

Modificações ao Sistema RVfpga

Neste documento, resumimos as alterações que deve efetuar no sistema RVfpga para completar os exercícios dos RVfpga Labs 6-10. Esta versão alargada do sistema RVfpga, disponível em *[RVfpgaPath]/RVfpga/Labs/RVfpgaLabsSolutions/RVfpga_Solutions/src*, inclui todas as alterações em conjunto. Descrevemos aqui as alterações específicas necessárias para completar cada um dos exercícios de laboratório.

Os exercícios que exigem alterações ao sistema RVfpga são:

- Lab 6 – Exercício 3
 - Lab 6 – Exercício 4
 - Lab 7 – Exercício 3
 - Lab 8 – Exercício 2
 - Lab 9 – Exercício 2
-

Lab 6 – Exercício 3. Expanda o RVfpgaNexys para aceder aos cinco botões de pressão da placa.

- rvpfganexys.xdc

```
89 ##Buttons
90 set_property -dict { PACKAGE_PIN M17 IOSTANDARD LVCMOS33 } [get_ports { i_pb[0] }]; #IO_L9P_T1_D05_14 Sch=btnc
91 set_property -dict { PACKAGE_PIN M18 IOSTANDARD LVCMOS33 } [get_ports { i_pb[1] }]; #IO_L4N_T0_D05_14 Sch=btnc
92 set_property -dict { PACKAGE_PIN P17 IOSTANDARD LVCMOS33 } [get_ports { i_pb[2] }]; #IO_L12P_T1_MRCC_14 Sch=btnc
93 set_property -dict { PACKAGE_PIN M17 IOSTANDARD LVCMOS33 } [get_ports { i_pb[3] }]; #IO_L10N_T1_D15_14 Sch=btnc
94 set_property -dict { PACKAGE_PIN P18 IOSTANDARD LVCMOS33 } [get_ports { i_pb[4] }]; #IO_L9N_T1_D05_D13_14 Sch=btnd
```

- rvpfganexys.v

```
50      inout wire [4:0] i_pb,
```

```
267      .io_data2      (i_pb[4:0]),
```

- swervolf_core.v

```
85      inout wire [4:0] io_data2,
```

```
374 // GPIO2 - 5 Pushbuttons
375 wire [4:0] en_gpio2;
376 wire      gpio2_irq;
377 wire [4:0] i_gpio2;
378 wire [4:0] o_gpio2;
379
380 bidirec gpio2_0 (.oe(en_gpio2[0]), .inp(o_gpio2[0]), .outp(i_gpio2[0]), .bidir(io_data2[0]));
381 bidirec gpio2_1 (.oe(en_gpio2[1]), .inp(o_gpio2[1]), .outp(i_gpio2[1]), .bidir(io_data2[1]));
382 bidirec gpio2_2 (.oe(en_gpio2[2]), .inp(o_gpio2[2]), .outp(i_gpio2[2]), .bidir(io_data2[2]));
383 bidirec gpio2_3 (.oe(en_gpio2[3]), .inp(o_gpio2[3]), .outp(i_gpio2[3]), .bidir(io_data2[3]));
384 bidirec gpio2_4 (.oe(en_gpio2[4]), .inp(o_gpio2[4]), .outp(i_gpio2[4]), .bidir(io_data2[4]));
385
386 gpio_top gpio2_module(
387     .wb_clk_i      (clk),
388     .wb_rst_i      (wb_rst),
389     .wb_cyc_i      (wb_m2s_gpio2_cyc),
390     .wb_adr_i      ({2'b0,wb_m2s_gpio2_adr[5:2],2'b0}),
391     .wb_dat_i      (wb_m2s_gpio2_dat),
392     .wb_sel_i      (4'b1111),
393     .wb_we_i      (wb_m2s_gpio2_we),
394     .wb_stb_i      (wb_m2s_gpio2_stb),
395     .wb_dat_o      (wb_s2m_gpio2_dat),
396     .wb_ack_o      (wb_s2m_gpio2_ack),
397     .wb_err_o      (wb_s2m_gpio2_err),
398     .wb_inta_o     (gpio2_irq),
399
400     // External GPIO Interface
401     .ext_pad_i     (i_gpio2[4:0]),
402     .ext_pad_o     (o_gpio2[4:0]),
403     .ext_padoe_o   (en_gpio2));
```

- wb_intercon.vh

```

76 // GPIO2
77 wire [31:0] wb_m2s_gpio2_adr;
78 wire [31:0] wb_m2s_gpio2_dat;
79 wire [3:0] wb_m2s_gpio2_sel;
80 wire      wb_m2s_gpio2_we;
81 wire      wb_m2s_gpio2_cyc;
82 wire      wb_m2s_gpio2_stb;
83 wire [2:0] wb_m2s_gpio2_cti;
84 wire [1:0] wb_m2s_gpio2_bte;
85 wire [31:0] wb_s2m_gpio2_dat;
86 wire      wb_s2m_gpio2_ack;
87 wire      wb_s2m_gpio2_err;
88 wire      wb_s2m_gpio2_rty;

```

```

236 // GPIO2
237 .wb_gpio2_adr_o      (wb_m2s_gpio2_adr),
238 .wb_gpio2_dat_o      (wb_m2s_gpio2_dat),
239 .wb_gpio2_sel_o      (wb_m2s_gpio2_sel),
240 .wb_gpio2_we_o       (wb_m2s_gpio2_we),
241 .wb_gpio2_cyc_o      (wb_m2s_gpio2_cyc),
242 .wb_gpio2_stb_o      (wb_m2s_gpio2_stb),
243 .wb_gpio2_cti_o      (wb_m2s_gpio2_cti),
244 .wb_gpio2_bte_o      (wb_m2s_gpio2_bte),
245 .wb_gpio2_dat_i      (wb_s2m_gpio2_dat),
246 .wb_gpio2_ack_i      (wb_s2m_gpio2_ack),
247 .wb_gpio2_err_i      (wb_s2m_gpio2_err),
248 .wb_gpio2_rty_i      (wb_s2m_gpio2_rty),

```

- wb_intercon.v

```

79 // GPIO2
80 output [31:0] wb_gpio2_adr_o,
81 output [31:0] wb_gpio2_dat_o,
82 output [3:0]  wb_gpio2_sel_o,
83 output        wb_gpio2_we_o,
84 output        wb_gpio2_cyc_o,
85 output        wb_gpio2_stb_o,
86 output [2:0]  wb_gpio2_cti_o,
87 output [1:0]  wb_gpio2_bte_o,
88 input  [31:0] wb_gpio2_dat_i,
89 input        wb_gpio2_ack_i,
90 input        wb_gpio2_err_i,
91 input        wb_gpio2_rty_i,

```

```

164 wb_mux
165 #(.num_slaves (1)),
166 .MATCH_ADDR ((32'h00000000, 32'h00001000, 32'h00001040, 32'h00001100, 32'h00001200, 32'h00001240, 32'h00001280, 32'h000012c0, 32'h00001400, 32'h00001800, 32'h00002000)),
167 .MATCH_MASK ((32'hfffff000, 32'hfffff000, 32'hfffff000, 32'hfffff000, 32'hfffff000, 32'hfffff000, 32'hfffff000, 32'hfffff000, 32'hfffff000, 32'hfffff000, 32'hfffff000)),
168 wb_mux_io
169 (wb_clk_i (wb_clk_i),
170  wb_rst_i (wb_rst_i),
171  wb_m2s_i (wb_m2s_i),
172  wb_m2s_o (wb_m2s_o),
173  wb_m2s_sel_i (wb_m2s_sel_i),
174  wb_m2s_we_i (wb_m2s_we_i),
175  wb_m2s_cyc_i (wb_m2s_cyc_i),
176  wb_m2s_stb_i (wb_m2s_stb_i),
177  wb_m2s_cti_i (wb_m2s_cti_i),
178  wb_m2s_bte_i (wb_m2s_bte_i),
179  wb_m2s_dat_o (wb_m2s_dat_o),
180  wb_m2s_ack_o (wb_m2s_ack_o),
181  wb_m2s_err_o (wb_m2s_err_o),
182  wb_m2s_rty_o (wb_m2s_rty_o),
183  wb_s2m_adr_o (wb_s2m_adr_o, wb_sys_adr_o, wb_spi_flash_adr_o, wb_spi_accel_adr_o, wb_ptc_adr_o, wb_ptc2_adr_o, wb_ptc3_adr_o, wb_ptc4_adr_o, wb_gpio2_adr_o, wb_uart_adr_o),
184  wb_s2m_dat_o (wb_s2m_dat_o, wb_sys_dat_o, wb_spi_flash_dat_o, wb_spi_accel_dat_o, wb_ptc_dat_o, wb_ptc2_dat_o, wb_ptc3_dat_o, wb_ptc4_dat_o, wb_gpio2_dat_o, wb_uart_dat_o),
185  wb_s2m_sel_o (wb_s2m_sel_o, wb_sys_sel_o, wb_spi_flash_sel_o, wb_spi_accel_sel_o, wb_ptc_sel_o, wb_ptc2_sel_o, wb_ptc3_sel_o, wb_ptc4_sel_o, wb_gpio2_sel_o, wb_uart_sel_o),
186  wb_s2m_we_o (wb_s2m_we_o, wb_sys_we_o, wb_spi_flash_we_o, wb_spi_accel_we_o, wb_ptc_we_o, wb_ptc2_we_o, wb_ptc3_we_o, wb_ptc4_we_o, wb_gpio2_we_o, wb_uart_we_o),
187  wb_s2m_cyc_o (wb_s2m_cyc_o, wb_sys_cyc_o, wb_spi_flash_cyc_o, wb_spi_accel_cyc_o, wb_ptc_cyc_o, wb_ptc2_cyc_o, wb_ptc3_cyc_o, wb_ptc4_cyc_o, wb_gpio2_cyc_o, wb_uart_cyc_o),
188  wb_s2m_stb_o (wb_s2m_stb_o, wb_sys_stb_o, wb_spi_flash_stb_o, wb_spi_accel_stb_o, wb_ptc_stb_o, wb_ptc2_stb_o, wb_ptc3_stb_o, wb_ptc4_stb_o, wb_gpio2_stb_o, wb_uart_stb_o),
189  wb_s2m_cti_o (wb_s2m_cti_o, wb_sys_cti_o, wb_spi_flash_cti_o, wb_spi_accel_cti_o, wb_ptc_cti_o, wb_ptc2_cti_o, wb_ptc3_cti_o, wb_ptc4_cti_o, wb_gpio2_cti_o, wb_uart_cti_o),
190  wb_s2m_bte_o (wb_s2m_bte_o, wb_sys_bte_o, wb_spi_flash_bte_o, wb_spi_accel_bte_o, wb_ptc_bte_o, wb_ptc2_bte_o, wb_ptc3_bte_o, wb_ptc4_bte_o, wb_gpio2_bte_o, wb_uart_bte_o),
191  wb_s2m_dat_i (wb_s2m_dat_i, wb_sys_dat_i, wb_spi_flash_dat_i, wb_spi_accel_dat_i, wb_ptc_dat_i, wb_ptc2_dat_i, wb_ptc3_dat_i, wb_ptc4_dat_i, wb_gpio2_dat_i, wb_uart_dat_i),
192  wb_s2m_ack_i (wb_s2m_ack_i, wb_sys_ack_i, wb_spi_flash_ack_i, wb_spi_accel_ack_i, wb_ptc_ack_i, wb_ptc2_ack_i, wb_ptc3_ack_i, wb_ptc4_ack_i, wb_gpio2_ack_i, wb_uart_ack_i),
193  wb_s2m_err_i (wb_s2m_err_i, wb_sys_err_i, wb_spi_flash_err_i, wb_spi_accel_err_i, wb_ptc_err_i, wb_ptc2_err_i, wb_ptc3_err_i, wb_ptc4_err_i, wb_gpio2_err_i, wb_uart_err_i),
194  wb_s2m_rty_i (wb_s2m_rty_i, wb_sys_rty_i, wb_spi_flash_rty_i, wb_spi_accel_rty_i, wb_ptc_rty_i, wb_ptc2_rty_i, wb_ptc3_rty_i, wb_ptc4_rty_i, wb_gpio2_rty_i, wb_uart_rty_i));
195

```

Lab 6 – Exercício 4. Desenhe outro controlador para os cinco botões de pressão da placa. Ao contrário do exercício anterior, neste caso deve implementar o seu próprio controlador GPIO em Verilog/SystemVerilog.

- rvpfganexys.xdc

```
89 ##Buttons
90 set_property -dict { PACKAGE_PIN N17 IOSTANDARD LVCMOS33 } [get_ports { i_pb[0] }]; #IO_L9P_T1_DQS_14 Sch=btnc
91 set_property -dict { PACKAGE_PIN M18 IOSTANDARD LVCMOS33 } [get_ports { i_pb[1] }]; #IO_L4N_T0_D05_14 Sch=btnc
92 set_property -dict { PACKAGE_PIN P17 IOSTANDARD LVCMOS33 } [get_ports { i_pb[2] }]; #IO_L12P_T1_MRCC_14 Sch=btnc
93 set_property -dict { PACKAGE_PIN M17 IOSTANDARD LVCMOS33 } [get_ports { i_pb[3] }]; #IO_L10N_T1_D15_14 Sch=btnc
94 set_property -dict { PACKAGE_PIN P18 IOSTANDARD LVCMOS33 } [get_ports { i_pb[4] }]; #IO_L9N_T1_DQS_D13_14 Sch=btnc
```

- rvpfganexys.sv

```
50      inout wire [4:0] i_pb,
```

```
267      .io_data2      (i_pb[4:0]),
```

- swervolf_core.v

```
85      inout wire [4:0] io_data2,
```

```
236      .i_gpio2      (i_gpio2[4:0]),
```

- swervolf_syscon.v

```
34      input wire [4:0] i_gpio2,
```

```
77      reg [4:0] i_gpio2_reg;
78
79
80      always @(posedge i_clk) begin
81          i_gpio2_reg[4:0] <= i_gpio2[4:0];
82      end
```

```
270      //0x1B-0x1F
271      7 : o_wb_rdt <= {27'd0, i_gpio2_reg[4:0]};
```

Lab 7 – Exercício 3. Modifique o controlador descrito neste laboratório para que os 8 mostradores de 7 segmentos possam mostrar qualquer combinação de LEDs (segmentos) ON/OFF.

- swervolf_syscon.v

```

236     14 : begin
237         if (i_wb_sel[0]) Segments_Digit0 <= i_wb_dat[7:0];
238         if (i_wb_sel[1]) Segments_Digit1 <= i_wb_dat[15:8];
239         if (i_wb_sel[2]) Segments_Digit2 <= i_wb_dat[23:16];
240         if (i_wb_sel[3]) Segments_Digit3 <= i_wb_dat[31:24];
241     end
242     15 : begin
243         if (i_wb_sel[0]) Segments_Digit4 <= i_wb_dat[7:0];
244         if (i_wb_sel[1]) Segments_Digit5 <= i_wb_dat[15:8];
245         if (i_wb_sel[2]) Segments_Digit6 <= i_wb_dat[23:16];
246         if (i_wb_sel[3]) Segments_Digit7 <= i_wb_dat[31:24];
247     end

```

```

298 // Eight-Digit 7 Segment Displays
299
300 reg [6:0] Segments_Digit0;
301 reg [6:0] Segments_Digit1;
302 reg [6:0] Segments_Digit2;
303 reg [6:0] Segments_Digit3;
304 reg [6:0] Segments_Digit4;
305 reg [6:0] Segments_Digit5;
306 reg [6:0] Segments_Digit6;
307 reg [6:0] Segments_Digit7;
308
309 SevSegDisplays_Controller SegDispl_Ctr(
310     .clk          (i_clk),
311     .rst_n        (i_rst),
312     .Segments_Digit0 (Segments_Digit0),
313     .Segments_Digit1 (Segments_Digit1),
314     .Segments_Digit2 (Segments_Digit2),
315     .Segments_Digit3 (Segments_Digit3),
316     .Segments_Digit4 (Segments_Digit4),
317     .Segments_Digit5 (Segments_Digit5),
318     .Segments_Digit6 (Segments_Digit6),
319     .Segments_Digit7 (Segments_Digit7),
320     .AN             (AN),
321     .Digits_Bits    (Digits_Bits)
322 );
323

```

```

331 module SevSegDisplays_Controller(
332     input wire      clk,
333     input wire      rst_n,
334     input wire [6:0] Segments_Digit0,
335     input wire [6:0] Segments_Digit1,
336     input wire [6:0] Segments_Digit2,
337     input wire [6:0] Segments_Digit3,
338     input wire [6:0] Segments_Digit4,
339     input wire [6:0] Segments_Digit5,
340     input wire [6:0] Segments_Digit6,
341     input wire [6:0] Segments_Digit7,
342     output wire [7:0] AN,
343     output wire [6:0] Digits_Bits);
344
345 wire [(COUNT_MAX-1):0] countSelection;
346 wire overflow_o_count;
347
348
349 counter #(COUNT_MAX) counter20(clk, ~rst_n, 1'b0, 1'b1, 1'b0, 1'b0, 16'b0, countSelection, overflow_o_count);
350
351
352 wire [7:0] [7:0] enable;
353
354 assign enable[0] = (8'hfe);
355 assign enable[1] = (8'hfd);
356 assign enable[2] = (8'hfb);
357 assign enable[3] = (8'hf7);
358 assign enable[4] = (8'hcf);
359 assign enable[5] = (8'hdf);
360 assign enable[6] = (8'hbf);
361 assign enable[7] = (8'h7f);
362
363 SevSegMux
364 #(
365     .DATA_WIDTH(8),
366     .N_IN(8)
367 )
368 Select_Enables
369 (
370     .IN_DATA(enable),
371     .OUT_DATA(AN),
372     .SEL(countSelection[(COUNT_MAX-1):(COUNT_MAX-3)])
373 );
374
375
376 wire [7:0] [6:0] digits_concat;
377
378 assign digits_concat[0] = Segments_Digit0;
379 assign digits_concat[1] = Segments_Digit1;
380 assign digits_concat[2] = Segments_Digit2;
381 assign digits_concat[3] = Segments_Digit3;
382 assign digits_concat[4] = Segments_Digit4;
383 assign digits_concat[5] = Segments_Digit5;
384 assign digits_concat[6] = Segments_Digit6;
385 assign digits_concat[7] = Segments_Digit7;
386
387 SevSegMux
388 #(
389     .DATA_WIDTH(7),
390     .N_IN(8)
391 )
392 Select_Digits
393 (
394     .IN_DATA(digits_concat),
395     .OUT_DATA(Digits_Bits),
396     .SEL(countSelection[(COUNT_MAX-1):(COUNT_MAX-3)])
397 );
398
399 endmodule

```

Lab 8 – Exercício 2. Modifique o RVfpgaNexys de modo a ligar o sinal de saída PWM do temporizador a um dos dois LEDs tricolores disponíveis na placa Nexys A7.

- rvpfganexys.xdc

```
60 ## RGB LEDs
61 set_property -dict { PACKAGE_PIN R12 IOSTANDARD LVCMOS33 } [get_ports { LED16_B }]; #IO L5P_T0_D06_14 Sch=led16_b
62 set_property -dict { PACKAGE_PIN M16 IOSTANDARD LVCMOS33 } [get_ports { LED16_G }]; #IO L10P_T1_D14_14 Sch=led16_g
63 set_property -dict { PACKAGE_PIN N15 IOSTANDARD LVCMOS33 } [get_ports { LED16_R }]; #IO L11P_T1_SRCC_14 Sch=led16_r
```

- rvpfganexys.sv

```
53 output wire LED16_B,
54 output wire LED16_G,
55 output wire LED16_R,
56 output wire o_accel_cs_n,
57 output wire o_accel_mosi,
58 input wire i_accel_miso,
59 output wire accel_sclk);
60
61 wire [15:0] gpio_out;
62
63 wire pwm_pad_o_ptc2;
64 wire pwm_pad_o_ptc3;
65 wire pwm_pad_o_ptc4;
```

```
270 .pwm_pad_o_ptc2 (pwm_pad_o_ptc2),
271 .pwm_pad_o_ptc3 (pwm_pad_o_ptc3),
272 .pwm_pad_o_ptc4 (pwm_pad_o_ptc4),
```

```
282 assign LED16_B = pwm_pad_o_ptc2;
283 assign LED16_R = pwm_pad_o_ptc3;
284 assign LED16_G = pwm_pad_o_ptc4;
```

- swervolf_core.v

```
90 output wire pwm_pad_o_ptc2,
91 output wire pwm_pad_o_ptc3,
92 output wire pwm_pad_o_ptc4,
```

```

431 // PTC2
432 wire      ptc2_irq;
433
434 ptc_top timer_ptc2(
435     .wb_clk_i      (clk),
436     .wb_rst_i      (wb_rst),
437     .wb_cyc_i      (wb_m2s_ptc2_cyc),
438     .wb_adr_i      ({2'b0,wb_m2s_ptc2_adr[5:2],2'b0}),
439     .wb_dat_i      (wb_m2s_ptc2_dat),
440     .wb_sel_i      (4'b1111),
441     .wb_we_i      (wb_m2s_ptc2_we),
442     .wb_stb_i      (wb_m2s_ptc2_stb),
443     .wb_dat_o      (wb_s2m_ptc2_dat),
444     .wb_ack_o      (wb_s2m_ptc2_ack),
445     .wb_err_o      (wb_s2m_ptc2_err),
446     .wb_inta_o     (ptc2_irq),
447     // External PTC Interface
448     .gate_clk_pad_i (),
449     .capt_pad_i (),
450     .pwm_pad_o     (pwm_pad_o_ptc2),
451     .oen_padoen_o ()
452 );
453
454
455 // PTC3
456 wire      ptc3_irq;
457
458 ptc_top timer_ptc3(
459     .wb_clk_i      (clk),
460     .wb_rst_i      (wb_rst),
461     .wb_cyc_i      (wb_m2s_ptc3_cyc),
462     .wb_adr_i      ({2'b0,wb_m2s_ptc3_adr[5:2],2'b0}),
463     .wb_dat_i      (wb_m2s_ptc3_dat),
464     .wb_sel_i      (4'b1111),
465     .wb_we_i      (wb_m2s_ptc3_we),
466     .wb_stb_i      (wb_m2s_ptc3_stb),
467     .wb_dat_o      (wb_s2m_ptc3_dat),
468     .wb_ack_o      (wb_s2m_ptc3_ack),
469     .wb_err_o      (wb_s2m_ptc3_err),
470     .wb_inta_o     (ptc3_irq),
471     // External PTC Interface
472     .gate_clk_pad_i (),
473     .capt_pad_i (),
474     .pwm_pad_o     (pwm_pad_o_ptc3),
475     .oen_padoen_o ()
476 );
477
478
479 // PTC4
480 wire      ptc4_irq;
481
482 ptc_top timer_ptc4(
483     .wb_clk_i      (clk),
484     .wb_rst_i      (wb_rst),
485     .wb_cyc_i      (wb_m2s_ptc4_cyc),
486     .wb_adr_i      ({2'b0,wb_m2s_ptc4_adr[5:2],2'b0}),
487     .wb_dat_i      (wb_m2s_ptc4_dat),
488     .wb_sel_i      (4'b1111),
489     .wb_we_i      (wb_m2s_ptc4_we),
490     .wb_stb_i      (wb_m2s_ptc4_stb),
491     .wb_dat_o      (wb_s2m_ptc4_dat),
492     .wb_ack_o      (wb_s2m_ptc4_ack),
493     .wb_err_o      (wb_s2m_ptc4_err),
494     .wb_inta_o     (ptc4_irq),
495     // External PTC Interface
496     .gate_clk_pad_i (),
497     .capt_pad_i (),
498     .pwm_pad_o     (pwm_pad_o_ptc4),
499     .oen_padoen_o ()
500 );

```

- wb_intercon.vh

```

104 // PTC2
105 wire [31:0] wb_m2s_ptc2_adr;
106 wire [31:0] wb_m2s_ptc2_dat;
107 wire [3:0] wb_m2s_ptc2_sel;
108 wire wb_m2s_ptc2_we;
109 wire wb_m2s_ptc2_cyc;
110 wire wb_m2s_ptc2_stb;
111 wire [2:0] wb_m2s_ptc2_cti;
112 wire [1:0] wb_m2s_ptc2_bte;
113 wire [31:0] wb_s2m_ptc2_dat;
114 wire wb_s2m_ptc2_ack;
115 wire wb_s2m_ptc2_err;
116 wire wb_s2m_ptc2_rty;
117
118 // PTC3
119 wire [31:0] wb_m2s_ptc3_adr;
120 wire [31:0] wb_m2s_ptc3_dat;
121 wire [3:0] wb_m2s_ptc3_sel;
122 wire wb_m2s_ptc3_we;
123 wire wb_m2s_ptc3_cyc;
124 wire wb_m2s_ptc3_stb;
125 wire [2:0] wb_m2s_ptc3_cti;
126 wire [1:0] wb_m2s_ptc3_bte;
127 wire [31:0] wb_s2m_ptc3_dat;
128 wire wb_s2m_ptc3_ack;
129 wire wb_s2m_ptc3_err;
130 wire wb_s2m_ptc3_rty;
131
132 // PTC4
133 wire [31:0] wb_m2s_ptc4_adr;
134 wire [31:0] wb_m2s_ptc4_dat;
135 wire [3:0] wb_m2s_ptc4_sel;
136 wire wb_m2s_ptc4_we;
137 wire wb_m2s_ptc4_cyc;
138 wire wb_m2s_ptc4_stb;
139 wire [2:0] wb_m2s_ptc4_cti;
140 wire [1:0] wb_m2s_ptc4_bte;
141 wire [31:0] wb_s2m_ptc4_dat;
142 wire wb_s2m_ptc4_ack;
143 wire wb_s2m_ptc4_err;
144 wire wb_s2m_ptc4_rty;

```

```

262 // PTC2
263 .wb_ptc2_adr_o      (wb_m2s_ptc2_adr),
264 .wb_ptc2_dat_o      (wb_m2s_ptc2_dat),
265 .wb_ptc2_sel_o      (wb_m2s_ptc2_sel),
266 .wb_ptc2_we_o       (wb_m2s_ptc2_we),
267 .wb_ptc2_cyc_o      (wb_m2s_ptc2_cyc),
268 .wb_ptc2_stb_o      (wb_m2s_ptc2_stb),
269 .wb_ptc2_cti_o      (wb_m2s_ptc2_cti),
270 .wb_ptc2_bte_o      (wb_m2s_ptc2_bte),
271 .wb_ptc2_dat_i      (wb_s2m_ptc2_dat),
272 .wb_ptc2_ack_i      (wb_s2m_ptc2_ack),
273 .wb_ptc2_err_i      (wb_s2m_ptc2_err),
274 .wb_ptc2_rty_i      (wb_s2m_ptc2_rty),
275
276 // PTC3
277 .wb_ptc3_adr_o      (wb_m2s_ptc3_adr),
278 .wb_ptc3_dat_o      (wb_m2s_ptc3_dat),
279 .wb_ptc3_sel_o      (wb_m2s_ptc3_sel),
280 .wb_ptc3_we_o       (wb_m2s_ptc3_we),
281 .wb_ptc3_cyc_o      (wb_m2s_ptc3_cyc),
282 .wb_ptc3_stb_o      (wb_m2s_ptc3_stb),
283 .wb_ptc3_cti_o      (wb_m2s_ptc3_cti),
284 .wb_ptc3_bte_o      (wb_m2s_ptc3_bte),
285 .wb_ptc3_dat_i      (wb_s2m_ptc3_dat),
286 .wb_ptc3_ack_i      (wb_s2m_ptc3_ack),
287 .wb_ptc3_err_i      (wb_s2m_ptc3_err),
288 .wb_ptc3_rty_i      (wb_s2m_ptc3_rty),
289
290 // PTC4
291 .wb_ptc4_adr_o      (wb_m2s_ptc4_adr),
292 .wb_ptc4_dat_o      (wb_m2s_ptc4_dat),
293 .wb_ptc4_sel_o      (wb_m2s_ptc4_sel),
294 .wb_ptc4_we_o       (wb_m2s_ptc4_we),
295 .wb_ptc4_cyc_o      (wb_m2s_ptc4_cyc),
296 .wb_ptc4_stb_o      (wb_m2s_ptc4_stb),
297 .wb_ptc4_cti_o      (wb_m2s_ptc4_cti),
298 .wb_ptc4_bte_o      (wb_m2s_ptc4_bte),
299 .wb_ptc4_dat_i      (wb_s2m_ptc4_dat),
300 .wb_ptc4_ack_i      (wb_s2m_ptc4_ack),
301 .wb_ptc4_err_i      (wb_s2m_ptc4_err),
302 .wb_ptc4_rty_i      (wb_s2m_ptc4_rty),

```

- wb_intercon.v

```

107 // PTC2
108 output [31:0] wb_ptc2_adr_o,
109 output [31:0] wb_ptc2_dat_o,
110 output [3:0] wb_ptc2_sel_o,
111 output wb_ptc2_we_o,
112 output wb_ptc2_cyc_o,
113 output wb_ptc2_stb_o,
114 output [2:0] wb_ptc2_cti_o,
115 output [1:0] wb_ptc2_bte_o,
116 input [31:0] wb_ptc2_dat_i,
117 input wb_ptc2_ack_i,
118 input wb_ptc2_err_i,
119 input wb_ptc2_rty_i,
120
121 // PTC3
122 output [31:0] wb_ptc3_adr_o,
123 output [31:0] wb_ptc3_dat_o,
124 output [3:0] wb_ptc3_sel_o,
125 output wb_ptc3_we_o,
126 output wb_ptc3_cyc_o,
127 output wb_ptc3_stb_o,
128 output [2:0] wb_ptc3_cti_o,
129 output [1:0] wb_ptc3_bte_o,
130 input [31:0] wb_ptc3_dat_i,
131 input wb_ptc3_ack_i,
132 input wb_ptc3_err_i,
133 input wb_ptc3_rty_i,
134
135 // PTC4
136 output [31:0] wb_ptc4_adr_o,
137 output [31:0] wb_ptc4_dat_o,
138 output [3:0] wb_ptc4_sel_o,
139 output wb_ptc4_we_o,
140 output wb_ptc4_cyc_o,
141 output wb_ptc4_stb_o,
142 output [2:0] wb_ptc4_cti_o,
143 output [1:0] wb_ptc4_bte_o,
144 input [31:0] wb_ptc4_dat_i,
145 input wb_ptc4_ack_i,
146 input wb_ptc4_err_i,
147 input wb_ptc4_rty_i,

```

```

164 wb_mux
165 # (num_slaves (1)),
166 MATCH_ADDR ((32'h00000000, 32'h00001000, 32'h00001040, 32'h00001100, 32'h00001200, 32'h00001240, 32'h00001280, 32'h000012c0, 32'h00001400, 32'h00001800, 32'h00002000)),
167 MATCH_MASK ((32'hffffffc0, 32'hffffffc0, 32'hffffffc0, 32'hffffffc0, 32'hffffffc0, 32'hffffffc0, 32'hffffffc0, 32'hffffffc0, 32'hffffffc0, 32'hffffffc0, 32'hffffffc0)),
168 wb_mux_io
169 (.wb_clk_i (wb_clk_i),
170 .wb_rst_i (wb_rst_i),
171 .wbm_adr_i (wb_io_adr_i),
172 .wbm_dat_i (wb_io_dat_i),
173 .wbm_sel_i (wb_io_sel_i),
174 .wbm_we_i (wb_io_we_i),
175 .wbm_cyc_i (wb_io_cyc_i),
176 .wbm_stb_i (wb_io_stb_i),
177 .wbm_cti_i (wb_io_cti_i),
178 .wbm_bte_i (wb_io_bte_i),
179 .wbm_dat_o (wb_io_dat_o),
180 .wbm_ack_o (wb_io_ack_o),
181 .wbm_err_o (wb_io_err_o),
182 .wbm_rty_o (wb_io_rty_o),
183 .wbs_adr_o (wb_rom_adr_o, wb_sys_adr_o, wb_spi_flash_adr_o, wb_spi_accel_adr_o, wb_ptc_adr_o, wb_ptc2_adr_o, wb_ptc3_adr_o, wb_ptc4_adr_o, wb_gpio_adr_o, wb_gpio2_adr_o, wb_uart_adr_o),
184 .wbs_dat_o (wb_rom_dat_o, wb_sys_dat_o, wb_spi_flash_dat_o, wb_spi_accel_dat_o, wb_ptc_dat_o, wb_ptc2_dat_o, wb_ptc3_dat_o, wb_ptc4_dat_o, wb_gpio_dat_o, wb_gpio2_dat_o, wb_uart_dat_o),
185 .wbs_sel_o (wb_rom_sel_o, wb_sys_sel_o, wb_spi_flash_sel_o, wb_spi_accel_sel_o, wb_ptc_sel_o, wb_ptc2_sel_o, wb_ptc3_sel_o, wb_ptc4_sel_o, wb_gpio_sel_o, wb_gpio2_sel_o, wb_uart_sel_o),
186 .wbs_we_o (wb_rom_we_o, wb_sys_we_o, wb_spi_flash_we_o, wb_spi_accel_we_o, wb_ptc_we_o, wb_ptc2_we_o, wb_ptc3_we_o, wb_ptc4_we_o, wb_gpio_we_o, wb_gpio2_we_o, wb_uart_we_o),
187 .wbs_cyc_o (wb_rom_cyc_o, wb_sys_cyc_o, wb_spi_flash_cyc_o, wb_spi_accel_cyc_o, wb_ptc_cyc_o, wb_ptc2_cyc_o, wb_ptc3_cyc_o, wb_ptc4_cyc_o, wb_gpio_cyc_o, wb_gpio2_cyc_o, wb_uart_cyc_o),
188 .wbs_stb_o (wb_rom_stb_o, wb_sys_stb_o, wb_spi_flash_stb_o, wb_spi_accel_stb_o, wb_ptc_stb_o, wb_ptc2_stb_o, wb_ptc3_stb_o, wb_ptc4_stb_o, wb_gpio_stb_o, wb_gpio2_stb_o, wb_uart_stb_o),
189 .wbs_cti_o (wb_rom_cti_o, wb_sys_cti_o, wb_spi_flash_cti_o, wb_spi_accel_cti_o, wb_ptc_cti_o, wb_ptc2_cti_o, wb_ptc3_cti_o, wb_ptc4_cti_o, wb_gpio_cti_o, wb_gpio2_cti_o, wb_uart_cti_o),
190 .wbs_bte_o (wb_rom_bte_o, wb_sys_bte_o, wb_spi_flash_bte_o, wb_spi_accel_bte_o, wb_ptc_bte_o, wb_ptc2_bte_o, wb_ptc3_bte_o, wb_ptc4_bte_o, wb_gpio_bte_o, wb_gpio2_bte_o, wb_uart_bte_o),
191 .wbs_dat_i (wb_rom_dat_i, wb_sys_dat_i, wb_spi_flash_dat_i, wb_spi_accel_dat_i, wb_ptc_dat_i, wb_ptc2_dat_i, wb_ptc3_dat_i, wb_ptc4_dat_i, wb_gpio_dat_i, wb_gpio2_dat_i, wb_uart_dat_i),
192 .wbs_ack_i (wb_rom_ack_i, wb_sys_ack_i, wb_spi_flash_ack_i, wb_spi_accel_ack_i, wb_ptc_ack_i, wb_ptc2_ack_i, wb_ptc3_ack_i, wb_ptc4_ack_i, wb_gpio_ack_i, wb_gpio2_ack_i, wb_uart_ack_i),
193 .wbs_err_i (wb_rom_err_i, wb_sys_err_i, wb_spi_flash_err_i, wb_spi_accel_err_i, wb_ptc_err_i, wb_ptc2_err_i, wb_ptc3_err_i, wb_ptc4_err_i, wb_gpio_err_i, wb_gpio2_err_i, wb_uart_err_i),
194 .wbs_rty_i (wb_rom_rty_i, wb_sys_rty_i, wb_spi_flash_rty_i, wb_spi_accel_rty_i, wb_ptc_rty_i, wb_ptc2_rty_i, wb_ptc3_rty_i, wb_ptc4_rty_i, wb_gpio_rty_i, wb_gpio2_rty_i, wb_uart_rty_i));
195
196 endmodule

```

Lab 9 – Exercício 2. Expanda o RVfpgaNexys de modo a incluir uma segunda fonte de interrupção através da IRQ4 que vem do segundo GPIO que incluiu no Laboratório 6 para controlar os botões de pressão na placa.

- swervolf_core.v

```
234      .gpio2_irq      (gpio2_irq),
```

- swervolf_syscon.v

```
32      input wire      gpio2_irq,
```

```
143      // GPIO Interrupt through IRQ4. Enable by setting bit 0 of word 0x80001018
144      if (irq_gpio_enable & (gpio_irq | gpio2_irq)) begin
145          sw_irq4 <= 1'b1;
146      end
```