HDCompiler

- Translate Esterel
- From PDG to Optimal Circuit
  Jia Zeng
What’s the Aim

- Optimal Circuitry
  - Approaches: PDG, State Machine
  - Procedures:
    - Encoding;
    - Circuit Translation;
    - Optimization;
What we did

- **State Encoding**
  - One-hot, Compact, Combining
- **Turning State Machine into Circuitry**
  - Encoding State (i.e. decoding bits)
    - $\text{bit}[i] = \text{OR}(\sum j \text{ Wires for state-value-emit statements for state_value}[j] \text{ where bit}[i] = 1 \text{ in the value code})$
  - Decoding State
    - AND (State machine’s entry, State value decoded)
What we did (cont)

- Flip-flops Sharing
  - Qualification:
    (Exact Common Parent \(! =\) Parallel Fork)
  - For all shareable State Machines, Which should be chosen?

The exact common parents of S2 and S3 are: N1, N2. Since N1 is a parallel fork, S2 and S3 can’t share flip-flops.
What we did (cont)

- Slack Computing
  - Why compute slacks
    - An upper bound of possible delay increase without violating the timing constrain
    - Represent of the potential capability of obtaining area/power reduction
  - How to compute
    - $arrival\_t (v) = \max_{u \in FI(v)} (arrival\_t(u)) + delay(v)$
    - $required\_t (v) = \min_{w \in FO(v)} (required\_t(u) - delay(w))$
    - $slack (v) = required\_t (v) - arrival\_t (v)$
How well we did

- Circuits comparing before/after optimizing
  - see web page: http://www.cs.columbia.edu/~jia/testrecord/

- Circuits comparing between different encoding means:

<table>
<thead>
<tr>
<th></th>
<th>Flip-flops</th>
<th>Gates</th>
<th>Wires</th>
<th>Slacks</th>
</tr>
</thead>
<tbody>
<tr>
<td>One-hot</td>
<td>10</td>
<td>56</td>
<td>80</td>
<td>16</td>
</tr>
<tr>
<td>Compact</td>
<td>6</td>
<td>59</td>
<td>85</td>
<td>22</td>
</tr>
</tbody>
</table>
Future Work

- Just a structure for real hardware translator, much more future work:
  - Choose state encoding means
  - Optimize based on slacks
  - Take real PDG input
  - Catch bugs