Automatic Controller Detection for Large Scale RTL Designs

Wei Song and Jim Garside
School of Computer Science
The University of Manchester
Motivation

• **Problem**
  – Matching of coding styles (DC: Design Compiler).
  – Only Finite State Machines (FSMs)

• **Target**
  – An algorithm that can detect **ALL** controllers.

• **Solution**
  – Signal-level data flow graph
  – Common pattern of controllers
Traffic Light Controller

always @(posedge clk or negedge rstn)
  if(!rstn)
    state <= R;
  else
    state <= state_nxt;

always @(state or cnt) // next state
  if(cnt == 0)
    case(state)
      R: state_nxt = YR;
      YR: state_nxt = G;
      G: state_nxt = YG;
      default:
        state_nxt = R;
    endcase // case (state)
  else
    state_nxt = state;

always @(posedge clk or negedge rstn)
  if(!rstn)
    cnt <= 0;
  else if(cnt == 0)
    case(state)
      R: cnt <= 2;
      YR: cnt <= 49;
      G: cnt <= 4;
      default:
        cnt <= 49;
    endcase // case (state)
  else
    cnt <= cnt - 1;

assign red = state == R ? 1 : 0;
assign green = state == G ? 1 : 0;
assign yellow =
  (state == YR || state == YG) ? 1 : 0;
DC: Matching of Coding Styles

- Never be assigned by a value other than predefined states.
- Can be used in only `==` and `!=` statements.
- Never be used as a port.

(Recognize all the counters used as controllers.)
Criteria of Controllers

• **Requirement 1**
  – A self-loop.

• **Requirement 2**
  – A controlling output.

• **Requirement 3**
  – No data input other than itself or constants.
Examples of Counters

Not a controller. Does not meet Requirement 3.

Not a controller. Does not meet Req. 2 & 3.

A controller.
Detection Flow

Multi-file Verilog RTL designs

Hierarchical internal abstract semantic tree

Hierarchical signal-level data flow graphs (DFGs) (Connections between **signals**)

Register relation graph (Connections between **Flip-Flops**)

Controller detection and report
Signal-Level DFG

- R: 50 sec
- GY: 3 sec
- G: 50 sec

- cnt
- state
- cnt == 0

- R G RY || GY
- red
- green
- yellow

- rstn clk
- state
- state_nxt
- red
- green
- yellow

- i_port
- o_port
- combi_block
- seq_block
- reset
- clock
- control
- data
if (a == 0)
case (state)
dout = mem[addr];
sum = da * time + extra;

dout = b > 3 ? da : db;
if(b > 3 ? da : db)    // control has a higher priority
Both *state* and *cnt* are controllers.
Dynamic Programming

B = A\(==0\) \(? \ 5 : 12\);
C = B + 10;

C = (A\(==0\) \(? \ 5 : 12\)) + 10;

Partial path reduction
## Test Cases

<table>
<thead>
<tr>
<th>Design Name</th>
<th>Description</th>
<th>Reason to choose</th>
<th>No. of Regs</th>
</tr>
</thead>
<tbody>
<tr>
<td>OR1200</td>
<td>A 32-bit 5-stage OpenRISC microprocessor</td>
<td>Data path controlled PC Scan chain Combinational forward loop</td>
<td>124</td>
</tr>
<tr>
<td>Reed-Solomon</td>
<td>A claimed industrial standard Reed-Solomon decoder IP</td>
<td>A not so well-written design Single block FSMs with irrelevant signal assignments</td>
<td>325</td>
</tr>
<tr>
<td>H.264/AVC</td>
<td>A 196K gate H.264/AVC baseline decoder</td>
<td>A well-written and large-scale design</td>
<td>855</td>
</tr>
</tbody>
</table>
## Test Results

<table>
<thead>
<tr>
<th>Name</th>
<th>Time</th>
<th>Reported</th>
<th>Verified</th>
<th>FSM</th>
<th>Counter</th>
<th>Flag</th>
<th>Error</th>
<th>Error Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>OR1200</td>
<td>1s</td>
<td>19</td>
<td>17</td>
<td>7</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>89%</td>
</tr>
<tr>
<td>Reed-Solomon</td>
<td>2.0s</td>
<td>56</td>
<td>54</td>
<td>6</td>
<td>36</td>
<td>12</td>
<td>2</td>
<td>96%</td>
</tr>
<tr>
<td>H.264/AVC</td>
<td>7.1s</td>
<td>55</td>
<td>49</td>
<td>13</td>
<td>30</td>
<td>6</td>
<td>6</td>
<td>89%</td>
</tr>
</tbody>
</table>

Environment: Intel Core™ 2 Duo 3.00 GHz PC with 2GB memory

All FSMs are detected with a small number of false errors.

Limitations:
Combinational loops, separated assignment (a[0]=b; a[1]=c;), Rom-style tables, etc.
Conclusion

• A pattern-matching algorithm has been proposed to detect all controllers including FSM, controlling counters and flags.

• Advantages:
  – Pattern matching
  – No restriction on coding styles
  – Recognize controlling counters
  – Automatic type estimation
Thanks for your listening!
Any questions?