High-level synthesis of asynchronous circuits from control data flow graph representations

Sune F. Nielsen, Jan Madsen and Jens Sparsø
Informatics and Mathematical Modelling, DTU
Email: {sfn,jm,jsp}@imm.dtu.dk
J.S. Jensen (now at MIPS Denmark)
J.P. Hammerstoft

Motivation

- Today, design of large scale asynchronous circuits is dominated by syntax directed compilation:
  - Non-standard HDL-languages.
  - One-to-One translation.
  - Higher-level of abstraction required.
- Our Aim:
  - Use of resource sharing methods from traditional high-level behavioral synthesis of synchronous circuit to asynchronous circuits.
Related work

- J.Cortedellia:
- B.M.Bachmann:
  - Synthesis from pure DFG segments (modifications to ASAP/ALAP scheduling and resource allocation). [Proc.ICCD,oct,1999]
- S.Tyermann and P.J.Ashenden:
- We address synthesis from CDFGs:
  - Performance non-degrading resource sharing
  - Performance degrading resource sharing

Outline

- The design flow
- CDFG
- Performance non-degrading resource sharing
- Performance degrading resource sharing
- Results and Conclusion
A CDFG is

- Abstract representation of computation:
  - Representing only inherent dependencies/orderings in computation.
- Formally: its a Coloured Petri-net
  Vertices: operators \{+,-,\times,\ldots\}
  Edges: dependencies
CDFG Example

Sequential Code

\[ x = a + b; \]
\[ \text{tmp} = (b - c) + e; \]
\[ y = \text{tmp} + a + d; \]
\[ \text{if} \ (c > e) \ \text{then} \]
\[ z = \text{tmp} - (d + f); \]
\[ \text{else} \]
\[ z = \text{tmp} - (d - f); \]
\[ \text{endif} \]

Performance Non-Degradning Resource Sharing

- Minimum area without speed reduction
- Use of operator disjunctiveness
- Compatibility Graph:
  - If two operators are path/branch disjunctive there is an edge between them.
- Clique partitioning (of compatibility graph):
  "covering of disjoint completely connected sets"
CDFG example disjunctiveness

Compatibility graph
Clique Partitioning
• 3 possible clique partitions:

CDFG partitioning

ACID-WG 2002
Disjunctive resource sharing

- Transformation mechanism:

\[ \begin{align*}
\text{Input} & \quad \quad \text{Transformed CDFG} \\
\text{4 operators} & 
\end{align*} \]
Performance degrading resource sharing

- Further area reduction at speed penalty.
- DFG subsets in the CDFG have static relations between operators.
  - Given a maximum set of resources construct a schedule determining an exact execution order.
  - Imposing additional dependencies to the original DFG subset.
- List scheduling:
  - Urgency (=T_ASA-P-T_ALAP) in continous time.

CDFG partitioned

ACID-WG 2002
Deterministic resource sharing

- Transformation mechanism:

CDFG transformed

3 operators
Results

- Implementation: 4-phase bundled data async. handshake components in a 0.25 um process.
- Synopsys simulation data for 3 synthesized real-life async. circuits:

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Operators</th>
<th>Area (um2)</th>
<th>$&lt;T&gt;$ (ns)</th>
<th>$P$ (mW)</th>
<th>Resource Sharing</th>
<th>Operators</th>
<th>Area (um2)</th>
<th>$&lt;T&gt;$ (ns)</th>
<th>$P$ (mW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GCD</td>
<td>1 &quot;&quot;&gt;&quot;, 2 &quot;:&quot;</td>
<td>38691</td>
<td>745</td>
<td>1.63</td>
<td>Disjunctive</td>
<td>1ALU</td>
<td>38628</td>
<td>818</td>
<td>1.79</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(-0.12%)</td>
<td></td>
<td></td>
<td>(+9.7%)</td>
<td>(+22%)</td>
</tr>
<tr>
<td>Diffeq</td>
<td>6 &quot;&quot;&gt;&quot;, 2 &quot;:&quot;, 2 &quot;+&quot;, 1&quot;&gt;&quot;</td>
<td>115956</td>
<td>95</td>
<td>2.49</td>
<td>Disjunctive</td>
<td>4&quot;&quot;&gt;&quot;, 1&quot;&gt;&quot;, 1&quot;&gt;&quot;, 1ALU</td>
<td>93744</td>
<td>109</td>
<td>4.46</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(-19%)</td>
<td></td>
<td></td>
<td>(+14%)</td>
<td>(+79%)</td>
</tr>
<tr>
<td>FIR1</td>
<td>7 &quot;&quot;&gt;&quot;, 14 &quot;+&quot;</td>
<td>149067</td>
<td>173</td>
<td>1.74</td>
<td>Deterministic</td>
<td>1 1&quot;&gt;&quot;, 2 &quot;+&quot;</td>
<td>77805</td>
<td>334</td>
<td>3.88</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(-48%)</td>
<td></td>
<td></td>
<td>(+92%)</td>
<td>(+122%)</td>
</tr>
</tbody>
</table>

Conclusion

- Synthesis from CDFGs:
- High-level of abstraction with maximum concurrency.
- It is possible to apply automatic resource-sharing methods from high-level behavioral synthesis to asynchronous design:
- Trading execution time against area.
- Area gains for large designs.
- Speed overhead introduced in performance non-degrading resource sharing acceptable.
- High energy penalty for resource shared circuits.
- Well-known from low-power design.
Questions...

● Why little related work ?!

● How to reduce power consumption for resource shared components ?!
  – Same problem in syntax direct synthesis ?
  – Lower power consumption for different asynchronous handshake protocols (dual-rail) ?

ACID-WG 2002