FAQ

How to adapt the firmware to my implementation?

Answer:

1. edit minsoc/sw/support/orp.ld line 14 LENGTH = 0x00006E00 to
   a) your memory amount in Bytes $4 \cdot 2^{MEMORYADRWIDTH}$, where MEMORYADRWIDTH is defined in `define MEMORY_ADR_WIDTH in "minsoc/rtl/verilog/minsoc_defines.v"

      $4 \cdot 2^{MEMORYADRWIDTH}$ minus ORIGIN = 0x00001200

      (e.g. 4*2^13 = 32,768 Bytes = 0x8000 | LENGTH = 0x8000 – 0x1200 = 0x6E00)
2. select your STACK size on minsoc/sw/support/board.h line 16 #define STACK_SIZE 0x01000
   a) change your IN_CLK if not using 25000000 (25MHz)

How to configure the simulation

Answer:

1. configure your system: minsoc/rtl/verilog/minsoc_defines.v
   a) **Note:** you can uncomment ETHERNET on minsoc_defines.v to input data to the SoC's Ethernet interface and read data from it
2. configure minsoc/bench/verilog/minsoc_bench_defines.v
   a) Your testbench will use a memory model, not actually the same memory controller the implementation uses. This enables the option "`define INITIALIZE_MEMORY_MODEL", where the firmware is loaded to the memory before testbench start.
   b) You may use the actual implementation memory:
      - comment "`define INITIALIZE_MEMORY_MODEL"
      - edit minsoc/sim/run/generate_bench
        - substitute "../bin/minsoc_model.txt" for "../bin/minsoc_memory.txt"
      - You might want to uncomment "`define START_UP", it loads the firmware to a SPI memory. At start of testbench the system reads this memory and loads the firmware to main memory. Takes +-3 min. This is possible to be used for a real system, all you have to do is uncomment "`define START_UP" from minsoc/rtl/verilog/minsoc_defines.v.
3. Modify testbench

Is it possible to debug the simulation as I debug the firmware running on my board?

Answer: Yes:

Open 3 terminals:

1. terminal 1: from minsoc/sim/run/
   a) ./generate_bench
   b) ./run_bench <your_firmware.hex>
      - ./run_bench ../../sw/uart/uart-nocache-twobyte-sizefirst.hex
2. terminal 2: from minsoc/sim/run
   a) ./start_server
3. terminal 3: at minsoc/sw/uart
   a) or32-elf-gdb uart-nocache.or32
   b) target remote :9999
   c) load
      - if you have INITIALIZE_MEMORY_MODEL enabled you don't have to do this
      - if you have START_UP and waited for the message: “Memory start-up completed...” you also don't need this
   d) set $pc=0x100
   e) c

My device is full, can I reduce the used logic of the SoC?

Answer: yes

1. configure minsoc/rtl/verilog/or1200/rtl/verilog/or1200_defines.v (recommended values for different devices under synthesis_examples.pdf)
   a) Target FPGA memories (OR1200_XILINX_RAMB16 for Xilinx, Spartan 3 and above, OR1200_ALTERA_LPM for all Altera)

      (if you do this, check: I have generate bench errors, what happened?)

   b) Type of register file RAM (OR1200_RFRAM_GENERIC, OR1200_RFRAM_TWOPORT or OR1200_RFRAM_DUALPORT) (dual port is supported by Xilinx BRAM and Altera)

      (select only one of the three)

      - if altera: include `define OR1200_ALTERA_LPM_XXX (if you wish right under `define OR1200_ALTERA_LPM)

   c) comment `define OR1200_PM_IMPLEMENTED
   d) If not using Linux you can:
      - uncomment `define OR1200_NO_DC

- uncomment `define OR1200_NO_IC
- uncomment `define OR1200_NO_DMMU
- uncomment `define OR1200_NO_IMMU
- comment out `define OR1200_CFGR_IMPLEMENTED

e) If you don't need multiplication, mac operations or divisions

- comment out `define OR1200_MULT_IMPLEMENTED
- comment out `define OR1200_MAC_IMPLEMENTED
- comment out `define OR1200_DIV_IMPLEMENTED

(If you do this, change sw/support/Makefile.inc line 7: GCC_OPT=-mhard-mul -g to GCC_OPT=-msoft-mul -g)

I have generate bench errors, what happened?

Answer:

```
foo@ubuntu:~/minsoc/sim/run$ ./generate_bench
../../bench/verilog/minsoc_bench.v:590: error: Could not find variable
``minsoc_top_0.or1200_top.or1200_cpu.or1200_rf.rf_a.ramb16_s36_s36.mem" in ``minsoc_bench.init_fpga_memory"
../../bench/verilog/minsoc_bench.v:591: error: Could not find variable
``minsoc_top_0.or1200_top.or1200_cpu.or1200_rf.rf_b.ramb16_s36_s36.mem" in ``minsoc_bench.init_fpga_memory"
2 error(s) during elaboration.
foo@ubuntu:~/minsoc/sim/run$
```

You tried to use the Xilinx RAM or your specific memory, by uncommenting the 'define OR1200_XILINX_RAMB16 or others in the minsoc/rtl/verilog/or1200/rtl/verilog/or1200_defines.v file.

On or1200_r3 the register file, or1200_rf.v, always instantiates a generic memory for DUAL PORT RAM. Previously it instantiated a target specific or generic memory depending on your sets of or1200_defines.v.
Since the CPU does not work if the registers aren't set to zero previous to simulation start, my testbench especifically set the memory content of the registers to zero, before simulation start. I didn't try to find out why this is like that, I only noticed it was that way.
First, I commented out the initialization for dual port RAM to test if the new memory would work. The simulation failed as it did before.
I could include a new initialization for the new memory. Because the new memory is generic and I believe target specific memory should be used whenever possible, I'd recommend you to switch it back to the way it was before and not to touch the memory initializations:

On or1200_rf.v: lines 304 and 280 edit this way:
304:

```
or1200_dpram_32x32
rf_b
(
.rst_a(rst),
.rst_b(rst),
.oe_a(1'b1),
```

```
// Port A
.clk_a(clk),
.ce_a(rf_enb),
...
280:
/* or1200_dpram #
(
.aw(5),
.dw(32)
)*/
or1200_dpram_32x32
rf_a
(
.rst_a(rst),
.rst_b(rst),
.oe_a(1'b1),
// Port A
.clk_a(clk),
.ce_a(rf_ena),
...
```

This might be a typo or maybe a work around for something else. That's something the OpenRISC developers have to tell us. It is remarkable that both, OR1200_RFRAM_GENERIC and OR1200_TWOPORT are exactly the way there were before. However, the OR1200_RFRAM_DUALPORT uses now a new module, which is basically an adaptation of the OR1200_RFRAM_GENERIC one, but with DUALPORT, of course.