Routing Congestion Removing Of CMOL FPGA Circuits By A Recursive Method

HOSSEIN HAMIDIPOUR, PARVIZ KESHAVARZI, ALI NADERI
Electrical and Computer Engineering Faculty
Semnan university
Semnan
IRAN
h.hamidipour@yahoo.com, pkeshavarzi@semnan.ac.ir, alinaderi1998@gmail.com

Abstract: In this paper we have proposed a recursive method for removing the routing congestion in CMOL FPGA circuits where CMOLCAD tool cannot route them successfully. CMOL FPGA architecture, with T basic cells and a latch cell per tile, uses K basic cells (predefined by user) and a latch cell for logic implementation and (T-K) cells for routing. When the circuit encountered congestion, CMOLCAD tool decreases K to route the circuit. This is a drawback for CMOLCAD tool that cannot route the circuit with predefined K. In proposed method, we keep and rank the placement solutions in some of the last iterations of placement algorithm, according to the cost and use them for routing the circuits with more options. If the routing on the highest priority placement solution has failed, this solution will be removed from ranking and another placement solution will be used according to the ranking. This procedure will be continued until circuit routed without congestion by predefined K. The results show that we can remove 9.7% of congestions by applying the proposed method beside CMOLCAD.

Key-Words: - CMOL, CMOLCAD, FPGA, Congestion, Placement, Routing.

1 Introduction
Silicon technology continues to advance towards the end of Moore’s Law, predicted with the end of CMOS (complementary metal–oxide–semiconductor) scaling only 10–15 years away. So, the semiconductor digital integrated circuit [1] growth will decrease. Accurate lithographic definition of some parameters such as length and width of conducting channel is the main reason for this growth reduction. By decrease in scale, accuracy grows exponentially and the cost of lithography becomes very expensive. [2] The theoretical and experimental [3-5] results indicate that the plausible alternative for current technology is hybrid semiconductor/ nanodevices circuits. This structure uses the functionality of programmable diodes (latching-switches) in crossbar structure (Fig. 1) beside the silicon chip. The functionality of such devices is illustrated in Fig.1.1b. At low applied voltages, the device behaves as a usual diode, but a higher voltage may switch it between low-resistive (ON) and high-resistive (OFF) states.

CMOS Molecular (CMOL) [6] architecture basically called hybrid nanowire/ molecular/ semiconductor circuit which incorporates the nanodevices and nanowire crossbar into the CMOS IC implementation. CMOL technology uses the advantages of nano and CMOS technology together and shows some advantages in device density and fault tolerant compare with custom CMOS FPGA. In this structure, the nanowire crossbar

Fig. 1: (a) Nanowire/nanodevice crossbar (b) I-V curve of a two-terminal crosspoint nanodevice-programmable diode (schematically).
with molecular switches at the crosspoint, are fabricated on top of a CMOS die to establish the basic logic and memory cells with programmable capability. (Fig. 1) The generic CMOL circuit is shown in Fig. 2. This structure contains 3 sections: nanowire crossbar, interface pins and CMOS stack. (Fig. 2.a) The idea of addressing a particular nanodevice is shown in Fig. 2.b. Any nanodevice may be addressed via the appropriate pin pair (e.g., pins 1 and 2 for the left of the two shown devices, and pins 1 and 2’ for the right device- Fig. 2.c). In (b), only the activated CMOS lines and nanowires are shown, while (c) shows only two devices.

Fig. 2. CMOL interface. (a) Schematic side view. (b) The idea of addressing a particular nanodevice. (c) Zoom-in on several adjacent pins.

F_{nano} and F_{CMOS} are nano and CMOS half pitch, respectively and β is a dimensionless factor. CMOS stack and nanowire crossbars are turned by angle α for addressing issues. [8]

The interface pins are used for connecting nanowire crossbar and CMOS stack. A zoom in top view on the circuit near several adjacent interface pins is shown in Fig. 2.c.

### 2 CMOL FPGA

The Field Programmable Gate Arrays (FPGAs) are becoming increasingly important platforms for digital circuits [9].

Fig. 3: CMOL FPGA logic circuits: (a) the basic CMOS cell (b) the latch cell.

For utilizing FPGA advantages, the CMOL FPGA circuit has been proposed by Strukov and Likarev [8]. They proposed two structures for applying on CMOL FPGA, the basic cell (Fig. 3-a) and the latch cell (Fig. 3-b).

The basic cell or the basic CMOL inverter cell (I-Cell), consists of a CMOS inverter, two CMOS pass transistors, two nanowires and two pins. In configuration stage, the selected nanodevices are programmed. When the configuration has been completed, the pass transistors are used as pull down resistors, while the nanodevices set into ON (low resistive) state as pull up resistors. Three I-Cells (shaded) and two programmable molecular switches can be configured as a two-input NOR gate (Fig. 4). The two molecular switches work as two diodes and will establish an OR function of the two inputs A and B. The inverter output F will be the inverse of
(A+B), thus implements a NOR function. It is noted that the two input cells can be shared with other functions in a circuit.

![Fig. 4: Implementing A NOR gate. (a) CMOL configuration (b) schematic diagram of a NOR gate.](image)

Therefore, the NOR function will require only one I-cell. Using this NOR gate as a building block, different combinational logic functions can be obtained. The latch cell has 8 interface pins and 2 pass transistors for circuit configuration. For using CAD tool in design automation, CMOL FPGA CAD utilizes basic and latch cells in square shaped “tiles”. Each tile has a latch cell and T basic cell surrounding the latch (Fig. 5).

![Fig. 5: A fragment of two-cell CMOL FPGA fabric](image)

If we assume each basic cell needs a unit area, CMOS layout estimation [10] shows that the latch cell has four times larger area. The default number for T is 12, thus we have a basic tile area 16(12+4) times of a basic cell. The design flow for CMOLCAD has been proposed in [11]. Beside K, there is a specific concept for CMOL FPGA, the cell connectivity domain (A). This concept is defined as such fabric fragment that any cell within it can be directly connected to any cell of the initial tile. In CMOLCAD, K basic cells and a latch cell are used for logic implementation and (T-K) basic cells are reserved for routing purposes. The placer tries to locate logics in tiles. After placement, the reserved basic cells and idle cells will be used for global routing. If there is congestion after the global routing, i.e. the number of reserved cells and idle cells are not sufficient to route the circuit, the algorithm tries to route the circuit by reducing K [12]. So, the routing algorithm cannot route the circuit successfully with predefined K and this is a drawback for CMOLCAD tool.

### 2.1 CMOLCAD Placement algorithm

The placement algorithm in CMOLCAD is very close to the VPR [13] tool. In CMOLCAD, Simulated annealing (S.A) is main algorithm to place the circuits which has been optimized to obtain a faster and better solution according to CMOL FPGA architecture. The placement cost function contains wiring cost and timing cost. The timing cost is calculated similar to timing calculations in [14] and the wiring cost is calculated by using the following equations [12]:

\[
\text{SimpleHOP}(\text{tile}_a, \text{tile}_b) = \frac{2 \max(|x_a - x_b|, |y_a - y_b|) - 1}{A - 1} \quad (1)
\]

\[
\text{Hop}(\text{tile}_a, \text{tile}_b, \text{Polarity}) = 2 \left[ \frac{\text{SimpleHOP}(\text{tile}_a, \text{tile}_b) + \text{Polarity}}{2} \right] \quad (2)
\]

\[
\text{Wiring cost} = \sum_{i=1}^{\text{Connections}} \text{Hop}(\text{tile}_a(i), \text{tile}_b(i), \text{Polarity}(i)) \quad (3)
\]

Here (x, y) is the location of a tile with input and output gates of the considered connection, and Connections is the total number of connections in the circuit. Polarity is just additional information for placer and global router to specify whether an odd or even number of routing inverters should be used when interconnecting nets [12]. The total cost is the combination of these timing and area cost functions.

### 2.2 CMOLCAD Routing Algorithm

CMOL FPGA uses nanowire/ nanodevice/ nanowire links and inverters to route the circuits. If the destination tile is in the source tile connectivity domain, the router algorithm connects them by a nanowire/ nanodevice/ nanowire link. Otherwise, it uses inverter(s) to route the connection. The routing process in CMOLCAD is accomplished in two steps, global routing and detailed routing. In global routing, the circuit is routed generally and after that, in detailed routing, the defects of global routing will be modified. CMOLCAD global routing is done by a Greedy algorithm that is a heuristic one which is close to RSA [15] with logarithmic time complexity.
3 Proposed method

As described before in section 2, when the circuit encountered congestion after routing process, the routing algorithm reduces K to route the circuit without congestion. This is a defect for routing algorithm that cannot route the circuit with predefined K.

Fig.6 shows the flowchart of the proposed method. This method is similar to CMOL CAD design flow except in additional steps for saving and ranking the placement solutions in each iteration of the placement algorithm. The placement is done by S.A same as CMOL CAD placement. In each step of temperature reduction, the accepted placements are saved and ranked according to their placement cost. The ranked placement solutions will be used to route the circuits with more options. The ranking is according to the placement cost, i.e. the placement solution which has lower cost, has higher priority for being the platform for routing. The routing algorithm tries to route the circuit (same as CMOL CAD) on highest priority placement solution. If the circuit is routed without congestion, this placement solution is accepted and the algorithm is terminated. Otherwise the current placement solution is excluded from ranking and another placement is selected according to the ranking. This procedure will be continued until the algorithm routes the circuit without congestion or all of the ranked placement solutions tested.

4 Implementation results

For testing the feasibility and efficiency of the proposed method we have found some circuits (MCNC benchmark and some of other circuits) that CMOLCAD tool could not route them without congestion by predefined values of A and K. We have selected one of these values of A and K for each circuit. The results of applying CMOLCAD and proposed method are mentioned in table 1.

In original CMOLCAD, the circuits could not be routed successfully on final placement platform but the proposed method route these failed routing circuits successfully. The area and timing costs for both implementations are presented in the table. The recurrent steps column shows the number of placement solutions that the proposed method changes them to remove the congestion recursively.

For routing the circuits that CMOLCAD cannot route them without congestion, the proposed method needs some additional costs. The last row of the table shows total costs in both implementations. In average, these results show that our method for ten circuits results in only 1.5% and 1% increase in area and timing cost, respectively. The proposed method needs additional time for recurrent steps which is mentioned in table 1 for each circuit. Total analysis shows that the additional CPU time for running our method (for ten circuits) is 26.5 seconds while total time for CMOLCAD is 313 seconds. This means only 8.5% increase in CPU time for running our method.

Now, by testing all of possible values of A and K, we have obtained all of conditions that each circuit cannot be routed by CMOLCAD. The proposed method can route some of these failed routing conditions. Fig.7 shows congestion removing percentage of each circuit separately which is defined as the relation of the number of removed congestions to the total number of congestions for each circuit. The results show that our proposed method can remove 9.7% of congestions in average. The relation of the number of return steps to the total number of placement steps is illustrated in Fig.8. The results show that we must return 22% of total steps averagely to route the congested circuits successfully.

5 Conclusion

The proposed method by a recursive process routes some of the circuits which had faced congestion in CMOLCAD. The average results for ten circuits show that we can remove 9.7% of congestions by applying the proposed method beside CMOLCAD. It can be seen that by paying negligible additional costs, we route the failed routing circuits. This method can be added to CMOLCAD or other similar algorithms for removing congestion and routing the circuits successfully with higher probability.

References:
Fig 7 congestion removing percentage of each circuit

Fig 8. The relation of the number of return steps to the total number of placement steps


Table 1: The Implementation results of the CMOLCAD and proposed method

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Parameters</th>
<th>Original CMOLCAD</th>
<th>Proposed method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Circuit</td>
<td>K</td>
<td>A</td>
</tr>
<tr>
<td>alu4.blif</td>
<td>6</td>
<td>4</td>
<td>4768</td>
</tr>
<tr>
<td>misex3.blif</td>
<td>5</td>
<td>4</td>
<td>3574</td>
</tr>
<tr>
<td>s298.blif</td>
<td>5</td>
<td>4</td>
<td>3574</td>
</tr>
<tr>
<td>tseng.blif</td>
<td>9</td>
<td>4</td>
<td>1910</td>
</tr>
<tr>
<td>duke2.blif*</td>
<td>10</td>
<td>3</td>
<td>215</td>
</tr>
<tr>
<td>5xp1.blif</td>
<td>10</td>
<td>2</td>
<td>93</td>
</tr>
<tr>
<td>bl12.blif</td>
<td>10</td>
<td>4</td>
<td>359</td>
</tr>
<tr>
<td>apex7.blif</td>
<td>11</td>
<td>2</td>
<td>151</td>
</tr>
<tr>
<td>sao2.blif</td>
<td>10</td>
<td>2</td>
<td>142</td>
</tr>
<tr>
<td>sq16.blif</td>
<td>11</td>
<td>1</td>
<td>42</td>
</tr>
<tr>
<td>Total</td>
<td>-</td>
<td>-</td>
<td>16544</td>
</tr>
</tbody>
</table>

* Other parameters are same as CMOLCAD default.

** The timing cost and area cost of this circuit in both implementations (original CMOLCAD and the proposed method) are equal, but their placements are different.