Metro-on-chip: an efficient physical design technique for congestion reduction

Ali Jahanian and Morteza Saheb Zamani

Computer Engineering and IT Department, Amirkabir University of Technology,
Hafez St., Tehran 15914, Iran

Abstract: Routing congestion is one of the main factors in designing in deep submicron technology that may cause unroutability of the design, signal integrity problems and large delays in detoured wires. In this paper, a new methodology is presented which multiplexes regular nets by asynchronous serial transceivers in the physical design flow in order to improve the congestion of the design. Experimental results show that for the attempted benchmarks, the overflow congestion was reduced by up to 40.03% without any degradation in clock frequency and negligible power consumption overhead.

Keywords: asynchronous serial transmission, congestion, physical design

Classification: Integrated circuits

References


1 Introduction

Routing congestion in a VLSI design may cause unroutability or make large delays due to detoured long wires, especially for large and congested circuits.
In a congested design, many wires have to be detoured in global routing stage and a large number of long wires are generated. These global wires can affect the performance of the design and may also result in crosstalk and manufacturability problems. Therefore, congestion reduction has a key role in the performance, manufacturability and signal integrity improvement of designs.

The increasing impact of latency of global wires and their variation have driven some designers to delay insensitive paradigms [1]. Carloni and Vincentelli suggested delay-insensitive on-chip communications to omit clock distribution problems in System-on-chip design [1]. In [2], asynchronous serial bus systems in Network-on-Chip (NOC) structures were analyzed. Teifel and Manohar [3] presented a high-speed and clock-less serial link transceiver for inter-chip communication in asynchronous VLSI systems.

In [4] an NOC with a new serialization mechanism was proposed which provides high performance serialization with automatic bandwidth adjustment. Dobkin et al. [5] presented a novel bit-serial interconnect structure with a new data encoding style in a pipelined structure.

In this paper, a new method is presented which uses the asynchronous serial transmission to reduce the congestion and total wirelength of deep submicron circuits in application specific integrated circuits design flow. We call this mechanism as metro-on-chip (MOC) because its semantic is similar to metro transportation systems in large cities.

2 Metro-on-chip mechanism

In the conventional physical design flow, terminals of each net are connected via a dedicated wire. However, in circuits with large number of nets, congestion is an important factor and its reduction can improve the routability, signal integrity and even performance of the design. Multiplexing some nets is an effective solution for congestion reduction. The main idea of this paper is to automatically find non-critical and sufficiently long wires and multiplex them using asynchronous serial transmission hardware to improve the congestion. This idea is similar to the metro system in large cities that conveys many passengers in one trip from a source to a destination to avoid traffic congestion. Fig. 1 shows the conventional routing using dedicated wires and versus serial multiplexing mechanism.

The main contribution of this paper is improving the routing congestion

![Diagram](a) Conventional dedicated wires vs. (b) serial multiplexed wires
and routability of designs by reducing the number of long global nets but at the same time, it may improve the performance of the design by avoiding detoured global nets resulted from routing congestion. We use the fast serial communication mechanism proposed in [5] for the serial link module. The characteristics of the serial link module are important for the feasibility and quality of the proposed design flow. Therefore, in the following subsection, the structure of this module is briefly reviewed.

In this paper, MOC cells are inserted after global placement in which timing estimations are not very rough. On the other hand, postponing MOC insertion to the routing stage may impose large design costs.

### 2.1 Serial transmission system review

The fast serial link transceiver [5] uses low-latency synchronizers at the source and sinks with a two-phase non-return zero (NRZ) asynchronous protocol that allows non-uniform delay intervals between successive bits. The acknowledgment of transmission is returned only once per word, rather than bit by bit enable wave-pipelined transmission.

A MOC cluster with \( n \) wires consists of \( 2^n \) pipelined asynchronous shift registers to serialize and deserialize the data. These shift registers are named as XL in [5]. Each XL contains 12 transistors with about 10 \( \mu \text{m}^2 \) area in 0.13 \( \mu \text{m} \) technology. It was shown that the proposed circuit can send and receive each bit at a data cycle of a single FO4 (fan-out of 4) inverter delay [5] (about 33 ps in 0.13 \( \mu \text{m} \) technology).

### 2.2 Interfacing the MOC system and a general synchronous 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 \texttt{start} signal and the completion of the whole transmission process is indicated by a ‘1’ on \texttt{wordAck} signal.

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 timing constraint of the design). However, if the paths of the selected wires have enough delay slack, the multiplexing delay will not affect the critical paths. Therefore, in selecting signals for multiplexing, their criticality and length are taken into account to avoid timing violation.
3 MOC-based physical design process

In the flow proposed in this paper, MOC clusters are inserted after detailed placement where the final position of cells have been determined and delay estimations are not too rough. At this stage, fixed cells may become obstacles for the placement of MOC devices. Since, the number of MOC modules are chosen to be very small compared with the total number of cells, MOC devices can be placed in white spaces by an incremental post placement algorithm. The MOC-based physical design flow is described in the following subsections.

3.1 Floorplanning and placement

In this stage, the design is floorplanned and the final locations of cells and pins are determined. Dragon placer [6] is used to perform floorplanning and placement of the design.

3.2 Static timing analysis

In this phase, a static timing analysis (STA) is performed to estimate the delay of paths after placement. Then, the delay of each path in the design is calculated using Elmore delay model and the critical paths of the design are determined.

3.3 Net selection for multiplexing

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. \( T_d \) can be specified as:

\[
T_d = T_w - \alpha T_w
\]

where \( T_w \) is the most critical delay of the design and \( \alpha \) is a number between 0 and 1 which defines a margin for near critical nets. In other words, 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, we estimate an appropriate value 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 \( 2m_iD_{FO4} \) 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:

\[
2m_iD_{FO4} \leq \alpha T_w \Rightarrow m_i \leq \frac{\alpha T_w}{2D_{FO4}}
\]  

\( \star \) signal is activated on rising edge of the clock with the delay of a few gates. Therefore, its delay is negligible compared with the total delay of a cluster. The delay of \( \text{world}_{ack} \) can be also ignored because the receiver circuit has not to wait for the sender to receive the acknowledge signal. Our experimental results show that the error of this approximation is low (about...
7% on average) and Eq. 2 can be used as a good initial approximation for \( m_i \) and \( \alpha \). The detailed proof of this approximation and experimental results are not included here due to space limitation.

### 3.4 Clustering 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 sources and close sinks. Let \( G = \{g_1, g_2, \ldots, g_n\} \) be a set of MOC clusters where cluster \( g_i \) has \( m_i \) nets. 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 distance between sources/sinks in \( g_i \). The developed clustering algorithm is shown in Fig. 2.

<table>
<thead>
<tr>
<th>Clustering algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: Create initial CS matrix and find minimum CS.</td>
</tr>
<tr>
<td>2: WHILE (there are any uncommitted nets)</td>
</tr>
<tr>
<td>3: Merge the net pair with minimum CS and update the CS matrix</td>
</tr>
<tr>
<td>4: Find new minimum CS in CS matrix.</td>
</tr>
<tr>
<td>5: IF (size of cluster &gt; MAX.CLUSTER.CAP)</td>
</tr>
<tr>
<td>6: Mark the cluster as final cluster.</td>
</tr>
<tr>
<td>7: END</td>
</tr>
<tr>
<td>8: END</td>
</tr>
</tbody>
</table>

Fig. 2. Clustering algorithm for MOC insertion

At the start of the algorithm, each net is considered as a cluster with one net. First, a CS matrix is generated showing the companionship cost of each pair of nets. Then, two clusters with minimum CS are found and merged in each loop of the algorithm and the CS matrix is updated. If the size of the merged cluster reaches a maximum offset, the cluster is marked as final cluster. The algorithm is continued until there are no unmarked clusters. The complexity of this algorithm is \( O(n^2) \).

### 3.5 Repairing MOC groups

After clustering the nets, the clusters are evaluated in order to gain more benefits. We first remove too short nets and secondly, the clusters with large delays are repaired by iteratively removing the far nets which violate timing constraints.

### 3.6 Inserting MOC devices into netlist

In this phase, multiplexer and demultiplexer macro models are inserted into the cell library and respective instances are added to the netlist. Then, the netlist is updated to connect these MOC devices to the source and the sink nodes.
### 3.7 Incremental placement of MOC devices

In this phase, an incremental placement is performed to determine the final locations of MOC devices in the netlist.

### 4 Experimental results

We implemented our algorithm in C++ on an Intel P4 workstation with 2 GB of memory. It was applied to twelve circuits randomly selected from the IWLS-2005 benchmarks with various design sizes from 3227 to 92048 cells. They were synthesized in 130 nm technology with six layers of metals. All benchmarks are placed by Dragon placer with 90% of cell utilization (10% white spaces). Two different design flows were attempted to test the proposed idea. In the first design flow, namely conventional net routing (CNR), each wire is connected via a dedicated net and in the second flow (MOC), some nets were selected and serialized asynchronously.

Table I shows the results of our experiments on the benchmarks in terms of design congestion, total wirelength and power consumption. In this table, \#Cells shows the number of cells, \#MOCs represents the number of MOC clusters and \#Members is the average number of nets in each cluster. Critical path delay shows the most critical delay of each circuit in picoseconds. Congestion and TWL show the congestion reduction and total wirelength improvement, respectively and finally, PWR represents the increase in power consumption after MOC insertion.

<table>
<thead>
<tr>
<th>Circuit</th>
<th>#Cells</th>
<th>#MOCs</th>
<th>#Members</th>
<th>Critical path delay (ps)</th>
<th>Congestion (%)</th>
<th>TWL (%)</th>
<th>PWR (%)</th>
<th>CPU time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>spi</td>
<td>3227</td>
<td>0</td>
<td>0</td>
<td>2524.24</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>tv80</td>
<td>7161</td>
<td>0</td>
<td>0</td>
<td>4164.69</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>58</td>
</tr>
<tr>
<td>sima</td>
<td>16816</td>
<td>9</td>
<td>23.4</td>
<td>1872.75</td>
<td>21.05</td>
<td>1.01</td>
<td>0.055</td>
<td>61</td>
</tr>
<tr>
<td>bridge32</td>
<td>19131</td>
<td>14</td>
<td>21.21</td>
<td>9916.51</td>
<td>4.2</td>
<td>1.43</td>
<td>0.089</td>
<td>23</td>
</tr>
<tr>
<td>b22</td>
<td>28337</td>
<td>24</td>
<td>24.75</td>
<td>10796.53</td>
<td>2.48</td>
<td>0.38</td>
<td>0.108</td>
<td>50</td>
</tr>
<tr>
<td>bridge45</td>
<td>1923344</td>
<td>10</td>
<td>31.3</td>
<td>1923344</td>
<td>0</td>
<td>0.0</td>
<td>0.0</td>
<td>37</td>
</tr>
<tr>
<td>wireminmax</td>
<td>29034</td>
<td>0</td>
<td>0</td>
<td>8816.92</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>97</td>
</tr>
<tr>
<td>b17</td>
<td>37111</td>
<td>28</td>
<td>26.59</td>
<td>27297.41</td>
<td>22.3</td>
<td>0</td>
<td>0.125</td>
<td>63</td>
</tr>
<tr>
<td>b17</td>
<td>37983</td>
<td>36</td>
<td>10.86</td>
<td>27391.29</td>
<td>20.8</td>
<td>0.9</td>
<td>0.182</td>
<td>38</td>
</tr>
<tr>
<td>ethernet</td>
<td>46773</td>
<td>36</td>
<td>10.08</td>
<td>23359.74</td>
<td>4.4</td>
<td>0.158</td>
<td>0.089</td>
<td>38</td>
</tr>
<tr>
<td>b18</td>
<td>92048</td>
<td>24</td>
<td>34.7</td>
<td>76038.44</td>
<td>21.3</td>
<td>0.0</td>
<td>0.0</td>
<td>94.16</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>16.99%</td>
<td>3.08%</td>
<td>0.12%</td>
<td>94.16</td>
</tr>
</tbody>
</table>

When the lengths of multiplexed nets are too short, MOC mechanism may have little wirelength benefit. On the other hand, when the delay of paths is too small, MOC insertion may violate the timing constraints of the design. As can be seen in Table I, the circuits that either have no long nets or long critical paths are not good candidates for MOC insertion.

Table I shows that after MOC insertion, congestion overflow is improved by 16.99% on average, without any degradation of performance. We use the method proposed in [7] for overflow congestion estimation that represents the routability of design. Congestion reduction in benchmark ethernet is very high (40.03%) since its netlist has large busses which generate very beneficial MOC clusters. As can be seen in this table, total wirelength improvement
is small because the number of candidate nets for MOC is normally much smaller than the total number of nets in a design. The experimental results show that the increase in power consumption of the benchmarks after MOC insertion is very small (0.12% on average). In Table I, CPU time shows the actual run time of MOC insertion process. This overhead is 7.01% of the placement runtime on average.

5 Conclusion

In this paper, a new approach for congestion reduction was proposed inspired with the metro transportation systems in large cities. In the proposed method, non-critical and sufficiently long wires are selected to be multiplexed at the source, transmitted via an asynchronous serial link and demultiplexed at the destination. Experimental results show that this technique can improve the overflow congestion by up to 40.03% and total wirelength is decreased 3.1% on average. On the other hand, MOC insertion cause to a small increase in power consumption of the benchmarks (0.12% on average) and total CPU time (7.01% on average).