Blockage and Voltage Island-Aware Dual-VDD Buffered Tree Construction

Bruce Tseng
Faraday Technology Cor.
Hsinchu, Taiwan

Hung-Ming Chen
Dept of EE
National Chiao Tung U.
Hsinchu, Taiwan

April 14, 2008 ISPD
Outline

- Introduction
- Modified RMP Algorithm
- Voltage Island Aware Buffered Tree Construction (ViaBuf)
- Experimental Results
- Conclusions
Motivation of This Work

- Voltage island architecture is getting popular, however corresponding EDA tools development is still very few.
- We develop approaches to solving the buffer insertion and level converter assignment problem in the presence of voltage island in a low-power design.
Our Contributions

- We have modified the RMP approach\(^1\) so that it can be applied on those designs which contain voltage islands.

- Our method ViaBuf has provided massive speedup over modified RMP, and even produced lower power buffered trees.

- As the number of sinks increases, our approach can effectively find feasible solutions within reasonable runtime.

Previous Work: DVB Algorithm

- First in depth study on applying dual Vdd buffers in buffer insertion. (DAC’05\(^1\))
  - With restrictions on the ordering of buffers, DVB neglects the necessity of level converter.
  - But DVB can’t fit a design with voltage island because of the restrictions.
- DVB is realized on a tree based VG’s style buffer insertion and a graph based RMP algorithm.
  - Compared with single voltage, it reduces 18\%–26\% power consumption.
- With RMP algorithm, DVB uses long time to complete both routing and buffer insertion for a net with less than 10 sinks.

Previous Work: DVB Algorithm

- It is not practical to have no level converters (LCs) presented in the Dual-Vdd designs
  - If $C_1$ is a high Vdd device, we still need LC
  - DVB inserts both kinds of buffers anywhere, which makes P/G routing very difficult
Problem Formulation

- Given a design with voltage island(s), a net with:
  - A source node
  - Multiple sink node with RAT (required arrival time) at each sink
  - Feasible buffer locations
  - Buffer library
  - Wire obstacles (such as hard IPs)

- We want to construct buffered routing tree with buffer insertion and level converter assignment under the following constraints:
  - RAT at each sink should be met.
  - The design works during power saving mode.
  - Signal levels are maintained for all devices.
Modified RMP Algorithm: Routing Grid Construction

Partition the graph into a grid graph by using the vertical and horizontal lines intersect at:

- Source and sink nodes
- Buffer locations
- 4 corners of the wiring blockages

::Source  Symbol  :Sink  Symbol  :Buffer location  Symbol  :Blockage  Symbol  :voltage island
Modified RMP Algorithm: Initial Solution Fill

- There are ten items (cap, rat, pow, rn, rs, B, signalV, Cbl, bend, totLength) in each solution

1. **cap**: capacitive load
2. **rat**: require arrival time
3. **pow**: power consumption
4. **rn**: reachable nodes (preventing from traversing the same path)
5. **rs**: reachable sinks (the farthest sink contained in solutions)
6. **B**: buffer type and corresponding location
7. **signalV**: signal voltage level
8. **Cbl**: extra load capacitance that the buffer needs to drive (when solutions merged at buffer location)
9. **bend**: The accumulated number of bending (solution pruning)
10. **totLength**: The accumulated wirelength
Modified RMP Algorithm: Initial Solution Fill (cont’d)

1. For a sink $p$, there is only one solution that states a buffer routing tree with zero wirelength.

2. For a source $p$, there is only one solution that models a driver as a specialized buffer.

3. For other kinds of node $p$:
   (Assume there are $n_H$ high $V_{dd}$ buffers, $n_L$ low $V_{dd}$ buffers, $m$ voltage level converters)
   a. If it is not a feasible buffer location, there is only one solution.
   b. If it is a feasible buffer location and within voltage island (low $V_{dd}$ region), fill $1+n_L$ solutions.
   c. If it is a feasible buffer location and outside the voltage island, fill $1+n_H+m$ solutions.
Modified RMP Algorithm: Solution Propagation (1/5)

- ◼️: solution with rs={1}
- ◼: solution with rs={2}
- ◼️ː solution with rs={1,2}
- ◼️ːː Source
- ◼️ːːː Sink
- ◼️ːːːː Buffer feasible location

Diagram: B1 and B2 regions with different solution indications.
Modified RMP Algorithm: Solution Propagation (2/5)

- Solution with rs={1}
- Solution with rs={2}
- Solution with rs={1,2}
- Source
- Sink
- Buffer feasible location
Modified RMP Algorithm: Solution Propagation (3/5)

- Use the wave propagation style to propagate the solutions from sink nodes to source node.

- Some restrictions:
  1. If both source and sink nodes are out of island, buffer cannot be placed within island. (in case voltage island turns off)
  2. If \( \text{signalV} \) (signal voltage level) is high, low Vdd buffer cannot be placed at target node. (otherwise large leakage will occur)
  3. \( r_n_A \cap r_n_B = \emptyset \) (solutions propagating from A to B) (to avoid path overlapping)
Modified RMP Algorithm: Solution Propagation (4/5)

- We propagate a solution within node A to its neighbor node B

- If $B_B=0$, (No buffer placed at node B):
  \[
  \begin{align*}
  \text{cap}_{\text{new}} &= \text{cap}_B + \text{cap}_A + C_W \\
  \text{rat}_{\text{new}} &= \min(\text{rat}_B, \text{rat}_A - D_W) \\
  \text{pow}_{\text{new}} &= \text{pow}_A + \text{pow}_B + E_w \\
  \text{rn}_{\text{new}} &= \text{rn}_A \cup \text{rn}_B \\
  \text{rs}_{\text{new}} &= \text{rs}_A \cup \text{rs}_B \\
  B_{\text{new}} &= B_A \cup B_B \\
  \text{signalV}_{\text{new}} &= \text{signalV}_A \\
  \text{Cbl}_{\text{new}} &= 0 \\
  \text{bend}_{\text{new}} &= \text{bend}_A + \text{bend}_B + ((\text{turn direction})?1:0) \\
  \text{totLength}_{\text{new}} &= \text{totLength}_A + \text{totLength}_B + (\text{Length between A, B})
  \end{align*}
\]
If $B_B \neq 0$, (Assume buffer $B_B$ placed at node B)

- $\text{cap}_{\text{new}} = \text{buffer } B_B$’s input capacitance
- $\text{rat}_{\text{new}} = \min(D_1, D_2)$ where $D_1 = \text{rat}_B \cdot R_w \cdot (C_w + \text{cap}_A)$; $D_2 = \text{rat}_A \cdot (D_w + D_B + R_w \cdot C_{bl_{\text{new}}})$
- $\text{pow}_{\text{new}} = \text{pow}_A + E_w$ (Vdd bases on driver)$+E_B$
- $\text{rn}_{\text{new}} = \text{rn}_A \cup \text{rn}_B$
- $\text{rs}_{\text{new}} = \text{rs}_A \cup \text{rs}_B$
- $B_{\text{new}} = B_B$
- $\text{signalV}_{\text{new}} = (B_B \text{ is a level converter}) \ ? \ \text{low} : (V_A \ | \ | \ V_B)$
- $C_{bl_{\text{new}}} = \text{cap}_A + C_w + C_{bl_B}$
- $\text{bend}_{\text{new}} = \text{bend}_A + \text{bend}_B + ((\text{turn direction})?1:0)$
- $\text{totLength}_{\text{new}} = \text{totLength}_A + \text{totLength}_B + (\text{Length between A, B})$
Modified RMP Algorithm: Solution Pruning

- For two solutions $s_A$ and $s_B$
  - Prune with VG approach:
    - If $\text{signal}_{V_A} = \text{signal}_{V_B}$, $\text{pow}_A > \text{pow}_B$, $\text{cap}_A \geq \text{cap}_B$, $\text{rat}_A \leq \text{rat}_B$, then $s_A$ is dominated and can be pruned.
  - Prune with bends and wirelength:
    - If $\text{bend}_A > \text{bend}_B$, $\text{totLength}_A \geq \text{totLength}_B$, $\text{rat}_A \leq \text{rat}_B$, then $s_A$ is dominated and can be pruned
Modified RMP Algorithm: Complexity Analysis

- Almost all the nodes in the graph could be a Steiner point for merging two buffered routing subtree with non-overlap reachable sink.

- Assume that a net with $n$ sinks, a grid graph has size $M \times N$ and each node has $K$ solutions, then the modified RMP has $O(2^nMNK)$ solutions during propagation, which grows exponentially.
Voltage Island Aware Buffered Tree Construction (ViaBuf)

- Perform modified RMP to deal with one sink only during each iteration.
- Erase the useless solutions besides the following solutions:
  - Initial solutions (to propagate solution from sink to source)
  - For the node on the desired path, keep solution with
    1. $rs=$ {sinks that were processed}
    2. Solutions with different buffer insertion solutions on the desired path. (useful Steiner points!)

Algorithm Voltage Island Aware Buffered Tree Construction (VIABuf)

Input: A routing grid graph and a wave pool $W$
Output: Solutions at source node, each one corresponds to a buffered routing tree topology

1. While ($W$ is not empty) {
   2. get a wave $w$ with sink nearest to source node
   3. for each node $ni$ in $w$ {
      4. for each solution $si$ in $ni$ {
         5. for each node $nk$ which is a neighbor of $ni$ {
            6. propagate $si$ to the solutions at neighbor node $nk$
            7. store new generated solutions in temporary container $Q$
            8. prune redundant solutions in $Q$
            9. if $Q$ is not empty {
               10. store new generated solution from $Q$ to $nk$
               11. put $nk$ to a temporary wave $wtemp$
            } 12. }}
   13. if $wtemp$ contains source node {
      14. choose a desired solution with least power consumption
      15. erase useless solutions in the routing grid graph
      16. } else {
         17. $W = W \setminus wtemp$
      18. }
   19. }
ViaBuf (cont.)

- Keep the following solutions in our approach:
  - $rs=\{\text{sinks that were processed}\}$
  - Solutions can be used when the path is possibly shared by handling next sink.

- Besides the above solutions and initial solutions, all the solutions of each node on the grid graph can be pruned.
### Comparisons Between Approaches

<table>
<thead>
<tr>
<th>Differences</th>
<th>Key steps</th>
<th>Solutions keeping and pruning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithms</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RMP</td>
<td>Pop solution with maximum RAT during each iteration</td>
<td>1. Keep exact one solution with the smallest cap for each reachable sink set</td>
</tr>
<tr>
<td>DVB</td>
<td>The same as RMP</td>
<td>1. Solution sampling. 2. Store solutions with a balanced tree.</td>
</tr>
<tr>
<td>Modified RMP</td>
<td>Classify solutions with the same reachable sink set as a wave, pop a wave during each iteration.</td>
<td>1. Prune with bends</td>
</tr>
<tr>
<td>ViaBuf</td>
<td>The same as modified RMP</td>
<td>1. Prune with bends 2. Greedy heuristic</td>
</tr>
</tbody>
</table>
Experimental Results

- Each of these cases has the 6 obstacles, 1 voltage island, 10 buffer locations, and grid graph is about 25*25 nodes on a 17*17mm design.
- A massive speed up over modified RMP could be obtained, while RATs are met.
- Our approach also achieves lower power with slightly worse phase delay.

<table>
<thead>
<tr>
<th>Benchmarks</th>
<th>source out of voltage island</th>
<th>manipulated RMP</th>
<th>ViaBuf</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>delay(ps)</td>
<td>power(fJ)</td>
<td>CPU time(sec)</td>
</tr>
<tr>
<td>net4</td>
<td>no</td>
<td>1162</td>
<td>9253</td>
</tr>
<tr>
<td>net5</td>
<td>no</td>
<td>903</td>
<td>8918</td>
</tr>
<tr>
<td>net6</td>
<td>yes</td>
<td>1199</td>
<td>12504</td>
</tr>
<tr>
<td>net10</td>
<td>no</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>net15</td>
<td>yes</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Instead of MRST, our algorithm intends to find a buffered routing tree meeting timing requirement and also signal integrity.
Conclusions

- We have implemented modified RMP algorithm to deal with the designs in the presence of voltage island.

- ViaBuf is much faster than modified RMP algorithm and can deal with multiple sinks net as the number of sinks increases.

- With RAT constraints, we can produce lower power buffered routing tree suitable for voltage island designs.
References (cont.)

References (cont.)


