Boolean Satisfiability-Based Routing and Its Application to Xilinx UltraScale Clock Network

Henri Fraisse
Abhishek Joshi
Dinesh Gaitonde
Alireza Kaviani

Xilinx Inc.

@xilinx.com
Agenda

- Introduction
- UltraScale clocking architecture
- Vivado clock routing
- Comparison of routing algorithms
- Conclusion
Clock routing challenges in modern FPGA

- The clocking resources of modern FPGAs are rapidly increasing in both size and complexity
  - More than 600 clocking buffers in the largest Xilinx UltraScale devices

- Conventional routing algorithms are running out of steam
  - Routability issues are emerging during clock tree synthesis

- New routing algorithms are required
  - SAT-based routing is proposed to address these challenges
UltraScale clocking architecture
A simplified Zync UltraScale device

FSR: Fabric Sub Region
GT: Gigabit Transceiver
PS: Processor Sub System
Under the hood of a Fabric Sub Region

Hal FSR

Clocking resources

- Interconnect Tile
- Configurable Logic Block (LUT, FF)
Under the hood of a Fabric Sub Region

Half-column

Clock leaves
Vivado clock routing
Clock routing

Global Clock Routing:
Generate partial tree from clock sources to Half-columns

Detailed Clock Routing:
Route each Half-column
Comparison of routing algorithms
SAT routing versus Iterative algorithms

- Created more than 1000 designs to stress clock routing
- Extracted the 51 designs not fully routed with standard iterative routing algorithms
- Collected the 1803 unsolved routing problems

<table>
<thead>
<tr>
<th></th>
<th>Designs</th>
<th>Routing problems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>51</td>
<td>1803</td>
</tr>
<tr>
<td>Routed SAT</td>
<td>43 (84.3%)</td>
<td>1786 (99.0%)</td>
</tr>
</tbody>
</table>

SAT-based routing can route efficiently 99% of the clock routing problems unsolved with iterative algorithms.
Comparison of two SAT encodings

<table>
<thead>
<tr>
<th>Initial number of variables</th>
<th>Binary</th>
<th>Unary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>HIGH</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Additional variables for CNF</th>
<th>Binary</th>
<th>Unary</th>
</tr>
</thead>
<tbody>
<tr>
<td>MANY</td>
<td>NO</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Reduction of variables from graph reachability</th>
<th>Binary</th>
<th>Unary</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO</td>
<td>YES</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Binary</th>
<th>Unary</th>
<th>Avg. of ratios</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg. runtime</td>
<td>1.4 sec</td>
<td>0.13 sec</td>
<td>17.9</td>
</tr>
<tr>
<td>Avg. variable count</td>
<td>10226</td>
<td>990</td>
<td>10.2</td>
</tr>
<tr>
<td>Avg. clause count</td>
<td>51494</td>
<td>3102</td>
<td>16.5</td>
</tr>
<tr>
<td>Avg. literal count</td>
<td>150780</td>
<td>7500</td>
<td>19.9</td>
</tr>
</tbody>
</table>

Unary SAT encoding is 18 times faster and produces formulas 20 times smaller
Concluding Remarks

- Improved Vivado clock routing using SAT-based routing
  - Both Routability and Runtime

- Demonstrated that Unary encoding is more efficient than conventional Binary encoding
  - Evaluated two SAT encodings and provided a comparative analysis

- Exploring SAT-Based routing for other applications
Thank you!