Finding and understanding bugs in FPGA synthesis tools

Yann Herklotz and John Wickerson

Department of Electrical and Electronic Engineering
Imperial College London

27 November 2019
The FPGA synthesis toolflow

Verilog design → synthesis tool → Verilog netlist → place and route → bitstream generator → FPGA

Can we trust this?
John Wickerson
Finding and understanding bugs in FPGA synthesis tools

Verismith

Xilinx Vivado
Yosys
Intel Quartus
Xilinx XST

Verilog design

synthesis tool

Verilog netlist

place and route

bitstream generator

FPGA

Verilog netlist

module top (parameter param = $timescale 1ns/1ps)
endmodule
c

Verilog code snippet:

```verilog
module top (parameter param = $timescale 1ns/1ps)
input clk, wire0, wire1, wire2, wire3,
output wire [1:0] wire0,
output wire [1:0] wire1,
output wire [1:0] wire2,
output wire [1:0] wire3,
input wire [1:0] wire0,
input wire [1:0] wire1,
input wire [1:0] wire2,
input wire [1:0] wire3;
wire [1:0] wire0,
wire [1:0] wire1,
wire [1:0] wire2,
wire [1:0] wire3;
always @(posedge clk)
begin
  for (reg1 = 0; reg1 < 10; reg1 = reg1 + 1)
    begin
      if (wire1 == reg1-1)
        begin
          reg1 = (reg1 + 1) % 10;
        end
      end
      else
        reg1 = (reg1 + 1) % 10;
    end
end
always @(posedge clk)
begin
  if (wire1 == reg1)
    begin
      reg1 = (reg1 + 1) % 10;
    end
end
always @(posedge clk)
begin
  if (wire1 == reg1)
    begin
      reg1 = (reg1 + 1) % 10;
    end
end
always @(posedge clk)
begin
  if (wire1 == reg1)
    begin
      reg1 = (reg1 + 1) % 10;
    end
end
always @(posedge clk)
begin
  if (wire1 == reg1)
    begin
      reg1 = (reg1 + 1) % 10;
    end
end
endmodule
```
# Results

<table>
<thead>
<tr>
<th>Tool</th>
<th>Total test cases</th>
<th>Failing test cases</th>
<th>Distinct failing test cases</th>
<th>Bug reports</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yosys 0.8</td>
<td>26400</td>
<td>7164 (27.1%)</td>
<td>≥ 1</td>
<td>0</td>
</tr>
<tr>
<td>Yosys 3333e00</td>
<td>51000</td>
<td>7224 (14.2%)</td>
<td>≥ 4</td>
<td>3</td>
</tr>
<tr>
<td>Yosys 70d0f38 (crash)</td>
<td>11</td>
<td>1 (9.09%)</td>
<td>≥ 1</td>
<td>1</td>
</tr>
<tr>
<td>Yosys 0.9</td>
<td>26400</td>
<td>611 (2.31%)</td>
<td>≥ 1</td>
<td>1</td>
</tr>
<tr>
<td>Vivado 18.2</td>
<td>47992</td>
<td>1134 (2.36%)</td>
<td>≥ 5</td>
<td>3</td>
</tr>
<tr>
<td>Vivado 18.2 (crash)</td>
<td>47992</td>
<td>566 (1.18%)</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>XST 14.7</td>
<td>47992</td>
<td>539 (1.12%)</td>
<td>≥ 2</td>
<td>0</td>
</tr>
<tr>
<td>Quartus Prime 19.2</td>
<td>80300</td>
<td>0 (0%)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Quartus Prime Lite 19.1</td>
<td>43</td>
<td>17 (39.5%)</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Quartus Prime Lite 19.1 (No $signed)</td>
<td>137</td>
<td>0 (0%)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Icarus Verilog 10.3</td>
<td>26400</td>
<td>616 (2.33%)</td>
<td>≥ 1</td>
<td>1</td>
</tr>
</tbody>
</table>
Tracking bugs over time

Vivado version

2016.1
2016.2
2017.4
2018.2

22
15
6
22
28
11
28
17
43
Generating good testcases

[probability]
expr.binary = 5
expr.concatenation = 3
expr.number = 1
expr.rangeselect = 5
expr.signed = 5
expr.string = 0
expr.ternary = 5
expr.unary = 5
expr.unsigned = 5
expr.variable = 5
moditem.assign = 5
moditem.combinational = 1
moditem.instantiation = 1
moditem.sequential = 1
statement.blocking = 0
statement.conditional = 1
statement.forloop = 0
statement.nonblocking = 3

[property]
module.depth = 2
module.max = 5
output.combine = false
sample.method = "random"
sample.size = 10
size = 20
statement.depth = 3

Average lines of code in generated programs
Evaluating reduction
Do these bugs matter?

Re: Vivado 2019.1 Bit selection synthesis mismatch

Does this bug affect any priori Vivado versions, i.e. 2018.*?

This looks to me to be a rather critical bug. @ymherklotz - did you try your test on any other Vivado version?
Next steps

- Test ASIC synthesis tools
- Metamorphic testing
- Test other stages of the hardware synthesis process, e.g. place-and-route