SAT-based Synthesis of Clock Gating Functions Using 3-Valued Abstraction

Oleg Rokhlenko
Joint work with Eli Arbel and Karen Yorav
IBM Haifa Research Labs
Clock-Gating

- Saving power by stopping the clock
- Clocks consume up to 50% of dynamic power
- Clock gating
  - Reduces dynamic power consumption
  - Prevents unnecessary switching of parts of the clock network

- Fine-grained clock gating
  - Analyze each latch/FF separately
    - Gate-level analysis
Clock gating analysis approaches

- Structural analysis
  - RTL-coding/gate-level structures
  - Scalable but limited in strength

- Functional analysis
  - Simulation based
    - Partial coverage of design behavior
  - Formal
    - Finds all opportunities but capacity is an issue

```vhdl
always @posedge clk
if ( sel )
  d = i
```
Typical (functional) clock-gating algorithm

- Feedback Loop Elimination (combinatorial clock gating):
  - Based on hold conditions

- A valid gating function – but may be infeasible
  - E.g. timing/area constraints

- There are other types of clock gating
  - Sequential clock gating, e.g. based on unobservability conditions
### BDD-based clock-gating

- Build the strongest function
  - typically based on one or more copies of the next-state function
- Minimize the function
  - by building its BDD
- Synthesize a net-list implementation
  - translation from BDD to net-list
- Timing constraint:
  - a CG signal may arrive too late, skewing the clock signal
BDD-based clock-gating

- Build the strongest function
  - typically based on one or more copies of the next-state function

- Minimize the function
  - by building its BDD

- Approximate the function
  - to allow for timing/area constraints
  - e.g. by “trimming” the BDD

- Synthesize a net-list implementation
  - translation from BDD to net-list
Our contribution: SAT-based clock-gating

Overview:

- Send the algorithmic gating function to a SAT-solver
  - each satisfying assignment is a gating opportunity
- Use an “all-SAT-like” algorithm to produce assignments
  - the result is the disjunction of all assignments

\[ CNF(f \land \neg \sigma_j) \]

What about the depth problem?
Our contribution: SAT-based clock-gating

Overview:

- Send the algorithmic gating function to a SAT-solver
  - each satisfying assignment is a gating opportunity
- Use an “all-SAT-like” algorithm to produce assignments
  - the result is the disjunction of all solutions
- Make the solver produce bounded-size clauses
  - directly generating the approximated solution
- Further optimize if needed
  - possibly using BDD-based solutions
3-valued logic

- In 3-valued logic the value $X$ stands for *unknown*

<table>
<thead>
<tr>
<th>$\land$</th>
<th>0</th>
<th>1</th>
<th>$X$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>$X$</td>
</tr>
<tr>
<td>$X$</td>
<td>0</td>
<td>$X$</td>
<td>$X$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\lor$</th>
<th>0</th>
<th>1</th>
<th>$X$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>$X$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$X$</td>
<td>$X$</td>
<td>1</td>
<td>$X$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\neg$</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$X$</td>
<td>$X$</td>
</tr>
</tbody>
</table>

- If for some assignment $\overline{\sigma}$, $\overline{\sigma}(i) = X$, $i \in \text{inputs}(f)$ then

  $$f(\overline{\sigma}) = b \rightarrow \forall i : f(\overline{\sigma}) = b \quad b \in \{0,1\}$$

- Since it’s a one way implication, it’s an *approximation*.
  - Xs values imply universal quantification but not every quantification can be done with Xs
SAT-based Synthesis of Clock Gating Functions

\[ \alpha_i = 1 \iff i \neq X \]

\[ g = 1 \iff \sum_i \alpha_i \leq n \]
SAT-based Synthesis of Clock Gating Functions
Example

\[
\begin{align*}
\sum_{i} \alpha_i & \leq 2 \rightarrow g = 1 \\
(i_1 \land \bar{i}_3) & \\
\Psi & = 1 \\
n & = 2
\end{align*}
\]
Example

\[ f = 1 \]

\[ \sum_{i} \alpha_{i} \leq 2 \rightarrow g = 1 \]

\[ \Psi = 1 \]

\[ (i_1 \land i_3) \lor (i_2) \]

\[ n = 2 \]
## Experimentation

Latches on which BDD timed-out

We show only designs with > 10% of hard latches

<table>
<thead>
<tr>
<th>Design</th>
<th>Candidates</th>
<th>Hard</th>
<th>% Hard</th>
<th>SAT solved</th>
<th>% Solved</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>585</td>
<td>418</td>
<td>71.45%</td>
<td>418</td>
<td>100.00%</td>
</tr>
<tr>
<td>D2</td>
<td>576</td>
<td>273</td>
<td>47.40%</td>
<td>273</td>
<td>100.00%</td>
</tr>
<tr>
<td>D3</td>
<td>397</td>
<td>243</td>
<td>61.21%</td>
<td>243</td>
<td>100.00%</td>
</tr>
<tr>
<td>D4</td>
<td>1096</td>
<td>126</td>
<td>11.50%</td>
<td>126</td>
<td>100.00%</td>
</tr>
<tr>
<td>D5</td>
<td>234</td>
<td>72</td>
<td>30.77%</td>
<td>72</td>
<td>100.00%</td>
</tr>
<tr>
<td>D6</td>
<td>409</td>
<td>62</td>
<td>15.16%</td>
<td>62</td>
<td>100.00%</td>
</tr>
<tr>
<td>D7</td>
<td>328</td>
<td>60</td>
<td>18.29%</td>
<td>60</td>
<td>100.00%</td>
</tr>
<tr>
<td>D8</td>
<td>219</td>
<td>48</td>
<td>21.92%</td>
<td>48</td>
<td>100.00%</td>
</tr>
<tr>
<td>D9</td>
<td>1735</td>
<td>251</td>
<td>14.47%</td>
<td>250</td>
<td>99.60%</td>
</tr>
<tr>
<td>D10</td>
<td>626</td>
<td>134</td>
<td>21.41%</td>
<td>132</td>
<td>98.51%</td>
</tr>
<tr>
<td>D11</td>
<td>390</td>
<td>54</td>
<td>13.85%</td>
<td>53</td>
<td>98.15%</td>
</tr>
<tr>
<td>D12</td>
<td>212</td>
<td>99</td>
<td>46.70%</td>
<td>97</td>
<td>97.98%</td>
</tr>
<tr>
<td>D13</td>
<td>1580</td>
<td>202</td>
<td>12.78%</td>
<td>194</td>
<td>96.04%</td>
</tr>
<tr>
<td>D14</td>
<td>2507</td>
<td>270</td>
<td>10.77%</td>
<td>259</td>
<td>95.93%</td>
</tr>
<tr>
<td>D15</td>
<td>107</td>
<td>13</td>
<td>12.15%</td>
<td>10</td>
<td>76.92%</td>
</tr>
<tr>
<td>D16</td>
<td>247</td>
<td>54</td>
<td>21.86%</td>
<td>38</td>
<td>70.37%</td>
</tr>
<tr>
<td>D17</td>
<td>195</td>
<td>30</td>
<td>15.38%</td>
<td>5</td>
<td>16.67%</td>
</tr>
</tbody>
</table>

The SAT-based approach succeeded in finding a clock gating condition for more than 73% of all the hard latches.
Over abstraction

- 3-valued abstraction is an *approximation* of universal quantification, but is not exact.
  - It is possible for there to be a term of size \( n \) that implies \( f \) while there exists no satisfying assignment to \( \Psi \)

- Example: \( f = (i \lor \neg i) \land j \) \( \Rightarrow \) \( j \) is a legal clock-gating function

\[
\sigma = (X,1)
\]

\[
\begin{array}{ccc}
X & \rightarrow & X \\
& \downarrow & \downarrow \\
1 & \rightarrow & X \quad \Rightarrow \quad \text{For n=1 our approach won’t find it}
\end{array}
\]

- Solution:
  - use higher values of \( n \) than our actual depth limit and then use BDD to optimize further
Experimentation (over abstraction)

- For 27% of hard latches, the SAT solver reported unsatisfiability.

- Using stronger machine with much more time and memory, the BDD-based approach reported the following:
  - 2% - no gating possible
  - 78% - BDD exploded
  - 20% - BDD solved, having an average depth 50.6. After approximation (up to depth=6), only 2% remain having on-set probability > 0.1

- All-in-all only 0.5% of hard latches were missed!
Experimentation (iterative approach)

- Iterative approach for a specific latch, when \( n = 1 \) to 6
  - Start with \( n=1 \) and increment by one each time we get UNSAT
- After 6 seconds, on-set probability is 93% of the optimum.
- The iterative approach allows us getting the strongest result.
Conclusion

- Using SAT when BDD fails allows handling much larger designs than before.
- Using 3-valued abstraction, we are able to directly generate the (strongest) approximation.
- Our approach produces partial results even if the computation is not completed within a set time limit.
- Over approximation is present, but we are fine with it. Moreover, extending $n$ beyond our target depth overcomes the over-approximation.
- In general, our approach allows universal quantification using SAT.
Thank you!

Questions?