TSEA44 - Potential pitfalls

Andreas Ehliar <ehliar@isy.liu.se>

2015-11-17

Outline

- What can go wrong?
  - Design mistakes
  - Synthesis errors
  - Runtime errors
- Crossing clock domains
  - Handshaking
  - Asynchronous FIFOs

A design bug

- Symptom: The boot sequence of uClinux hangs after a second when the Icache is on.
- uClinux boots ok with Icache off
- No problems detected in the monitor when the icache is on

First try

- Modify the testbench so uClinux is present in SDRAM models
- Add interesting signals to the wave window
- Run the simulation over night

Oops. . .

- In the morning the simulation was not running any longer
- The log files had filled up all free space on the fileserver. . .
  - . . . which promptly crashed, causing all sorts of merriment

Handling long simulation runtimes

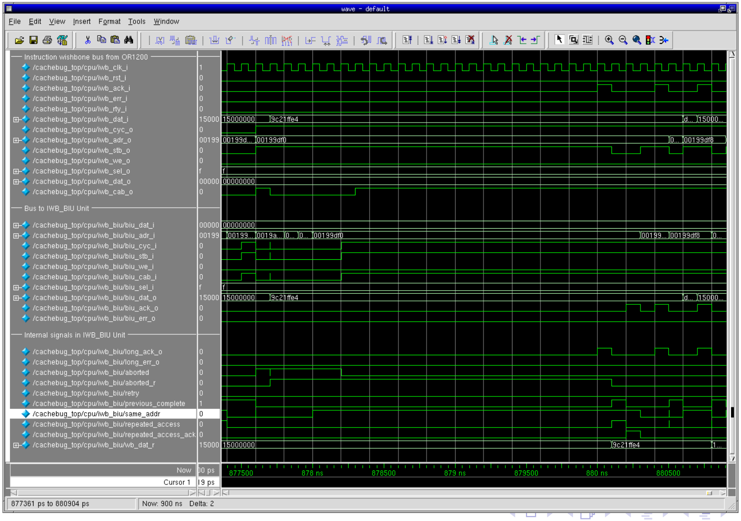
- Use checkpointing to reduce/eliminate the need for logging
  - Add no signals to wave window (and log for that matter)
  - Modify UART so printouts are displayed in the transcript window (using `$display()`)
  - `run 100ms; checkpoint 100ms.chk`
  - `run 100ms; checkpoint 200ms.chk`
  - `run 100ms; checkpoint 300ms.chk`
  - ...

Handling long simulation runtimes

- Now you can pinpoint the time interval where the crash happened
  - Restore the checkpoint in Modelsim that occured closest before the actual crash
  - `vsim -restore 600ms.chk`
  - Debug as usual (by adding signals to wave window/etc)

So what was the bug?

Wrong content:	AAAA	AAAA	CCCC	DDDD
Correct content:	AAAA	BBBB	CCCC	DDDD



```
Instruction wishbone bus from OR1200
/cachebug_top/cpu/iwb_clk_i                        1
/cachebug_top/cpu/iwb_rst_i                        0
/cachebug_top/cpu/iwb_ack_i                        0
/cachebug_top/cpu/iwb_err_i                        0
/cachebug_top/cpu/iwb_rty_i                        0
/cachebug_top/cpu/iwb_dat_i                        15000
/cachebug_top/cpu/iwb_cyc_o                        0
/cachebug_top/cpu/iwb_adr_o                        00199
/cachebug_top/cpu/iwb_stb_o                        0
/cachebug_top/cpu/iwb_we_o                         0
/cachebug_top/cpu/iwb_sel_o                        f
/cachebug_top/cpu/iwb_dat_o                        00000
/cachebug_top/cpu/iwb_cab_o                        0
Bus to IWB_BIU Unit
/cachebug_top/cpu/iwb_biu/biu_dat_i                00000
/cachebug_top/cpu/iwb_biu/biu_adr_i                00199
/cachebug_top/cpu/iwb_biu/biu_cyc_i                0
/cachebug_top/cpu/iwb_biu/biu_stb_i                0
/cachebug_top/cpu/iwb_biu/biu_we_i                 0
/cachebug_top/cpu/iwb_biu/biu_cab_i                0
/cachebug_top/cpu/iwb_biu/biu_sel_i                f
/cachebug_top/cpu/iwb_biu/biu_dat_o                15000
/cachebug_top/cpu/iwb_biu/biu_ack_o                0
/cachebug_top/cpu/iwb_biu/biu_err_o                0
Internal signals in IWB_BIU Unit
/cachebug_top/cpu/iwb_biu/long_ack_o               0
/cachebug_top/cpu/iwb_biu/long_err_o               0
/cachebug_top/cpu/iwb_biu/aborted                  0
/cachebug_top/cpu/iwb_biu/aborted_r                0
/cachebug_top/cpu/iwb_biu/retry                    0
/cachebug_top/cpu/iwb_biu/previous_complete        1
/cachebug_top/cpu/iwb_biu/same_addr                0
/cachebug_top/cpu/iwb_biu/repeated_access          0
/cachebug_top/cpu/iwb_biu/repeated_access_ack      0
/cachebug_top/cpu/iwb_biu/wb_dat_r                 15000

Now       00 ps
Cursor 1  19 ps

877361 ps to 880904 ps    Now: 900 ns   Delta: 2
```

What if you can’t find a bug during simulation?

- Very likely you have some undefined behavior in your design
  - Race condition in RTL code (blocking vs non-blocking assignment)
  - Incorrect use of “don’t cares”
  - You are not crossing clock domains correctly
  - etc
- Not so likely:
  - You have triggered a bug in the CAD tools (more on this later)

Clock domain crossings

- Why do we need synchronous designs?
  - Race conditions
  - Metastability
- Crossing clock domains
  - (Avoid it if possible)
  - Using handshakes
  - Using asynchronous FIFOs
  - Your own solution
    - (Only if you like debugging systems where bugs cannot be deterministically reproduced. . .)
- Don't forget that the reset signal has to be passed to each clock domain!

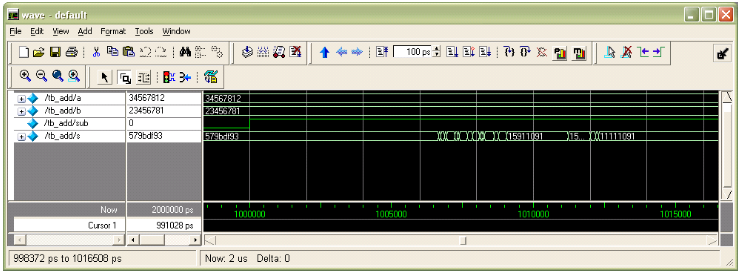
An example of a synthesis bug

- RTL simulation works fine
- Real hardware dosen't work
- (There are no clock domain crossings involved)

```
// Work around possible synthesis bug in an
// old version of synthesis tool
logic signed [27:0] foo;
logic signed [27:0] bar;
assign          foo = x2[2] * S6;
assign          bar = x2[3] * C6;
// ********* Third stage ************
always_ff @(posedge clk_i) begin
   if (en) begin
      x3_0 <= x2[0] + x2[1];
      x3_1 <= x2[0] - x2[1];
      x3_2 <= x2[2]*C6 + x2[3]*S6;
      x3_3 <= bar - foo;
```

Post synthesis simulation

- Synthesize the design
- Convert the synthesized design back to Verilog (`netgen`)
- The simulation is done using FPGA components like LUTs instead of at the behavioral level
- Can even be done after place and route if you want to simulate timing (very useful for power simulations!)



/tb_add/a	34567812	34567812
/tb_add/b	23456781	23456781
/tb_add/sub	0	
/tb_add/s	579bdf93	579bdf93 … 15911091 … 15... … 11111091
Now	2000000 ps	1000000 1005000 1010000 1015000
Cursor 1	991028 ps	

```
998372 ps to 1016508 ps        Now: 2 us   Delta: 0
```

Writing testbenches for post synthesis netlists 1

```
// Original testbench code

initial begin // Test adder
    @(posedge clk);
    a <= 5;
    b <= 3;
    @(posedge clk);
    if (result !== 8) begin
        $display("Adder has funny ideas of addition");
        $stop;
    end
end

// This code will probably lead to unknown values all over
// the simulation.
```

Writing testbenches for post synthesis netlists 1

```
// Modified testbench

initial begin // Test adder
    @(posedge clk);
    #4; // (Use an appropriate delay so the setup/hold
        //  of flip-flops in the circuit are honored.)
    a <= 5;
    b <= 3;
    @(posedge clk);
    // However, we should still check the result on the
    // clock edge!
    if (result !== 8) begin
        $display("Adder has funny ideas of addition");
        $stop;
    end
end
```

Dealing with bugs that you are unable to reproduce in simulation

- Add extra debugging logic to the FPGA design
- Chipscope/SignalTap
  - Logic analyzer that store samples to built in BlockRAMs
  - Communicates with the host computer via JTAG
- Warning!
  - Many people use ChipScope/SignalTap as a substitute for a comprehensive self-checking testbench.
  - Don't! You will most likely waste time better spent writing high quality testbenches.

Single event upset - SEU

- Very important in aerospace applications
- Can be important for ground based safety critical applications.
- (Fortunately not a problem in this course)
- Some (partial) solutions:
  - Triplicate logic
  - Reprogram FPGA often