

GRUPO 5 - TURMA A

PROJETO FINAL
PACMAN

MC 613 - Primeiro Semestre de 2010

PROFESSOR: GUIDO ARAÚJO

HENRIQUE SERAPIÃO GONZALES	RA: 083636
MARCELO GALVÃO PÓVOA	RA: 082115
TIAGO CHEDRAOUI SILVA	RA: 082941

18 de junho de 2010

1 Código

1.1 Top-level

Listing 1: Display

```
1 LIBRARY ieee;
2 USE ieee.STD_LOGIC_1164.all;
3 USE ieee.NUMERIC_STD.all;
4 USE work.PAC_DEFS.all;
5 USE work.PAC_SPRITES.all;
6
7 ENTITY pacman is
8   PORT (
9     clk27M, reset_button      : in  STD_LOGIC;
10    red, green, blue           : out STD_LOGIC_vector(3 downto 0);
11    hsync, vsync               : out STD_LOGIC;
12    LEDG                       : BUFFER STD_LOGIC_VECTOR (7 downto 5);
13    PS2_DAT                    : inout STD_LOGIC;
14    PS2_CLK                    : inout STD_LOGIC;
15    SEG0, SEG1, SEG2, SEG3     : OUT  STD_LOGIC_VECTOR(6 downto 0);
16    LEDR                       : BUFFER STD_LOGIC_VECTOR (2 downto 0);
17    endgame                   : OUT  STD_LOGIC
18  );
19 END pacman;
20
21 ARCHITECTURE comportamento of pacman is
22   SIGNAL rstn: STD_LOGIC;           -- reset active low
23   SIGNAL restartn: STD_LOGIC;       -- Usado quando o pacman morre (active low)
24   SIGNAL load_cenario: STD_LOGIC;   -- Informa quando o cenário está sendo recarregado
25
26   -- Interface com a memória de vídeo do controlador
27   SIGNAL we : STD_LOGIC;             -- write enable ('1' p/ escrita)
28   SIGNAL addr : INTEGER
29     range 0 to SCR_HGT*SCR_WDT-1;    -- ENDereco mem. vga
30   SIGNAL block_in, block_out : t_blk_sym; -- dados trocados com a mem. vga
31   SIGNAL vga_pixel_out: t_color_3b;
32
33   -- Sinais dos contadores de linhas e colunas utilizados para percorrer
34   -- as posições da memória de vídeo (pixels) no momento de construir um quadro.
35   SIGNAL line : INTEGER range 0 to SCR_HGT-1; -- linha atual
36   SIGNAL col : INTEGER range 0 to SCR_WDT-1;  -- coluna atual
37   SIGNAL col_rstn : STD_LOGIC;               -- reset do contador de colunas
38   SIGNAL col_enable : STD_LOGIC;             -- enable do contador de colunas
39   SIGNAL line_rstn : STD_LOGIC;              -- reset do contador de linhas
40   SIGNAL line_enable, line_inc : STD_LOGIC;  -- enable do contador de linhas
41   SIGNAL fim_escrita : STD_LOGIC;            -- '1' quando um quadro terminou de ser
42                                           -- escrito na memória de vídeo
43
44   -- Especificação dos tipos e sinais da máquina de estados de controle
45   TYPE estado_t is (SHOW_SPLASH, CARREGA_MAPA, INICIO_JOGO, PERCORRE_QUADRO,
46                     ATUALIZA_LOGICA_1, ATUALIZA_LOGICA_2, ATUALIZA_LOGICA_3, MEMORIA_WR,
47                     REINICIO, FIM_JOGO, PACMAN_VENCE);
48   SIGNAL estado: estado_t := SHOW_SPLASH;
49   SIGNAL pr_estado: estado_t := SHOW_SPLASH;
50
51   -- sinais que servem como enable de várias velocidades
52   SIGNAL atua_en: STD_LOGIC_VECTOR(2 downto 0);
53   SIGNAL display_en: STD_LOGIC;
54   SIGNAL disp_count: INTEGER range 0 to 4194304;
55   SIGNAL sig_blink: UNSIGNED(6 downto 0); -- enables com duty de 50%
56
57   -- Sinais de desenho em overlay sobre o cenário do jogo
58   SIGNAL overlay: STD_LOGIC;
59   SIGNAL ovl_blk_in: t_ovl_blk_sym;
60
61   -- Sinais para um contador utilizado para atrasar
```

```

62     -- a frequência da atualização
63     SIGNAL contador, long_cont : INTEGER range 0 to DIV_FACT-1;
64     SIGNAL timer, long_timer : STD_LOGIC;      -- vale '1' quando o contador chegar ao fim
65     SIGNAL timer_rstn, timer_enable : STD_LOGIC;
66
67     COMPONENT counter IS
68     PORT (clk, rstn, en: IN STD_LOGIC;
69           max: IN INTEGER;
70           q: OUT INTEGER);
71     END COMPONENT counter;
72
73     -----
74     -- Sinais de controle da lógica do jogo
75     -----
76     SIGNAL got_coin, got_spc_coin: STD_LOGIC; -- informa se obteve moeda no ultimo movimento
77     SIGNAL reg_coin_we: STD_LOGIC;
78     SIGNAL q_rem_moedas: INTEGER range 0 to 255 := 240; -- quantidade de moedas normais para vencer
79     SIGNAL q_vidas: INTEGER range 0 to 5 := 3;
80     SIGNAL q_pontos: INTEGER range 0 to 9999 := 0;
81     SIGNAL vidas_arr: STD_LOGIC_VECTOR(2 downto 0);
82
83     -- Controle do pacman
84     SIGNAL pac_pos_x: t_pos := PAC_START_X;
85     SIGNAL pac_pos_y: t_pos := PAC_START_Y;
86     SIGNAL pac_cur_dir: t_direcao;
87     SIGNAL pac_area: t_blk_sym_3x3;
88     SIGNAL pacman_dead: STD_LOGIC;
89     SIGNAL pac_fans_hit: UNSIGNED(0 to FAN_NO-1);
90     SIGNAL pac_atua: STD_LOGIC;
91
92     -- Controle dos fantasmas
93     SIGNAL fan_pos_x: t_fans_pos;
94     SIGNAL fan_pos_y: t_fans_pos;
95     SIGNAL fan_cur_dir: t_fans_dirs;
96     SIGNAL fan_state: t_fans_states;
97     SIGNAL fan_area: t_fans_blk_sym_3x3;
98     SIGNAL fan_atua: STD_LOGIC;
99     SIGNAL fan_died: STD_LOGIC;
100
101     SIGNAL pac_key_dir: t_direcao; -- sinais lidos pelo teclado
102     SIGNAL fan_key_dir: t_fans_dirs;
103 BEGIN
104
105     -- Controlador VGA com duas camadas (RAMs) de blocos:
106     -- cenário e overlay, isto é, o pacman e os fantasmas
107     -- Devolve os pixels convertidos pelos sprites e os
108     -- sinais de controle do monitor
109     vga_controller: entity work.vgacon port map (
110         clk27M      => clk27M,
111         rstn        => '1',
112         vga_pixel   => vga_pixel_out,
113         data_block  => block_out,
114         hsync       => hsync,
115         vsync       => vsync,
116         write_clk   => clk27M,
117         write_enable => we,
118         write_addr  => addr,
119         data_in     => block_in,
120         ovl_in      => ovl_blk_in,
121         ovl_we      => overlay);
122
123     -- Atribuição capada das cores 3b -> 12b
124     red   <= (OTHERS => vga_pixel_out(0));
125     green <= (OTHERS => vga_pixel_out(1));
126     blue  <= (OTHERS => vga_pixel_out(2));
127
128     -- Controlador do teclado. Devolve os sinais síncronos das teclas
129     -- de interesse pressionadas ou não.
130     kbd: ENTITY WORK.kbd_key PORT MAP (

```

```

131     CLOCK_27  => clk27M,
132     KEY       => reset_button,
133     LEDG      => LEDG(7 downto 5),
134     PS2_DAT   => PS2_DAT,
135     PS2_CLK   => PS2_CLK,
136     p1_dir    => pac_key_dir,
137     p2_dir    => fan_key_dir(0),
138     p3_dir    => fan_key_dir(1)
139 );
140
141 disp_counter: COMPONENT counter
142     PORT MAP (clk => clk27M,
143               rstn => '1',
144               en  => '1',
145               max => DISP_DIV_FACT-1,
146               q  => disp_count);
147
148 display_en <= '1' WHEN (disp_count = DISP_DIV_FACT-1)
149 ELSE '0';
150
151 -- Módulo que controla os displays 7-seg imprimindo
152 -- mensagens e a pontuação atual
153 display: ENTITY WORK.disp PORT MAP (
154     CLK      => clk27M,
155     EN       => display_en,
156     VIDAS    => q_vidas,
157     PNT      => q_pontos,
158     PEDRAS   => q_rem_moedas,
159     seg0     => SEG0,
160     seg1     => SEG1,
161     seg2     => SEG2,
162     seg3     => SEG3
163 );
164
165 -- Contadores de varredura da tela
166 conta_coluna: COMPONENT counter
167     PORT MAP (clk => clk27M,
168               rstn => col_rstn,
169               en  => col_enable,
170               max => SCR_WDT-1,
171               q  => col);
172
173 -- o contador de linha só incrementa quando o contador de colunas
174 -- chegou ao fim
175 line_inc <= '1' WHEN (line_enable='1' and col = SCR_WDT-1)
176 ELSE '0';
177
178 conta_linha: COMPONENT counter
179     PORT MAP (clk => clk27M,
180               rstn => line_rstn,
181               en  => line_inc,
182               max => SCR_HGT-1,
183               q  => line);
184
185 -- podemos avançar para o próximo estado?
186 fim_escrita <= '1' WHEN (line = SCR_HGT-1) and (col = SCR_WDT-1)
187 ELSE '0';
188
189 -- Controlador dos fantasmas
190 ctrl_fans_inst: ENTITY work.ctrl_fans PORT MAP (
191     clk27M      => clk27M,      rstn => rstn and restartn,
192     atualiza    => fan_atua,     atua_en => atua_en,
193     keys_dir   => fan_key_dir,   fan_died => fan_died,
194     fan_area   => fan_area,      pacman_dead => pacman_dead,
195     spc_coin   => got_spc_coin,   pac_fans_hit=> pac_fans_hit,
196     fan_pos_x  => fan_pos_x,     fan_pos_y  => fan_pos_y,
197     fan_state  => fan_state,     fan_cur_dir => fan_cur_dir
198 );
199

```

```

200 -- Controlador do pacman
201 ctrl_pac_inst: ENTITY work.ctrl_pacman PORT MAP (
202   clk27M => clk27M,      rstn => rstn and restartn,
203   key_dir => pac_key_dir,  atualiza => pac_atua and atua_en(0),
204   pac_area => pac_area,    pac_cur_dir => pac_cur_dir,
205   pac_pos_x => pac_pos_x,  pac_pos_y => pac_pos_y,
206   got_coin => got_coin,    got_spc_coin=> got_spc_coin
207 );
208
209 -- Preenche as matrizes 3x3 das vizinhanças pac_area
210 -- e fans_area durante PERCORRE_QUADRO
211 -- type : sequential
212 p_fill_memarea: PROCESS (clk27M)
213 VARIABLE x_offset, y_offset: t_offset;
214 BEGIN
215   IF (clk27M'event and clk27M='1') THEN
216     IF (estado = PERCORRE_QUADRO) THEN
217       --Leitura atrasada devido ao ciclo de clock da ram
218       y_offset := line - pac_pos_y;
219       x_offset := col - pac_pos_x;
220       IF (x_offset >=0 and x_offset <=2 and y_offset >=-1 and y_offset<=1) THEN
221         pac_area(y_offset, x_offset-1) <= block_out;
222       END IF;
223
224       FOR i in 0 to FAN_NO-1 LOOP
225         y_offset := line - fan_pos_y(i);
226         x_offset := col - fan_pos_x(i);
227         IF (x_offset >=0 and x_offset <=2 and y_offset >=-1 and y_offset<=1) THEN
228           fan_area(i)(y_offset, x_offset-1) <= block_out;
229         END IF;
230       END LOOP;
231     END IF;
232   END IF;
233 END PROCESS;
234
235 -- Atualiza parâmetros de informação atual do jogo
236 -- type: sequential
237 param_jogo: PROCESS (clk27M, rstn)
238 BEGIN
239   IF (rstn = '0') THEN
240     q_vidas <= 3;
241     q_pontos <= 0;
242     q_rem_moedas <= 240;
243   ELSIF (clk27M'event and clk27M = '1') THEN
244     IF (pacman_dead = '1' and fan_atua = '1') THEN--estado = fan_atua) THEN
245       q_vidas <= q_vidas - 1;
246     END IF;
247
248     IF (fan_died = '1') THEN
249       q_pontos <= q_pontos + 200;
250   ELSIF (pac_atua = '1' and atua_en(0) = '1') THEN
251     IF (got_coin = '1') THEN
252       q_pontos <= q_pontos + 10;
253       q_rem_moedas <= q_rem_moedas - 1;
254     ELSIF (got_spc_coin = '1') THEN
255       q_pontos <= q_pontos + 50;
256     END IF;
257
258     IF (got_coin = '1' or got_spc_coin = '1') THEN
259       reg_coin_we <= '1'; --registra uma moeda comida
260     ELSE
261       reg_coin_we <= '0';
262     END IF;
263   END IF;
264 END IF;
265 END PROCESS param_jogo;
266
267 -- purpose: Processo para que gera todos os sinais de desenho de overlay
268 -- (ie, sobre o fundo) da vidas, do pacman e dos fantasmas de acordo

```

```

269 --          com a varredura de line e col durante PERCORRE_QUADRO
270 -- type      : combinational
271 des_overlay: PROCESS (pac_pos_x, pac_pos_y, pac_cur_dir, sig_blink, vidas_arr,
272                      fan_pos_x, fan_pos_y, fan_state, fan_cur_dir, line, col)
273 VARIABLE x_offset, y_offset: t_offset;
274 VARIABLE ovl_blk_tmp: t_ovl_blk_sym;
275 BEGIN
276 ovl_blk_tmp := BLK_NULL; -- este será o bloco que vai pra VGA
277
278 FOR i in 0 to FAN_NO-1 LOOP -- Desenho dos fantasmas
279     y_offset := line - fan_pos_y(i) + 2;
280     x_offset := col - fan_pos_x(i) + 2;
281     IF (x_offset>=0 and x_offset<5 and
282         y_offset>=0 and y_offset<5) THEN
283         IF (fan_state(i) = ST_VULN_BLINK) THEN
284             IF (sig_blink(4) = '0') THEN --pisca no final do modo vulnerável
285                 ovl_blk_tmp := FAN_VULN_BITMAP(y_offset, x_offset);
286             ELSE
287                 ovl_blk_tmp := BLK_NULL;
288             END IF;
289         ELSIF (fan_state(i) = ST_VULN) THEN
290             ovl_blk_tmp := FAN_VULN_BITMAP(y_offset, x_offset);
291         ELSIF (fan_state(i) = ST_DEAD) THEN
292             ovl_blk_tmp := FAN_DEAD_BITMAPS(fan_cur_dir(i))(y_offset, x_offset);
293         ELSE
294             ovl_blk_tmp := FAN_BITMAPS(i)(fan_cur_dir(i))(y_offset, x_offset);
295         END IF;
296     y_offset := line - fan_pos_y(1) + 2;
297     END IF;
298 END LOOP;
299
300 -- Desenho do pacman
301 y_offset := line - pac_pos_y + 2;
302 x_offset := col - pac_pos_x + 2;
303 IF (x_offset>=0 and x_offset<5 and
304     y_offset>=0 and y_offset<5) THEN
305     IF (sig_blink(4) = '0') THEN
306         ovl_blk_tmp := PAC_BITMAPS(pac_cur_dir)(y_offset, x_offset);
307     ELSE
308         IF (pac_cur_dir = DIREI or pac_cur_dir = ESQUE) THEN
309             ovl_blk_tmp := PAC_FECH_BITMAP(y_offset, x_offset);
310         ELSE
311             ovl_blk_tmp := PAC_FECV_BITMAP(y_offset, x_offset);
312         END IF;
313     END IF;
314 END IF;
315
316 FOR i in 0 to 2 LOOP --Desenho dos ícones de vida
317     IF (vidas_arr(i) = '1') THEN
318         y_offset := line - VIDA_ICONS_Y(i) + 2;
319         x_offset := col - VIDA_ICONS_X(i) + 2;
320         IF (x_offset>=0 and x_offset<5 and
321             y_offset>=0 and y_offset<5) THEN
322             ovl_blk_tmp := PAC_BITMAPS(DIREI)(y_offset, x_offset);
323         END IF;
324     END IF;
325 END LOOP;
326
327 ovl_blk_in <= ovl_blk_tmp;
328 END PROCESS;
329
330 -- Determina quando o pacman colidiu com cada um dos fantasmas
331 -- type: combinational
332 PROCESS (pac_pos_x, pac_pos_y, fan_pos_x, fan_pos_y)
333 VARIABLE off_x, off_y: t_offset;
334 BEGIN
335 FOR i in 0 to FAN_NO-1 LOOP
336     off_x := pac_pos_x - fan_pos_x(i);
337     off_y := pac_pos_y - fan_pos_y(i);

```

```

338     -- a tolerância para colisão é uma região 3x3
339     IF (off_x >=-1 and off_x <=1 and off_y>=-1 and off_y<=1) THEN
340         pac_fans_hit(i) <= '1';
341     ELSE
342         pac_fans_hit(i) <= '0';
343     END IF;
344 END LOOP;
345 END PROCESS;
346
347 -- Define dado que entra na ram de cenário
348 def_block_in: PROCESS (load_cenario, addr)
349 BEGIN
350     IF (load_cenario = '1') THEN
351         block_in <= CONV_TAB_BLK(MAPA_INICIAL(addr));
352     ELSE
353         block_in <= BLK_PATH; --Caso que a moeda é comida pelo pacman
354     END IF;
355 END PROCESS;
356
357 led_vidas: PROCESS (q_vidas)
358 BEGIN
359     IF (q_vidas = 3) THEN
360         vidas_arr <= "111";
361     ELSIF (q_vidas = 2) THEN
362         vidas_arr <= "011";
363     ELSIF (q_vidas = 1) THEN
364         vidas_arr <= "001";
365     ELSE
366         vidas_arr <= "000";
367     END IF;
368 END PROCESS led_vidas;
369
370 LEDR <= vidas_arr;
371
372 -----
373 -- Processos que definem a FSM principal. Alguns sinais de controle são definidos
374 -- apenas para um estado e portanto estão localizados no process seguinte
375 -- type : combinational
376 logica_mealy: PROCESS (estado, fim_escrita, timer, long_timer, q_rem_moedas, q_vidas,
377                        col, line, pac_pos_x, pac_pos_y, pacman_dead, reg_coin_we, fan_died)
378 BEGIN
379     case estado is
380     when CARREGA_MAPA => IF (fim_escrita = '1') THEN
381         pr_estado <= INICIO_JOGO;
382     ELSE
383         pr_estado <= CARREGA_MAPA;
384     END IF;
385     line_rstn <= '1';
386     line_enable <= '1';
387     col_rstn <= '1';
388     col_enable <= '1';
389     we <= '1';
390     timer_rstn <= '0';
391     timer_enable <= '0';
392     addr <= col + SCR_WDT*line;
393
394     when REINICIO => IF (long_timer = '1') THEN
395         pr_estado <= INICIO_JOGO;
396     ELSE
397         pr_estado <= REINICIO;
398     END IF;
399     line_rstn <= '1';
400     line_enable <= '1';
401     col_rstn <= '1';
402     col_enable <= '1';
403     we <= '0';
404     timer_rstn <= '1';
405     timer_enable <= '1';
406     addr <= 0;

```

```

407
408 when FIM_JOGO => pr_estado <= FIM_JOGO; --não sai disso
409         line_rstn    <= '1';
410         line_enable   <= '1';
411         col_rstn     <= '1';
412         col_enable    <= '1';
413         we            <= '0';
414         timer_rstn    <= '0';
415         timer_enable  <= '0';
416         addr         <= 0;
417
418 when PACMAN_VENCE => pr_estado <= PACMAN_VENCE; --não sai disso
419         line_rstn    <= '1';
420         line_enable   <= '1';
421         col_rstn     <= '1';
422         col_enable    <= '1';
423         we            <= '0';
424         timer_rstn    <= '0'; -- reset é active low!
425         timer_enable  <= '0';
426         addr         <= 0;
427
428 when INICIO_JOGO => IF (timer = '1') THEN
429         pr_estado <= PERCORRE_QUADRO;
430     ELSE
431         pr_estado <= INICIO_JOGO;
432     END IF;
433     line_rstn    <= '0'; -- reset é active low!
434     line_enable  <= '0';
435     col_rstn     <= '0'; -- reset é active low!
436     col_enable   <= '0';
437     we           <= '0';
438     timer_rstn   <= '1'; -- reset é active low!
439     timer_enable <= '1';
440     addr         <= 0;
441
442 when PERCORRE_QUADRO => IF (fim_escrita = '1') THEN
443     pr_estado <= ATUALIZA_LOGICA_1;
444 ELSE
445     pr_estado <= PERCORRE_QUADRO;
446 END IF;
447 line_rstn    <= '1';
448 line_enable  <= '1';
449 col_rstn     <= '1';
450 col_enable   <= '1';
451 we           <= '0';
452 timer_rstn   <= '0';
453 timer_enable <= '0';
454 addr         <= col + SCR_WDT*line;
455
456 when ATUALIZA_LOGICA_1 => IF (pacman_dead = '1') THEN
457     IF (q_vidas = 0) THEN
458         pr_estado <= FIM_JOGO;
459     ELSE
460         pr_estado <= REINICIO;
461     END IF;
462 ELSIF (q_rem_moedas <= 0) THEN
463     pr_estado <= PACMAN_VENCE;
464 ELSE
465     pr_estado <= ATUALIZA_LOGICA_2;
466 END IF;
467 line_rstn    <= '1';
468 line_enable  <= '0';
469 col_rstn     <= '1';
470 col_enable   <= '0';
471 we           <= '0';
472 timer_rstn   <= '0';
473 timer_enable <= '0';
474 addr         <= 0;
475

```



```

476 when ATUALIZA_LOGICA_2 => pr_estado <= ATUALIZA_LOGICA_3;
477     line_rstn      <= '1';
478         line_enable <= '0';
479         col_rstn    <= '1';
480         col_enable  <= '0';
481         we          <= '0';
482         timer_rstn  <= '0';
483         timer_enable <= '0';
484         addr        <= 0;
485
486 when ATUALIZA_LOGICA_3 => pr_estado <= MEMORIA_WR;
487     line_rstn      <= '1';
488         line_enable <= '0';
489         col_rstn    <= '1';
490         col_enable  <= '0';
491         we          <= '0';
492         timer_rstn  <= '0';
493         timer_enable <= '0';
494         addr        <= 0;
495
496     when MEMORIA_WR => pr_estado <= INICIO_JOGO;
497         line_rstn      <= '0';
498             line_enable <= '0';
499             col_rstn    <= '0';
500             col_enable  <= '0';
501             we          <= reg_coin_we;
502             timer_rstn  <= '0';
503             timer_enable <= '0';
504             addr        <= pac_pos_x + SCR_WDT * pac_pos_y;
505
506 when others => pr_estado <= CARREGA_MAPA;
507     line_rstn      <= '0';
508     line_enable    <= '0';
509     col_rstn       <= '0';
510     col_enable     <= '0';
511     we             <= '0';
512     timer_rstn     <= '1';
513     timer_enable   <= '0';
514     addr           <= 0;
515
516     END case;
517 END PROCESS logica_mealy;
518
519 -- Define sinais de controle da FSM usados em apenas UM ESTADO
520 -- type: combinational
521 sinais_extras: PROCESS (estado, atua_en)
522 BEGIN
523     IF (estado = PERCORRE_QUADRO)
524     THEN overlay <= '1';
525     ELSE overlay <= '0';
526     END IF;
527
528     IF (estado = CARREGA_MAPA)
529     THEN load_cenario <= '1';
530     ELSE load_cenario <= '0';
531     END IF;
532
533     IF (estado = REINICIO)
534     THEN restartn <= '0';
535     ELSE restartn <= '1';
536     END IF;
537
538     IF (estado = ATUALIZA_LOGICA_2)
539     THEN fan_atua <= '1';
540     ELSE fan_atua <= '0';
541     END IF;
542
543     IF (estado = ATUALIZA_LOGICA_1)
544     THEN pac_atua <= '1';
545     ELSE pac_atua <= '0';

```

```

545     END IF;
546
547     IF (estado = FIM_JOGO)
548     THEN endgame <= '1';
549     ELSE endgame <= '0';
550     END IF;
551 END PROCESS;
552
553 -- Avança a FSM para o próximo estado
554 -- type : sequential
555 seq_fsm: PROCESS (clk27M, rstn)
556 BEGIN
557     IF (rstn = '0') THEN
558         estado <= SHOW_SPLASH;
559     elsif (clk27M'event and clk27M = '1') THEN
560         estado <= pr_estado;
561     END IF;
562 END PROCESS seq_fsm;
563
564 -- Atualiza contadores de número de atualizações
565 -- Gera enables de atualizações para cada velocidade de atualização
566 -- type: sequential
567 atual_counters: PROCESS (clk27M, rstn)
568 VARIABLE atual_cont: t_vet_velocs;
569 BEGIN
570     IF (rstn = '0') THEN
571         atual_cont := (OTHERS => 0);
572         sig_blink <= (OTHERS => '0');
573     ELSIF (clk27M'event and clk27M = '1') THEN
574         IF (estado = ATUALIZA_LOGICA_2) THEN
575             FOR i IN 0 to 2 LOOP
576                 IF (atual_cont(i) = VEL_DIV(i)-1) THEN
577                     atual_cont(i) := 0;
578                     atua_en(i) <= '1';
579                 ELSE
580                     atual_cont(i) := atual_cont(i) + 1;
581                     atua_en(i) <= '0';
582                 END IF;
583             END LOOP;
584             sig_blink <= sig_blink + 1;
585         END IF;
586     END IF;
587 END PROCESS;
588
589 -- Contadores utilizados para atrasar a animação (evitar
590 -- que a atualização de quadros fique muito veloz).
591 p_contador0: COMPONENT counter
592 PORT MAP (clk => clk27M,
593           rstn => timer_rstn,
594           en => timer_enable,
595           max => DIV_FACT - 1,
596           q => contador);
597
598 p_contador1: COMPONENT counter
599 PORT MAP (clk => clk27M,
600           rstn => timer_rstn, --mesmo reset do contador 0, porém
601           en => timer, --contagem a cada término do contador 0
602           max => 127,
603           q => long_cont);
604
605 --O sinal "timer" indica a hora de fazer nova atualização
606 timer <= '1' WHEN (contador = DIV_FACT - 1)
607 ELSE '0';
608
609 --Timer para mostrar um evento na tela
610 long_timer <= '1' WHEN (long_cont = 127)
611 ELSE '0';
612
613 -- Processos que sincronizam o reset assíncrono, de preferência com mais

```

```

614     -- de 1 flipflop, para evitar metaestabilidade.
615     -- type      : sequential
616     build_rstn: PROCESS (clk27M)
617         VARIABLE temp : STD_LOGIC;          -- flipflop intermediario
618     BEGIN
619         IF (clk27M'event and clk27M = '1') THEN
620             rstn <= temp;
621             temp := reset_button;
622         END IF;
623     END PROCESS build_rstn;
624 END comportamento;

```

1.2 Definições

Listing 2: Definições para pacman

```

1  LIBRARY ieee;
2  USE ieee.STD_LOGIC_1164.all;
3  USE ieee.NUMERIC_STD.all;
4  --USE work.PAC_SPRITES.all;
5
6  PACKAGE pac_defs IS
7      -----
8      -- Definições de dados, constantes e tipos para o jogo
9      -----
10
11     --Resolução de blocos usada (hgt linhas por wdt colunas)
12     CONSTANT SCR_HGT : INTEGER := 96;
13     CONSTANT SCR_WDT : INTEGER := 128;
14
15     --Maior dimensao do tabuleiro (em blocos)
16     CONSTANT TAB_LEN: INTEGER := 91;
17
18     SUBTYPE t_color_3b is std_logic_vector(2 downto 0);
19
20     TYPE t_direcao is (CIMA,DIREI,BAIXO,ESQUE,NADA);
21
22     --A legenda pros elementos no tabuleiro é dada por t_tab_sym
23     --Cuidado para manter a mesma sequencia observada em t_blk_sym (FIXME)
24     --Os números representam elementos visuais na tela e o resto
25     --representa posições especiais
26     --' ': vazio, '.' : caminho, 6 tipos de parede de acordo com a orientação,
27     --C: moeda, P: moeda especial, D: porta
28     TYPE t_tab_sym is (' ', '.', '|', '-', 'Q', 'W', 'E', 'R', 'C', 'P', 'D');
29
30     SUBTYPE t_blk_id is STD_LOGIC_VECTOR(3 downto 0);
31     SUBTYPE t_ovl_blk_id is STD_LOGIC_VECTOR(8 downto 0);
32
33     TYPE t_blk_sym is (BLK_NULL, BLK_PATH, BLK_WALL_V, BLK_WALL_H, BLK_WALL_Q, BLK_WALL_W, BLK_WALL_E,
34         BLK_WALL_R, BLK_COIN, BLK_SPC_COIN, BLK_DOOR);
35
36     TYPE t_blk_bool is array(t_blk_sym) of boolean;
37     CONSTANT WALKABLE: t_blk_bool := --define quais blocos são percorríveis
38         (BLK_PATH => true, BLK_COIN => true, BLK_SPC_COIN => true, OTHERS => false);
39
40     TYPE t_ovl_blk_sym is (Ver figura XXXXXX);
41     CONSTANT FAN_NO: INTEGER := 2; --Número de fantasmas no jogo
42
43     TYPE c_tab_blk is array(t_tab_sym) of t_blk_sym;
44     CONSTANT CONV_TAB_BLK: c_tab_blk :=
45         (' ' => BLK_NULL, '.' => BLK_PATH, '|' => BLK_WALL_V, '-' => BLK_WALL_H, 'Q' => BLK_WALL_Q, 'W' =>
46             BLK_WALL_W,
47             'E' => BLK_WALL_E, 'R' => BLK_WALL_R, 'C' => BLK_COIN, 'P' => BLK_SPC_COIN, 'D' => BLK_DOOR);
48
49     TYPE t_tab is array(0 to SCR_HGT-1, 0 to SCR_WDT-1) of t_tab_sym;
50     TYPE t_blk_sym_3x3 is array(-1 to 1, -1 to 1) of t_blk_sym;
51

```

```

51 TYPE t_sprite5 is array(0 to 4, 0 to 4) of STD_LOGIC;
52 TYPE t_ovl_blk_5x5 is array(0 to 4, 0 to 4) of t_ovl_blk_sym;
53 TYPE t_ovl_blk_dir_vet is array(t_direcao) of t_ovl_blk_5x5;
54 TYPE t_fans_ovl_blk_dir_vet is array(0 to FAN_NO-1) of t_ovl_blk_dir_vet;
55
56 TYPE t_sprite5_vet is array(t_blk_sym) of t_sprite5;
57 TYPE t_ovl_sprite5_vet is array(t_ovl_blk_sym) of t_sprite5;
58
59 --Tipos em array para os fantasmas
60 SUBTYPE t_pos is INTEGER range 0 to TAB_LEN-1;
61 SUBTYPE t_offset is INTEGER range -TAB_LEN to TAB_LEN;
62 SUBTYPE t_fan_time is INTEGER range 0 to 1000;
63 TYPE t_fan_state is (ST_VIVO, ST_VULN, ST_VULN_BLINK, ST_DEAD, ST_PRE_DEAD, ST_FIND_EXIT, ST_FUGA);
64
65 TYPE t_fans_pos is array(0 to FAN_NO-1) of t_pos;
66 TYPE t_fans_dirs is array(0 to FAN_NO-1) of t_direcao;
67 TYPE t_fans_blk_sym is array(0 to FAN_NO-1) of t_blk_sym;
68 TYPE t_fans_blk_sym_3x3 is array(0 to FAN_NO-1) of t_blk_sym_3x3;
69 TYPE t_fans_states is array(0 to FAN_NO-1) of t_fan_state;
70 TYPE t_fans_times is array(0 to FAN_NO-1) of t_fan_time;
71 SUBTYPE t_fans_bits is STD_LOGIC_VECTOR(0 to FAN_NO-1);
72
73 TYPE t_vidas_pos is array(0 to 2) of t_pos;
74
75 SUBTYPE t_velocs is INTEGER range 0 to 20;
76 TYPE t_vet_velocs is array(0 to 2) of t_velocs;
77
78 --Fator de divisão do clock de 27MHz, usada para atualização do
79 --estado do jogo ("velocidade de execução")
80 CONSTANT DIV_FACT: INTEGER := 202500;
81 CONSTANT DISP_DIV_FACT: INTEGER := 20*DIV_FACT;
82
83 subtype sentido is INTEGER range -1 to 1;
84 TYPE t_direc is array(0 to 1) of sentido;
85 TYPE t_direc_vet is array(t_direcao) of t_direc;
86
87 CONSTANT DIRS: t_direc_vet := (CIMA => (-1, 0), DIREI => ( 0, 1),
88                               BAIXO => ( 1, 0), ESQUE => ( 0,-1),
89                               NADA  => ( 0, 0));
90
91 CONSTANT PAC_START_X : INTEGER := 42;
92 CONSTANT PAC_START_Y : INTEGER := 71;
93 CONSTANT FANS_START_X : t_fans_pos := (40, 45);
94 CONSTANT FANS_START_Y : t_fans_pos := (44, 44);
95 CONSTANT FAN_TIME_VULN_START_BLINK : INTEGER := 600;
96 CONSTANT FAN_TIME_VULN_END : INTEGER := 750;
97 CONSTANT FAN_TIME_DEAD : INTEGER := 700;
98 CONSTANT CELL_IN_X : INTEGER := 42;
99 CONSTANT CELL_IN_Y : INTEGER := 44;
100 CONSTANT CELL_OUT_Y : INTEGER := 35;
101 CONSTANT TELE_DIR_POS : INTEGER := 82;
102 CONSTANT TELE_ESQ_POS : INTEGER := 2;
103 CONSTANT VIDA_ICONS_X: t_vidas_pos := (90, 90, 90);
104 CONSTANT VIDA_ICONS_Y: t_vidas_pos := (89, 83, 77);
105 --velocidades de atualização para: 0=pacman, 1=fantasma, 2=fantasma morto
106 CONSTANT VEL_DIV: t_vet_velocs := (6, 5, 4);
107
108 TYPE t_tab_array is array(0 to SCR_WDT*SCR_HGT-1) of t_tab_sym;
109 TYPE t_tab_mapa is array(0 to SCR_HGT-1, 0 to SCR_WDT-1) of t_tab_sym;
110 --Mapa de inicialização da RAM inferior, a legenda está acima
111 CONSTANT MAPA_INICIAL: t_tab_array := VER ----
112
113 --Neste mapa, estão armazenados apenas a próxima direção do percurso de um fantasma
114 --quando este é comido. A legenda é Q: CIMA, W: BAIXO, E: ESQUERDA, R: DIREITA
115 CONSTANT FAN_PERCURSO: t_tab_mapa := ver -----
116
117
118 END pac_defs;

```

Listing 3: Definicoes para pacman

```

1  TYPE t_ovl_blk_sym is (BLK_NULL,
2      BLK_PAC_CTM_00, BLK_PAC_CTM_01, BLK_PAC_CTM_02, BLK_PAC_CTM_03, BLK_PAC_CTM_04,
3      BLK_PAC_CTM_10, BLK_PAC_CTM_11, BLK_PAC_CTM_12, BLK_PAC_CTM_13, BLK_PAC_CTM_14,
4      BLK_PAC_CTM_20, BLK_PAC_CTM_21, BLK_PAC_CTM_22, BLK_PAC_CTM_23, BLK_PAC_CTM_24,
5      BLK_PAC_CTM_30, BLK_PAC_CTM_31, BLK_PAC_CTM_32, BLK_PAC_CTM_33, BLK_PAC_CTM_34,
6      BLK_PAC_CTM_40, BLK_PAC_CTM_41, BLK_PAC_CTM_42, BLK_PAC_CTM_43, BLK_PAC_CTM_44,
7      BLK_PAC_DIR_00, BLK_PAC_DIR_01, BLK_PAC_DIR_02, BLK_PAC_DIR_03, BLK_PAC_DIR_04,
8      BLK_PAC_DIR_10, BLK_PAC_DIR_11, BLK_PAC_DIR_12, BLK_PAC_DIR_13, BLK_PAC_DIR_14,
9      BLK_PAC_DIR_20, BLK_PAC_DIR_21, BLK_PAC_DIR_22, BLK_PAC_DIR_23, BLK_PAC_DIR_24,
10     BLK_PAC_DIR_30, BLK_PAC_DIR_31, BLK_PAC_DIR_32, BLK_PAC_DIR_33, BLK_PAC_DIR_34,
11     BLK_PAC_DIR_40, BLK_PAC_DIR_41, BLK_PAC_DIR_42, BLK_PAC_DIR_43, BLK_PAC_DIR_44,
12     BLK_PAC_BAI_00, BLK_PAC_BAI_01, BLK_PAC_BAI_02, BLK_PAC_BAI_03, BLK_PAC_BAI_04,
13     BLK_PAC_BAI_10, BLK_PAC_BAI_11, BLK_PAC_BAI_12, BLK_PAC_BAI_13, BLK_PAC_BAI_14,
14     BLK_PAC_BAI_20, BLK_PAC_BAI_21, BLK_PAC_BAI_22, BLK_PAC_BAI_23, BLK_PAC_BAI_24,
15     BLK_PAC_BAI_30, BLK_PAC_BAI_31, BLK_PAC_BAI_32, BLK_PAC_BAI_33, BLK_PAC_BAI_34,
16     BLK_PAC_BAI_40, BLK_PAC_BAI_41, BLK_PAC_BAI_42, BLK_PAC_BAI_43, BLK_PAC_BAI_44,
17     BLK_PAC_ESQ_00, BLK_PAC_ESQ_01, BLK_PAC_ESQ_02, BLK_PAC_ESQ_03, BLK_PAC_ESQ_04,
18     BLK_PAC_ESQ_10, BLK_PAC_ESQ_11, BLK_PAC_ESQ_12, BLK_PAC_ESQ_13, BLK_PAC_ESQ_14,
19     BLK_PAC_ESQ_20, BLK_PAC_ESQ_21, BLK_PAC_ESQ_22, BLK_PAC_ESQ_23, BLK_PAC_ESQ_24,
20     BLK_PAC_ESQ_30, BLK_PAC_ESQ_31, BLK_PAC_ESQ_32, BLK_PAC_ESQ_33, BLK_PAC_ESQ_34,
21     BLK_PAC_ESQ_40, BLK_PAC_ESQ_41, BLK_PAC_ESQ_42, BLK_PAC_ESQ_43, BLK_PAC_ESQ_44,
22     BLK_PAC_FECH_00, BLK_PAC_FECH_01, BLK_PAC_FECH_02, BLK_PAC_FECH_03, BLK_PAC_FECH_04,
23     BLK_PAC_FECH_10, BLK_PAC_FECH_11, BLK_PAC_FECH_12, BLK_PAC_FECH_13, BLK_PAC_FECH_14,
24     BLK_PAC_FECH_20, BLK_PAC_FECH_21, BLK_PAC_FECH_22, BLK_PAC_FECH_23, BLK_PAC_FECH_24,
25     BLK_PAC_FECH_30, BLK_PAC_FECH_31, BLK_PAC_FECH_32, BLK_PAC_FECH_33, BLK_PAC_FECH_34,
26     BLK_PAC_FECH_40, BLK_PAC_FECH_41, BLK_PAC_FECH_42, BLK_PAC_FECH_43, BLK_PAC_FECH_44,
27     BLK_PAC_FECV_00, BLK_PAC_FECV_01, BLK_PAC_FECV_02, BLK_PAC_FECV_03, BLK_PAC_FECV_04,
28     BLK_PAC_FECV_10, BLK_PAC_FECV_11, BLK_PAC_FECV_12, BLK_PAC_FECV_13, BLK_PAC_FECV_14,
29     BLK_PAC_FECV_20, BLK_PAC_FECV_21, BLK_PAC_FECV_22, BLK_PAC_FECV_23, BLK_PAC_FECV_24,
30     BLK_PAC_FECV_30, BLK_PAC_FECV_31, BLK_PAC_FECV_32, BLK_PAC_FECV_33, BLK_PAC_FECV_34,
31     BLK_PAC_FECV_40, BLK_PAC_FECV_41, BLK_PAC_FECV_42, BLK_PAC_FECV_43, BLK_PAC_FECV_44,
32     BLK_FAN_GRN_00, BLK_FAN_GRN_01, BLK_FAN_GRN_02, BLK_FAN_GRN_03, BLK_FAN_GRN_04,
33     BLK_FAN_GRN_10, BLK_FAN_GRN_11, BLK_FAN_GRN_12, BLK_FAN_GRN_13, BLK_FAN_GRN_14,
34     BLK_FAN_GRN_20, BLK_FAN_GRN_21, BLK_FAN_GRN_22, BLK_FAN_GRN_23, BLK_FAN_GRN_24,
35     BLK_FAN_GRN_30, BLK_FAN_GRN_31, BLK_FAN_GRN_32, BLK_FAN_GRN_33, BLK_FAN_GRN_34,
36     BLK_FAN_GRN_40, BLK_FAN_GRN_41, BLK_FAN_GRN_42, BLK_FAN_GRN_43, BLK_FAN_GRN_44,
37     BLK_EYE_GRN_CTM_00, BLK_EYE_GRN_CTM_01, BLK_EYE_GRN_CTM_02, BLK_EYE_GRN_CTM_10, BLK_EYE_GRN_CTM_11, BLK_EYE_GRN_CTM_12,
38     BLK_EYE_GRN_DIR_00, BLK_EYE_GRN_DIR_01, BLK_EYE_GRN_DIR_02, BLK_EYE_GRN_DIR_10, BLK_EYE_GRN_DIR_11, BLK_EYE_GRN_DIR_12,
39     BLK_EYE_GRN_BAI_00, BLK_EYE_GRN_BAI_01, BLK_EYE_GRN_BAI_02, BLK_EYE_GRN_BAI_10, BLK_EYE_GRN_BAI_11, BLK_EYE_GRN_BAI_12,
40     BLK_EYE_GRN_ESQ_00, BLK_EYE_GRN_ESQ_01, BLK_EYE_GRN_ESQ_02, BLK_EYE_GRN_ESQ_10, BLK_EYE_GRN_ESQ_11, BLK_EYE_GRN_ESQ_12,
41     BLK_FAN_RED_00, BLK_FAN_RED_01, BLK_FAN_RED_02, BLK_FAN_RED_03, BLK_FAN_RED_04,
42     BLK_FAN_RED_10, BLK_FAN_RED_11, BLK_FAN_RED_12, BLK_FAN_RED_13, BLK_FAN_RED_14,
43     BLK_FAN_RED_20, BLK_FAN_RED_21, BLK_FAN_RED_22, BLK_FAN_RED_23, BLK_FAN_RED_24,
44     BLK_FAN_RED_30, BLK_FAN_RED_31, BLK_FAN_RED_32, BLK_FAN_RED_33, BLK_FAN_RED_34,
45     BLK_FAN_RED_40, BLK_FAN_RED_41, BLK_FAN_RED_42, BLK_FAN_RED_43, BLK_FAN_RED_44,
46     BLK_EYE_RED_CTM_00, BLK_EYE_RED_CTM_01, BLK_EYE_RED_CTM_02, BLK_EYE_RED_CTM_10, BLK_EYE_RED_CTM_11, BLK_EYE_RED_CTM_12,
47     BLK_EYE_RED_DIR_00, BLK_EYE_RED_DIR_01, BLK_EYE_RED_DIR_02, BLK_EYE_RED_DIR_10, BLK_EYE_RED_DIR_11, BLK_EYE_RED_DIR_12,
48     BLK_EYE_RED_BAI_00, BLK_EYE_RED_BAI_01, BLK_EYE_RED_BAI_02, BLK_EYE_RED_BAI_10, BLK_EYE_RED_BAI_11, BLK_EYE_RED_BAI_12,
49     BLK_EYE_RED_ESQ_00, BLK_EYE_RED_ESQ_01, BLK_EYE_RED_ESQ_02, BLK_EYE_RED_ESQ_10, BLK_EYE_RED_ESQ_11, BLK_EYE_RED_ESQ_12,
50     BLK_EYE_BLK_CTM_00, BLK_EYE_BLK_CTM_01, BLK_EYE_BLK_CTM_02, BLK_EYE_BLK_CTM_10, BLK_EYE_BLK_CTM_11, BLK_EYE_BLK_CTM_12,
51     BLK_EYE_BLK_DIR_00, BLK_EYE_BLK_DIR_01, BLK_EYE_BLK_DIR_02, BLK_EYE_BLK_DIR_10, BLK_EYE_BLK_DIR_11, BLK_EYE_BLK_DIR_12,
52     BLK_EYE_BLK_BAI_00, BLK_EYE_BLK_BAI_01, BLK_EYE_BLK_BAI_02, BLK_EYE_BLK_BAI_10, BLK_EYE_BLK_BAI_11, BLK_EYE_BLK_BAI_12,
53     BLK_EYE_BLK_ESQ_00, BLK_EYE_BLK_ESQ_01, BLK_EYE_BLK_ESQ_02, BLK_EYE_BLK_ESQ_10, BLK_EYE_BLK_ESQ_11, BLK_EYE_BLK_ESQ_12,
54     BLK_FAN_VULN_00, BLK_FAN_VULN_01, BLK_FAN_VULN_02, BLK_FAN_VULN_03, BLK_FAN_VULN_04,
55     BLK_FAN_VULN_10, BLK_FAN_VULN_11, BLK_FAN_VULN_12, BLK_FAN_VULN_13, BLK_FAN_VULN_14,
56     BLK_FAN_VULN_20, BLK_FAN_VULN_21, BLK_FAN_VULN_22, BLK_FAN_VULN_23, BLK_FAN_VULN_24,
57     BLK_FAN_VULN_30, BLK_FAN_VULN_31, BLK_FAN_VULN_32, BLK_FAN_VULN_33, BLK_FAN_VULN_34,
58     BLK_FAN_VULN_40, BLK_FAN_VULN_41, BLK_FAN_VULN_42, BLK_FAN_VULN_43, BLK_FAN_VULN_44
59 );
60

```

Listing 4: Mapa inicial

```

1  CONSTANT MAPA_INICIAL: t_tab_array :=
2  "
3  "
4  " Q-----W Q-----W
5  "
6  "
7  " C..C..C..C..C..C..C..C..C..C C..C..C..C..C..C..C..C..C..C
8  " . . . . . . . . . . . . . . . . . . . . . . . . . . . .
9  " . . . . . . . . . . . . . . . . . . . . . . . . . . . .
10 " C Q-----W C Q-----W C C Q-----W C Q-----W C
11 " . . . . . . . . . . . . . . . . . . . . . . . . . . . .
12 " P . . . . . . . . . . . . . . . . . . . . . . . . . . . .
13 " . . . . . . . . . . . . . . . . . . . . . . . . . . . .
14 " . . . . . . . . . . . . . . . . . . . . . . . . . . . .
15 " C E-----R C E-----R C E-----R C E-----R C E-----R C
16 " . . . . . . . . . . . . . . . . . . . . . . . . . . . .
17 " . . . . . . . . . . . . . . . . . . . . . . . . . . . .
18 " C..C..C..C..C..C..C..C..C..C C..C..C..C..C..C..C..C..C..C
19 " . . . . . . . . . . . . . . . . . . . . . . . . . . . .
20 " . . . . . . . . . . . . . . . . . . . . . . . . . . . .
21 " . . . . . . . . . . . . . . . . . . . . . . . . . . . .
22 " C Q-----W C Q-----W C Q-----W C Q-----W C Q-----W C
23 " . | . | . | . | . | . | . | . | . | . | . | . | . |

```


14

Listing 6: sprites pacman

```

1 LIBRARY ieee;
2 USE ieee.STD_LOGIC_1164.all;
3 USE ieee.NUMERIC_STD.all;
4 USE work.PAC_DEFS.all;
5
6 PACKAGE pac_sprites IS
7   --Desenhos do pacman nas quatro possiveis direcoes
8   CONSTANT PAC_BITMAPS: t_ovl_blk_dir_vet := (
9     CIMA => ((BLK_PAC_CIM_00, BLK_PAC_CIM_01, BLK_PAC_CIM_02,
10              BLK_PAC_CIM_03, BLK_PAC_CIM_04),
11              (BLK_PAC_CIM_10, BLK_PAC_CIM_11, BLK_PAC_CIM_12,
12               BLK_PAC_CIM_13, BLK_PAC_CIM_14),
13              (BLK_PAC_CIM_20, BLK_PAC_CIM_21, BLK_PAC_CIM_22,
14               BLK_PAC_CIM_23, BLK_PAC_CIM_24),
15              (BLK_PAC_CIM_30, BLK_PAC_CIM_31, BLK_PAC_CIM_32,
16               BLK_PAC_CIM_33, BLK_PAC_CIM_34),
17              (BLK_PAC_CIM_40, BLK_PAC_CIM_41, BLK_PAC_CIM_42,
18               BLK_PAC_CIM_43, BLK_PAC_CIM_44)),
19     DIREI => ((BLK_PAC_DIR_00, BLK_PAC_DIR_01, BLK_PAC_DIR_02,
20                BLK_PAC_DIR_03, BLK_PAC_DIR_04),
21               (BLK_PAC_DIR_10, BLK_PAC_DIR_11, BLK_PAC_DIR_12,
22                BLK_PAC_DIR_13, BLK_PAC_DIR_14),
23               (BLK_PAC_DIR_20, BLK_PAC_DIR_21, BLK_PAC_DIR_22,
24                BLK_PAC_DIR_23, BLK_PAC_DIR_24),
25               (BLK_PAC_DIR_30, BLK_PAC_DIR_31, BLK_PAC_DIR_32,
26                BLK_PAC_DIR_33, BLK_PAC_DIR_34),
27               (BLK_PAC_DIR_40, BLK_PAC_DIR_41, BLK_PAC_DIR_42,
28                BLK_PAC_DIR_43, BLK_PAC_DIR_44)),
29     BAIXO => ((BLK_PAC_BAI_00, BLK_PAC_BAI_01, BLK_PAC_BAI_02,
30                BLK_PAC_BAI_03, BLK_PAC_BAI_04),
31               (BLK_PAC_BAI_10, BLK_PAC_BAI_11, BLK_PAC_BAI_12,
32                BLK_PAC_BAI_13, BLK_PAC_BAI_14),
33               (BLK_PAC_BAI_20, BLK_PAC_BAI_21, BLK_PAC_BAI_22,
34                BLK_PAC_BAI_23, BLK_PAC_BAI_24),
35               (BLK_PAC_BAI_30, BLK_PAC_BAI_31, BLK_PAC_BAI_32,
36                BLK_PAC_BAI_33, BLK_PAC_BAI_34),
37               (BLK_PAC_BAI_40, BLK_PAC_BAI_41, BLK_PAC_BAI_42,
38                BLK_PAC_BAI_43, BLK_PAC_BAI_44)),
39     ESQUE => ((BLK_PAC_ESQ_00, BLK_PAC_ESQ_01, BLK_PAC_ESQ_02,
40                BLK_PAC_ESQ_03, BLK_PAC_ESQ_04),
41               (BLK_PAC_ESQ_10, BLK_PAC_ESQ_11, BLK_PAC_ESQ_12,
42                BLK_PAC_ESQ_13, BLK_PAC_ESQ_14),
43               (BLK_PAC_ESQ_20, BLK_PAC_ESQ_21, BLK_PAC_ESQ_22,
44                BLK_PAC_ESQ_23, BLK_PAC_ESQ_24),
45               (BLK_PAC_ESQ_30, BLK_PAC_ESQ_31, BLK_PAC_ESQ_32,
46                BLK_PAC_ESQ_33, BLK_PAC_ESQ_34),
47               (BLK_PAC_ESQ_40, BLK_PAC_ESQ_41, BLK_PAC_ESQ_42,
48                BLK_PAC_ESQ_43, BLK_PAC_ESQ_44)),
49     OTHERS => (OTHERS => (OTHERS => BLK_NULL))
50   );
51
52   CONSTANT PAC_FECH_BITMAP: t_ovl_blk_5x5 := (
53     (BLK_PAC_FECH_00, BLK_PAC_FECH_01, BLK_PAC_FECH_02,
54      BLK_PAC_FECH_03, BLK_PAC_FECH_04),
55     (BLK_PAC_FECH_10, BLK_PAC_FECH_11, BLK_PAC_FECH_12,
56      BLK_PAC_FECH_13, BLK_PAC_FECH_14),
57     (BLK_PAC_FECH_20, BLK_PAC_FECH_21, BLK_PAC_FECH_22,
58      BLK_PAC_FECH_23, BLK_PAC_FECH_24),
59     (BLK_PAC_FECH_30, BLK_PAC_FECH_31, BLK_PAC_FECH_32,
60      BLK_PAC_FECH_33, BLK_PAC_FECH_34),
61     (BLK_PAC_FECH_40, BLK_PAC_FECH_41, BLK_PAC_FECH_42,
62      BLK_PAC_FECH_43, BLK_PAC_FECH_44));
63
64   CONSTANT PAC_FECV_BITMAP: t_ovl_blk_5x5 := (
65     (BLK_PAC_FECV_00, BLK_PAC_FECV_01, BLK_PAC_FECV_02,
66      BLK_PAC_FECV_03, BLK_PAC_FECV_04),
67     (BLK_PAC_FECV_10, BLK_PAC_FECV_11, BLK_PAC_FECV_12,
68      BLK_PAC_FECV_13, BLK_PAC_FECV_14),
69     (BLK_PAC_FECV_20, BLK_PAC_FECV_21, BLK_PAC_FECV_22,
70      BLK_PAC_FECV_23, BLK_PAC_FECV_24),
71     (BLK_PAC_FECV_30, BLK_PAC_FECV_31, BLK_PAC_FECV_32,
72      BLK_PAC_FECV_33, BLK_PAC_FECV_34),
73     (BLK_PAC_FECV_40, BLK_PAC_FECV_41, BLK_PAC_FECV_42,
74      BLK_PAC_FECV_43, BLK_PAC_FECV_44));
75
76   CONSTANT FAN_BITMAPS: t_fans_ovl_blk_dir_vet := (
77     0 => ( --primeiro fantasma:
78       CIMA => ((BLK_FAN_GRN_00, BLK_FAN_GRN_01, BLK_FAN_GRN_02,
79                 BLK_FAN_GRN_03, BLK_FAN_GRN_04),
80               (BLK_FAN_GRN_10, BLK_FAN_GRN_11, BLK_FAN_GRN_12,
81                 BLK_FAN_GRN_13, BLK_FAN_GRN_14),
82               (BLK_FAN_GRN_20, BLK_FAN_GRN_21, BLK_FAN_GRN_22,
83                 BLK_FAN_GRN_23, BLK_FAN_GRN_24),
84               (BLK_FAN_GRN_30, BLK_FAN_GRN_31, BLK_FAN_GRN_32,
85                 BLK_FAN_GRN_33, BLK_FAN_GRN_34),
86               (BLK_FAN_GRN_40, BLK_FAN_GRN_41, BLK_FAN_GRN_42,
87                 BLK_FAN_GRN_43, BLK_FAN_GRN_44)),
88       DIREI=> ((BLK_FAN_GRN_00, BLK_FAN_GRN_01, BLK_FAN_GRN_02,
89                  BLK_FAN_GRN_03, BLK_FAN_GRN_04),
90                (BLK_FAN_GRN_10, BLK_FAN_GRN_11, BLK_FAN_GRN_12,
91                 BLK_FAN_GRN_13, BLK_FAN_GRN_14),
92                (BLK_FAN_GRN_20, BLK_FAN_GRN_21, BLK_FAN_GRN_22,
93                 BLK_FAN_GRN_23, BLK_FAN_GRN_24),
94                (BLK_FAN_GRN_30, BLK_FAN_GRN_31, BLK_FAN_GRN_32,
95                 BLK_FAN_GRN_33, BLK_FAN_GRN_34),
96                (BLK_FAN_GRN_40, BLK_FAN_GRN_41, BLK_FAN_GRN_42,
97                 BLK_FAN_GRN_43, BLK_FAN_GRN_44)),
98       BAIXO=> ((BLK_FAN_RED_00, BLK_FAN_RED_01, BLK_FAN_RED_02,
99                  BLK_FAN_RED_03, BLK_FAN_RED_04),
100                (BLK_FAN_RED_10, BLK_FAN_RED_11, BLK_FAN_RED_12,
101                 BLK_FAN_RED_13, BLK_FAN_RED_14),
102                (BLK_FAN_RED_20, BLK_FAN_RED_21, BLK_FAN_RED_22,
103                 BLK_FAN_RED_23, BLK_FAN_RED_24),
104                (BLK_FAN_RED_30, BLK_FAN_RED_31, BLK_FAN_RED_32,
105                 BLK_FAN_RED_33, BLK_FAN_RED_34),
106                (BLK_FAN_RED_40, BLK_FAN_RED_41, BLK_FAN_RED_42,
107                 BLK_FAN_RED_43, BLK_FAN_RED_44)),
108       OTHERS => (OTHERS => (OTHERS => BLK_NULL))
109     );
110
111   CONSTANT FAN_DEAD_BITMAPS: t_ovl_blk_dir_vet := (
112     CIMA => (( BLK_NULL, BLK_NULL, BLK_NULL, BLK_NULL,
113                BLK_NULL),
114              ( BLK_NULL, BLK_FAN_CIM_00, BLK_FAN_CIM_01,
115               BLK_FAN_CIM_02, BLK_NULL),
116              ( BLK_NULL, BLK_FAN_CIM_10, BLK_FAN_CIM_11,
117               BLK_FAN_CIM_12, BLK_NULL),
118              ( BLK_NULL, BLK_NULL, BLK_NULL, BLK_NULL,
119               BLK_NULL)),
120     DIREI=> (( BLK_NULL, BLK_NULL, BLK_NULL, BLK_NULL,
121                BLK_NULL),
122              ( BLK_NULL, BLK_FAN_DIR_00, BLK_FAN_DIR_01,
123               BLK_FAN_DIR_02, BLK_NULL),
124              ( BLK_NULL, BLK_FAN_DIR_10, BLK_FAN_DIR_11,
125               BLK_FAN_DIR_12, BLK_NULL),
126              ( BLK_NULL, BLK_NULL, BLK_NULL, BLK_NULL,
127               BLK_NULL))
128   );

```



```

103      ( BLK_NULL,      BLK_NULL,      BLK_NULL,      BLK_NULL,      179 BLK_PAC_DTR_12 => ("11111", "11111", "11111", "11111", "11111"),
104      BAIXO=> (( BLK_NULL,      BLK_NULL,      BLK_NULL,      BLK_NULL,      180 BLK_PAC_DTR_13 => ("11111", "11111", "11111", "11100", "10000"),
105      , BLK_NULL), 181 BLK_PAC_DTR_14 => ("11000", "11000", "00000", "00000", "00000"),
106      ( BLK_NULL, BLK_EYE_BLK_BAI_00, BLK_EYE_BLK_BAI_01, 182 BLK_PAC_DTR_20 => ("00111", "00111", "00111", "00111", "00111"),
107      BLK_EYE_BLK_BAI_02, BLK_NULL), 183 BLK_PAC_DTR_21 => ("11111", "11111", "11111", "11111", "11111"),
108      ( BLK_NULL, BLK_EYE_BLK_BAI_10, BLK_EYE_BLK_BAI_11, 184 BLK_PAC_DTR_22 => ("11111", "11100", "11000", "11100", "11111"),
109      BLK_EYE_BLK_BAI_12, BLK_NULL), 185 BLK_PAC_DTR_23 => ("00000", "00000", "00000", "00000", "00000"),
110      ( BLK_NULL, BLK_NULL, BLK_NULL, BLK_NULL, 186 BLK_PAC_DTR_24 => ("00000", "00000", "00000", "00000", "00000"),
111      BLK_NULL), 187 BLK_PAC_DTR_30 => ("00111", "00011", "00001", "00001", "00000"),
112      ( BLK_NULL, BLK_NULL, BLK_NULL, BLK_NULL, 188 BLK_PAC_DTR_31 => ("11111", "11111", "11111", "11111", "11111"),
113      BLK_NULL), 189 BLK_PAC_DTR_32 => ("11111", "11111", "11111", "11111", "11111"),
114      ( BLK_NULL), 190 BLK_PAC_DTR_33 => ("10000", "11100", "11111", "11111", "11111"),
115      ESQUE=> (( BLK_NULL,      BLK_NULL,      BLK_NULL,      BLK_NULL,      191 BLK_PAC_DTR_34 => ("00000", "00000", "00000", "11000", "11000"),
116      , BLK_NULL), 192 BLK_PAC_DTR_40 => ("00000", "00000", "00000", "00000", "00000"),
117      ( BLK_NULL, BLK_EYE_BLK_ESQ_00, BLK_EYE_BLK_ESQ_01, 193 BLK_PAC_DTR_41 => ("01111", "00111", "00011", "00000", "00000"),
118      BLK_EYE_BLK_ESQ_02, BLK_NULL), 194 BLK_PAC_DTR_42 => ("11111", "11111", "11111", "11111", "00000"),
119      ( BLK_NULL, BLK_EYE_BLK_ESQ_10, BLK_EYE_BLK_ESQ_11, 195 BLK_PAC_DTR_43 => ("11111", "11111", "11110", "11000", "00000"),
120      BLK_EYE_BLK_ESQ_12, BLK_NULL), 196 BLK_PAC_DTR_44 => ("10000", "00000", "00000", "00000", "00000"),
121      ( BLK_NULL, BLK_NULL, BLK_NULL, BLK_NULL, 197 BLK_PAC_BAI_00 => ("00000", "00000", "00000", "00000", "00000"),
122      BLK_NULL), 198 BLK_PAC_BAI_01 => ("00000", "00000", "00001", "00011", "01111"),
123      ( BLK_NULL, BLK_NULL, BLK_NULL, BLK_NULL, 199 BLK_PAC_BAI_02 => ("00000", "00000", "11111", "11111", "11111"),
124      BLK_NULL), 200 BLK_PAC_BAI_03 => ("00000", "00000", "10000", "11000", "11110"),
125      OTHERS => (OTHERS => (OTHERS => BLK_NULL)) 201 BLK_PAC_BAI_04 => ("00000", "00000", "00000", "00000", "00000"),
126      ); 202 BLK_PAC_BAI_10 => ("00000", "00001", "00011", "00111", "00111"),
127      203 BLK_PAC_BAI_11 => ("11111", "11111", "11111", "11111", "11111"),
128      204 BLK_PAC_BAI_12 => ("11111", "11111", "11111", "11111", "11111"),
129      205 BLK_PAC_BAI_13 => ("11111", "11111", "11111", "11111", "11111"),
130      206 BLK_PAC_BAI_14 => ("00000", "10000", "11000", "11100", "11100"),
131      207 BLK_PAC_BAI_20 => ("01111", "01111", "01110", "01111", "01111"),
132      208 BLK_PAC_BAI_21 => ("11111", "11111", "11111", "11111", "11111"),
133      209 BLK_PAC_BAI_22 => ("11111", "11111", "11011", "10001", "10001"),
134      210 BLK_PAC_BAI_23 => ("11111", "11111", "11111", "11111", "11111"),
135      211 BLK_PAC_BAI_24 => ("11110", "11110", "11110", "11110", "11110"),
136      212 BLK_PAC_BAI_30 => ("01111", "01111", "00111", "00111", "00011"),
137      213 BLK_PAC_BAI_31 => ("11111", "11110", "11110", "11100", "11100"),
138      214 BLK_PAC_BAI_32 => ("00000", "00000", "00000", "00000", "00000"),
139      215 BLK_PAC_BAI_33 => ("11111", "01111", "01111", "01111", "01111"),
140      216 BLK_PAC_BAI_34 => ("11110", "11110", "11100", "11100", "11000"),
141      217 BLK_PAC_BAI_40 => ("00001", "00000", "00000", "00000", "00000"),
142      218 BLK_PAC_BAI_41 => ("11000", "11000", "00000", "00000", "00000"),
143      219 BLK_PAC_BAI_42 => ("00000", "00000", "00000", "00000", "00000"),
144      220 BLK_PAC_BAI_43 => ("00011", "00011", "00000", "00000", "00000"),
145      221 BLK_PAC_BAI_44 => ("00000", "00000", "00000", "00000", "00000"),
146      222 BLK_PAC_ESQ_00 => ("00000", "00000", "00000", "00000", "00001"),
147      223 BLK_PAC_ESQ_01 => ("00000", "00011", "01111", "11111", "11111"),
148      224 BLK_PAC_ESQ_02 => ("00000", "11111", "11111", "11111", "11111"),
149      225 BLK_PAC_ESQ_03 => ("00000", "00000", "11000", "11100", "11110"),
150      226 BLK_PAC_ESQ_04 => ("00000", "00000", "00000", "00000", "00000"),
151      227 BLK_PAC_ESQ_10 => ("00011", "00011", "00000", "00000", "00000"),
152      228 BLK_PAC_ESQ_11 => ("11111", "11111", "11111", "00111", "00001"),
153      229 BLK_PAC_ESQ_12 => ("11111", "11111", "11111", "11111", "11111"),
154      230 BLK_PAC_ESQ_13 => ("11111", "11111", "11111", "11111", "11111"),
155      231 BLK_PAC_ESQ_14 => ("00000", "10000", "10000", "11000", "11100"),
156      232 BLK_PAC_ESQ_20 => ("00000", "00000", "00000", "00000", "00000"),
157      233 BLK_PAC_ESQ_21 => ("00000", "00000", "00000", "00000", "00000"),
158      234 BLK_PAC_ESQ_22 => ("11111", "00111", "00011", "00111", "11111"),
159      235 BLK_PAC_ESQ_23 => ("11111", "11111", "11111", "11111", "11111"),
160      236 BLK_PAC_ESQ_24 => ("11100", "11100", "11100", "11100", "11100"),
161      237 BLK_PAC_ESQ_30 => ("00000", "00000", "00000", "00011", "00011"),
162      238 BLK_PAC_ESQ_31 => ("00001", "00111", "11111", "11111", "11111"),
163      239 BLK_PAC_ESQ_32 => ("11111", "11111", "11111", "11111", "11111"),
164      240 BLK_PAC_ESQ_33 => ("11111", "11111", "11111", "11111", "11111"),
165      241 BLK_PAC_ESQ_34 => ("11100", "11000", "10000", "10000", "00000"),
166      242 BLK_PAC_ESQ_40 => ("00001", "00000", "00000", "00000", "00000"),
167      243 BLK_PAC_ESQ_41 => ("11111", "11111", "01111", "00011", "00000"),
168      244 BLK_PAC_ESQ_42 => ("11111", "11111", "11111", "11111", "00000"),
169      245 BLK_PAC_ESQ_43 => ("11110", "11100", "11000", "00000", "00000"),
170      246 BLK_PAC_ESQ_44 => ("00000", "00000", "00000", "00000", "00000"),
171      247 BLK_PAC_FECV_00 => ("00000", "00000", "00000", "00000", "00000"),
172      248 BLK_PAC_FECV_01 => ("00000", "00000", "00001", "00011", "01111"),
173      249 BLK_PAC_FECV_02 => ("00000", "00000", "11111", "11111", "11111"),
174      250 BLK_PAC_FECV_03 => ("00000", "00000", "10000", "11000", "11110"),
175      251 BLK_PAC_FECV_04 => ("00000", "00000", "00000", "00000", "00000"),
176      252 BLK_PAC_FECV_10 => ("00000", "00001", "00011", "00111", "00111"),
177      253 BLK_PAC_FECV_11 => ("11111", "11111", "11111", "11111", "11111"),
178      254 BLK_PAC_FECV_12 => ("11111", "11111", "11111", "11111", "11111"),
179      255 BLK_PAC_FECV_13 => ("11111", "11111", "11111", "11111", "11111"),
180      256 BLK_PAC_FECV_14 => ("00000", "10000", "10000", "11000", "11100"),
181      257 BLK_PAC_FECV_20 => ("01111", "01111", "01111", "01111", "01111"),
182      258 BLK_PAC_FECV_21 => ("11111", "11111", "11111", "11111", "11111"),
183      259 BLK_PAC_FECV_22 => ("11111", "11111", "11111", "11111", "11111"),
184      260 BLK_PAC_FECV_23 => ("11111", "11111", "11111", "11111", "11111"),
185      261 BLK_PAC_FECV_24 => ("11110", "11110", "11110", "11110", "11110"),
186      262 BLK_PAC_FECV_30 => ("00111", "00111", "00011", "00001", "00000"),
187      263 BLK_PAC_FECV_31 => ("11111", "11111", "11111", "11111", "11111"),
188      264 BLK_PAC_FECV_32 => ("11111", "11111", "11111", "11111", "11111"),
189      265 BLK_PAC_FECV_33 => ("11111", "11111", "11111", "11111", "11111"),
190      266 BLK_PAC_FECV_34 => ("11100", "11100", "11000", "10000", "00000"),
191      267 BLK_PAC_FECV_40 => ("00000", "00000", "00000", "00000", "00000"),
192      268 BLK_PAC_FECV_41 => ("01111", "00011", "00001", "00000", "00000"),
193      269 BLK_PAC_FECV_42 => ("11111", "11111", "11111", "00000", "00000"),
194      270 BLK_PAC_FECV_43 => ("11110", "11000", "10000", "00000", "00000"),

```


456	B1K_PAC_D_24	=>	"00000"	"00000"	"00000"	"00000"	"00000"
457	B1K_PAC_DTR_30	=>	"001111"	"00011"	"00001"	"00001"	"00000"
458	B1K_PAC_DTR_32	=>	"111111"	"11111"	"11111"	"11111"	"11111"
459	B1K_PAC_DTR_33	=>	"111111"	"11111"	"11111"	"11111"	"11111"
460	B1K_PAC_DTR_34	=>	"100000"	"11100"	"11111"	"11111"	"11100"
461	B1K_PAC_DTR_40	=>	"000000"	"00000"	"00000"	"00000"	"00000"
462	B1K_PAC_DTR_41	=>	"001111"	"00111"	"00011"	"00000"	"00000"
463	B1K_PAC_DTR_42	=>	"111111"	"11111"	"11111"	"11111"	"00000"
464	B1K_PAC_DTR_43	=>	"111111"	"11111"	"11110"	"10000"	"00000"
465	B1K_PAC_DTR_44	=>	"100000"	"00000"	"00000"	"00000"	"00000"
466	B1K_PAC_BAI_00	=>	"000000"	"00000"	"00000"	"00000"	"00000"
467	B1K_PAC_BAI_01	=>	"000000"	"00000"	"00001"	"00011"	"00111"
468	B1K_PAC_BAI_02	=>	"000000"	"00000"	"11111"	"11111"	"11111"
469	B1K_PAC_BAI_03	=>	"000000"	"00000"	"10000"	"10000"	"11110"
470	B1K_PAC_BAI_04	=>	"000000"	"00000"	"00000"	"00000"	"00000"
471	B1K_PAC_BAI_10	=>	"000000"	"00001"	"00011"	"00111"	"00111"
472	B1K_PAC_BAI_11	=>	"111111"	"11111"	"11111"	"11111"	"11111"
473	B1K_PAC_BAI_12	=>	"111111"	"11111"	"11111"	"11111"	"11111"
474	B1K_PAC_BAI_13	=>	"111111"	"11111"	"11111"	"11111"	"11111"
475	B1K_PAC_BAI_14	=>	"000000"	"00000"	"10000"	"10000"	"11100"
476	B1K_PAC_BAI_20	=>	"011111"	"01111"	"01111"	"01111"	"01111"
477	B1K_PAC_BAI_21	=>	"111111"	"11111"	"11111"	"11111"	"11111"
478	B1K_PAC_BAI_22	=>	"111111"	"11111"	"10111"	"10001"	"10001"
479	B1K_PAC_BAI_23	=>	"111111"	"11111"	"11111"	"11111"	"11111"
480	B1K_PAC_BAI_24	=>	"111110"	"11110"	"11110"	"11110"	"11110"
481	B1K_PAC_BAI_30	=>	"011111"	"01111"	"00111"	"00111"	"00011"
482	B1K_PAC_BAI_31	=>	"111111"	"11110"	"11110"	"11000"	"11100"
483	B1K_PAC_BAI_32	=>	"000000"	"00000"	"00000"	"00000"	"00000"
484	B1K_PAC_BAI_33	=>	"111111"	"01111"	"01111"	"00111"	"00111"
485	B1K_PAC_BAI_34	=>	"111110"	"11110"	"11100"	"11000"	"10000"
486	B1K_PAC_BAI_40	=>	"000001"	"00000"	"00000"	"00000"	"00000"
487	B1K_PAC_BAI_41	=>	"10000"	"10000"	"00000"	"00000"	"00000"
488	B1K_PAC_BAI_42	=>	"000000"	"00000"	"00000"	"00000"	"00000"
489	B1K_PAC_BAI_43	=>	"000011"	"00011"	"00000"	"00000"	"00000"
490	B1K_PAC_BAI_44	=>	"10000"	"00000"	"00000"	"00000"	"00000"
491	B1K_PAC_ESQ_00	=>	"000000"	"00000"	"00000"	"00000"	"00001"
492	B1K_PAC_ESQ_01	=>	"000000"	"00011"	"01111"	"11111"	"11111"
493	B1K_PAC_ESQ_02	=>	"000000"	"11111"	"11111"	"11111"	"11111"
494	B1K_PAC_ESQ_03	=>	"000000"	"00000"	"10000"	"11000"	"11110"
495	B1K_PAC_ESQ_04	=>	"000000"	"00000"	"00000"	"00000"	"00000"
496	B1K_PAC_ESQ_10	=>	"000011"	"00011"	"00000"	"00000"	"00000"
497	B1K_PAC_ESQ_11	=>	"111111"	"11111"	"11111"	"00111"	"00001"
498	B1K_PAC_ESQ_12	=>	"111111"	"11111"	"11111"	"11111"	"11111"
499	B1K_PAC_ESQ_13	=>	"111111"	"11111"	"11111"	"11111"	"11111"
500	B1K_PAC_ESQ_14	=>	"000000"	"10000"	"10000"	"10000"	"11100"
501	B1K_PAC_ESQ_20	=>	"000000"	"00000"	"00000"	"00000"	"00000"
502	B1K_PAC_ESQ_21	=>	"000000"	"00000"	"00000"	"00000"	"00000"
503	B1K_PAC_ESQ_22	=>	"111111"	"00111"	"00011"	"00111"	"11111"
504							

```

547 BLK_PAC_FCH_11 => ("11111", "11111", "11111", "11111", "11111", "11111"),
548 BLK_PAC_FCH_12 => ("11111", "11111", "11111", "11111", "11111", "11111"),
549 BLK_PAC_FCH_13 => ("11111", "11111", "11111", "11111", "11111", "11111"),
550 BLK_PAC_FCH_14 => ("00000", "10000", "10000", "10000", "11000", "11100"),
551 BLK_PAC_FCH_20 => ("00111", "00111", "00111", "00111", "00111", "00111"),
552 BLK_PAC_FCH_21 => ("11111", "11111", "11111", "11111", "11111", "11111"),
553 BLK_PAC_FCH_22 => ("11111", "11111", "11111", "11111", "11111", "11111"),
554 BLK_PAC_FCH_23 => ("11111", "11111", "11111", "11111", "11111", "11111"),
555 BLK_PAC_FCH_24 => ("11100", "11100", "11100", "11100", "11100", "11100"),
556 BLK_PAC_FCH_30 => ("00111", "00011", "00001", "00001", "00000", "00000"),
557 BLK_PAC_FCH_31 => ("11111", "11111", "11111", "11111", "11111", "11111"),
558 BLK_PAC_FCH_32 => ("11111", "11111", "11111", "11111", "11111", "11111"),
559 BLK_PAC_FCH_33 => ("11111", "11111", "11111", "11111", "11111", "11111"),
560 BLK_PAC_FCH_34 => ("11100", "11000", "10000", "10000", "00000", "00000"),
561 BLK_PAC_FCH_40 => ("00000", "00000", "00000", "00000", "00000", "00000"),
562 BLK_PAC_FCH_41 => ("01111", "00111", "00011", "00000", "00000", "00000"),
563 BLK_PAC_FCH_42 => ("11111", "11111", "11111", "11111", "00000", "00000"),
564 BLK_PAC_FCH_43 => ("11110", "11100", "11000", "00000", "00000", "00000"),
565 BLK_PAC_FCH_44 => ("00000", "00000", "00000", "00000", "00000", "00000"),
566 BLK_EYE_RED_CIM_00 => ("00001", "10001", "10001", "10001", "11111", "11111"),
567 BLK_EYE_RED_CIM_01 => ("00000", "10001", "10001", "10001", "10001", "10001"),
568 BLK_EYE_RED_CIM_02 => ("00010", "00001", "00001", "11111", "11111", "11111"),
569 BLK_EYE_RED_CIM_10 => ("11111", "01111", "00000", "00000", "00000", "00000"),
570 BLK_EYE_RED_CIM_11 => ("10001", "00000", "00000", "00000", "00000", "00000"),
571 BLK_EYE_RED_CIM_12 => ("11111", "11110", "00000", "00000", "00000", "00000"),
572 BLK_EYE_RED_DIR_00 => ("01111", "11111", "11100", "11100", "11100", "11100"),
573 BLK_EYE_RED_DIR_01 => ("00000", "10001", "00001", "00001", "00001", "00001"),
574 BLK_EYE_RED_DIR_02 => ("11110", "11111", "11000", "11000", "11000", "11000"),
575 BLK_EYE_RED_DIR_10 => ("11111", "01111", "00000", "00000", "00000", "00000"),
576 BLK_EYE_RED_DIR_11 => ("10001", "00000", "00000", "00000", "00000", "00000"),
577 BLK_EYE_RED_DIR_12 => ("11111", "11110", "00000", "00000", "00000", "00000"),
578 BLK_EYE_RED_BAI_00 => ("01111", "11111", "11111", "11111", "11111", "11100"),
579 BLK_EYE_RED_BAI_01 => ("00000", "10001", "10001", "10001", "10001", "10001"),
580 BLK_EYE_RED_BAI_02 => ("11110", "11111", "11111", "11111", "11111", "10001"),
581 BLK_EYE_RED_BAI_10 => ("11000", "01000", "00000", "00000", "00000", "00000"),
582 BLK_EYE_RED_BAI_11 => ("10001", "00000", "00000", "00000", "00000", "00000"),
583 BLK_EYE_RED_BAI_12 => ("10001", "10000", "00000", "00000", "00000", "00000"),
584 BLK_EYE_RED_ESQ_00 => ("01111", "11111", "00011", "00011", "00011", "00011"),
585 BLK_EYE_RED_ESQ_01 => ("00000", "10001", "10000", "10000", "10000", "10000"),
586 BLK_EYE_RED_ESQ_02 => ("11110", "11111", "00011", "00011", "00011", "00011"),
587 BLK_EYE_RED_ESQ_10 => ("11111", "01111", "00000", "00000", "00000", "00000"),
588 BLK_EYE_RED_ESQ_11 => ("10001", "00000", "00000", "00000", "00000", "00000"),
589 BLK_EYE_RED_ESQ_12 => ("11111", "11110", "00000", "00000", "00000", "00000"),
590 BLK_FAN_GRN_00 => ("00000", "00000", "00000", "00000", "00000", "00000"),
591 BLK_FAN_GRN_01 => ("00000", "00111", "01111", "11111", "11111", "11111"),
592 BLK_FAN_GRN_02 => ("00000", "01111", "11111", "11111", "11111", "11111"),
593 BLK_FAN_GRN_03 => ("00000", "11000", "11100", "11110", "11110", "11110"),
594 BLK_FAN_GRN_04 => ("00000", "00000", "00000", "00000", "00000", "00000"),
595 BLK_FAN_GRN_10 => ("00001", "00011", "00011", "00111", "00111", "00111"),
596 BLK_FAN_GRN_14 => ("00000", "10000", "10000", "10000", "11000", "11000"),
597 BLK_FAN_GRN_20 => ("00111", "00111", "01111", "01111", "01111", "01111"),
598 BLK_FAN_GRN_24 => ("11000", "11000", "11100", "11100", "11100", "11100"),
599 BLK_FAN_GRN_30 => ("01111", "01111", "01111", "01111", "01111", "01111"),
600 BLK_FAN_GRN_31 => ("11111", "11111", "11111", "11111", "11111", "11111"),
601 BLK_FAN_GRN_32 => ("11111", "11111", "11111", "11111", "11111", "11111"),
602 BLK_FAN_GRN_33 => ("11111", "11111", "11111", "11111", "11111", "11111"),
603 BLK_FAN_GRN_34 => ("11100", "11100", "11100", "11100", "11100", "11100"),
604 BLK_FAN_GRN_40 => ("01111", "01111", "01111", "00110", "00000", "00000"),
605 BLK_FAN_GRN_41 => ("10011", "00001", "00000", "00000", "00000", "00000"),
606 BLK_FAN_GRN_42 => ("11111", "11111", "11110", "01100", "00000", "00000"),
607 BLK_FAN_GRN_43 => ("10011", "00001", "00001", "00000", "00000", "00000"),
608 BLK_FAN_GRN_44 => ("11100", "11100", "11100", "11000", "00000", "00000"),
609 BLK_EYE_GRN_CIM_00 => ("10001", "10001", "10001", "10001", "11111", "11111"),
610 BLK_EYE_GRN_CIM_01 => ("11111", "11111", "11111", "11111", "11111", "11111"),
611 BLK_EYE_GRN_CIM_02 => ("00011", "00011", "00011", "11111", "11111", "11111"),
612 BLK_EYE_GRN_CIM_10 => ("11111", "11111", "11111", "11111", "11111", "11111"),
613 BLK_EYE_GRN_CIM_11 => ("11111", "11111", "11111", "11111", "11111", "11111"),
614 BLK_EYE_GRN_CIM_12 => ("11111", "11111", "11111", "11111", "11111", "11111"),
615 BLK_EYE_GRN_DIR_00 => ("11111", "11111", "11100", "11100", "11100", "11100"),
616 BLK_EYE_GRN_DIR_01 => ("11111", "11111", "01111", "01111", "01111", "01111"),
6
```



```

10     key_dir           :IN t_direcao; --tecla de ação lida pelo teclado
11     pac_area          :IN t_blk_sym_3x3; --mapa 3x3 em torno da posição atual
12     pac_pos_x, pac_pos_y :BUFFER t_pos;
13     pac_cur_dir        :BUFFER t_direcao;
14     got_coin, got_spc_coin :OUT STD_LOGIC
15 );
16 END ctrl_pacman;
17
18 ARCHITECTURE behav OF ctrl_pacman IS
19     SIGNAL pac_nxt_cel, pac_dir_cel, pac_esq_cel, pac_cim_cel, pac_bai_cel: t_blk_sym;
20 BEGIN
21     --Calcula possíveis parâmetros envolvidos no próximo movimento
22     --do pacman
23     PROCESS (pac_cur_dir, pac_area)
24     BEGIN
25         --calcula qual seriam as proximas celulas visitadas pelo pacman
26         pac_nxt_cel <= pac_area(DIRS(pac_cur_dir)(0), DIRS(pac_cur_dir)(1));
27
28         IF (WALKABLE(pac_area(DIRS(pac_cur_dir)(0), DIRS(pac_cur_dir)(1)))) THEN
29             --o pacman conseguirá andar para a próxima casa
30             pac_dir_cel <= pac_area(DIRS(pac_cur_dir)(0), DIRS(pac_cur_dir)(1) + 1);
31             pac_esq_cel <= pac_area(DIRS(pac_cur_dir)(0), DIRS(pac_cur_dir)(1) - 1);
32             pac_cim_cel <= pac_area(DIRS(pac_cur_dir)(0)-1, DIRS(pac_cur_dir)(1));
33             pac_bai_cel <= pac_area(DIRS(pac_cur_dir)(0)+1, DIRS(pac_cur_dir)(1));
34         ELSE
35             pac_dir_cel <= pac_area( 0, 1);
36             pac_esq_cel <= pac_area( 0, -1);
37             pac_cim_cel <= pac_area(-1, 0);
38             pac_bai_cel <= pac_area( 1, 0);
39         END IF;
40     END PROCESS;
41
42     -- purpose: Este processo irá atualizar a posição do pacman e definir
43     -- suas ações no jogo. Opera no estado ATUALIZA_LOGICA_1
44     -- type : sequential
45     p_atualiza_pacman: PROCESS (clk27M, rstn)
46     VARIABLE nxt_move, key_dir_old: t_direcao;
47     BEGIN
48         IF (rstn = '0') THEN
49             pac_pos_x <= PAC_START_X;
50             pac_pos_y <= PAC_START_Y;
51             pac_cur_dir <= NADA;
52             nxt_move := NADA;
53         ELSIF (clk27M'event and clk27M = '1') THEN
54             IF (atualiza = '1') THEN
55                 --Checa teclado para "agendar" um movimento
56                 IF (key_dir /= NADA and key_dir_old = NADA) THEN
57                     nxt_move := key_dir;
58                 END IF;
59
60                 --atualiza direção
61                 IF (nxt_move = CIMA and WALKABLE(pac_cim_cel)) THEN
62                     pac_cur_dir <= CIMA;
63                     nxt_move := NADA;
64                 ELSIF (nxt_move = DIREI and WALKABLE(pac_dir_cel)) THEN
65                     pac_cur_dir <= DIREI;
66                     nxt_move := NADA;
67                 ELSIF (nxt_move = BAIXO and WALKABLE(pac_bai_cel)) THEN
68                     pac_cur_dir <= BAIXO;
69                     nxt_move := NADA;
70                 ELSIF (nxt_move = ESQUE and WALKABLE(pac_esq_cel)) THEN
71                     pac_cur_dir <= ESQUE;
72                     nxt_move := NADA;
73                 END IF;
74
75                 IF (WALKABLE(pac_nxt_cel)) THEN --atualiza posicao
76                     IF (pac_pos_x = TELE_DIR_POS) then --teletransporte
77                         pac_pos_x <= TELE_ESQ_POS + 1;
78                     ELSIF (pac_pos_x = TELE_ESQ_POS) then

```

```

79         pac_pos_x <= TELE_DIR_POS - 1;
80     ELSE
81         pac_pos_x <= pac_pos_x + DIRS(pac_cur_dir)(1);
82         pac_pos_y <= pac_pos_y + DIRS(pac_cur_dir)(0);
83     END IF;
84 END IF;
85     key_dir_old := key_dir;
86 END IF;
87 END IF;
88 END PROCESS;
89
90 got_coin <= '1' WHEN (pac_nxt_cel = BLK_COIN and atualiza = '1')
91 ELSE '0';
92
93 got_spc_coin <= '1' WHEN (pac_nxt_cel = BLK_SPC_COIN and atualiza = '1')
94 ELSE '0';
95 END behav;

```

Listing 8: Definições para pacman

```

1  LIBRARY ieee;
2  USE ieee.STD_LOGIC_1164.all;
3  USE ieee.NUMERIC_STD.all;
4  USE work.PAC_DEFS.all;
5
6  ENTITY ctrl_fans IS
7  PORT (
8      clk27M, rstn      :IN STD_LOGIC;
9      atualiza          :IN STD_LOGIC;
10     atua_en           :IN STD_LOGIC_VECTOR(2 downto 0); --velocidades para atualizar
11     keys_dir          :IN t_fans_dirs; --teclas de ação lidas pelo teclado
12     fan_area           :IN t_fans_blk_sym_3x3; --mapa 3x3 em torno da posição atual
13     spc_coin           :IN STD_LOGIC;
14     pac_fans_hit       :IN UNSIGNED(0 to FAN_NO-1);
15     fan_pos_x, fan_pos_y :BUFFER t_fans_pos;
16     fan_state          :BUFFER t_fans_states;
17     fan_cur_dir        :BUFFER t_fans_dirs;
18     pacman_dead        :OUT STD_LOGIC;
19     fan_died           :OUT STD_LOGIC
20 );
21 END ctrl_fans;
22
23 ARCHITECTURE behav OF ctrl_fans IS
24     SIGNAL fan_nxt_cel, fan_dir_cel, fan_esq_cel, fan_cim_cel, fan_bai_cel: t_fans_blk_sym;
25     SIGNAL fan_tempo: t_fans_times;
26     SIGNAL fan_rstn_tempo: t_fans_bits;
27     SIGNAL pr_fan_state: t_fans_states;
28 BEGIN
29     --Calcula possíveis parâmetros envolvidos no próximo movimento
30     --de todos os fantasmas
31     -- type: combinational
32     PROCESS (fan_area, fan_cur_dir)
33     BEGIN
34         FOR i in 0 to FAN_NO-1 LOOP
35             fan_nxt_cel(i) <= fan_area(i)(DIRS(fan_cur_dir(i))(0), DIRS(fan_cur_dir(i))(1));
36
37             IF (WALKABLE(fan_area(i)(DIRS(fan_cur_dir(i))(0), DIRS(fan_cur_dir(i))(1)))) THEN
38                 fan_dir_cel(i) <= fan_area(i)(DIRS(fan_cur_dir(i))(0), DIRS(fan_cur_dir(i))(1) + 1);
39                 fan_esq_cel(i) <= fan_area(i)(DIRS(fan_cur_dir(i))(0), DIRS(fan_cur_dir(i))(1) - 1);
40                 fan_cim_cel(i) <= fan_area(i)(DIRS(fan_cur_dir(i))(0) - 1, DIRS(fan_cur_dir(i))(1));
41                 fan_bai_cel(i) <= fan_area(i)(DIRS(fan_cur_dir(i))(0) + 1, DIRS(fan_cur_dir(i))(1));
42             ELSE
43                 fan_dir_cel(i) <= fan_area(i)(0,1);
44                 fan_esq_cel(i) <= fan_area(i)(0,-1);
45                 fan_cim_cel(i) <= fan_area(i)(-1,0);
46                 fan_bai_cel(i) <= fan_area(i)(1,0);
47             END IF;

```

```

48     END LOOP;
49 END PROCESS;
50
51 -- purpose: Este processo irá atualizar as posições dos fantasmas e definir
52 -- suas ações no jogo de acordo com seus estados.
53 -- type : sequential
54 -- inputs : clk27M, rstn, pac_area
55 --          fan_cur_dir, fan_pos_x, fan_pos_y
56 -- outputs: fan_cur_dir, fan_pos_x, fan_pos_y, got_coin
57 p_atualiza_fan: PROCESS (clk27M, rstn)
58 VARIABLE keys_dir_old, nxt_move: t_fans_dirs;
59 BEGIN
60     IF (rstn = '0') THEN
61         fan_pos_x <= FANS_START_X;
62         fan_pos_y <= FANS_START_Y;
63         fan_cur_dir <= (others => NADA);
64         nxt_move := (others => NADA);
65     ELSIF (clk27M'event and clk27M = '1') THEN
66         IF (atualiza = '1') THEN
67             FOR i in 0 to FAN_NO-1 LOOP
68                 CASE fan_state(i) IS
69                     WHEN ST_VIVO | ST_VULN | ST_VULN_BLINK =>
70                         IF (atua_en(1) = '1') THEN
71                             --Checa teclado para "agendar" um movimento
72                             IF (keys_dir(i) /= NADA and keys_dir_old(i) = NADA) THEN
73                                 nxt_move(i) := keys_dir(i);
74                             END IF;
75
76                             IF (nxt_move(i) = CIMA and WALKABLE(fan_cim_cel(i))) THEN
77                                 fan_cur_dir(i) <= CIMA;
78                                 nxt_move(i) := NADA;
79                             ELSIF (nxt_move(i) = DIREI and WALKABLE(fan_dir_cel(i))) THEN
80                                 fan_cur_dir(i) <= DIREI;
81                                 nxt_move(i) := NADA;
82                             ELSIF (nxt_move(i) = BAIXO and WALKABLE(fan_bai_cel(i))) THEN
83                                 fan_cur_dir(i) <= BAIXO;
84                                 nxt_move(i) := NADA;
85                             ELSIF (nxt_move(i) = ESQUE and WALKABLE(fan_esq_cel(i))) THEN
86                                 fan_cur_dir(i) <= ESQUE;
87                                 nxt_move(i) := NADA;
88                             END IF;
89
90                             IF (WALKABLE(fan_nxt_cel(i))) THEN --atualiza posicao
91                                 IF (fan_pos_x(i) = TELE_DIR_POS) then --teletransporte
92                                     fan_pos_x(i) <= TELE_ESQ_POS + 1;
93                                 ELSIF (fan_pos_x(i) = TELE_ESQ_POS) then
94                                     fan_pos_x(i) <= TELE_DIR_POS - 1;
95                                 ELSE
96                                     fan_pos_x(i) <= fan_pos_x(i) + DIRS(fan_cur_dir(i))(1);
97                                     fan_pos_y(i) <= fan_pos_y(i) + DIRS(fan_cur_dir(i))(0);
98                                 END IF;
99                             END IF;
100
101                             keys_dir_old(i) := keys_dir(i);
102                         END IF;
103                     WHEN ST_DEAD | ST_PRE_DEAD | ST_FIND_EXIT =>
104                         IF (atua_en(2) = '1') THEN
105                             -- Movimento automático do fantasma para a cela
106                             CASE FAN_PERCURSO(fan_pos_y(i), fan_pos_x(i)) IS
107                                 WHEN 'Q' =>
108                                     fan_pos_y(i) <= fan_pos_y(i) - 1;
109                                     fan_cur_dir(i) <= CIMA;
110                                 WHEN 'W' =>
111                                     fan_pos_y(i) <= fan_pos_y(i) + 1;
112                                     fan_cur_dir(i) <= BAIXO;
113                                 WHEN 'E' =>
114                                     fan_pos_x(i) <= fan_pos_x(i) - 1;
115                                     fan_cur_dir(i) <= ESQUE;
116                                 WHEN 'R' =>

```

```

117         fan_pos_x(i) <= fan_pos_x(i) + 1;
118         fan_cur_dir(i) <= DIREI;
119         WHEN OTHERS =>
120             END CASE;
121     END IF;
122     WHEN ST_FUGA => --Supõe que fan_pos_x já vale CELL_IN_X
123         IF (atua_en(1) = '1') THEN
124             fan_pos_y(i) <= fan_pos_y(i) - 1;
125             fan_cur_dir(i) <= CIMA;
126         END IF;
127     END CASE;
128 END LOOP;
129 END IF;
130 END IF;
131 END PROCESS p_atualiza_fan;
132
133 -- Gera o próximo estado de cada fantasma na atualização
134 -- type: combinational
135 p_fan_next_state: PROCESS (fan_state, spc_coin, pac_fans_hit, fan_tempo,
136                             fan_pos_x, fan_pos_y, atua_en)
137     VARIABLE pacman_dead_var, fan_died_var: STD_LOGIC;
138 BEGIN
139     pacman_dead_var := '0';
140     fan_died_var := '0';
141
142     FOR i in 0 to FAN_NO-1 LOOP
143         CASE fan_state(i) IS
144             WHEN ST_VIVO =>
145                 IF (pac_fans_hit(i) = '1') THEN
146                     pr_fan_state(i) <= ST_VIVO;
147                     pacman_dead_var := '1';
148                 ELSIF (spc_coin = '1') THEN
149                     pr_fan_state(i) <= ST_VULN;
150                 ELSE
151                     pr_fan_state(i) <= ST_VIVO;
152                 END IF;
153                 fan_rstn_tempo(i) <= '0';
154
155             WHEN ST_VULN =>
156                 IF (pac_fans_hit(i) = '1') THEN
157                     pr_fan_state(i) <= ST_PRE_DEAD;
158                     fan_died_var := '1';
159                 ELSIF (fan_tempo(i) > FAN_TIME_VULN_START_BLINK) THEN
160                     pr_fan_state(i) <= ST_VULN_BLINK;
161                 ELSE
162                     pr_fan_state(i) <= ST_VULN;
163                 END IF;
164                 fan_rstn_tempo(i) <= '1';
165
166             WHEN ST_VULN_BLINK =>
167                 IF (pac_fans_hit(i) = '1') THEN
168                     pr_fan_state(i) <= ST_PRE_DEAD;
169                     fan_died_var := '1';
170                 ELSIF (fan_tempo(i) > FAN_TIME_VULN_END) THEN
171                     pr_fan_state(i) <= ST_VIVO;
172                 ELSE
173                     pr_fan_state(i) <= ST_VULN_BLINK;
174                 END IF;
175                 fan_rstn_tempo(i) <= '1';
176
177             WHEN ST_PRE_DEAD => --apenas zera contador de tempo
178                 pr_fan_state(i) <= ST_DEAD;
179                 fan_rstn_tempo(i) <= '0';
180                 pacman_dead <= '0';
181
182             WHEN ST_DEAD =>
183                 IF (fan_tempo(i) > FAN_TIME_DEAD) THEN
184                     pr_fan_state(i) <= ST_FIND_EXIT;
185                 ELSE

```



```

186         pr_fan_state(i) <= ST_DEAD;
187     END IF;
188     fan_rstn_tempo(i) <= '1';
189
190     WHEN ST_FIND_EXIT =>
191         IF (fan_pos_x(i) = CELL_IN_X and fan_pos_y(i) = CELL_IN_Y) THEN
192             pr_fan_state(i) <= ST_FUGA;
193         ELSE
194             pr_fan_state(i) <= ST_FIND_EXIT;
195         END IF;
196         fan_rstn_tempo(i) <= '0';
197
198     WHEN ST_FUGA =>
199         IF (fan_pos_y(i) = CELL_OUT_Y) THEN
200             pr_fan_state(i) <= ST_VIVO;
201         ELSE
202             pr_fan_state(i) <= ST_FUGA;
203         END IF;
204         fan_rstn_tempo(i) <= '0';
205     END CASE;
206 END LOOP;
207
208 pacman_dead <= pacman_dead_var and atua_en(1);
209 fan_died <= fan_died_var;
210 END PROCESS p_fan_next_state;
211
212 -- Avança a FSM para o próximo estado
213 -- type: sequential
214 seq_fsm_fan: PROCESS (clk27M, rstn)
215 BEGIN
216     IF (rstn = '0') THEN -- asynchronous reset (active low)
217         fan_state <= (OTHERS => ST_FIND_EXIT);
218     ELSIF (clk27M'event and clk27M = '1') THEN
219         fan_state <= pr_fan_state;
220     END IF;
221 END PROCESS seq_fsm_fan;
222
223 -- Contadores de tempo para os fantasmas
224 fan_counters: PROCESS (clk27M, fan_rstn_tempo)
225 BEGIN
226     FOR i in 0 to FAN_NO-1 LOOP
227         IF (fan_rstn_tempo(i) = '0') THEN
228             fan_tempo(i) <= 0;
229         ELSIF (clk27M'event and clk27M = '1') THEN
230             IF (atualiza = '1') THEN
231                 fan_tempo(i) <= fan_tempo(i) + 1;
232             END IF;
233         END IF;
234     END LOOP;
235 END PROCESS fan_counters;
236 END behav;

```

Listing 9: Definicoes para pacman

```

1 --
2 -- decodifica tecla pressionada
3 -- em direcao
4 --
5
6 library ieee;
7 use ieee.std_logic_1164.all;
8 use ieee.numeric_std.all;
9 use work.pac_defs.all;
10
11 --entrada: codigo proveniente do teclado
12 --saida: mudancas de direcao do player
13 -- code entrada codificada das teclas

```

```

14 --(maximo tres teclas pressionadas 16 bits cada)
15 -- p1_dir - p2_dir representacao de cima,desce,esquerda,direita,nenhuma
16 entity player_dir is
17 port (
18     code: IN STD_LOGIC_VECTOR(47 downto 0);
19     p1_dir,p2_dir,p3_dir: OUT t_direcao
20 );
21 end entity player_dir;
22
23 architecture rtl of player_dir is
24     signal key_1,key_2,key_3 : std_logic_vector(15 downto 0);
25
26 begin
27     -- Modelo:
28     --> 2-players -> cada um aperta uma tecla
29     -- Codigo referente a cada tecla deve estar em key_1 ou key_2
30     --> Desconsiderarei key_3 -> implementacao futura mas sem grande utilidade.
31     -- Problemas referentes a entrada de teclas:
32     --> se um player aperta 3 teclas, o outro nao tera a sua teclalida
33     key_1<=code(47 downto 32);-- terceira tecla pressionada
34     key_2<=code(31 downto 16);-- segunda tecla pressionada
35     key_3<=code(15 downto 0); -- primeira tecla pressionada
36
37     -- P 1 Teclas
38     -- Movimentacao/ Tecla | codigo
39     -- Cima | (Numpad 8) | x"0075"
40     -- Baixo | (Numpad 5) | x"0073"
41     -- Esquerda | (Numpad 4) | x"006B"
42     -- Direita | (Numpad 6) | x"0074"
43
44     -- P 2 Teclas
45     -- Movimentacao/Tecla | codigo
46     -- Cima | (W) | x"001d"
47     -- Baixo | (S) | x"001b"
48     -- Esquerda | (A) | x"001c"
49     -- Direita | (D) | x"0023"
50
51
52     -- P 3 Teclas
53     -- Movimentacao/Tecla | codigo
54     -- Cima | (I) | x"0043"
55     -- Baixo | (K) | x"0042"
56     -- Esquerda | (J) | x"003B"
57     -- Direita | (L) | x"004B"
58
59     --Implementação mais simples para 2 players com 3 teclas
60     p1_dir <= CIMA WHEN (key_1 = x"0075" or key_2 = x"0075" or key_3 = x"0075")
61     ELSE DIREI WHEN (key_1 = x"0074" or key_2 = x"0074" or key_3 = x"0074")
62     ELSE BAIXO WHEN (key_1 = x"0073" or key_2 = x"0073" or key_3 = x"0073")
63     ELSE ESQUE WHEN (key_1 = x"006B" or key_2 = x"006B" or key_3 = x"006B")
64     ELSE NADA;
65
66     p2_dir <= CIMA WHEN (key_1 = x"001d" or key_2 = x"001d" or key_3 = x"001d")
67     ELSE DIREI WHEN (key_1 = x"0023" or key_2 = x"0023" or key_3 = x"0023")
68     ELSE BAIXO WHEN (key_1 = x"001b" or key_2 = x"001b" or key_3 = x"001b")
69     ELSE ESQUE WHEN (key_1 = x"001c" or key_2 = x"001c" or key_3 = x"001c")
70     ELSE NADA;
71
72     p3_dir <= CIMA WHEN (key_1 = x"0043" or key_2 = x"0043" or key_3 = x"0043")
73     ELSE DIREI WHEN (key_1 = x"004B" or key_2 = x"004B" or key_3 = x"004B")
74     ELSE BAIXO WHEN (key_1 = x"0042" or key_2 = x"0042" or key_3 = x"0042")
75     ELSE ESQUE WHEN (key_1 = x"003B" or key_2 = x"003B" or key_3 = x"003B")
76     ELSE NADA;
77 END rtl;

```

Conclusão