soc_ver arch
Spi12_i2c

TOP

hw_top dut();

```
spi_vif_config::set(null,"*tb.spienv1.**","vif",dut.in_spi1);
spi_vif_config::set(null,"*tb.spienv2.**","vif",dut.in_spi2);
i2c_vif_config::set(null,"*.tb.i2c.**","vif", hw_top.if);


wb_vif_config::set(null,"*tb.wbenv.**","vif",dut.in_wb);
clock_and_reset_vif_config::set(null , "*tb.clk_rst_env.** , "vif" , dut.clk_rst_if);
```

in_wb

clock_and_reset_if

clkgen

in_spi1

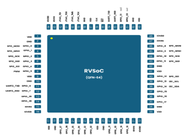
in_spi2

in_i2c

hw_top

```
 wb_if in_wb(clock,reset);
 spi_if in_spi1(clock,reset,sclk,cs);
 spi_if in_spi2(clock,reset,sclk,cs);
 i2c_if in_i2c (,clk(clock), .rst_n(reset));
```

```
top_rv32i_soc DUT (
.CLK_PAD      (clock),
.RESET_N_PAD  (reset),
.O_UART_TX_PAD (o_uart_tx_pad),
.I_UART_RX_PAD (i_uart_rx_pad),
.IO_DATA_PAD  (gpio_pads)
);

clock_and_reset_if clk_rst_if (
  .clock(clock),
  .reset(),
  .run_clock(run_clock),
  .clock_period(clock_period)
);

  clkgen clkgen (
    .clock(clock ),
    .run_clock(run_clock),
    .clock_period(32'd10)
  );

  always @(*) begin
      force DUT.u_rv32i_soc.wb_m2s_io_adr = in_wb.addr;
      force DUT.u_rv32i_soc.wb_m2s_io_dat = in_wb.din;
      force DUT.u_rv32i_soc.wb_m2s_io_sel = 4'b1111;
      force DUT.u_rv32i_soc.wb_m2s_io_we  = in_wb.we;
      force DUT.u_rv32i_soc.wb_m2s_io_stb = in_wb.stb;
      force DUT.u_rv32i_soc.wb_m2s_io_cyc = in_wb.cyc;
      force DUT.u_rv32i_soc.i_flash_miso = in_spi1.miso;

  end
assign in_wb.ack  = DUT.u_rv32i_soc.wb_s2m_io_ack;
assign in_wb.dout = DUT.u_rv32i_soc.wb_s2m_io_dat;
assign cs         = DUT.u_rv32i_soc.o_flash_cs_n;
assign sclk       = DUT.u_rv32i_soc.o_flash_sclk;
assign in_spi1.mosi = DUT.u_rv32i_soc.o_flash_mosi;

assign in_spi2.sclk = gpio_pads[5];        // GPIO_2  - SPI2_SCK
assign in_spi2.miso = gpio_pads[4];        // GPIO_1  - SPI2_MISO
assign gpio_pads[3] = in_spi2.mosi;        // GPIO_0  - SPI2_MOSI
assign in_spi2.cs   = gpio_pads[6];        // GPIO_3  - SPI2_SS0

assign in_spi1.sclk = gpio_pads[46];       // GPIO_10 - SPI1_SCK
assign in_spi1.miso = gpio_pads[45];       // GPIO_9  - SPI1_MISO
assign gpio_pads[44] = in_spi1.mosi;       // GPIO_8  - SPI1_MOSI
assign in_spi1.cs   = gpio_pads[43];       // GPIO_11 - SPI1_SS0

assign gpio_pads[39] = scl_padoen_oe ? 1'bz : scl_pad_o; // SCL open-drain
assign gpio_pads[38] = sda_padoen_oe ? 1'bz : sda_pad_o; // SDA open-drain

assign iif.scl = gpio_pads[39]; // SCL input sampling from pad
assign iif.sda = gpio_pads[38]; // SDA input sampling from pad

// Optional: pullups if not already externally on the board
pullup p1(gpio_pads[39]);
pullup p2(gpio_pads[38]);
```



soc_tb

```
uvm_config_int::set(this, "*wb*", "num_masters", 1);
uvm_config_int::set(this, "*wb*", "num_slaves", 0);

uvm_config_int::set(this, "*spi*", "enable_master", 0);
uvm_config_int::set(this, "*spi*", "enable_slave", 1);

uvm_config_int::set(this, "*i2c*", "num_masters", 0);
uvm_config_int::set(this, "*i2c*", "num_slaves", 1);

wbenv = wb_env::type_id::create("wbenv", this);

spienv1 = spi_env::type_id::create("spienv1", this);
spienv2 = spi_env::type_id::create("spienv2 ", this);

i2cenv = i2c_env::type_id::create("i2cenv ", this);

clk_rst_env = clock_and_reset_env::type_id::create("clk_rst_env", this);
soc_refenv = soc_ref_env::type_id::create("soc_refenv", this);
soc_mcseqr = soc_mcsequencer::type_id::create("soc_mcseqr", this);


soc_mcseqr.wb_seqr = wbenv.masters[0].sequencer;
soc_mcseqr.spi1_s_seqr = spienv1.slave_agent.seqr;
soc_mcseqr.spi2_s_seqr = spienv2.slave_agent.seqr;
soc_mcseqr.i2c_seqr= i2cenv.slaves[0].sequencer;
```

soc_mcsequencer

```
wb_master_sequencer wb_seqr;
spi_slave_sequencer spi1_s_seqr ;
spi_slave_sequencer spi2_s_seqr ;
i2c_slave_sequencer i2c_seqr;
```

wb_env

```
masters[i] = wb_master_agent::type_id::create(inst_name, this);
```

wb_master_agent

```
monitor = wb_master_monitor::type_id::create("monitor", this);
```

wb_master_monitor

uvm_analysis_port #(wb_transaction) item_collected_port;

```
wbenv.masters[0].monitor.item_collected_port.connect(soc_refenv.wb_ref.wb_in);
spienv.slave_agent[0].mon.spi_out.connect(soc_refenv.scb.spi_in1);
spienv.slave_agent[1].mon.spi_out.connect(soc_refenv.scb.spi_in2);
i2cenv.slaves[0].monitor. i2c_analysis_port.connect(soc_refenv.scb.i2c_in);
```

need to add the spi2_seqr to the mcseqr

```
soc_ref_env
scb = soc_scb::type_id::create("scb", this);
wb_ref = wb_ref_model::type_id::create("wb_ref", this) ;
spiref_model1 = wb_x_spi_module::type_id::create("spiref_model1", this);
spiref_model2 = wb_x_spi_module::type_id::create("spiref_model2", this);
i2cref_model = i2c_module ::type_id::create("i2cref_model", this);
```

```
wb_ref.wb2scbspi1_port.connect(scb.spi1ref);
wb_ref.wb2scbspi2_port.connect(scb.spi2ref_in);
wb_ref.wb2scbi2c_port.connect(scb.i2cref_in);
wb_ref.wb2spi1ref_port.connect(spiref_model1.wb_in);
wb_ref.wb2spi2ref_port.connect(spiref_model2.wb_in);
wb2i2cref_port.connect(i2cref_model.wb_in);
```

need to change to : wb_tlram_ref_model

wb_ref_model

```
uvm_analysis_port #(wb_transaction) wb2scbspi1_port,
uvm_analysis_port #(wb_transaction) wb2scbspi2_port,
uvm_analysis_port #(wb_transaction) wb2scbi2c_port,

uvm_analysis_port #(wb_transaction) wb2spi1ref_port,

uvm_analysis_port #(wb_transaction) wb2spi2ref_port,
uvm_analysis_port #(wb_transaction) wb2i2cref_port,

uvm_analysis_imp_decl(_wb)
uvm_analysis_imp_wb#(wb_transaction, wb_ref_model) wb_in;
```

```
void write_wb(wb_transaction t);

// SPI 1: 0x20000200 - 0x2000027F
if (tr.addr >= 32'h20000200 && tr.addr <= 32'h2000027F) begin
  wb2spi1ref_port.write(tr);
  wb2scbspi1_port.write(tr);

end
// SPI 2: 0x20000280 - 0x200002FF
else if (tr.addr >= 32'h20000280 && tr.addr <= 32'h200002FF) begin
  wb2spi2ref_port.write(tr);
  wb2scbspi2_port.write(tr);

end
// UART: 0x20000000 - 0x200000FF
else if (tr.addr >= 32'h20000000 && tr.addr <= 32'h200000FF) begin
  // wb2uartref_port.write(tr);
  // wb2scb_port.write(tr);

end
// GPIO: 0x20000100 - 0x200001FF
else if (tr.addr >= 32'h20000100 && tr.addr <= 32'h200001FF) begin
  //wb2gpioref_port.write(tr);
  // wb2scb_port.write(tr);

end
// I2C: 0x20000300 - 0x200003FF
else if (tr.addr >= 32'h20000300 && tr.addr <= 32'h200003FF) begin
  wb2i2cref_port.write(tr);
  wb2scbi2c_port.write(tr);

end
// PTC (PWM): 0x20000400 - 0x200004FF
else if (tr.addr >= 32'h20000400 && tr.addr <= 32'h200004FF) begin
  //wb2ptcref_port.write(tr);
  //wb2scb_port.write(tr);

end

endfunction: write_wb
```

wb_x_spi_module

```
uvm_analysis_imp_decl(_wb)
uvm_analysis_imp_wb#(wb_transaction, wb_x_spi_module) wb_in;
```

get functions

example func getreg1()

example func getreg2()

example func getreg3()

wb_x_spi_module

```
uvm_analysis_imp_decl(_wb)
uvm_analysis_imp_wb#(wb_transaction, wb_x_spi_module) wb_in;
```

get functions

example func getreg1()

example func getreg2()

example func getreg3()

wb_x_i2c_ref_model

```
uvm_analysis_imp_decl(_wb)
uvm_analysis_imp_wb#(wb_transaction, wb_x_i2c_ref_model) wb_in;
```

get functions

example func getreg1()

example func getreg2()

example func getreg3()

soc_scb

```
uvm_analysis_imp_decl(_spi1ref)
uvm_analysis_imp_spi1ref#(wb_transaction, soc_scb) spi1ref_in;

uvm_analysis_imp_decl(_spi2ref)
uvm_analysis_imp_spi2ref#(wb_transaction, soc_scb) spi2ref_in;

uvm_analysis_imp_decl(_i2c)
uvm_analysis_imp_i2cref#(wb_transaction, soc_scb) i2cref_in;
```

spiref_model1 spi1

spi1.getreg1();

spiref_model2 spi2

spi2.getreg1();

i2cref_model i2c

i2c.getreg1();

```
`uvm_analysis_imp_decl(_spi1)
uvm_analysis_imp_spi1#(spi_transaction, soc_scb) spi_in1;

`uvm_analysis_imp_decl(_spi2)
uvm_analysis_imp_spi1#(spi_transaction, soc_scb) spi_in2;

`uvm_analysis_imp_decl(_i2c)
uvm_analysis_imp_i2c#(spi_transaction, soc_scb) i2c_in;
```

spi_env

```
slave_agent = spi_slave_agent::type_id::create("slave_agent", this);
```

slave_agent

```
mon = spi_slave_monitor::type_id::create("mon",this);
```

spi_slave_monitor

uvm_analysis_port#(spi_transaction) spi_out;

spi_env

```
slave_agent = spi_slave_agent::type_id::create("slave_agent", this);
```

slave_agent

```
mon = spi_slave_monitor::type_id::create("mon",this);
```

spi_slave_monitor

uvm_analysis_port#(spi_transaction) spi_out;

i2c_env

```
slaves[i]  = i2c_slave_agent::type_id::create(inst_name, this);
```

slave_agent

```
monitor = i2c_slave_monitor::type_id::create("monitor", this);
```

spi_slave_monitor

uvm_analysis_port #(i2c_transaction) i2c_analysis_port;