Transactions D:
Computer Science & Engineering and
Electrical Engineering
Vol. 17, No. 1, pp. 11–24

SCIENTIA IRANICA

Higher Routability and Reduced Crosstalk Noise by Asynchronous Multiplexing of On-Chip Interconnects

A. Jahanian[1] and M. Saheb Zamani[2,*]

Abstract. *The signal integrity problem, especially crosstalk noise, is an important issue in physical design in the nanometer regime. Wire multiplexing is a recently proposed method to reduce these problems in current design methodologies. This technique is presented to increase the routability of the design and also reduce crosstalk noise by serializing parallel wires via delay insensitive asynchronous serial transceivers. In this paper, this technique is improved in terms of routability and computation time and, also, its impact on crosstalk is examined. Finally, it is evaluated with 180 nm and 130 nm technologies. Experimental results show that for the attempted benchmarks, congestion and routability are improved by 15.54% and 21.57% on average, respectively, and crosstalk noise is improved by 9.51% on average. These improvements are achieved at the cost of a slight increase in power consumption (0.12% on average) and runtime (less than 10.01% on average).*

Keywords: *Asynchronous serial transmission; Crosstalk; Routability.*

INTRODUCTION

Signal integrity is an important metric for the quality of a design and it has become dominant for large and complicated designs. In a congested design, many wires have to be detoured at the global routing stage and a large number of long wires are generated. These global wires can have a great impact on the routability and performance of the design and may also result in crosstalk and manufacturability problems. Therefore, congestion and crosstalk noise reduction have a key role in the performance, manufacturability and signal integrity of designs.

Routing congestion (or congestion) of a layout rectangular region (bin) is defined as the number of used routing tracks divided by the total number of routing tracks on the perimeter of the bin. If the congestion of a bin goes beyond 100%, those internal nets of the bin, which should be connected out, are not routable. In this situation, the design is not routable. Therefore, routing congestion and routability are dependent concepts.

Asynchronous serial transceivers can be used to improve congestion and routability in application specific integrated circuit design methodology. With the emergence of the globally asynchronous locally synchronous concept [1], some techniques for using asynchronous serial transceivers in Network-On-Chip (NOC) systems have been proposed [2]. These systems were mainly used to improve the data transmission delay between cores. Conventional asynchronous circuits used differential-signaling serial links that made them inappropriate for on-chip applications. This is because they require PLL-based clock and data recovery circuits, which consume excessive power and area [3].

In [4], an NOC with a new serialization mechanism was proposed, which provides high performance serialization with small overhead. In this approach, the data transmission bandwidth is automatically adjusted based on the workload of the network. This method was implemented and tested in 0.18 μm technology at 3 Gb/s.

In [5], the authors proposed a serial link transceiver for global on-chip communications, working

1. *Department of Electrical and Computer Engineering, Shahid Beheshti University, G.C., Tehran, P.O. Box 19839-63113, Iran.*
2. *Department of Information Technology and Computer Engineering, Amirkabir University of Technology, Tehran, P.O. Box 15875-4413, Iran.*
*. *Corresponding author. E-mail: jahanian@sbu.ac.ir*

Received 11 October 2009; received in revised form 14 December 2009; accepted 29 December 2009

at 3 Gb/s per wire in a standard 0.18 μm CMOS process, with low crosstalk-induced delay variability. They implemented a pure electrical link without any repeater in higher layers of metals.

Teifle and Manohar [6] presented a high-speed and clockless serial link transceiver for inter-chip communication. Their experiments showed that their transceiver operates at up to 3 Gb/s in 0.18 μm CMOS technology.

Dobkin et al. [7-9] presented a novel bit-serial interconnect structure with a new data encoding style to encode each bit in one transition in a pipelined structure. They showed that their asynchronous circuit can send or receive a new bit at a data cycle of a single FO4 (i.e. fan-out of 4) inverter delay.

The authors of this paper present a new approach for serializing the parallel wires using an asynchronous serial transceiver to improve routing congestion and the crosstalk noise of designs [10]. This mechanism was called metro-on-chip (MOC) because its semantic is similar to metro transportation systems in large cities that convey many passengers in one trip. In this methodology, non-critical and sufficiently long wires are selected and serialized at the signal sources and then, they are demultiplexed into parallel signals at the destinations after serial transmission. Finally, these demultiplexed signals travel to their sinks. It is worthwhile to note that a congestion metric is a good metric, which can be more useful if it is measured in terms of routability. This paper has two major improvements compared with [10]. The first is that in [10] no results were reported in terms of the routability of the design after MOC insertion. Due to importance of routability in modern complicated chips, routability analysis is very crucial for any proposed methodology in the physical design era.

It is noted that test coverage of multiplexed wires is lower than for parallel ones, because inserting a fault on multiplexed wire means that any of all the multiplexed lines may actually become stuck at that value.

In this paper, we improve the metro-on-chip mechanism in terms of routability and computation time and also examine its impact on crosstalk with a newer technology feature size.

OVERVIEW ON THE METRO-ON-CHIP

Metro-on-Chip Concept

In the conventional physical design flow, terminals of each net are connected via dedicated and specialized wires (Figure 1).

However, in new circuits which have a large number of nets, routing congestion raised issues in terms of routability, signal integrity and even the performance of the design. Multiplexing the nets in congested circuits

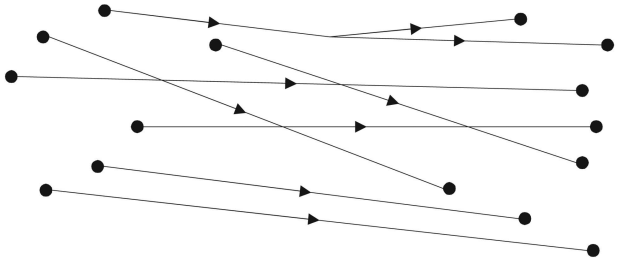

Figure 1. Conventional dedicated wires.

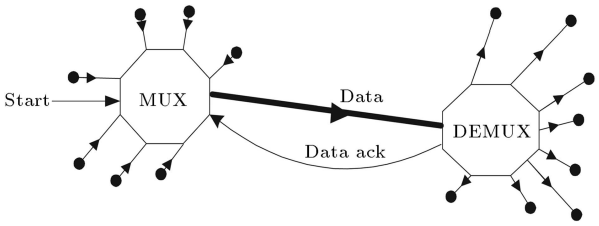

Figure 2. Serial multiplexed wires.

can be an effective technique to reduce congestion and improve routability. The main concept of this paper is to automatically find non-critical and sufficiently long wires and multiplex, and send them using asynchronous serial transmission hardware. This idea is similar to the metro system in large cities, which conveys many passengers in one trip to reduce traffic congestion. In this paper, a static paradigm of the metro is implemented in which the metro has fixed stations at the source and the destination without any intermediate stations (Figure 2).

Inserting the MOC cells (MOC multiplexers and demultiplexers) after detail or even global routing can impose large costs due to the need for updating the routing or placement results. On the other hand, MOC insertion at the earlier stages of physical design (e.g. partitioning or floorplanning) may not be effective because, at those stages, there is not enough information about cell locations and, therefore, the estimations of physical parameters may be very rough. In this paper, MOC modules are inserted after detailed placement where the locations of cells and white spaces have been determined.

The contributions of this approach are in decreasing routing congestion and total wirelength and also in increasing design routability by reducing the number of global nets. It should be noted that congestion reduction does not increase the performance of the design explicitly, but it may improve the timing characteristics by avoiding the detoured global nets resulted from routing congestion.

FAST SERIAL TRANSCEIVER

The fast serial communication mechanism proposed in [7] is used as the serial link hardware. The area,

delay and other characteristics of this system play a key role in the feasibility and quality of the proposed EDA design flow. In the following subsection, the hardware structure and transmission delay of this module have been briefly reviewed.

Serial Transmission System Review

The fast serial link transceiver [7] uses low-latency synchronizers at the sources and sinks with a two-phase non-return and a zero level encoded dual rails (LEDR) asynchronous protocol that allows non-uniform delay intervals between successive bits. Acknowledgment of transmission is returned only once per word, rather than bit by bit. This enables multiple bits to be transmitted in a wave-pipelined manner over the serial channel. The main structure of this mechanism is shown in Figure 3.

In [7], a high data rate shift-register structure for an asynchronous serializer and deserializer has been proposed. It operates at about one fan-out 4 inverter (FO4) delay and can be fully implemented in CMOS technology without any requirement to PLL or auxiliary clock generation circuitry.

Let MOC cluster, i, consist of m_i wires. The serial transmission hardware of this cluster contains a m_i-bit shift-register in the MUX module and also a m_i-bit shift-register in the DEMUX module. The total delay of MOC cluster, i, consists of five elements:

- Delay of wires from source nodes to the MUX input pins (input wires),
- Delay of multiplexing shift registers,
- Delay of transmission line from MUXes to DEMUXes,
- Delay of demultiplexing shift registers,
- Delay of wires from the DEMUX output pins to sink nodes (output wires).

In Figure 4, the five sub-elements of MOC delay are shown.

With a good clustering algorithm, source/sink nodes are close to the source/sink gravity centers and the delays of input/output wires are very small. Because of the pipeline structure of the serial link transceiver proposed in [7], each input signal can be applied to the MUX input while the previous signal is being multiplexed (except for the first input). Moreover, each output signal can be applied to its sink while the next signal is being demultiplexed (except for the last output). Therefore, total delay of the ith MOC cluster with m_i wires can be calculated as:

$$D_{(i)} = ID(i,1) + 2.m_i.\max(D_{\mathrm{FO4}}, TD(i)) + OD(i,m_i), \quad (1)$$

where $ID(i,1)$ is the delay of the first input signal, $OD(i,m_i)$ is the delay of the last output, $TD(i)$ is the transmission line delay in ith cluster and D_{FO4} represents an FO4 inverter delay. The first term in this equation is the delay of wires from source nodes to the MUX input pins. Because of overlapping the delay of shifting and transmission operations, the second term is maximum between the delay of one-bit shift register (D_{FO4}) and the transmission delay of one bit ($TD(i)$). Finally, the last term shows the delay of wires from the DEMUX output pins to sink nodes.

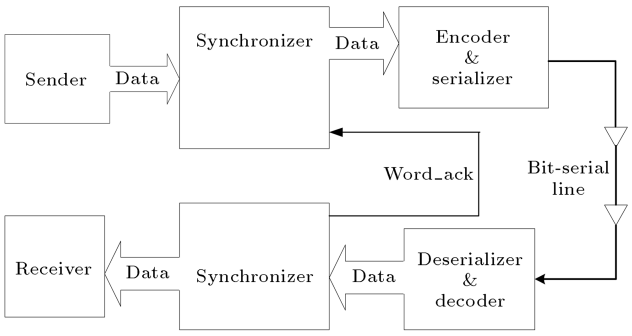

Figure 3. Fast bit-serial transmission scheme.

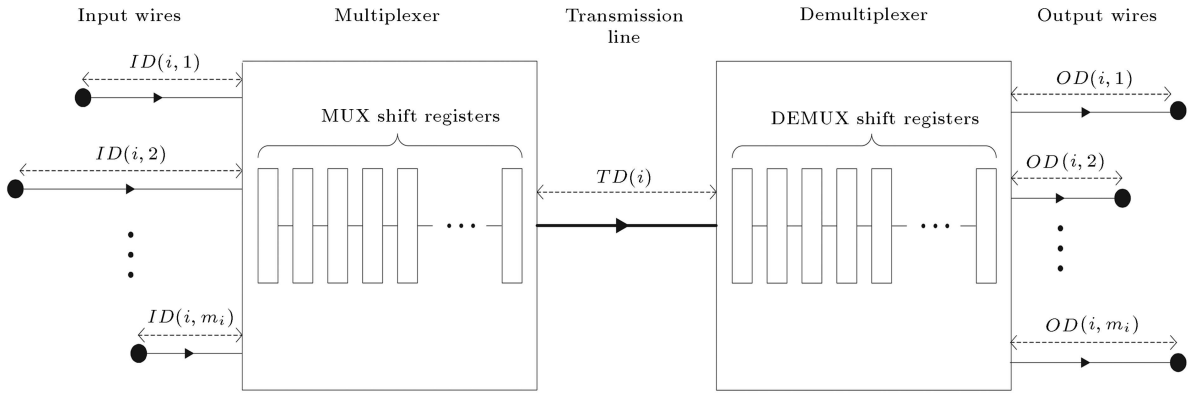

Figure 4. Delay sub-elements of cluster i.

Interfacing the MOC System in a General Design

Inserting an asynchronous module in a synchronous circuit may change the functionality of the whole system due to the violation of the setup and hold time requirements of synchronous registers. In this section, a feasible solution for interfacing the asynchronous transmission module and a synchronous circuit with a global clock is presented. In the proposed interface, on each rising edge of the system clock, the asynchronous serial link starts to convey a copy of signals at the source toward the destination and this operation is repeated for all input signals. At the end of the operation, a word-acknowledge signal is returned back to the sender. The process of serializing, transmission and deserializing is initiated by the activation of the start signal and the completion of the whole transmission process is indicated by '1' on word_ack signal. The signaling of the synchronous system after inserting the asynchronous serial link is shown in Figure 5.

In this mechanism, the clock period must be extended by the delay of MOC hardware if the paths containing the multiplexed wires do not have enough delay slack (i.e. their delays are close to the maximum clock period of the design). However, if the paths of the selected wires have enough delay slack, the multiplexing delay will not affect the critical path of the design. Therefore, the criticality of nets must be taken into account for inserting the MOC cells and multiplexing the wires. In our approach, only the wires which do not violate the design timing constraints after MOC insertion are chosen to be multiplexed.

MOC-BASED PHYSICAL DESIGN FLOW

In the proposed design flow, MOC modules are inserted after detailed placement where the position of cells have been determined and delay estimations are not too rough. The location of the MoC insertion stage is shown in Figure 6.

As highlighted in Figure 6, MoC insertion is performed after placement and before the routing stage. The flow of MOC insertion is shown in Figure 7.

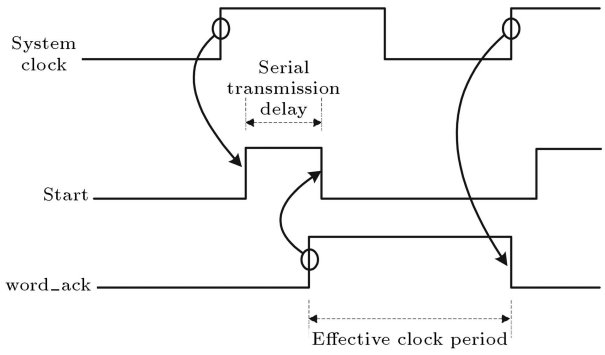

Figure 5. Signaling of serial link transmission.

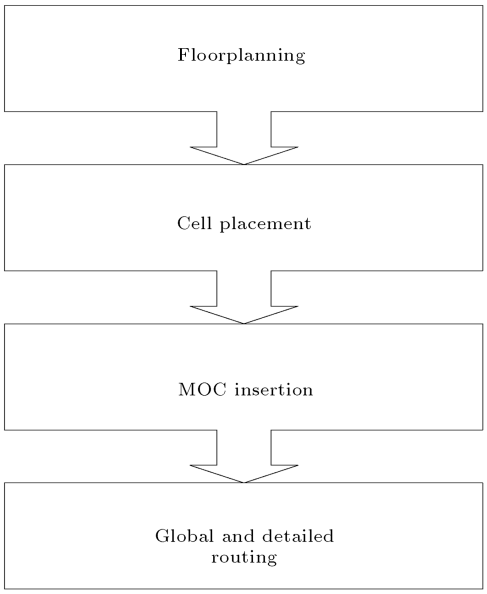

Figure 6. Location of MOC insertion in physical design flow.

MOC-based physical design flow	
A:	Floorplanning and placement
B:	Static timing analysis
C:	Select non-critical nets
D:	Cluster the multiplexed nets into MOC groups
E:	Repair the MOC groups
F:	Insert the MOC devices into the netlist
G:	Incremental placement of MOC devices

Figure 7. MOC insertion in the proposed physical design flow.

Floorplanning and Placement

In this stage, the design is floor-planned and the final locations of cells are determined. The quality of the placement algorithm has a great impact on the congestion and wirelength benefits of the MOC system. If the quality of the placer is poor, the number of long nets may be very high and, therefore, MOC insertion may result in large improvement; but these results may be misleading. Actually, MOC insertion benefits are more realistic using a placer with good localization features. Therefore, Dragon [11] is used for floorplanning and placement, which is a well-known placer with good results on total wirelength and congestion.

Static Timing Analysis

In this phase, a Static Timing Analysis (STA) is performed to estimate path delay. Before routing, the

topologies of nets have not been determined and they must be estimated for static timing analysis.

A Refined Single Trunk Steiner Tree (RSTST) model [12] is used to predict the topologies of nets before the routing stage. After estimating the topology of nets, the delay of each path in the design is calculated using the Elmore delay model and the critical paths of the design are determined. A critical path list will be used to select non-critical nets for multiplexing.

Select Non-Critical Nets

In this stage, a list of non-critical and sufficiently long wires is generated based on the results of the static timing analysis. After STA, a list of paths whose delays are greater than a specified threshold (T_d) has been generated. These nets cannot be multiplexed because they may violate the timing constraints of the design. Therefore, these nets are marked to be excluded from MOC groups.

The appropriate value for T_d has an important effect on the benefits of MOCs. In this subsection, an approximation technique to set its value is presented. T_d can be specified as:

$$T_d = T_w - \alpha.T_w, \tag{2}$$

where T_w is the most critical delay of the design (timing wall) and α is a number between 0 and 1, which defines a margin for near critical nets. In other words, the delay overhead of each MOC cluster must be less than $\alpha.\ T_w$. Therefore, all paths whose delays are greater than T_d are considered as critical paths and their nets are not selected for multiplexing. T_d must be adjusted by trial and error if no appropriate prediction could be made. However, an appropriate value is estimated for it based on the following analysis.

Consider cluster g_i with m_i nets. If the sources and sinks in g_i are sufficiently close to the MUX and the DEMUX, respectively, the total delay of g_i can be approximated as $2.m_i.D_{\mathrm{FO4}}$ in which D_{FO4} is one FO4 inverter delay. By this assumption, the following condition must be met to avoid any performance degradation in the design:

$$D_i \leq \alpha.T_w \Rightarrow m_i \leq \frac{\alpha.T_w}{2.D_{\mathrm{FO4}}}. \tag{3}$$

This equation shows the relation of average MOC cluster size and the delay threshold of the design. We used this approximation for choosing the value of α and cluster sizes. Table 1 shows the experimental results of this approximation.

In Table 1, *Approximated cluster size* shows the approximated values for cluster sizes, *Actual cluster size* shows the best found values for cluster sizes at the end of the test process, and column *Error* represents the absolute percentage of differences between approximated and actual cluster sizes. This table shows that the error of this approximation is small (7.01 on average) and Equation 3 can be used as a good initial approximation for m_i.

Cluster the Multiplexed Nets into MOC Groups

In this phase, the selected nets are clustered into some groups such that the nets in each group have close

Table 1. Approximated and actual cluster sizes.

Index	Circuit	TW (ps)	Approximated Cluster Size	Actual Cluster Size	Error (%)
1	spi	2524.24	0	0	0
2	tv80	4163.09	0	0	0
3	pci_bridge32	18727.58	31.21263	23.4	7.812
4	dma	18491.70	30.8195	24.5	6.319
5	b20_1	6961.51	11.60252	10.21	0.607
6	b22	10796.53	17.99422	18.75	0.755
7	b22_1	10233.41	17.05568	24.3	7.244
8	wb_conmax	8816.92	0	0	0
9	b17	27207.41	45.34568	26.55	18.795
10	b17_1	27291.20	45.48533	36.86	8.625
11	Ethernet	23330.71	38.88452	30.05	8.834
12	b18	73038.43	121.7307	96.7	25.03
Average					7.01

sources and close sinks. Let $G = \{g_1, g_2, \cdots, g_n\}$ be a set of MOC clusters, where cluster g_i has m_i nets. The *Companionship* of g_i, denoted as $CS(i)$, is defined as the cost of multiplexing the nets in g_i. $CS(i)$ can be calculated as $SrcD(i)+SnkD(i)$ where $SrcD(i)/SnkD(i)$ is the summation of distances between sources/sinks in g_i. The developed clustering algorithm, which uses Fibonacci min-heap [13], is shown in Figure 8.

At the start of the algorithm, each net is considered as a cluster with one net. First, a CS Fibonacci min-heap (FHeap) is generated showing the companionship cost of each pair of clusters. Then, two clusters with minimum CS are fetched from FHeap, merged in each loop of the algorithm and, then, the list of costs of cluster pairs in CS FHeap is updated. If the size of the merged clusters reaches a predefined offset, the cluster is marked as a final cluster. The algorithm is continued until there are no unmarked clusters. The complexity of this algorithm is $O(n^2 \log(n))$, which is better than the complexity of the clustering algorithm in [10] ($O(n^3)$).

Repairing the MOC Groups

After clustering the nets, the clusters are evaluated in order to gain more benefits from MOCs. In this paper, the repairing operations are applied to improve the quality of clustering:

- Clusters with a negative or small wirelength benefit are detected and removed from the final cluster list.
- Some clusters are beneficial in terms of wirelength, but they have delay overshoot behind the critical delay of the design. These clusters are repaired by iteratively removing their relatively far nets until the timing constraints are met.

Inserting MOC Devices into Netlist

In this phase, multiplexer and demultiplexer cells are added to the netlist and it is updated to connect the MOC devices to the source and the sink nodes of the clusters.

Clustering algorithm

```
1:  Create initial CS Fibonacci Min-heap
2:  WHILE (there are any uncommitted nets)
3:    Get minimum CS from FHeap
4:    Merge minimum CS pair to make new cluster
5:    Update the FHeap
6:    IF (size of cluster > MAX_CLUSTER_CAP)
7:      Mark the cluster as final cluster
8:    END
9:  END
10: Finalize the clusters
```

Figure 8. Clustering algorithm for MOC insertion.

G. Incremental Placement of MOC Devices

After inserting the MOC instances into the netlist, their locations in the layout must be determined. An incremental placement algorithm is used to determine the final locations of the MOC devices in the netlist. MOC devices have to be inserted in the white spaces throughout the chip. Since the size of MOC cells is very small and their number is much less than the total number of design cells, inserting them in the white spaces of the design does not degrade the total wirelength and performance of the placed results. After placing the MOC cells, if the delay of a cluster is increased beyond T_d, the cluster is repaired by iteratively removing its far nets from the cluster until the timing constraints are met.

EXPERIMENTAL RESULTS

Benchmarks and Test Strategies

We implemented our algorithm in C++ on an Intel 4300 workstation with 2 GB of memory. It was applied to twelve circuits randomly selected from the IWLS-2005 benchmarks [14] with various design sizes from 3227 to 92048 cells. They were synthesized in 130 nm technology with six layers of metals. All benchmarks were placed by the Dragon placer with 90% cell utilization (10% white spaces). Table 2 shows the characteristics of the benchmarks together with the results of MOC insertion.

In this table, #Cells show the number of cells in each benchmark, and #MOCs represents the number of generated MOC clusters in each design. Column #Members shows the average number of wires which are multiplexed in MOC clusters and TW represents the critical delay of the benchmarks.

Referring to the metro system analogy, it is worth noting that such a system is a good solution for traffic congestion reduction in large cities with large numbers of passengers traveling across the city. For small trips, the cost of the metro system may be higher than its benefits and, therefore, using the metro system for small trips may not be cost effective.

Analogously, in electronic circuits, if the lengths of multiplexed nets are too short, the benefits of MOC may be very little or it may even increase the total wirelength and congestion of the design. On the other hand, the delay of signal transmission in the MOC system is higher than that of dedicated wires in electronic circuits. Therefore, if the timing constraints of the design are too tight, MOC insertion may violate such constraints.

As can be seen in Table 2, circuits 1 and 2 have no long critical paths for MOC insertion; benchmark 8 has medium critical paths but its nets are not long

Table 2. Benchmarks characteristics.

Index	Benchmark	#Cells	#MOCs	#Members	TW(ps)
1	Spi	3227	0	0	2524.24
2	tv80	7161	0	0	4163.09
3	pci_bridge32	16816	9	23.4	18727.58
4	dma	19118	11	24.5	18491.70
5	b20_1	19131	14	10.21	6961.51
6	b22	28317	24	18.75	10796.53
7	b22_1	28128	45	24.3	10233.41
8	wb_conmax	29034	0	0	8816.92
9	b17	37117	28	26.55	27207.41
10	b17_1	37383	36	36.86	27291.20
11	Ethernet	46771	36	30.05	23330.71
12	b18	92048	24	96.7	73038.43

enough. Therefore, our algorithm did not insert any MOC modules into these circuits.

Two different design flows were attempted to test the proposed idea. In the first design flow, called Conventional Net Routing (CNR), terminals are connected via dedicated wires and in the second flow (MOC), nets were selected and serialized asynchronously.

Congestion and Routability

For congestion estimation, the method proposed in [15] was used. In [15], some metrics and methods were presented to estimate the congestion map of a design after placement by a probabilistic estimation of routing topologies of nets with consideration of router intelligence. Table 3 shows the results of our experiments on the benchmarks, in terms of design congestion in 130 nm and 180 nm. *Overflow* shows the percentage of bins that have a congestion level beyond the routing capacity of bins (congestion overflow) and *Congestion Improvement* represents the congestion reduction of MOC vs. CNR.

Table 3 shows that after MOC insertion, congestion overflow is improved by 15.54%, on average, in 130 nm and 13.74%, on average, in 180 nm. Our experimental results show that for small circuits, the improvements in 130 nm is not more than those in

Table 3. Congestion before and after MOC insertion.

Index	Benchmark	130 nm			180 nm		
		Over flow		Congestion	Over flow		Congestion
		CNR (%)	MOC (%)	Improvement (%)	CNR (%)	MOC (%)	Improvement (%)
1	spi	2.47	2.47	0	-	-	-
2	tv80	7.37	7.37	0	-	-	-
3	pci_bridge32	1.9	1.51	20.5	0.85	0.66	28
4	dma	15.45	13.96	9.6	17.94	16.33	9.8
5	b20_1	48.32	46.39	3.9	36.3	34.4	5.2
6	b22	46.14	45.02	2.4	34.2	32.6	4.6
7	b22_1	61.43	52.86	13.9	58.1	52.3	9.9
8	wb_conmax	34.47	34.47	0	-	-	-
9	b17	6.70	5.48	18.2	4.7	4.1	12.7
10	b17_1	6.94	5.74	17.2	59.96	49.3	17.7
11	Ethernet	11.27	7.98	29.1	59.96	54	19.35
12	b18	4.54	3.73	16.1	1.02	0.6	14
Average				15.54			13.47

Table 4. Design routability of benchmarks.

Index	Benchmark	#Nets	Misroute		Routability Improvement (%)
			CNR (%)	MOC (%)	
1	spi	4100	0	-	0
2	tv80	7220	1	-	0
3	pci_bridge32	17122	34	29	14.7
4	dma	19640	41	31	24.3
5	b20_1	13896	104	84	19.2
6	b22	270120	346	284	17.9
7	b22_1	270034	220	190	13.6
8	wb_conmax	29321	22	-	0
9	b17	39233	303	243	19.8
10	b17_1	39622	281	202	28.1
11	Ethernet	52816	583	375	35.6
12	b18	96842	632	501	20.7
Average					21.57%

180 nm but for large circuits, the benefits of MOC insertion in 130 nm is considerably more that those in 180 nm. This is because in small circuits there are not long enough nets and the delay of MOC modules is comparable with wire delays. Therefore, the benefits of MOC mechanism are small. However, in large circuits there are more long wires that have considerable wire delays so that the delay of MOC cells can be a small portion of the delay of these wires.

In this situation, more MOCs can be added into the design. On the other hand, in the scaling process, the clock period of the scaled design is decreased slower than gate delays. Therefore, in a scaled design, the delay of MOC MUX and DEMUX cells are decreased more than the critical delay of the design and, consequently, scaling will improve the benefit of MOC insertion. It is worth noting that congestion reduction in benchmarks 3 and 11 are higher than others, since their netlists have large busses that generate very beneficial MOC clusters.

To evaluate the routability improvement in MOC compared with CNR, we run global routing on benchmarks. We used a global router [16] and [17]; an improved version of the PathFinder algorithm [18] for standard-cell global routing. This tool routes the design by iterative rip-up & reroute operations. At the end of the rip-up & reroute process, mis-routed nets are routed by a post-process algorithm, if there are any. In this experiment, routing layers are restricted to only 2 layers at the global routing stage, in order to emphasize the benefits of MOC insertion. Ref [10] does not address any reports on design routability and, therefore, we only reported the routability of MOC insertion with the new clustering algorithm in 130 nm technology. Table 4 shows the routability improvement of the MOC insertion process.

In this table, #Nets shows the number of nets in the netlist before MOC insertion, *Misroute* represents the percentage of misrouted nets before the post-process phase of global routing and the column, *Routability Improvement*, shows the improvement of the routability of MOC compared with CNR. As can be seen in this table, routability of the design is increased by 21.57% on average.

Crosstalk Reduction

In this sub-section, the Metro-on-Chip mechanism is evaluated in terms of reducing the crosstalk noise. We estimated the crosstalk of initial parallel wires and also the crosstalk noise of the final multiplexed wires. Then, we compared them to show the crosstalk improvement in the proposed methodology. We used a 2-π noise model [19] to estimate the crosstalk after placement, which is a simple yet accurate close form solution for both peak noise and noise width [19]. Consider a victim net and also an aggressor net, as shown in Figure 9.

In the 2-π model, the *peak noise* and the *noise pulse* width can be estimated by the following equation:

$$V_m = \frac{t_x}{t_r}(1 - e^{-\frac{t_r}{t_v}}), \tag{4}$$

where t_x is the Elmore delay of the coupling capacitance with the near source wire, t_r is the transition

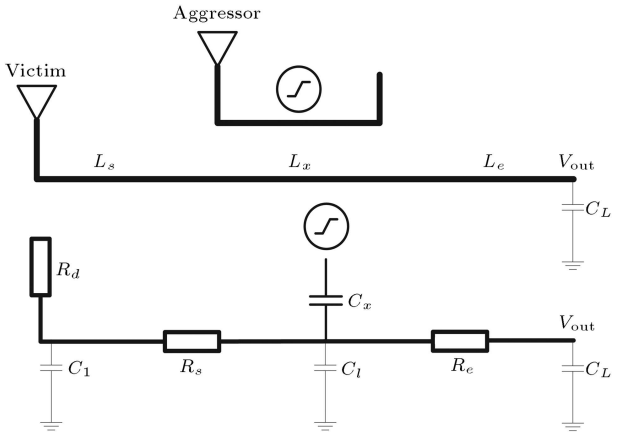

Figure 9. The layout of aggressor and victim nets corresponding with their crosstalk model.

time of the aggressor net, and t_v is the Elmore delay of the victim net. In practice, both peak noise and pulse width are used to determine the impact of noise. From [19], the peak noise amplitude and noise width product (AW) at the receiver can be written as:

$$AW = (R_d + R_s).C_x.f(x), \tag{5}$$

where R_d is the drive resistance, R_s is the near source wire resistance (i.e., from driver to the coupling segment) of the sink net, C_x is the coupling capacitance and $f(x)$ is as:

$$f(x) = \frac{e^x - e^{-x}}{1 - e^{-x}}. \ln \frac{e^x - e^{-x}}{1 - e^{-x}},$$

and:

$$x = \frac{t_r}{t_v}. \tag{6}$$

It has been shown in [19] that, with an acceptable approximate, we can estimate AW as:

$$AW = (R_d + R_s).C_x. \tag{7}$$

In this paper, we replace some parallel wires with a serial wire. In practice, layouts are very congested and in a congested design most tracks are used for routing and the number of empty tracks is very small. In this situation, it can be assumed that all parallel wires (before multiplexing) and serial wires after multiplexing have many adjacent wires that can be an aggressor to these wires. We used the following Lemma to estimate the upper bound of the crosstalk on a victim net.

Lemma

Assume that a victim net has n neighboring aggressors. The total amplitude and pulse width product of the coupling noise of all the aggressors on the victim net can be approximated as:

$$AW = (R_d + R_w).C_w, \tag{8}$$

where R_w is the total resistance of the victim net and C_w is the summation of all capacitances between the victim net and the aggressor nets.

Proof

Let a victim net have n neighboring aggressors (Figure 10).

The peak noise amplitude and noise width product (AW) can be calculated as:

$$AW = (R_d + R_{s1}).C_{x1} + (R_d + R_{s2}).C_{x2} + \cdots + (R_d + R_{sn}).C_{xn}. \tag{9}$$

Let the total resistance of the victim net be R_w and the total summation of capacitances between the victim net and the aggressors be C_w. Therefore, $R_s = \frac{R_w}{n}$ and $C_{xi} = \frac{C_w}{n}$ and AW can be calculated as:

$$\begin{aligned} AW &= \left(R_d + \frac{R_w}{n}\right).\frac{C_w}{n} + \left(R_d + \frac{2R_w}{n}\right).\frac{C_w}{n} \\ &\quad + \left(R_d + \frac{3R_w}{n}\right).\frac{C_w}{n} + \cdots + \left(R_d + \frac{(n-1)R_w}{n}\right).\frac{C_w}{n} \\ &= \frac{C_w}{n}\left[R_d + \frac{R_w}{n} + R_d + \frac{2R_w}{n} + R_d \right. \\ &\quad \left. + \frac{3R_w}{n} + \cdots + R_d + \frac{(n-1)R_w}{n}\right] \\ &= \frac{C_w}{n}\left[nR_d + \frac{n(n-1)}{2}.\frac{R_w}{n}\right] \\ &= C_w\left[R_d + \frac{n(n-1)}{n}.\frac{R_w}{n}\right]. \end{aligned} \tag{10}$$

For large values of n, the upper bound of AW can be calculated as:

$$\begin{aligned} \lim_{n\to\infty}(AW) &= \lim_{n\to\infty} C_w\left(R_d + \frac{n(n-1)}{n^2}.R_w\right) \\ &= C_w(R_d + R_w). \end{aligned} \tag{11}$$

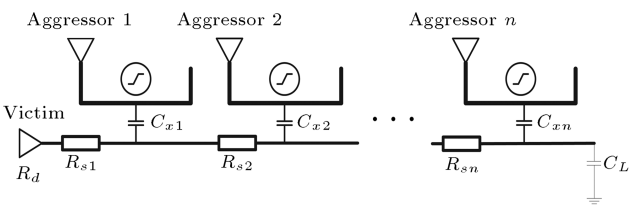

Figure 10. A victim net and n aggressor wire segments.

We used this approximation to estimate the upper bound of the crosstalk noise of the selected nets for multiplexing before and after MOC insertion, and the results are shown in Table 5.

In this table, columns CBM and CAM represent the crosstalk noise of multiplexed nets before and after multiplexing in volts, respectively. Column CRMN represents the crosstalk reduction of only multiplexed nets and column TCR shows the total crosstalk noise reduction of each design in percentage.

As can be seen in Table 5, Crosstalk Improvement of Multiplexed Nets (CRMN) is very high, but the number of multiplexed nets is much smaller than the total number of nets in each benchmark. Therefore, TCR is much smaller than CRMN in all attempted benchmarks. It is worthwhile to note that crosstalk improvement in multiplexed nets is proportional to the number of MOC groups multiplied by the average number of nets in each MOC group. Consequently, in those benchmarks wherein the number of MOC groups is more, more crosstalk improvement is achieved (as depicted in Table 2).

Wirelength Reduction

In Table 6, wirelength reduction after MOC insertion is reported. TWL represents the value of total wirelength and TWL Reduction shows the improvement in total wirelength. As can be seen in this table, the decrease in total wirelength is small because the number of candidate nets for MOC is much smaller than the total number of nets in a design.

Power Consumption Overhead

Power consumption is an important concern in current design methodologies and any new design flow may need to consider it. MOC insertion affects power consumption from the following points of view:

- Inserting new devices (MUXes and DEMUXes) increases the number of transistors in the design and, hence, increases the static power dissipation of the design.
- Multiplexing parallel nets into a serial link decreases the capacitance of nets and the dynamic power dissipated in the wires. However, the signal activity on multiplexed wire is almost equal to the summation of activities on the dedicated nets before multiplexing [7]. Therefore, reduction in the number of nets is neutralized by the increase in the signal activity of multiplexed wires.
- Figures 11 and 12 show examples of the MOC system with two nets. $Ci1$ and $Ci2$ are initial capacitances of two input wires before MOC insertion, respectively, and $Cm1$ and $Cm2$ are capacitances of the input wires after MOC insertion, respectively. The values of $Cm1$ and $Cm2$ are smaller than $Ci1$ and $Ci2$ because the lengths of input wires ($Wm1$ and $Wm2$) are much smaller than those of dedicated wires ($Wi1$ and $Wi2$). Therefore, the power consumption of $G1$ and $G2$ after MOC insertion is less than their power consumption before MOC insertion.

Table 5. Experimental results in terms of improvement in crosstalk upper bound.

Index	Benchmark	Crosstalk Noise		CRMN	TCR
		CBM (v)	CAM (v)		(%)
1	spi	-	-	-	-
2	tv80	-	-	-	-
3	pci_bridge32	1.30e-07	6.19e-10	210.6	3.2
4	dma	5.08e-07	1.88e-09	269.5	3.3
5	b20_1	3.32e-07	2.32e-09	142.94	3.7
6	b22	9.21e-08	2.04e-10	450	4.2
7	b22_1	2.09e-06	1.91e-09	1093.5	16.2
8	wb_conmax	-	-	-	-
9	b17	1.09e-07	1.47e-10	743.4	9.1
10	b17_1	4.18e-06	3.15e-09	1326.96	14.2
11	Ethernet	3.75e-06	3.46e-09	1081.8	14.2
12	b18	9.84e-06	4.24e-09	2320.8	17.5
Average				605.27	9.51%

Table 6. Experimental results for wirelength.

Index	Benchmark	TWL		TWL Reduction (%)
		CNR (μm)	MOC (μm)	
1	spi	5.221e8	5.221e8	0
2	tv80	1.350e9	5.221e8	0
3	pci_bridge32	4.000e9	3.967e9	1.01
4	dma	5.762e9	5.652e9	1.94
5	b20_1	4.895e9	4.826e9	1.43
6	b22	6.482e9	6.457e9	0.38
7	b22_1	7.577e9	7.126e9	6.3
8	wb_conmax	6.324e9	6.324e9	0
9	b17	1.243e10	1.240e10	0.2
10	b17_1	1.248e10	1.167e10	6.9
11	Ethernet	1.683e10	1.612e10	4.3
12	b18	3.269e10	3.092e10	5.5
Average				3.08%

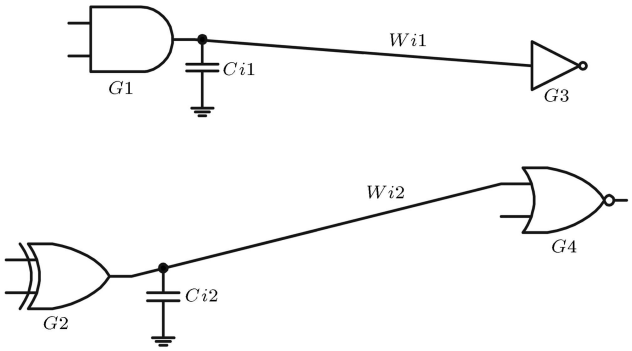

Figure 11. Capacitances of input wires before MOC insertion.

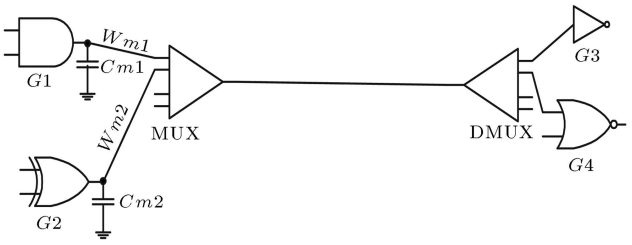

Figure 12. Capacitances of input wires after MOC insertion.

Our analysis shows that the final effect of the above factors is that the power consumption after MOC insertion is slightly more than that of the initial design, and a major part of the increased power is resulted from the static power consumption of MOC cells.

Table 7 shows that the power consumption of the benchmarks after MOC insertion is slightly more than the initial state (0.12% on average).

Runtime Overhead

As explained before, the computation time of the MOC algorithm is improved in this paper. In Table 8, the computation time of standard MOC insertion flow is reported and compared with the placement time for each benchmark.

In Table 9, the runtime improvement of the new version of the MOC clustering algorithm is compared with the clustering runtime reported in [10]. As can be seen in this table, the clustering runtime is improved about 24.3% compared with that of [10], on average. It is worthwhile to note that the clustering runtime is a major part of MOC algorithm computation time and improving it reduces the total computation time considerably.

Our analyses show that the average computation overhead after improvement is less than 8%. It is noted that clustering runtime improvement is more considerable for large circuits that have more MOC clusters. This can make the approach more practical for large and congested circuits.

CONCLUSION

In this paper, a new approach for crosstalk reduction and routability improvement inspired by the idea of the metro transportation systems in large cities was proposed. In the proposed method, a set of wires is marked as non-critical and sufficiently long and then these wires are multiplexed at the source transmitted via an asynchronous serial link and demultiplexed at the destination.

Table 7. Experimental results in terms of power consumption overhead.

Index	Benchmark	Power Consumption Before MOC Insertion (μW)	Power Consumption After MOC Insertion (μW)	Power Consumption Overhead (%)
1	Spi	87.67	87.67	0
2	tv80	93.71	93.71	0
3	pci_bridge32	102.87	108.3221	0.053
4	dma	113.16	120.4022	0.064
5	b20_1	118.04	128.5456	0.089
6	b22	143.93	159.4744	0.108
7	b22_1	188.79	235.4211	0.247
8	wb_conmax	247.36	247.36	0
9	b17	379.42	426.8475	0.125
10	b17_1	384.5	454.479	0.182
11	Ethernet	413.21	478.4972	0.158
12	b18	629.39	691.0702	0.098
Average				0.12%

Table 8. Experimental results for runtime.

Index	Circuit	Placement Runtime (sec)	MOC Insertion Runtime (sec)	Runtime Overhead (%)
1	spi	45	2	4.4
2	tv80	169	58	34.3
3	pci_bridge32	387	51	13.1
4	dma	472	86	18.2
5	b20_1	453	23	5.03
6	b22	965	50	5.18
7	b22_1	993	47	4.7
8	wb_conmax	1036	57	5.3
9	b17	1367	63	4.6
10	b17_1	1400	68	4.8
11	Ethernet	1175	66	5.6
12	b18	5124	759	14.8
Average				10.01%

We improved the metro-on-chip mechanism and evaluated it in 130 nm technology. Experimental results show that the overflow congestion is reduced by 15.54%, on average, and routability and total wirelength are increased by 21.57% and 3.08%, respectively, on average. Furthermore, crosstalk noise is reduced by 9.51%, on average.

On the other hand, the power consumption of the attempted benchmarks is increased slightly (0.12% on average). In addition, our analyses show that MOC-

Table 9. Experimental results for runtime of new version of MOC algorithm.

Index	Benchmark	Clustering Runtime in [10] (sec)	Improved Clustering Runtime (sec)	Clustering Improvement (%)
1	Spi	0.08	0.08	0
2	tv80	0.15	0.15	0
3	pci_bridge32	0.16	0.16	0
4	dma	0.39	0.38	23
5	b20_1	0.65	0.72	17.9
6	b22	0.91	0.81	10.9
7	b22_1	5.7	3.44	39
8	wb_conmax	0.123	0.123	0
9	b17	1.95	1.54	21
10	b17_1	7.23	3.1	57
11	Ethernet	36.91	12.18	67
12	b18	16	7	56
Average				24.3%

based design flow can result in valuable improvements in congestion and routability in large and congested circuits in a nano-scale regime. MOC insertion design flow may be improved if the distribution of long wires and white spaces for MOC insertion are planned at earlier stages of the physical design flow, such as floorplanning. The research on this topic is underway.

REFERENCES

1. Carloni, L.P. and Sangiovani-Vincentelli, A.L. "On-chip communication design: Roadblocks and avenues", *CODES+ISSS*, pp. 75-76 (2003).
2. Bjerregaard, T. and Mahadeven, S. "A survey of research and practices of network-on-chip", *ACM Computing Surveys*, **38**(1), pp. 1-51 (2006).
3. Meincke, T. et al. "Globally asynchronous locally synchronous architecture for large high-performance ASICs", *International Symposium on Circuits and Systems*, pp. 512-515 (1999).
4. Lee, S.J., Kim, K., Kim, H., Cho, N. and Yoo, H.J. "Adaptive network-on-chip with wave-front train serialization scheme", *VLSI Circuits*, pp. 104-107 (2005).
5. Caputa, P. and Svensson, C. "A 3Gb/s/wire global on-chip bus with near velocity-of-light latency", *International Conference on VLSI Design*, pp. 117-122 (2006).
6. Teifle, J. and Manohar, R. "A high-speed clockless serial link transceiver", *International Symposium on Asynchronous Design*, pp. 151-161 (2003).
7. Dobkin, R., Kolodny, A. and Morgenshtein, A. "Fast asynchronous bit-serial interconnects for network-on-chip", *International Symposium on Asynchronous Design*, pp. 117-127 (2006).
8. Dobkin, R., Morgenshtein, A., Kolodny, A. and Ginosar, R. "Parallel vs. serial on-chip communication", *International Workshop on System-Level Interconnect Prediction*, pp. 43-50 (2008).
9. Dobkin, R., Perelman, Y., Liran, T., Ginosar, R. and Kolodny, A. "High rate wave-pipelined asynchronous on-chip bit-serial data link", *IEEE International Symposium on Asynchronous Circuits and Systems* (2007).
10. Jahanian, A. and Saheb Zamani, M. "Metro-on-chip: An efficient physical design technique for congestion reduction", *IEICE Electronics Express*, **4**(16), pp. 510-516 (2007).
11. Taghavi, T., Yang, X. and Choi, B-K. "Dragon2005: Large scale mixed size placement tool", *International Symposium on Asynchronous Design*, pp. 245-247 (2005).
12. Chen, H., Qiao, C., Zhou, F. and Cheng, C.K., "Refined single trunk Steiner tree: A rectilinear Steiner tree generator for interconnect prediction", *System Level Interconnect Prediction*, pp. 85-89 (2002).
13. Horowitz, E., Sahni, S. and Mehta, D. "Fundamentals of data structures in C++", *W.H. Freeman & Co.* (1995).
14. "IWLS Benchmarks", available from World Wide Web: http://iwls.org/iwls2005/benchmarks.html (2005).
15. Saeedi, M., Saheb Zamani, M., and Jahanian, A. "Prediction and reduction of routing congestion", *International Symposium on Physical Design*, pp. 72-77 (2006).

16. Aghapour, M., Saeedi, M. and Saheb Zamani, M., "Global routing with crosstalk consideration", *Computer Society of Iran Computer Conference*, Iran (2006).
17. "STD-Cell PathFinder Global router", available from World Wide Web: http://ceit.aut.ac.ir/~eda (2007).
18. McMurchie, L. and Ebeling, C. "PathFinder: A negotiation-based performance-driven router for FPGAs", *International Symposium on Field Programmable Gate Arrays*, pp. 111-117 (1995).
19. Ren, H., Pan, D.Z. and Villarrubia, P.G. "True crosstalk aware incremental placement with noise map", *International Conference on Computer Aided Design*, pp. 402-409 (2004).

BIOGRAPHIES

Ali Jahanian received a BS degree in Computer Engineering from the University of Tehran, in Iran, in 1996, and MS and PhD degrees in Computer Engineering from Amirkabir University of Technology, Iran, in 1998 and 2008, respectively. His research interests are: VLSI Design Automation and Embedded System Design.

Morteza Saheb Zamani received a BS degree in Computer Engineering from Isfahan University of Technology, in Iran, in 1989, and Meng and PhD degrees in Computer Engineering from the University of New South Wales, Australia, in 1992 and 1996, respectively. He joined the faculty of Amirkabir University of Technology, Tehran, Iran in 1996. His current research interests are focused on VLSI Design Automation, Reconfigurable Computing Systems and Quantum Computing.