB DRAM. Although area of the peripheral circuits is only about 20 % of the whole chip area, it normally takes much longer to design the peripheral circuits compared to the memory array. This is because layout designers must carefully design the peripheral circuits to avoid crosstalk noise and meet critical path delay, often by exploiting a full-custom design flow. In addition, they must pay close attention to layout area minimization, which is quite important especially for mass-produced integrated circuits such as the memory devices.

We next describe a number of key facts related to the design of peripheral circuits for memory devices. Address and data signals should be routed with the shortest connections and be immune from crosstalk noise. Analog signals used for voltage converter circuitry should be shielded from crosstalk noise. Over-the-cell area should be effectively exploited in order to reduce the overall chip area. Notice that in these peripheral circuits, the size of transistors is large in order to provide a high-current drive strength [1]. In practice, the logic cell height in peripheral circuits tends to be about 100um whereas the standard cell height in ASIC designs is about 10um. Finally, the process technology used to fabricate memory devices (especially for DRAM devices) provides only two or three metal layers in order to reduce production cost.

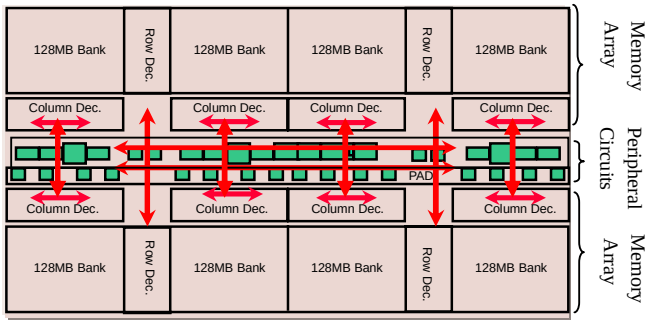


Figure 1. An example of 1GB DRAM architecture

In this paper, we solve the automatic routing problem of peripheral circuits with performance constraints as a performance-driven over-the-cell channel routing problem. The reasons for this approach are:

1. The crosstalk problem can be solved more easily in channel routing compared to area routing. This is because of the fact that wire segments in channel routing are processed track-by-track. This in turn allows us to do an efficient and accurate analysis of crosstalk correlations between neighboring nets.
2. The layout shape of a typical peripheral circuit is a long rectangle, which makes it more amenable to channel-based routing.
3. Memory devices use only two or three metal layers (except for high-speed SRAM devices which tend to use five or more metal layers). This makes channel routing an especially effective technique for routing the memory devices. Notice that channel routing is not normally used for multi-layer (more than 3-layers) routing problems, because its routing efficiency tends to be lower than that of area routing. However, for two or three layers routing problem, channel routing algorithms [2-4] have an edge over the area routing.

The remainder of the paper is organized as follows. Section 2 introduces the models of fixed net topology, timing, crosstalk noise, and cell layout style used in this paper. The proposed routing algorithm and some extensions are described in section 3 and 4, respectively. Experimental results are shown in Sections 5. We shall draw concluding remarks in Section 6.

* This research was supported in part by NSF under grant no. 9988441.

2. PRELIMINARIES

2.1 Fixed Topology Model

The peripheral circuits include nets with various performance constraints. To simultaneously meet these constraints and achieve our design goals, we introduce the notion of a *fixed topology net*, that is, a net with the predefined routing topology. In particular, we define four types of fixed routing topologies according to performance constraints. First, we classify signal nets in peripheral circuits into four types:

1. Delay critical and crosstalk noise sensitive nets (called *critical nets* and denoted by N_C): Signal nets that belong to address and data signal nets.

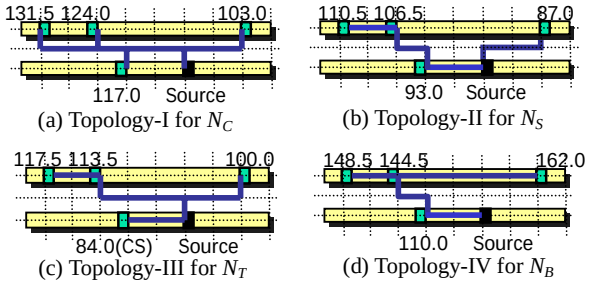
2. Crosstalk noise sensitive nets (called *sensitive nets* and denoted by N_S): Signal nets that reside in the circuitry of pre-charge, domino logic or are used to carry low voltage swing signals (especially for analog nets).

3. Delay critical nets (called *timing nets* and denoted by N_T): Signal nets that lie on the timing-critical paths.

4. Non-critical digital nets (called *base nets* and denoted by N_B): Signal nets that have sufficiently positive noise margin and timing slack so that the interconnect topology in channel routing for the nets does not affect the circuit performance.

Each type of nets has its corresponding routing topology. Figure 2 shows four types of routing topology with source-to-sink delay for each sink, and table 1 summarizes critical sink delay and consumed channel area for each of these routing topologies. Delay is calculated by using the Elmore delay [5] model to be reviewed in Section 2.2. For simplicity, we assume unit capacitance and resistance per unit length. The grid marks in this figure represent the unit length. The source has a drive resistance of 5 units, and all sinks have a load capacitance of 1 unit. The entries in the second row of table 1 correspond to the critical sink delay from a source, and the amount of delay increase compared to topology III.

Note that the routing results significantly vary depending on the routing topology that is being used. Topology-I for N_C has the single track from which branches connect to the pins. This topology improves the signal propagation and routability [6] because the bus line is routed with the shortest straight line and all the sinks have rather similar delays from the source. To minimize crosstalk noise, the routing topology-II, which results in the minimum total routing length, is desirable. This is because the coupling capacitance that causes crosstalk is proportional to the total coupling length, which will in turn be shorter if the total routing length is shorter. The delay between a source and a specific *critical sink* of a multi-terminal net is minimized via topology-III. Critical paths in peripheral circuits are usually well defined. Therefore, minimizing the critical sink delay is more effective in reducing the critical path delay than minimizing the maximum delay of a net.



(a) Topology-I for N_C (b) Topology-II for N_S

(c) Topology-III for N_T (d) Topology-IV for N_B

Figure 2. Routing solutions corresponding to different topologies for a net comprised of a source and four sinks.

TABLE 1. Comparisons of routing performances for four routing topologies.

Topology type	I	II	III	IV
Critical sink delay	117 (39%)	93 (10%)	84	110 (30%)
Total routing length	12	11	12	12
Consumed channel area	12	7	8	3

Topology-IV can significantly reduce channel height for a circuit with large cell heights and many base nets, which is the case for peripheral circuits of memory devices. Note that although the total routing length is nearly the same for the four topologies, the delay and routing length used inside the channel are quite different. In this paper, we use these observations by enforcing the notion of *fixed net topologies*, that is, each net has its optimal routing topology according to its performance constraint.

2.2 Delay Model

When computing interconnect delay, we use the Elmore delay [5]. Thus we briefly review this model here.

Elmore Delay Calculation: Consider a routing tree T for net n_i. Each edge e_v of the tree is modeled as a π-type RC circuit comprised of a resistor r_{ev} and two capacitors $c_{ev}/2$, where r_{ev} is the total wire resistance and c_{ev} is the total capacitance of edge e_v. The Elmore delay from source p_0 to sink p_i is calculated by:

$$D(p_0, p_i) = R_d C_{p0} + \sum_{e_v \in \text{path}(p_0, p_i)} r_{ev} (c_{ev}/2 + C_v) \qquad (1)$$

where R_d is the output drive resistance at the source of T and C_v is the total sub-tree capacitance at T_v.

2.3 Crosstalk Model

When designing crosstalk noise immune interconnection, we use the peak crosstalk noise amplitude equation proposed in [7]. This equation shows high accuracy and simplicity for its application to layout techniques like routing. For a victim RC tree coupled to an aggressor, the peak noise at any node is calculated by:

$$PN = \frac{\sum_{R_i \in \Omega} C_{xi} R_i}{\sum_{C_i \in \Lambda} C_i R_{ii}} \qquad (2)$$

where Cx_i is the sum of downstream coupling capacitances seen from node i, Ω is the union of the victim driver resistance and the set of resistances in the unique path from the root to the node, Λ is the set of all capacitors, and R_{ii} is the resistance seen across C_i with all capacitors open circuited (refer to [7] for details). Figure 3 shows two on-chip lines running parallel on the same metal layer and its equivalent circuit. We assume line 2 is quiescent (as a victim net) when line 1 is switches (as an aggressor net). The peak noise voltage on line 2 caused by crosstalk between two lines is expressed by using the equation (2). The lumped π model is used to model the interconnection. *Cx* denotes half of the total coupling capacitance. *C1* and *C2* are half of the line capacitances whereas *C3* and *C4* denote sums of the respective half line capacitances and receiver capacitances. *R1* is the aggressor driver resistance, *R2* is the victim output resistance, and *R* is the line resistance [7].

Switching window is in general one of the key factors when crosstalk noise effects are considered. In memory circuit design, however, noise sensitive nets are mostly analog signals such as reference voltage signals with small voltage swing. As a result, we may assume that all noise sensitive nets must be kept immune to crosstalk noise at all times.

2.4 Cell Model

It is necessary to efficiently exploit over-the-cell area and thereby minimize channel height. This is especially important for peripheral circuits of the memory devices, which have large cell height. The efficiency of routing algorithms is largely dependent on the physical constraints that exist over the cell. Figure 4 depicts the cell model of peripheral circuits. This is often called "target-based cell" and is carefully designed to allow effective utilization of over-the-cell area during channel routing. The model of Figure 4 is the same as that of the Target-Based Cell [9] except that here power and ground lines are implemented in M2 layer. Pins are in the form of long vertical strips in M1 layer. In addition, the cell is filled with M1 obstructions resulting from the internal wires, and dummy features that are added to achieve oxide planarization [10]. M2 can be therefore freely used over-the-cell area except where the power and ground lines are, whereas M1 is forbidden in the over-the-cell area except for where the pins are. In the case of using M3 layer, the cell model may be changed (described in Section 4).

3. PROPOSED ROUTING ALGORITHM

The automatic routing problem of peripheral circuits of memory devices can be defined as follows.

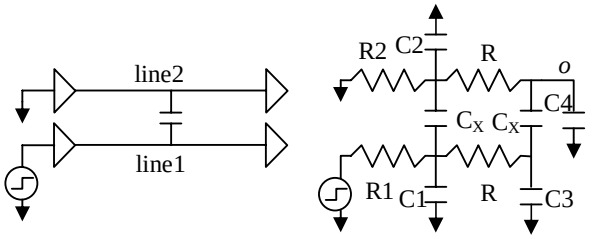


Figure 3. Noise coupling (a) Circuit configuration (b) Equivalent circuits; the peak noise is calculated by using (2) is as:

$$PN = \frac{(2R_2 + R)C_X}{R(C_1 + C_3 + 2C_X) + R_2(C_2 + C_4 + 2C_X) + R(C_3 + C_4 + 2C_X)}$$

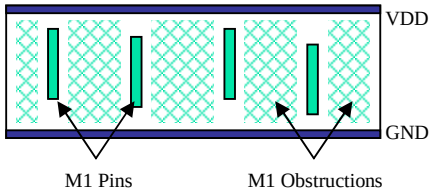


Figure 4. Target based cell

Problem Statement: Given a set of nets *N* to be routed inside a channel using two or three routing layers, the delay and noise information, do detailed routing of *N* such that the performance constraints are satisfied and channel height is optimized.

TANAR solves the given problem by applying two steps: a performance-driven net partitioning (PDNP) step, which decides a fixed topology for each net according to its net type and partitions the topology in order to build it during the routing, and a performance-driven track assignment (PDTA) step, which satisfies the delay and noise constraints while minimizing channel height.

3.1 PDNP Problem and Algorithms

The objective of the PDNP step is to construct an optimal routing topology for N_C, N_S, N_T and N_B, respectively, and then partition a net into sub-nets to build it during channel routing. The four types of nets have different performance constraints; therefore, the routing topology of each type of net should be optimized so that it satisfies the corresponding constraint. We specify the optimal routing topology for each net type and algorithm used to build the topology next.

- **BRT for N_C:** Given a net $n \in N_C$ with pins p_i (i = 1, 2, 3, …) in a channel, the Bus Routing Tree (*BRT*) is a steiner tree *T* which consists of a single track and a number of branches at steiner points on the track connecting p_i's. BRT can be constructed by laying a single track from the left-most pin to the right-most pin of the net in a channel during PDTA and connecting pins to the track.
- **MST for N_S:** Given a net $n \in N_S$ with pins p_i (i = 1, 2, 3, …) in a channel, the MST algorithm *MST(n)* finds a spanning tree *T* which minimizes the total length of the *T*.

This is because crosstalk noise is caused by the coupling capacitance, which is proportional to the coupling length; therefore, the routing topology for N_S should be optimum in terms of total length. The well-known Prim's algorithm of [11] is used for *MST(n)*.

- **CSRT for N_T:** Given a net $n \in N_T$ with a source p_0, a critical sink p_c and other sinks p_i (i = 1, 2, 3, …) in a channel, the Critical Sink Routing Tree algorithm *CSRT()* finds a spanning tree T which minimizes the Elmore delay from p_0 to p_c in the T. CSRT() originates from the ERT algorithm of [12] which generates a spanning tree such that the maximum Elmore delay from source to any sink is the minimum. CSRT() starts with a MST ($T - p_c$) and then connects p_c to T so that adding edge (p_c, p_i) or (p_c, p_0) yields a tree with the minimum critical sink delay from the source.

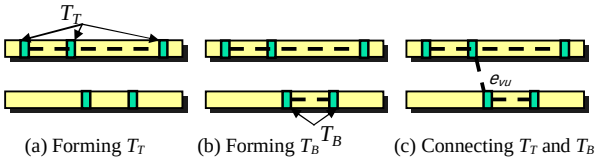
- **MCAT for N_B:** Given a net $n \in N_B$ with pins p_i (i = 1, 2, 3, …) in a channel, the Minimum Channel Area Tree algorithm *MCAT(n)* finds a spanning tree T which consumes the minimum channel area. Figure 5 shows the process of constructing this tree. *MCAT(n)* works as follows. First, it adds edges between top-pins P_T in the upper row, which forms the top-tree T_T (see Figure 5(a)). Next it adds edges between bottom-pins P_B in the lower row, which forms the bottom-tree T_B (see Figure 5(b)). Finally, it finds a pair of pins (u, v) with the minimum length, where $u \in T_T$, $v \in T_B$ (see Figure 5(c)).

Notice that BRT, CSRT, MST and MCAT correspond to topology-I, II, III and IV, respectively in Figure 2. In Figure 7, we illustrate the PDNP algorithm with an example. The algorithm first constructs T_{MST}, T_{CSRT}, and T_{MCAT} via MST(.), CSRT(.), and MCAT(.) for N_T, N_S, and N_B, respectively. We impose the upper bound on the increase in delay of the base nets. Therefore, the PDNP algorithm compares the delay of T_{MCAT} with that of T_{MST}. It will then use T_{MCAT} for nets in N_B only if the delay increase is equal to or less than some fixed threshold (say 30%). Otherwise, it uses T_{MST} for nets in N_B. Next, it partitions T into sub-trees T_{sub} in such a way that each edge e of T_{CSRT} and T_{MST} forms a sub-tree, resulting in a sub-net with two pins, and T_T, T_B and e_{vu} of T_{MCAT} forms each their sub-tree, generating three sub-nets. These sub-nets are independently assigned to a specific track during the track assignment. Figure 6 shows the results of partitioning a tree into sub-trees for these three trees. Note that BRT is not partitioned into sub-trees such that the whole pins are connected to the single track.

3.2 PDTA Problem and Algorithm

The objective of the PDTA is to assign nets to tracks in a channel so as to minimize the channel height while meeting the horizontal constraints (HC), vertical constraints (VC) and noise constraints (NC).

HC: If two horizontal segments with overlapping spans belong to different nets, then they should not be assigned to the same track.



(a) Forming T_T (b) Forming T_B (c) Connecting T_T and T_B

Figure 5. The process of constructing MCAT

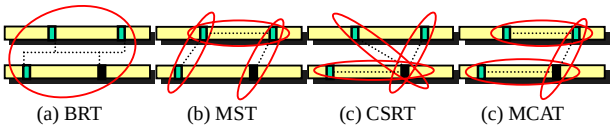

(a) BRT (b) MST (c) CSRT (c) MCAT

Figure 6. The results of partitioning a tree into sub-trees for four different routing trees

Algorithm Performance-Driven Net Partitioning (PDNP)
Given nets classified into three types: N_C, N_S, N_T and N_B
Begin
1. Construct a routing tree for each net $n \in N$ {
  If $n \in N_S$ **Then** Tree T_{MST} = MST(n);
  ElseIf $n \in N_T$ **Then** Tree T_{CSRT} = CSRT(n);
  ElseIf $n \in N_B$ **Then**
    T_{MST} = MST(n), T_{MCAT} = MCAT(n);
    Calculate the maximum delays of the two trees;
    If percentage of delay increase ≤ 30%
      Then Select T_{MCAT};
    Else Select T_{MST};
  EndIf
}
2. Partition T into sub-trees{
  If $T = T_{CSRT}$ or T_{MST}
    Then Generate sub-trees for each edge of T;
  Else /* $T = T_{MCAT}$ */
    Generate sub-trees for T_T, T_B and e_{sub}, respectively;
  EndIf

Figure 7. The performance-driven net partitioning algorithm

VC: When two pins belonging to different signal nets are in the same column, the track to which the upper pin in that column connects must be placed above the track to which the lower pin in that column connects.

NC: $PN(i) \le AN(i)$ for a net $n_i \in \{N_C, N_S\}$ should be satisfied, where $PN(i)$ is peak crosstalk and $AN(i)$ is the maximum allowable peak crosstalk noise for a net n_i.

The PDTA algorithm proceeds track by track from the over-the-cell areas towards the middle of channel. Figure 8 shows the order of tracks used by this algorithm, which is based on the work by Yoeli in [3]. The rationale for this track ordering is that over-the-cell areas can be used to the highest degree (tracks of the over-the-cell areas in the upper and lower rows are utilized first). In addition, unlike YACR2 [13] which explicitly constructs the vertical constraint graph, vertical constraint violations (VCV) can be simply checked and avoided by using a simple heuristic method [3]. In Figure 9, we present the new PDTA algorithm. It works as follows: first, subnets are sorted in ascending order of their leftmost end-points, then an unprocessed track T as per sequence of tracks (described

before) is picked. The algorithm then places the horizontal segment of a subnet having the lowest left-end position if placing the net to the current track satisfies all constraints such as HC, VC and NC. Otherwise, the next subnet with the lowest left-end position is attempted. This process is repeated until all subnets are assigned to tracks. Unassigned subnets may exist in real fields because of channel area shortages or cyclic VCs. To solve those problems, we first attempt to place the unassigned subnets to empty spaces of processed tracks with considering only HC and NC. If the unassigned subnets still remain, then we should increase the channel area and add a new track to the center of the channel, and then repeat the above process.

After the PDTA step, to connect pins to the track we used the maze routing routines presented in YACR2. Due to space limitations, we do not describe the maze routing routines in this paper; please see [13] for details.

4. EXTENSIONS

When a M3 layer is available for routing, we have limitations as to how we can exploit M3 with the cell model (cf. Figure 4). More precisely, the pin layer is M1, which in turn means that a HVH routing scheme cannot be used. Therefore, when connecting to an M1-pin from an M3-track, this connection may have a conflict with neighboring M2 wires. As a result, to improve the three-layer routing efficiency the pin layer should be changed from M1 layer to M2 layer. In such a case, the power/ground pins inside the cell also must be changed from M2 layer to M1 or M3 layers. In addition, in order to use HVH routing scheme, the VC definition should be modified as described in [3]; if at any column, the net, which enters the channel from the bottom, was assigned to track r, and the net, which enters the channel from the top, was assigned track r+1, then this is also considered a vertical-constraint violation.

Although our algorithm can be applied to the three-layer routing with the considerations given above, the current implementation only addresses the two-layer routing problem.

5. EXPERIMENTAL RESULTS

TANAR is implemented in C++ on an Ultra-2 Sun Spark workstation, and tested on the Deutsch difficult problem [14] and five other randomly generated benchmark circuits. The characteristics of the channel routing problems are summarized in Table 2. The circuits are generated with 100 nets to 500 nets. We assume that in each benchmark circuit, the member count of N_C, N_S, N_T and N_B is 10%, 15%, 25% and 50% of the total nets, respectively. The percentages for each net type are considered based on the peripheral circuits. Nets are randomly assigned a type to meet the aforementioned distribution profile.

The maximum allowable peak noise *AN* is specified for nets in N_C or N_S, that is, the 25% of all nets in each

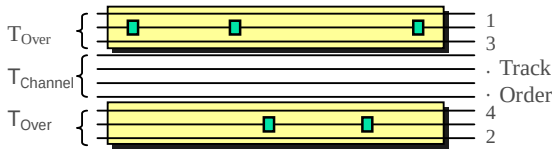


Figure 8. Track ordering

Algorithm Performance-Driven Track Assignment (PDTA)
/* Assume that there is no layout area limitations and cyclic VCs */
Begin
1. Sort all subnets in the increasing order of their leftmost end positions;
Repeat
2. Select a track *T* from the over-the-cell toward the middle of a channel;
3. Select the next subnet *n* with the lowest left-end position;
  If *n* satisfies HC, VC and NC
    Then Begin
      Place *n* on the current track *T*;
      Delete *n* from the sorted list;
      End
    Else Goto 3
  EndIf
Until all subnets are assigned
End

Figure 9. The performance-driven track assignment algorithm

benchmark circuit have the noise constraint, ranging from 10% to 30% of supply power voltage(we used uniformly distributed *AN* values in that range). In addition, we randomly selected the critical-sink among sinks of nets in N_T.

We compared two channel routing flows, TANAR (PDNP+PDTA) and CONV (NP+TA), in terms of the peak crosstalk noise for N_C and N_S, the critical-sink delay for N_T, and the number of tracks in the channel and over-the-cell. The conventional flow (CONV) consists of a MST-based partitioning step (NP) followed by a noise unaware track assignment step (TA). More precisely, the NP step partitions all nets into two-pin subnets using the MST topology in the same way as noise sensitive nets are partitioned by PDNP. TA used the same procedure as that employed in PDTA except it does not consider any noise constraints. As a result, the CONV flow tries to minimize the total routing length and the number of tracks used in the channel while satisfying the HC and VC.

TABLE 2. The characteristics of test examples

Test Examples	# of Nets	# of Columns	Channel Density
Deutsch	72	174	19
Rand100	100	220	20
Rand200	200	480	39
Rand300	300	720	60
Rand400	400	870	81
Rand500	500	1200	90

The results for the comparison are shown in Table 3 for test examples. Note that, compared to CONV, TANAR reduces the peak crosstalk noise for all sensitive nets by 40% or more for all test examples. The critical-sink delay from a source for timing-critical nets estimated using

HSPICE with TSMC 0.18µm/1.8v model: wire resistance 0.078Ω/µm, coupling capacitance 0.1fF/µm and bottom capacitance 0.02fF/µm. Furthermore, we assumed that the source of each net is driven by an inverter (Wp = 1.74µm, Wn = 1.2µm, Lp= Ln = 0.18µm), and each sink has 3.0 fF for the load capacitance. TANAR improves upon CONV by reducing this delay from 4% to 16% (depending on the test case). In addition, TANAR reduces the number of tracks used inside channel by effectively using over-the-cell tracks. Note that the relative delay performance of TANAR improves as the benchmark size increases.

In summary, the experimental results demonstrate that the proposed *fixed routing topology* for each net according to its net type and performance constraints works effectively for the channel routing problem. As a result, the key issues in the interconnect design of peripheral circuits for memory devices have been addressed by the proposed algorithm.

6. CONCLUSIONS

We solved the automatic routing problem of memory peripheral circuits as an over-the-cell channel routing problem with a novel interconnect design technique. Our approach handles noise sensitive nets and timing critical nets at the same time while efficiently utilizing the over-the-cell area. Experiments showed that the proposed routing algorithm reduces the routing channel height and at the same time reduces the peak crosstalk noise for noise sensitive nets and the critical-sink delay for timing critical nets. To the best of our knowledge, this is the first work that simultaneously reduces the crosstalk noise, delay and area during the over-the-cell channel routing.

References

[1] S. Shibatani, T. Sadakane, H Nakao, M. Terai and K. Okazaki, "A CMOS cell generation system for two-dimensional transistor placement," *Proc. IEEE Custom Integrated Circuits Conf.*, pp. 325-328, 1998.

[2] D. Braun, J. Burns, S Davadas, H. Ma, K. Mayaram, F. Romeo and A. L. Sangiovanni-Vincentelli,, "Chameleon: a new multi-layer channel router," *Proc. ACM/IEEE Design Automation Conference*, pp. 495-502, 1986.

[3] U. Yoeli, "A robust channel router," *IEEE Trans. Computer-Aided Design*, vol.10, 1991.

[4] R. Gidwani and N.A. Sherwani, "MISER: An integrated three layer gridless channel router and," *Proc. IEEE International Conf. Computer-Aided Design*, pp. 698-703, 1990.

[5] W. C. Elmore, "The transient response of damped linear network with particular regard to wideband amplifiers," *J. Applied Physics*, vol. 19, pp 55-63, 1948.

[6] H. P. Tseng, L. Scheffer and C. Sechen, "Timing- and crosstalk-driven area routing," *Proc. ACM/IEEE Design Automation Conference*, pp. 378-382, 1998.

[7] A. Vittal, L. H. Chen, M. Marek-Sadowska, K.P Wang, and S. Yang, "Crosstalk in VLSI Interconnections," *IEEE Trans. Computer-Aided Design*, vol.18, pp. 1817-1824, 1999.

[8] D. A. Kirkpatrick and A. L. Sangiovanni-Vincentelli, "Digital sensitivity: Predicting signal interaction using functional analysis," *Proc. IEEE International Conf. Computer-Aided Design*, pp. 536-541, 1996.

[9] B. M. Goni and T. Arslan, An Evolutionary 3D Over-the-Cell Router, *IEEE International ASIC/SOC Conference*, pp. 206 – 209, 1999.

[10] I. Ali, S. Roy and G. Shinn, "Chemical-mechanical polishing of interlayer dielectric: A review," *Solid State Technology*, 37(10), pp. 63-70, 1994.

[11] A. Prim, "Shortest connecting networks and some generalizations," *Bell System*, Tech. J., vol. 36, pp. 1389-1401, 1957.

[12] K. D. Boese, A. B. Kahng, B. A. McCoy and G Robins, "Near-optimal critical sink routing tree constructions," *IEEE Trans. Computer-Aided Design*, vol.14, pp1417-1436, 1995.

[13] J. Redd, A. Sangiovanni-Vaincentelli, and M. Santomauro, "A new symbolic channel router: YACR2," *IEEE Trans. Computer-Aided Design*, vol. CAD-4, pp. 208-219, 1985.

[14] Burstein and R. Pelavin, "Hierarchical channel router," *Integration VLSI J.*, vol. 1, pp. 21-38, 1983.

TABLE 3. Routing results comparison between TANAR and CONV

Test Examples	TANAR				CONV			
	Peak Noise(V) for *Ns* and N_C	CS-Delay(ps) for N_T	# of Used Tracks		Peak Noise(V) for *Ns* and N_C	CS-Delay (ps) for N_T	# of Used Tracks	
			In-the-Channel	Over-the-Cell			In-the-Channel	Over-the-Cell
Deutsch	0.17 (67.0%)	28.9(4.0%)	13	24	0.53	30.4	16	15
Rand100	0.06 (40.9%)	26.5(5.6%)	16	27	0.10	28.1	17	14
Rand200	0.20 (67.7%)	33.8(7.3%)	25	45	0.62	36.5	26	28
Rand300	0.36 (60.0%)	36.9(7.7%)	44	77	0.90	40.0	46	40
Rand400	0.41 (62.0%)	40.7(13.5%)	41	86	1.08	47.1	45	47
Rand500	0.52 (57.4%)	46.4(16.5%)	54	92	1.22	55.6	56	56