

I2C Core RTL Simulation / FPGA mismatch...:

Subject: I2C Core RTL Simulation / FPGA mismatch...
msg#00024, hardware.opencores.cores

I'm using an FPGA to verify an ASIC which is using the I2C core. I'm getting a mismatch between the RTL simulation & the actual FPGA gates when writing to the I2C CR (command register). I setup the 2 prer bytes with known values. Then when I write the Cr register, the value written to the cr overwrites the prer(7:0). This is the case only in the FPGA. In rtl simulation, it works as expected.

Upon further examination of the rtl code, I think I have found an issue. Please correct me if I'm wrong.

Here's the code snippet with the case statement...

```
// generate registers
always @(posedge wb_clk_i or negedge rst_i)
  if (!rst_i)
    begin
      prer <= #1 16'hffff;
      ctr  <= #1 8'h0;
      txr  <= #1 8'h0;
    end
  else if (wb_rst_i)
    begin
      prer <= #1 16'hffff;
      ctr  <= #1 8'h0;
      txr  <= #1 8'h0;
    end
  else
    if (wb_wacc)
      case (wb_adr_i) // synopsys full_case parallel_case
        3'b000 : prer [ 7:0] <= #1 wb_dat_i;
        3'b001 : prer [15:8] <= #1 wb_dat_i;
        3'b010 : ctr          <= #1 wb_dat_i;
        3'b011 : txr          <= #1 wb_dat_i;
      endcase
```

Does the // synopsys full_case parallel_case not tell the synthesis Tool that ALL the cases are listed? All other cases are considered "don't care" by the synthesis tool, right?

The "cr" register is mapped at address 3'b100. This case is not

Listed in the case statement, so this bit is optimized away and is not considered important.

That's the problem... now when address 3'b100 is written, the case statement sees it as 3'bx00 and the prer(7:0) is written as well as cr.

Does the case statement not need a default line which keeps the

Previous values for the 4 registers? Am I misunderstanding this concept?

Figure 2 In this posting from opencores.org, an engineer wanted to produce a better gate-level implementation netlist, but the code behaves differently in synthesis from the way it behaves in simulation.