Hacking an FPGA Design Until It Breaks (and Fixing It Again)

Jakub Duchniewicz

Supercon Berlin 2025

 j.duchniewicz@gmail.com www.jduchniewicz.com

How it began

My friend Krzysztof (wave to him) and I needed to calculate some signals *quite* fast - around 100 MHz, so regular MCUs started to be too slow for that.

Solution?

FPGA generated signals.

But(t)s

But FPGA's have quite costly divisions and multiplications (especially if you need to do a lot of them).

Solution?

RISCV softcore + FPGA.

(Re)using existing open source marvels

Fortunately we remembered about excellent "Doom on Ice40" project by Sylvain.

Figure: Screenshot from the Hackaday blogpost

Though many things did not work out of the box, we managed to get it running through *riscv_usb* subproject and get the UART console with meaningful logs.

Rough outline of the system

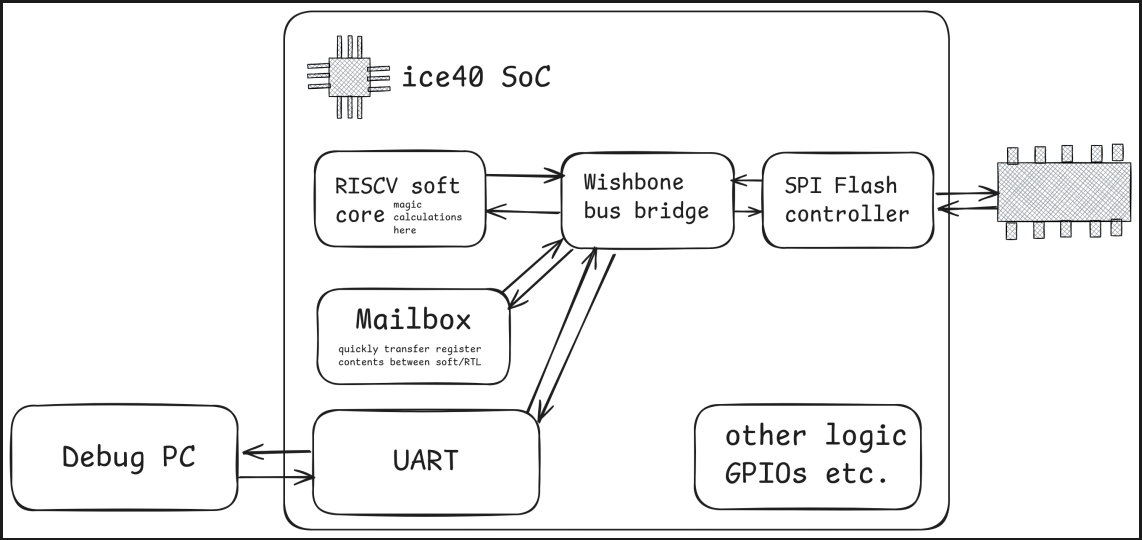

Figure: System outline

So you want to write a mailbox

We use mailboxes to transfer data between softcore and RTL. Since we are programmers, we debug stuff with blinking and printf'ing to UART.

What does one do when one sees such logs:

The logs in question

For 0x1 being written:

```
Eone!wriuing

Bootjng Bpp Jmagf..
```

For 0x2:

```
Fone"wrvtng

Flasj Manufacturgr.
```

For 0x4:

```
Gone#wriwing

Glask Maoufacturgr :#Glask Unkque#ID #  :#Commcnd>#Cootkng Cpp kmagg..
```

Further symptoms

We blame some memory corruption first!

The bug occurs only when we have already changed the value of at least one register via UART. Writing values higher than 0x1 and 0x2 crashes UART but RTL keeps running.

Weird right?

In plain view

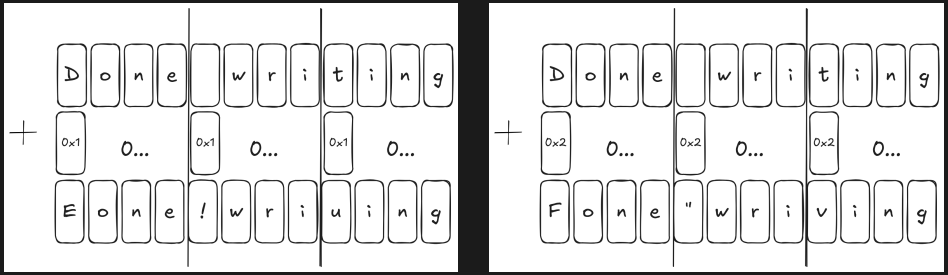

Figure: Illustrated plainly and clearly - ORing happened

Root cause and solution

Turns out the bus arbiter was written in such a way that it OR'ed data coming from various Wishbone clients.

We can blame either the arbiter not being bulletproof enough or my mailbox implementation being sloppy (choose one). The original bus code looks like below:

```
always @(*)
    begin : wb_or
            integer i;
            wb_rdata_or = 0;
            for (i=0; i<WB_N; i=i+1)
                    wb_rdata_or[WB_DW-1:0] = wb_rdata_or[WB_DW-1:0] | wb_rdata[WB_DW*i+:WB_DW];
end
```

Corrected arbiter

Changing it to choosing wb_rdata only from one client solved the issue.

```
reg [31:0] wb_rdata_sel;
always @(*) begin : wb_mux
        wb_rdata_sel = 32'h00000000;
        integer j;
        for (j = 0; j < WB_N; j = j + 1) begin
                if (wb_ack[j]) begin
                        wb_rdata_sel = wb_rdata[j*32 +: 32];
                end
        end
end
```

Other bugs

Bug when burst reading values from flash and saving them in mailboxes. Lack of bursting capabilities in the wishbone. Enabling hardware clock

counters in RISCV caused the clock constraints to be not met (especially with 100 MHz constraints). Had to implement the counter manually in RTL. Other shenanigans with meeting 100 MHz clock

constraints - happy to share some findings afterwards.

That's all folks

Thanks for listening!

Re-use open source :) Big thanks to Sylvain for providing excellent base for hacking around!

~Jakub