Low-Level Loop Analysis and Pipelining of Applications mapped to Xilinx FPGAs

Hossein Omidian and Guy Lemieux
FPL conference
Problem

Prior Work:
> Feed-forward pipelining is easy
Using report_pipeline_analysis

This Work:
> Pipelining of sequential loops

(1) Loop splitting optimization
Loops are not always obvious in the HLS/HDL

- Different designers working on different parts

- Can be created when using HLS

```
If (C1)
  counter <= counter + 1;
always @(posedge clk)
begin
  if (reset)
    C3 <= 1;
  else if (counter != N)
    C3 <= 0;
  else if
    C3 <=1;
```

```
always @(posedge clk)
begin
  if (reset)
    C2 <= 0;
  else if (C3)
    C2 <= 1;
```

```
always @(posedge clk)
begin
  if (reset)
    C1 <= 0;
  else if (C2)
    C1 <= 1;
```

(a) Synthesized netlist
(b) FSM
(c) Simplified FSM
(d) Optimized FSM after loop splitting
Loop Analysis

Create a tool which can:

• Locate low-level loops in the netlist (after HLS, after logic synthesis)
• Identify and simplify 11 different loop patterns
• Automatically optimize ~8 loop patterns, user hints for ~3 patterns

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Method</th>
<th>Automated Mod.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Counter</td>
<td>Splitting</td>
<td>Yes</td>
</tr>
<tr>
<td>2. Combinational chain</td>
<td>Refactoring</td>
<td>Yes</td>
</tr>
<tr>
<td>3. Combinational chain</td>
<td>Combining</td>
<td>Yes</td>
</tr>
<tr>
<td>4. FSM</td>
<td>Early condition</td>
<td>Some</td>
</tr>
<tr>
<td>5. Triangle Accumulator</td>
<td>Cutting</td>
<td>Yes</td>
</tr>
<tr>
<td>6. Combined counters w/ controller</td>
<td>Cutting</td>
<td>Yes</td>
</tr>
<tr>
<td>7. Counter with a controller loop</td>
<td>Cutting</td>
<td>Yes</td>
</tr>
<tr>
<td>8. CRC lookalike</td>
<td>Combining</td>
<td>Yes</td>
</tr>
<tr>
<td>9. FIFO chain</td>
<td>Cutting</td>
<td>Yes</td>
</tr>
<tr>
<td>10. FIFO with read/write pointer</td>
<td>Cutting</td>
<td>No</td>
</tr>
<tr>
<td>11. FIFOs with a controller</td>
<td>Cutting</td>
<td>No</td>
</tr>
</tbody>
</table>
Pattern finding and simplifying

Loop from netlist

Loop simplified by the tool
Techniques

- Splitting
- Associative refactoring
- Combining (C-slow retiming, saves area)
- Early condition

\[
Y(t) = f(C_N, f(...f(C_1, f(C_0, Y(t-1))))...))
\]

\[
Y(t) = f(Y(t-1), f(C_N, f(...f(C_1, C_0))...)))
\]

New condition:

\[
(State = Sn-1 AND Counter > M-1) OR (State = Sn AND Counter < N-1)
\]
Results

Fmax Optimization

- Loop Fmax increase (%)
- Achieved loop Fmax (MHz)

<table>
<thead>
<tr>
<th>design_1</th>
<th>design_2</th>
<th>design_3</th>
<th>design_4</th>
<th>design_5</th>
<th>design_6</th>
<th>design_7</th>
</tr>
</thead>
<tbody>
<tr>
<td>60</td>
<td>10</td>
<td>70</td>
<td>80</td>
<td>90</td>
<td>70</td>
<td>50</td>
</tr>
</tbody>
</table>

Achieved loop Fmax (MHz)

<table>
<thead>
<tr>
<th>design_1</th>
<th>design_2</th>
<th>design_3</th>
<th>design_4</th>
<th>design_5</th>
<th>design_6</th>
<th>design_7</th>
</tr>
</thead>
<tbody>
<tr>
<td>600</td>
<td>700</td>
<td>800</td>
<td>900</td>
<td>700</td>
<td>600</td>
<td>500</td>
</tr>
</tbody>
</table>

Area+Fmax Optimization

- LUT (before)
- LUT (after)
- Loop Fmax increase (%)

<table>
<thead>
<tr>
<th>design_4</th>
<th>design_5</th>
<th>design_6</th>
<th>design_7</th>
</tr>
</thead>
<tbody>
<tr>
<td>350000</td>
<td>250000</td>
<td>200000</td>
<td>150000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>design_4</th>
<th>design_5</th>
<th>design_6</th>
<th>design_7</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>35</td>
<td>30</td>
<td>25</td>
</tr>
</tbody>
</table>

Applied after physical synthesis process

- Fmax optimized: average Fmax of 714MHz (+57%)
- Area+Fmax optimized: Fmax +15% to +41%, Average area -18%
Adaptable. Intelligent.