

Escola Politécnica da Universidade de São Paulo



Eduardo Thomaz dos Santos

NUSP: 11260961

Gabriel Pereira de Carvalho

NUSP: 11257668

Documentação Final

Projeto Reflexos

PCS 3635 - Laboratório de Circuitos Digitais 1

Professor: Edson Midorikawa

São Paulo

2021

1. Motivação do Projeto

Reflexos apurados e bons tempos de reação são essenciais para pilotos de Fórmula 1. Não apenas sendo um fator determinante em sua performance na corrida, mas também para sua segurança. Durante seu treinamento, os pilotos buscam tempos de reação pequenos e, principalmente, consistentes.



Imagem 1: Piloto treinando seus reflexos

O objetivo do projeto é medir o tempo de reação do jogador, exibindo a média dos tempos ao final de jogo e também a sequência de tempos com a função REPETIR.

2. Projeto Lógico das Extensões Propostas

Neste projeto de grupo, foram propostas **oito** modificações sobre o circuito base da semana 1.

- A cada jogada realizada, armazenar tempo de reação do jogador
- Modificar função REPETIR para exibir tempos de reação nos displays HEX
- Estender função REPETIR para o caso de vitória
- Adicionar um timer dedicado à função REPETIR
- Converter os tempos de reação de hexadecimal para BCD
- Adicionar tempo limite para o término do jogo
- Calcular média dos tempos de reação e exibir nos estados *TerminouCerto* e *TerminouErrado*
- Exibir mensagens durante o jogo para guiar experiência do usuário

Algumas considerações iniciais são importantes. Observe que nos estados *EsperaJogada* e *EsperaEscrita*, a transição para o estado seguinte no fluxo normal de funcionamento do circuito é disparada pela detecção de jogada através do *edge_detector*. Portanto, acionando o enable de um timer nesses dois estados de espera e desativando o enable após a transição, somos capazes de medir o tempo de reação do jogador como desejado.

Uma observação importante é que a medição é realizada apenas com a detecção de jogada. Portanto, no caso de *timeout* onde o jogador esgota o tempo limite de 5 segundos e perde antes de realizar sua jogada, esse tempo não é armazenado na RAM. Esse cenário será verificado posteriormente na fase de testes.

Vale observar que esses tempos são medidos em função de bordas de clock, portanto para serem interpretados da forma correta é necessário conhecer o período do clock utilizado. Neste projeto foi utilizada a frequência de 1kHz, portanto os tempos são medidos em milissegundos.

A função REPETIR funciona de forma análoga tanto para a vitória quanto para a derrota. Uma observação importante é que como serão exibidos os tempos de reação correspondentes a todas as jogadas realizadas na partida, os contadores utilizados para gerar os sinais de controle das transições foram modificados. Foram utilizados dois contadores: um para contar o número de jogadas realizadas e outro que percorre a memória RAM até atingir esse valor limite. Portanto foi necessário adicionar um *comparador8bits*, monitorando essa contagem.

Um desafio nessa fase final do projeto foi tentar propor novas funcionalidades dentro da limitação de pinos de entrada fornecidos pela FPGA. Uma vez que nosso objetivo é testar e treinar o tempo de reação do usuário, colocamos nosso foco em aumentar a dificuldade do jogo e fornecer uma estatística que possa ser usada para avaliar o desempenho do jogador ao final do jogo.

No nível *II* o jogador realiza um total de 152 jogadas, portanto somar os tempos de reação não era possível para exibição nos display HEX. A média no entanto pode ser exibida mantendo o mesmo número de displays HEX, o que motivou a ideia. Além disso, com a média podemos comparar partidas diferentes e até mesmo partidas jogadas em níveis diferentes.

O timer dedicado à função REPETIR foi proposto por razões práticas. Em versões anteriores, a função REPETIR utilizava o mesmo timer utilizado para indicar timeout na espera de jogada. No entanto, para exibir 152 jogadas do nível *II*, esse mecanismo leva aproximadamente 13 minutos, o que não é prático. Por isso, a função REPETIR utiliza um timer próprio com limite reduzido.

O algoritmo para conversão de binário para BCD funciona em duas etapas.

Primeiro o número é convertido para base 2 para base 2. Fazemos isso percorrendo os bits, um por vez, a partir do mais significativo. Ao processar um bit, estamos multiplicando nosso número por 2 e somando o novo bit. Processar cada bit requer um ciclo de clock, portanto o processo de conversão não é instantâneo. No nosso caso, são necessários dezesseis ciclos de clock.

Devido ao tempo necessário para conversão dos tempos de reação de hexadecimal para BCD antes da exibição no display, os tempos são exibidos apenas ao final do jogo nos estados *TerminouCerto*, *TerminouErrado* e durante a repetição. Tivemos problemas na prática tentando exibir os tempos durante o jogo como uma forma de feedback imediato.

Vale observar que as mudanças propostas requerem pequenas mudanças nas transições da máquina de estados, mas não houve acréscimo ou decréscimo de estados.

3. Implementação no Quartus

3.1 Novas componentes

3.1.1 timer Rep

A maior diferença entre este timer e o timer5k utilizado para timeout na espera de jogada são os sinais de controle, que justificam a utilização de duas componentes diferentes.

```

1  library IEEE;
2  use IEEE.std_logic_1164.all;
3  use IEEE.numeric_std.all;
4
5  entity timerRep is
6      port (
7          clock          : in  std_logic;
8          clr             : in  std_logic; -- ativo em alto
9          enable          : in  std_logic;
10         Q               : out std_logic_vector (13 downto 0);
11         timeout         : out std_logic
12     );
13 end entity;
14
15 architecture comportamental of timerRep is
16     signal IQ: integer range 0 to 1999;
17 begin
18     asyncreset: process (clr, clock,IQ,enable) -- async reset
19     begin
20         if clr = '1' then
21             IQ <= 0;
22         elsif clock'event and clock='1' then
23             if IQ=1999 then
24                 IQ <= 1999;
25             elsif enable='1' then
26                 IQ <= IQ + 1;
27             end if;
28         else
29             IQ <= IQ;
30         end if;
31     end process;
32
33     with IQ select
34         timeout <=  '1' when 1999,
35                    '0' when others;
36     Q <= std_logic_vector(to_unsigned(IQ, Q'length));
37 end architecture;

```

3.1.2 timer Total

A maior diferença entre este timer e os outros dois utilizados no circuito é o seu limite, que é muito maior e é variável em função do nível selecionado.

```

1  library IEEE;
2  use IEEE.std_logic_1164.all;
3  use IEEE.numeric_std.all;
4
5  entity timerTotal is
6      port (
7          clock          : in  std_logic;

```

```

8         nivel          : in std_logic_vector(1 downto 0);
9         clr             : in std_logic; -- ativo em alto
10        enable          : in std_logic;
11        timeout         : out std_logic
12    );
13 end entity;
14
15 architecture comportamental of timerTotal is
16     signal lim, IQ: integer range 0 to 610000; --limite do nivel selecionado
17 begin
18
19     with nivel select
20         lim <= 35000 when "00", --total de 14 jogadas
21             110000 when "01", --total de 44 jogadas
22             225000 when "10", --total de 90 jogadas
23             380000 when "11"; --total de 152 jogadas
24
25     asyncretaset: process (clr, clock,IQ,enable) -- async reset
26     begin
27         if clr = '1' then
28             IQ <= 0;
29         elsif clock'event and clock='1' then
30             if IQ=lim then
31                 IQ <= lim;
32             elsif enable='1' then
33                 IQ <= IQ + 1;
34             end if;
35         else
36             IQ <= IQ;
37         end if;
38         if(IQ = lim) then
39             timeout <= '1';
40         else
41             timeout <= '0';
42         end if;
43     end process;
44
45 end architecture;

```

3.1.3 calculaMedia

Os sinais de controle desta componente podem ser reaproveitados da RAM_tempos, que armazena os tempos de reação do jogador. Afinal, a média é atualizada quando a RAM_tempos escreve um novo tempo de reação.

```

1  library IEEE;
2  use IEEE.std_logic_1164.all;
3  use IEEE.numeric_std.all;
4

```

```

5  entity calcula_media is
6      port (
7          clock          : in  std_logic;
8          clr             : in  std_logic; -- ativo em alto
9          enable          : in  std_logic; -- ativo em alto
10         D               : in  std_logic_vector (13 downto 0); --
11         tempo de reacao da mais nova jogada
12         Q               : out std_logic_vector (13 downto 0) --media das
13         entradas fornecidas
14     );
15 end entity;
16
17 architecture comportamental of calcula_media is
18     signal nova_jogada, media: integer range 0 to 4999;
19     signal soma: integer range 0 to 800000;
20     signal num_jogadas: integer range 0 to 152;
21 begin
22
23     nova_jogada <= to_integer(unsigned(D));
24
25     asyncreset: process (clr, clock, enable) -- async reset
26     begin
27         if clr = '1' then
28             soma <= 0;
29             num_jogadas <= 0;
30             media <= 0;
31         elsif clock'event and clock='1' and enable='1' then
32             soma <= soma + nova_jogada;
33             num_jogadas <= num_jogadas + 1;
34             media <= soma/num_jogadas;
35         else
36             soma <= soma;
37             num_jogadas <= num_jogadas;
38             media <= media;
39         end if;
40     end process;
41
42     Q <= std_logic_vector(to_unsigned(media, Q'length));
43 end architecture;

```

3.1.4 conversão hexadecimal para BCD

O código utilizado nesta seção não é de autoria da dupla, ele é utilizado respeitando direitos autorais. O site fonte é citado nas referências deste documento.

```

1  -----
2  -----
3  --
4  --   FileName:          binary_to_bcd_digit.vhd
5  --   Dependencies:      none

```

```

6  -- Design Software:  Quartus II 64-bit Version 13.1.0 Build 162 SJ Web
7  Edition
8  --
9  -- HDL CODE IS PROVIDED "AS IS."  DIGI-KEY EXPRESSLY DISCLAIMS ANY
10 -- WARRANTY OF ANY KIND, WHETHER EXPRESS OR IMPLIED, INCLUDING BUT NOT
11 -- LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
12 -- PARTICULAR PURPOSE, OR NON-INFRINGEMENT.  IN NO EVENT SHALL DIGI-KEY
13 -- BE LIABLE FOR ANY INCIDENTAL, SPECIAL, INDIRECT OR CONSEQUENTIAL
14 -- DAMAGES, LOST PROFITS OR LOST DATA, HARM TO YOUR EQUIPMENT, COST OF
15 -- PROCUREMENT OF SUBSTITUTE GOODS, TECHNOLOGY OR SERVICES, ANY CLAIMS
16 -- BY THIRD PARTIES (INCLUDING BUT NOT LIMITED TO ANY DEFENSE THEREOF),
17 -- ANY CLAIMS FOR INDEMNITY OR CONTRIBUTION, OR OTHER SIMILAR COSTS.
18 --
19 -- Version History
20 -- Version 1.0 6/15/2017 Scott Larson
21 -- Initial Public Release
22 --
23 -----
24 -----
25
26 LIBRARY ieee;
27 USE ieee.std_logic_1164.all;
28
29 ENTITY binary_to_bcd_digit IS
30     PORT(
31         clk          :    IN          STD_LOGIC;
32                     --system clock
33         reset_n      :    IN          STD_LOGIC;
34                     --active low asynchronous reset
35         ena          :    IN          STD_LOGIC;
36                     --activate operation
37         binary       :    IN          STD_LOGIC;
38                     --bit shifted into digit
39         c_out        :    BUFFER STD_LOGIC;
40                     --carry out shifted to next larger digit
41         bcd          :    BUFFER STD_LOGIC_VECTOR(3 DOWNTO 0));
42 --resulting BCD output
43 END binary_to_bcd_digit;
44
45 ARCHITECTURE logic OF binary_to_bcd_digit IS
46     SIGNAL prev_ena    :    STD_LOGIC;  --keeps track of the previous
47     enable to identify when enable is first asserted
48 BEGIN
49
50     c_out <= bcd(3) OR (bcd(2) AND bcd(1)) OR (bcd(2) AND bcd(0));
51 --assert carry out when register value exceeds 4
52
53     PROCESS(reset_n, clk)
54     BEGIN
55         IF(reset_n = '0') THEN
56             --asynchronous reset asserted

```

```

57         prev_ena <= '0';
58         --clear ena history
59         bcd <= "0000";
60         --clear output
61         ELSIF(clk'EVENT AND clk = '1') THEN           --rising
62 edge of system clock
63         prev_ena <= ena;
64         --keep track of last enable
65         IF(ena = '1') THEN
66         --operation activated
67             IF(prev_ena = '0') THEN
68             --first cycle of activation
69                 bcd <= "0000";
70                 --initialize the register
71             ELSIF(c_out = '1') THEN
72             --register value exceeds 4
73                 bcd(0) <= binary;
74                 --shift new bit into first register
75                 bcd(1) <= NOT bcd(0);
76                 --set second register to adjusted value
77                 bcd(2) <= NOT (bcd(1) XOR bcd(0));
78             --set third register to adjusted value
79                 bcd(3) <= bcd(3) AND bcd(0);
80             --set fourth register to adjusted value
81             ELSE
82                 --register value does not exceed 4
83                 bcd <= bcd(2 DOWNT0 0) & binary;
84             --shift register values up and shift in new bit
85             END IF;
86         END IF;
87     END IF;
88 END PROCESS;
89
90 END logic;

```

```

1  -----
2  -----
3  --
4  --   FileName:      binary_to_bcd.vhd
5  --   Dependencies:  binary_to_bcd_digit.vhd
6  --   Design Software: Quartus II 64-bit Version 13.1.0 Build 162 SJ Web
7  Edition
8  --
9  --   HDL CODE IS PROVIDED "AS IS." DIGI-KEY EXPRESSLY DISCLAIMS ANY
10 --   WARRANTY OF ANY KIND, WHETHER EXPRESS OR IMPLIED, INCLUDING BUT NOT
11 --   LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
12 --   PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL DIGI-KEY
13 --   BE LIABLE FOR ANY INCIDENTAL, SPECIAL, INDIRECT OR CONSEQUENTIAL
14 --   DAMAGES, LOST PROFITS OR LOST DATA, HARM TO YOUR EQUIPMENT, COST OF
15 --   PROCUREMENT OF SUBSTITUTE GOODS, TECHNOLOGY OR SERVICES, ANY CLAIMS

```



```

16  -- BY THIRD PARTIES (INCLUDING BUT NOT LIMITED TO ANY DEFENSE THEREOF),
17  -- ANY CLAIMS FOR INDEMNITY OR CONTRIBUTION, OR OTHER SIMILAR COSTS.
18  --
19  -- Version History
20  -- Version 1.0 6/15/2017 Scott Larson
21  --   Initial Public Release
22  -- Version 1.1 6/23/2017 Scott Larson
23  --   Fixed small corner-case bug
24  -- Version 1.2 1/16/2018 Scott Larson
25  --   Fixed reset logic to include resetting the state machine
26  --
27  -----
28  -----
29
30  LIBRARY ieee;
31  USE ieee.std_logic_1164.all;
32
33  ENTITY binary_to_bcd IS
34      GENERIC(
35          bits          :    INTEGER := 10;           --size of the
36          binary input numbers in bits
37          digits :      INTEGER := 3);              --number of BCD
38          digits to convert to
39      PORT(
40          clk          :    IN          STD_LOGIC;
41                                     --system clock
42          reset_n      :    IN          STD_LOGIC;
43                                     --active low
44          asynchronous reset
45          ena          :    IN          STD_LOGIC;
46                                     --latches in new
47          binary number and starts conversion
48          binary :      IN          STD_LOGIC_VECTOR(bits-1 DOWNT0 0);
49                                     --binary number to convert
50          busy        :    OUT        STD_LOGIC;
51                                     --indicates conversion in
52          progress
53          bcd         :    OUT        STD_LOGIC_VECTOR(digits*4-1 DOWNT0
54          0)); --resulting BCD number
55  END binary_to_bcd;
56
57  ARCHITECTURE logic OF binary_to_bcd IS
58      TYPE          machine IS(idle, convert);
59                                     --needed states
60      SIGNAL state          :    machine;
61                                     --state machine
62      SIGNAL binary_reg      :    STD_LOGIC_VECTOR(bits-1
63      DOWNT0 0); --latched in binary number
64      SIGNAL bcd_reg          :
65      STD_LOGIC_VECTOR(digits*4-1 DOWNT0 0); --bcd result register
66      SIGNAL converter_ena    :    STD_LOGIC;

```

```

67     SIGNAL converter_inputs    :    STD_LOGIC_VECTOR(digits DOWNTO 0);
68         --inputs into each BCD single digit converter
69
70     --binary to BCD single digit converter component
71     COMPONENT binary_to_bcd_digit IS
72         PORT(
73             clk            :    IN                STD_LOGIC;
74             reset_n        :    IN                STD_LOGIC;
75             ena            :    IN                STD_LOGIC;
76             binary :        IN                STD_LOGIC;
77             c_out          :    BUFFER STD_LOGIC;
78             bcd            :    BUFFER STD_LOGIC_VECTOR(3 DOWNTO
79 0));
80     END COMPONENT binary_to_bcd_digit;
81
82 BEGIN
83
84     PROCESS(reset_n, clk)
85         VARIABLE bit_count :    INTEGER RANGE 0 TO bits+1 := 0;
86         --counts the binary bits shifted into the converters
87         BEGIN
88             IF(reset_n = '0') THEN
89                 --asynchronous reset asserted
90                 bit_count := 0;
91                 --reset bit counter
92                 busy <= '1';
93                 --indicate not available
94                 converter_ena <= '0';
95                 --disable the converter
96                 bcd <= (OTHERS => '0');
97                 --clear BCD result port
98                 state <= idle;
99                 --reset state machine
100             ELSIF(clk'EVENT AND clk = '1') THEN                --system clock
101 rising edge
102                 CASE state IS
103
104                     WHEN idle =>
105                         --idle state
106                         IF(ena = '1') THEN
107                             --converter is enabled
108                             busy <= '1';
109                             --indicate conversion in progress
110                             converter_ena <= '1';
111                             --enable the converter
112                             binary_reg <= binary;
113                             --latch in binary number for conversion
114                             bit_count := 0;
115                             --reset bit counter
116                             state <= convert;
117                             --go to convert state

```

```

118             ELSE
119                 --converter is not enabled
120                 busy <= '0';
121                 --indicate available
122                 converter_ena <= '0';
123                 --disable the converter
124                 state <= idle;
125                 --remain in idle state
126                 END IF;
127
128             WHEN convert =>
129                 --convert state
130                 IF(bit_count < bits+1) THEN
131                     --not all bits shifted in
132                     bit_count := bit_count + 1;
133                     --increment bit
134 counter
135                     converter_inputs(0) <=
136 binary_reg(bits-1);
137                     --shift next bit into converter
138                     binary_reg <= binary_reg(bits-2
139 DOWNT0 0) & '0';
140                     --shift binary number register
141                     state <= convert;
142                     --remain
143 in convert state
144             ELSE
145                 --all bits shifted in
146                 busy <= '0';
147                 --indicate conversion is complete
148                 converter_ena <= '0';
149                 --disable the
150 converter
151                 bcd <= bcd_reg;
152                 --output
153 result
154                 state <= idle;
155
156                 --return to idle state
157                 END IF;
158
159             END CASE;
160         END IF;
161     END PROCESS;
162
163     --instantiate the converter logic for the specified number of
164 digits
165     bcd_digits: FOR i IN 1 to digits GENERATE
166         digit_0: binary_to_bcd_digit
167             PORT MAP (clk, reset_n, converter_ena,
168 converter_inputs(i-1), converter_inputs(i), bcd_reg(i*4-1 DOWNT0 i*4-4));

```

```

169         END GENERATE;
170
171     END logic;

```

Uma preocupação inicial da dupla era o delay necessário para conversão, no entanto ao realizar testes vimos que a resposta era praticamente instantânea e a conversão pode ser usada na prática sem problemas.

3.1.5 comparador_8bits

Como a memória RAM, tempos e vetores no geral utilizam indexação com base zero, note que a condição de parada da função REPETIR não é quando o contador atinge o número de jogadas realizadas, mas sim quando ele atinge (número de jogadas realizadas - 1).

Essa consideração é fundamental para o comparador_8bits e ressalta a importância da ordem das entradas durante a atribuição.

```

1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  entity comparador_8bits is
6      port (
7          --numero binario A de 8 bits
8          i_A : in std_logic_vector(7 downto 0);
9
10         --numero binario B de 8 bits
11         i_B : in std_logic_vector(7 downto 0);
12
13         --saidas
14         --representam conclusoes realizadas com o circuito
15         iguais : out std_logic
16     );
17 end entity comparador_8bits;
18
19 architecture dataflow of comparador_8bits is
20
21     signal A, B: integer range 0 to 4999;
22
23     begin
24         A <= to_integer(unsigned(i_A));
25         B <= to_integer(unsigned(i_B));
26         process(i_A, i_B)
27         begin
28             if( A+1 = B)
29                 then iguais<='1';
30             else iguais<='0';
31             end if;
32         end process;
33
34
35     end architecture dataflow;

```

3.1.6 contador_tempos

O contador_tempos é bastante semelhante a implementação do contador_163 fornecida. No entanto como os limites associados a RAM_tempos são diferentes, pequenos detalhes tiveram de ser ajustados.

```
1  library IEEE;
2  use IEEE.std_logic_1164.all;
3  use IEEE.numeric_std.all;
4
5  entity contador_tempos is
6      port (
7          clock : in  std_logic;
8          clr    : in  std_logic;
9          ld     : in  std_logic;
10         ent    : in  std_logic;
11         enp    : in  std_logic;
12         D      : in  std_logic_vector (7 downto 0);
13         limite: in  std_logic_vector (7 downto 0);
14         Q      : out std_logic_vector (7 downto 0);
15         rco    : out std_logic
16     );
17 end contador_tempos;
18
19 architecture comportamental of contador_tempos is
20     signal IQ: integer range 0 to 255;
21     signal int_limite : integer range 0 to 255;
22 begin
23
24     -- convertemos para integer para realizar comparacao com IQ
25     int_limite <= to_integer(unsigned(limite));
26
27     process (clock,IQ,ent,int_limite)
28     begin
29
30         if clock'event and clock='1' then
31             if clr='0' then    IQ <= 0;
32             elsif ld='0' then IQ <= to_integer(unsigned(D));
33             elsif ent='1' and enp='1' then
34                 --paramos de contar quando atingimos limite para deixar rco
35                 ativado
36                 if IQ=int_limite then    IQ <= int_limite;
37                 else                      IQ <= IQ + 1;
38                 end if;
39             else                      IQ <= IQ;
40             end if;
41         end if;
42         -- assim que atingimos limite atual, encerramos contagem
43         if IQ=int_limite then rco <= '1';
44         else                  rco <= '0';
```

```

45     end if;
46
47     Q <= std_logic_vector(to_unsigned(IQ, Q'length));
48
49     end process;
50 end comportamental;

```

3.1.7 ram_tempos

Vale observar que no arquivo .mif, a RAM_tempos é inicializada com zero em todas as posições. O que é condizente com o estado obtido após o clear de cada execução. Como os tempos são obtidos a partir da entidade *timer_5k*, os vetores de entrada e saída da RAM_tempos utilizam limites condizentes com o timer.

```

1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  entity ram_tempos is
6      port (
7          clk          : in  std_logic;
8          endereco     : in  std_logic_vector(7 downto 0);
9          dado_entrada : in  std_logic_vector(13 downto 0); --tempo de reacao
10 do jogador
11         we           : in  std_logic;
12         ce           : in  std_logic;
13         dado_saida   : out std_logic_vector(13 downto 0)
14     );
15 end entity ram_tempos;
16
17 architecture ram_mif of ram_tempos is
18     type arranjo_memoria is array(0 to 255) of std_logic_vector(13 downto
19 0);
20     signal memoria : arranjo_memoria;
21
22     -- Configuracao do Arquivo MIF
23     attribute ram_init_file: string;
24     attribute ram_init_file of memoria: signal is "ram_tempos_inicial.mif";
25
26 begin
27
28     process(clk)
29     begin
30         if (clk = '1' and clk'event) then
31             if ce = '0' then -- dado armazenado na subida de "we" com "ce=0"
32
33                 -- Detecta ativacao de we (ativo baixo)
34                 if (we = '0')
35                     then memoria(to_integer(unsigned(endereco))) <=
36 dado_entrada;
37             end if;

```

[illegible]

89		"00000000000000",
90		"00000000000000",
91		"00000000000000",
92		"00000000000000",
93		"00000000000000",
94		"00000000000000",
95		"00000000000000",
96		"00000000000000",
97		"00000000000000",
98		"00000000000000",
99		"00000000000000",
100		"00000000000000",
101		"00000000000000",
102		
103	"00000000000000",	
104		"00000000000000",
105		"00000000000000",
106		"00000000000000",
107		"00000000000000",
108		"00000000000000",
109		"00000000000000",
110		"00000000000000",
111		"00000000000000",
112		"00000000000000",
113		"00000000000000",
114		"00000000000000",
115		"00000000000000",
116		"00000000000000",
117		"00000000000000",
118		"00000000000000",
119		
120	"00000000000000",	
121		"00000000000000",
122		"00000000000000",
123		"00000000000000",
124		"00000000000000",
125		"00000000000000",
126		"00000000000000",
127		"00000000000000",
128		"00000000000000",
129		"00000000000000",
130		"00000000000000",
131		"00000000000000",
132		"00000000000000",
133		"00000000000000",
134		"00000000000000",
135		"00000000000000",
136		
137	"00000000000000",	
138		"00000000000000",
139		"00000000000000",


```
140 "000000000000000",
141 "000000000000000",
142 "000000000000000",
143 "000000000000000",
144 "000000000000000",
145 "000000000000000",
146 "000000000000000",
147 "000000000000000",
148 "000000000000000",
149 "000000000000000",
150 "000000000000000",
151 "000000000000000",
152 "000000000000000",
153
154 "000000000000000",
155 "000000000000000",
156 "000000000000000",
157 "000000000000000",
158 "000000000000000",
159 "000000000000000",
160 "000000000000000",
161 "000000000000000",
162 "000000000000000",
163 "000000000000000",
164 "000000000000000",
165 "000000000000000",
166 "000000000000000",
167 "000000000000000",
168 "000000000000000",
169 "000000000000000",
170
171 "000000000000000",
172 "000000000000000",
173 "000000000000000",
174 "000000000000000",
175 "000000000000000",
176 "000000000000000",
177 "000000000000000",
178 "000000000000000",
179 "000000000000000",
180 "000000000000000",
181 "000000000000000",
182 "000000000000000",
183 "000000000000000",
184 "000000000000000",
185 "000000000000000",
186 "000000000000000",
189
190 "000000000000000",
191 "000000000000000",
192 "000000000000000",
```

193		"00000000000000",
194		"00000000000000",
195		"00000000000000",
196		"00000000000000",
197		"00000000000000",
198		"00000000000000",
199		"00000000000000",
200		"00000000000000",
201		"00000000000000",
202		"00000000000000",
203		"00000000000000",
204		"00000000000000",
205		"00000000000000",
206		
207	"00000000000000",	
208		"00000000000000",
209		"00000000000000",
210		"00000000000000",
211		"00000000000000",
212		"00000000000000",
213		"00000000000000",
214		"00000000000000",
215		"00000000000000",
216		"00000000000000",
217		"00000000000000",
218		"00000000000000",
219		"00000000000000",
220		"00000000000000",
221		"00000000000000",
222		"00000000000000",
223		
224	"00000000000000",	
225		"00000000000000",
226		"00000000000000",
227		"00000000000000",
228		"00000000000000",
229		"00000000000000",
230		"00000000000000",
231		"00000000000000",
232		"00000000000000",
233		"00000000000000",
234		"00000000000000",
235		"00000000000000",
236		"00000000000000",
237		"00000000000000",
238		"00000000000000",
239		"00000000000000",
240		
241	"00000000000000",	
242		"00000000000000",
243		"00000000000000",

244		"00000000000000",
245		"00000000000000",
246		"00000000000000",
247		"00000000000000",
248		"00000000000000",
249		"00000000000000",
250		"00000000000000",
251		"00000000000000",
252		"00000000000000",
253		"00000000000000",
254		"00000000000000",
255		"00000000000000",
256		"00000000000000",
257		
258	"00000000000000",	
259		"00000000000000",
260		"00000000000000",
261		"00000000000000",
262		"00000000000000",
263		"00000000000000",
264		"00000000000000",
265		"00000000000000",
266		"00000000000000",
267		"00000000000000",
268		"00000000000000",
269		"00000000000000",
270		"00000000000000",
271		"00000000000000",
272		"00000000000000",
273		"00000000000000",
274		
275	"00000000000000",	
276		"00000000000000",
277		"00000000000000",
278		"00000000000000",
279		"00000000000000",
280		"00000000000000",
281		"00000000000000",
282		"00000000000000",
283		"00000000000000",
284		"00000000000000",
285		"00000000000000",
286		"00000000000000",
287		"00000000000000",
288		"00000000000000",
289		"00000000000000",
290		"00000000000000",
291		
292	"00000000000000",	
293		"00000000000000",
294		"00000000000000",

```

295         "00000000000000",
296         "00000000000000",
297         "00000000000000",
298         "00000000000000",
299         "00000000000000",
300         "00000000000000",
301         "00000000000000",
302         "00000000000000",
303         "00000000000000",
304         "00000000000000",
305         "00000000000000",
306         "00000000000000",
307         "00000000000000",
308
309         "00000000000000",
310         "00000000000000",
311         "00000000000000",
312         "00000000000000",
313         "00000000000000",
314         "00000000000000",
315         "00000000000000",
316         "00000000000000",
317         "00000000000000",
318         "00000000000000",
319         "00000000000000",
320         "00000000000000",
321         "00000000000000",
322         "00000000000000",
323         "00000000000000",
324         "00000000000000");
325
326 begin
327
328     process(clk)
329     begin
330         if (clk = '1' and clk'event) then
331             if ce = '0' then -- dado armazenado na subida de "we" com "ce=0"
332                 -- Detecta ativacao de we (ativo baixo)
333                 if (we = '0')
334                     then memoria(to_integer(unsigned(endereco))) <=
335 dado_entrada;
336                 end if;
337
338             end if;
339         end if;
340     end process;
341
342     -- saida da memoria
343     dado_saida <= memoria(to_integer(unsigned(endereco)));
344
345 end architecture ram_modelsim;

```

3.2 Fluxo de dados

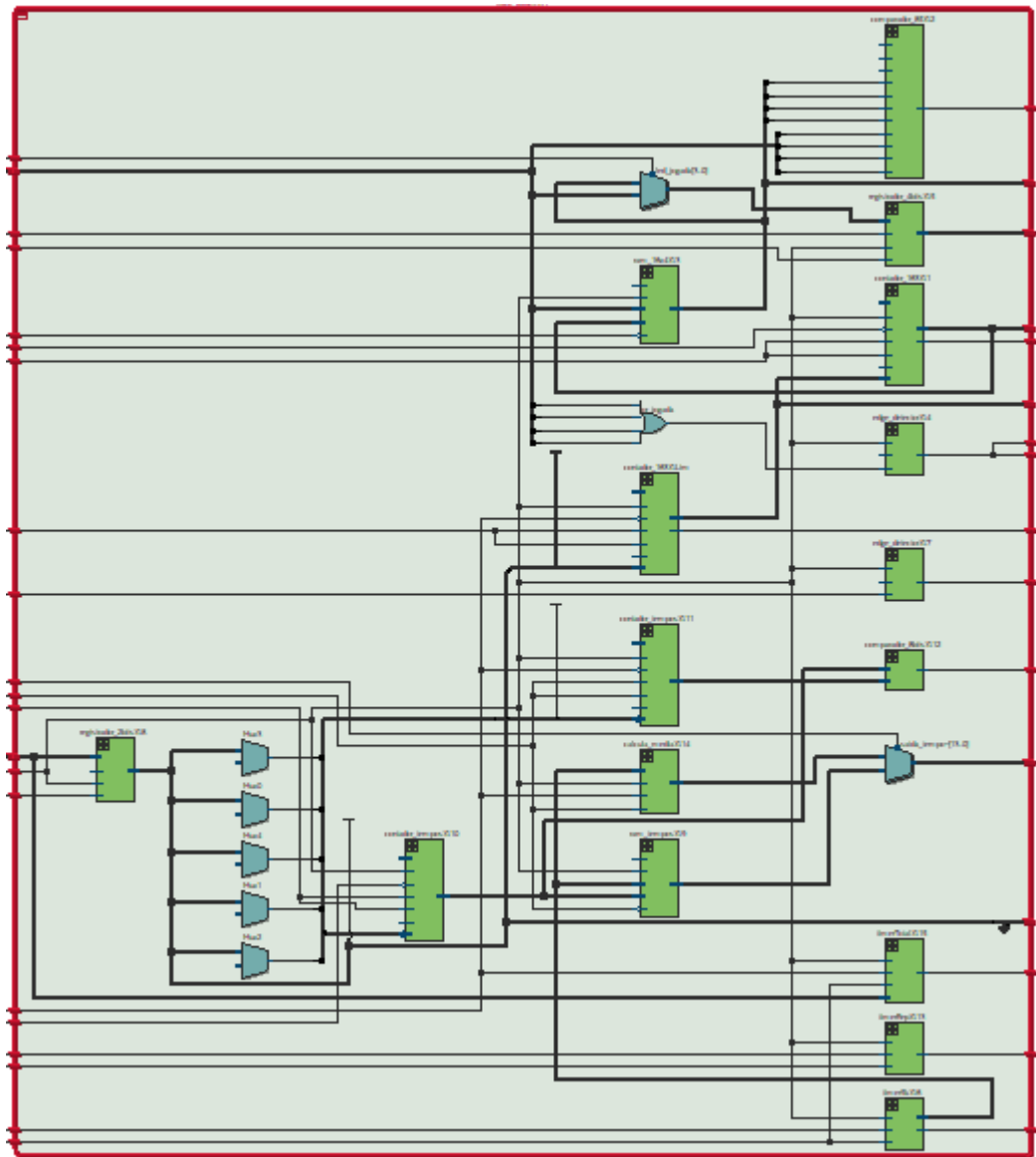


Imagem 2: Diagrama do fluxo de Dados obtido no RTL Viewer

A estratégia de construção incremental do circuito utilizada na disciplina foi muito valiosa especialmente no fluxo de dados. A cada semana foi construída uma base sólida para a edificação da semana seguinte. O diagrama atual é bastante complexo e de muito difícil leitura, mas conhecendo bem a base lançada nas primeiras sete semanas fomos capazes de navegar pelo circuito adicionando nossos novos componentes.

Observe entre as linhas 310 e 313 que o limite dos contador de tempos responsável por percorrer a RAM é calculado em função do nível de jogo selecionado. Vale observar que no último nível, composto por 16 jogadas, o cenário de vitória é composto por 152 jogadas. Logo a capacidade da RAM tempos deve ser maior ou igual a 152.

Outra boa observação é que os dois contadores associados às modificações do tempo de reação podem ser diferenciados pelos seus sinais de enable. O sinal *escreve_tempo* é usado no contador que congela após o final do jogo, armazenando o total de jogadas realizadas. Já o sinal *conta_jogadas* é ativado também durante a função REPETIR.

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity fluxo_dados is
5      port(
6          clock:                in std_logic;
7          zerac:                 in std_logic;
8          zeraLim:               in std_logic;
9          contac:                in std_logic;
10         escrevem:              in std_logic;
11         repetir:               in std_logic;
12         nivel:                  in std_logic_vector(1 downto 0);
13         enable_nivel:          in std_logic;
14         chaves:                 in std_logic_vector(3 downto 0);
15         escreve_tempo:          in std_logic;
16         enable_rep:             in std_logic; --NOVIDADE
17         reset_rep:              in std_logic; --NOVIDADE
18         zeraR:                  in std_logic;
19         enableR:                in std_logic;
20         incrementalLimite: in std_logic;
21         treset:                 in std_logic;
22         tenable:                in std_logic;
23         pronto_UC:              in std_logic;
24         zera_tempo:              in std_logic;
25         conta_jogadas:          in std_logic;
26         mostra_media:          in std_logic; --NOVIDADE
27         timeout:                 out std_logic;
28         timeout_rep:            out std_logic; --NOVIDADE
29         timeout_total:          out std_logic; --NOVIDADE
30         limiteMaximo:           out std_logic;
31         igual:                  out std_logic;
32         fimc:                   out std_logic;
33         db_tem_jogada:          out std_logic;
34         jogada_feita:           out std_logic;
35         repetir_edge:           out std_logic;
36         acabou_repetir:         out std_logic;
37         saida_tempo:            out std_logic_vector(13 downto 0);
38         db_contagem:            out std_logic_vector(3 downto 0);
39         db_memoria:             out std_logic_vector(3 downto 0);
40         db_limite:              out std_logic_vector(3 downto 0);
41         db_nivel:               out std_logic_vector(3 downto 0);
42         db_jogada:              out std_logic_vector(3 downto 0)
43     );
44 end entity;
45
```

```

46     architecture dataflow of fluxo_dados is
47     component contador_163
48     port (
49         clock : in  std_logic;
50         clr   : in  std_logic;
51         ld    : in  std_logic;
52         ent   : in  std_logic;
53         enp   : in  std_logic;
54         D     : in  std_logic_vector (3 downto 0);
55             limite: in  std_logic_vector (3 downto 0);
56         Q     : out std_logic_vector (3 downto 0);
57         rco   : out std_logic
58     );
59 end component;
60
61 component comparador_85
62 port (
63     i_A3 : in  std_logic;
64     i_B3 : in  std_logic;
65     i_A2 : in  std_logic;
66     i_B2 : in  std_logic;
67     i_A1 : in  std_logic;
68     i_B1 : in  std_logic;
69     i_A0 : in  std_logic;
70     i_B0 : in  std_logic;
71     i_AGTB : in  std_logic;
72     i_ALTB : in  std_logic;
73     i_AEQB : in  std_logic;
74     o_AGTB : out std_logic;
75     o_ALTB : out std_logic;
76     o_AEQB : out std_logic
77 );
78 end component;
79
80 component ram_16x4
81 port (
82     clk : in  std_logic;
83     endereco : in  std_logic_vector(3 downto 0);
84     dado_entrada : in  std_logic_vector(3 downto 0);
85     we : in  std_logic;
86     ce : in  std_logic;
87     dado_saida : out std_logic_vector(3 downto 0)
88 );
89 end component;
90
91 component registrador_4bits
92 port
93 (
94     clock : in std_logic;
95     clear : in std_logic;
96     enable : in std_logic;

```

```

97         D : in std_logic_vector(3 downto 0);
98         Q : out std_logic_vector(3 downto 0)
99     );
100 end component;
101
102 component edge_detector
103     port
104     (
105         clock : in std_logic;
106         reset : in std_logic;
107         sinal : in std_logic;
108         pulso : out std_logic
109     );
110 end component;
111
112 component timer5k is
113     port
114     (
115         clock      : in std_logic;
116         clr         : in std_logic; -- ativo em alto
117         enable      : in std_logic;
118         Q           : out std_logic_vector (13 downto 0);
119         timeout     : out std_logic
120     );
121 end component;
122
123 component timerRep
124     port (
125         clock      : in std_logic;
126         clr         : in std_logic; -- ativo em alto
127         enable      : in std_logic;
128         Q           : out std_logic_vector (13 downto 0);
129         timeout     : out std_logic
130     );
131 end component;
132
133 component registrador_2bits
134     port (
135         clock: in std_logic;
136         clear: in std_logic;
137         enable: in std_logic;
138         D: in std_logic_vector(1 downto 0);
139         Q: out std_logic_vector(1 downto 0)
140     );
141 end component;
142
143 component ram_tempos
144     port (
145         clk      : in std_logic;
146         endereco : in std_logic_vector(7 downto 0);
147         dado_entrada : in std_logic_vector(13 downto 0); --tempo de reacao

```



```

148     do jogador
149         we          : in  std_logic;
150         ce          : in  std_logic;
151         dado_saida  : out std_logic_vector(13 downto 0)
152     );
153 end component;
154
155 component contador_tempos
156     port (
157         clock : in  std_logic;
158         clr   : in  std_logic;
159         ld    : in  std_logic;
160         ent   : in  std_logic;
161         enp   : in  std_logic;
162         D     : in  std_logic_vector (7 downto 0);
163         limite: in  std_logic_vector (7 downto 0);
164         Q     : out std_logic_vector (7 downto 0);
165         rco   : out std_logic
166     );
167 end component;
168
169 component comparador_8bits
170     port (
171         --numero binario A de 14 bits
172         i_A : in std_logic_vector(7 downto 0);
173
174         --numero binario B de 14 bits
175         i_B : in std_logic_vector(7 downto 0);
176
177         --saidas
178         --representam conclusoes realizadas com o circuito
179         iguais : out std_logic
180     );
181 end component;
182
183 component calcula_media
184     port (
185         clock      : in  std_logic;
186         clr        : in  std_logic; -- ativo em alto
187         enable     : in  std_logic; -- ativo em alto
188         D          : in  std_logic_vector (13 downto 0); --
189         tempo de reacao da mais nova jogada
190         Q          : out std_logic_vector (13 downto 0) --media
191         das entradas fornecidas
192     );
193 end component;
194
195 component timerTotal
196     port (
197         clock      : in  std_logic;
198         nivel      : in  std_logic_vector(1 downto 0);

```

```

201         clr                : in std_logic; -- ativo em alto
202         enable              : in std_logic;
203         timeout             : out std_logic
204     );
205 end component;
206
207 signal s_endereco, led_jogada : std_logic_vector(3 downto 0);
208 signal s_dados: std_logic_vector(3 downto 0);
209 signal or_jogada : std_logic;
210 signal nzerac : std_logic;
211 signal nzeralim: std_logic;
212 signal limite, valor_nivel_4bits : std_logic_vector (3 downto 0);
213 signal valor_nivel_2bits : std_logic_vector(1 downto 0);
214 signal tem_jogada_feita: std_logic;
215 signal not_escrevem : std_logic;
216
217 --NOVIDADES
218 signal not_escreve_tempo, not_zera_tempo: std_logic;
219 signal limite_tempo, endereco_tempo, ultima_jogada: std_logic_vector(7
220     downto 0);
221 signal tempo_reacao, saida_ram_tempos, media: std_logic_vector(13 downto
222     0);
223
224 begin
225     nzerac <= not zerac;
226     nzeralim <= not zeralim;
227     not_escrevem <= not escrevem;
228     not_escreve_tempo <= not escreve_tempo;
229     not_zera_tempo <= not zera_tempo;
230
231     or_jogada <= chaves(0) or chaves(1) or chaves(2) or chaves(3);
232
233     G1: contador_163 port map(
234         clock => clock,
235         clr   => nzerac,
236         ld    => '1',
237         ent   => contac,
238         enp   => contac,
239         D     => "0000",
240         limite=> limite,
241         Q     => s_endereco,
242         rco   => fimc);
243
244     GLim: contador_163 port map(
245         clock => clock,
246         clr   => nzeralim,
247         ld    => '1',
248         ent   => incrementalimite,
249         enp   => incrementalimite,
250         D     => "0000",
251         limite=> valor_nivel_4bits,

```

```

252         Q      => limite,
253         rco     => limiteMaximo);
254
255     G2: comparador_85 port map(
256         i_A3     => s_dados(3),
257         i_B3     => chaves(3),
258         i_A2     => s_dados(2),
259         i_B2     => chaves(2),
260         i_A1     => s_dados(1),
261         i_B1     => chaves(1),
262         i_A0     => s_dados(0),
263         i_B0     => chaves(0),
264         i_AGTB   => '0',
265         i_ALTB   => '0',
266         i_AEQB   => '1',
267         o_AGTB   => open,
268                     o_ALTB => open,
269         o_AEQB   => igual);
270
271     G3: ram_16x4 port map(
272         clk                      => clock,
273         endereco                => s_endereco,
274         dado_entrada            => chaves,
275         we                      => not_escrevem,
276         ce                      => '0',
277         dado_saida              => s_dados);
278
279     G4 : edge_detector port map
280     (
281         clock => clock,
282         reset => '0',
283         sinal => or_jogada,
284         pulso => tem_jogada_feita
285     );
286
287     G5 : registrador_4bits port map
288     (
289         clock => clock,
290         clear => zeraR,
291         enable => enableR,
292         D => led_jogada,
293         Q => db_jogada
294     );
295
296     G6 : timer5k port map
297     (
298         clock => clock,
299         clr      => treset,
300         enable => tenable,
301         Q      => tempo_reacao,
302         timeout => timeout

```

```

303     );
304     G7 : edge_detector port map
305     (
306         clock => clock,
307         reset => '0',
308         sinal => repetir,
309         pulso => repetir_edge
310     );
311     G8: registrador_2bits port map
312     (
313         clock => clock,
314         clear => '0',
315         enable => enable_nivel,
316         D => nivel,
317         Q => valor_nivel_2bits);
318
319     db_tem_jogada <= tem_jogada_feita;
320     jogada_feita <= tem_jogada_feita;
321     db_contagem <= s_endereco;
322     db_memoria <= s_dados;
323     db_limite <= limite;
324
325     with valor_nivel_2bits select
326         valor_nivel_4bits <= "0011" when "00",
327                                     "0111" when "01",
328                                     "1011" when "10",
329                                     "1111" when others;
330
331     db_nivel <= "00" & valor_nivel_2bits;
332
333     with pronto_UC select
334         led_jogada <= chaves when '0',
335                                     s_dados when '1';
336
337     --NOVIDADES PROJETO DE GRUPO
338
339     G9: ram_tempos port map(
340         clk          => clock,
341         endereco     => endereco_tempo,
342         dado_entrada => tempo_reacao, --tempo de reacao do jogador
343         we           => not_escreve_tempo,
344         ce           => '0',
345         dado_saida   => saida_ram_tempos);
346
347     with valor_nivel_2bits select
348         limite_tempo <= "00001110" when "00", --1+2+3+4+4=14
349                                     "00101100" when "01",
350                                     --1+2+...+8+8=44
351                                     "01011001" when "10",
352                                     --1+2+...+12+12=89
353                                     "10011000" when

```

```

354
355 G10: contador_tempos port map(
356     clock => clock,
357     clr   => not_zera_tempo,
358     ld    => '1',
359     ent   => conta_jogadas,
360     enp   => conta_jogadas,
361     D     => "00000000",
362           limite=> limite_tempo,
363     Q     => endereco_tempo,
364     rco   => open);
365
366 G11: contador_tempos port map(
367     clock => clock,
368     clr   => nzerelim,
369     ld    => '1',
370     ent   => escreve_tempo,
371     enp   => escreve_tempo,
372     D     => "00000000",
373           limite=> limite_tempo,
374     Q     => ultima_jogada,
375     rco   => open);
376
377 -- se ultima_jogada e endereco tempo sao iguais, acabou
378 G12: comparador_8bits port map(
379     --numero binario A de 14 bits
380     i_A => endereco_tempo,
381
382     --numero binario B de 14 bits
383     i_B => ultima_jogada,
384
385     --saidas
386     --representam conclusoes realizadas com o circuito
387     iguais => acabou_repetir
388 );
389
390 G13: timerRep port map(
391     clock      => clock,
392     clr        => reset_rep, -- ativo em alto
393     enable     => enable_rep,
394     Q          => open,
395     timeout    => timeout_rep
396 );
397
398 with mostra_media select
399     saida_tempo <= saida_ram_tempos when '0',
400     media when others;
401
402 G14: calcula_media port map(
403     clock      => clock,
404     clr        => zeralim, -- ativo em alto

```

```

405         enable      => escreve_tempo, -- ativo em alto
406         D            => tempo_reacao, -- tempo de reacao da
407     mais nova jogada
408     Q                => media --media das entradas fornecidas
409 );
410
411 G15: timerTotal port map(
412     clock      => clock,
413     nivel      => valor_nivel_2bits,
414     clr        => zeraLim, --so damos reset entre execucoes
415     enable     => tenable, --mas contamos junto com timer5k
416     timeout    => timeout_total
417 );
418
419 end architecture;

```

3.3 Unidade de controle

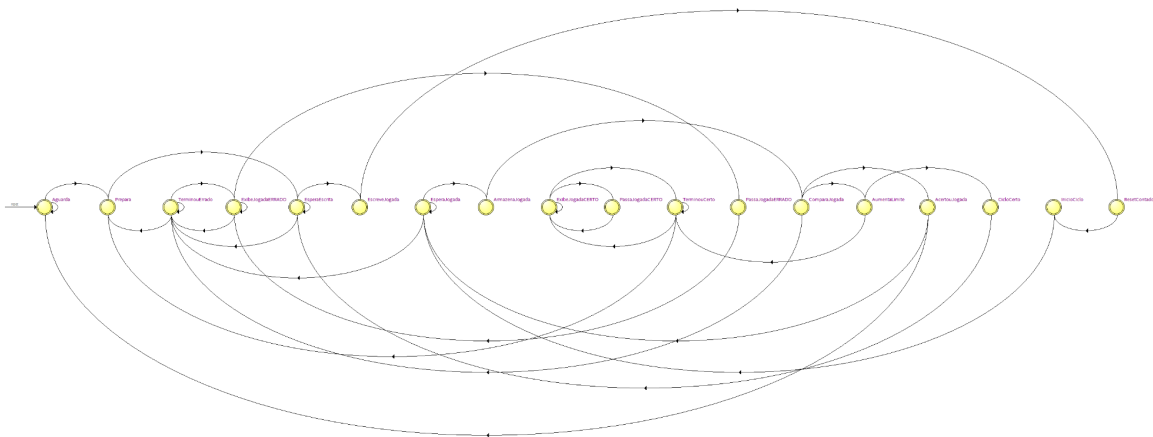


Imagem 3: Diagrama de Transição da Máquina de Estados obtido no Quartus

A estrutura da máquina de estados é a mesma, no entanto, ao adicionar o *timeout_total* foi necessário rever as transições que levam ao final do jogo. Foi necessário também realizar a atribuição de sinais de controle utilizados no fluxo de dados.

O sinal *escreve_tempo* é acionado nos estados que seguem detecção de jogada, funcionando como *write_enable* na RAM_tempos.

O sinal *conta_jogadas* atua como *enable* do contador que percorre a RAM_tempos. Não apenas ele deve ser acionada na detecção de jogada, mas também para percorrer a RAM durante a execução da função REPETIR.

O sinal *zera_tempo* é utilizado para *clear* da RAM_tempos entre os diferentes jogos que são realizados na placa. É importante retornar o circuito para suas condições iniciais após cada execução.

```

1  library ieee;
2  use ieee.std_logic_1164.all;
3

```

```

4  entity unidade_controle is
5      port (
6          clock:                in std_logic;
7          reset:                 in std_logic;
8          jogar:                 in std_logic;
9          repetir_edge:          in std_logic;
10         fim:                    in std_logic;
11         igual:                  in std_logic;
12         jogada:                  in std_logic;
13         limiteMaximo:           in std_logic;
14         timeout:                 in std_logic;
15         acabou_repetir:         in std_logic;
16         timeout_rep:            in std_logic;--NOVIDADE
17         timeout_total:          in std_logic;--NOVIDADE
18         treset:                  out std_logic;
19         tenable:                 out std_logic;
20         enable_rep:              out std_logic;--NOVIDADE
21         reset_rep:              out std_logic;--NOVIDADE
22         mostra_media:           out std_logic;--NOVIDADE
23         esta_jogando:           out std_logic;--NOVIDADE
24         escreve:                 out std_logic;
25         zera:                    out std_logic;
26         conta:                   out std_logic;
27         pronto:                  out std_logic;
28         registra:                out std_logic;
29         espera:                  out std_logic;
30         enable_nivel:           out std_logic;
31         escreve_tempo:          out std_logic;
32         zera_tempo:              out std_logic;
33         conta_jogadas:          out std_logic;
34         acertou:                out std_logic;
35         errou:                   out std_logic;
36         incrementalimite:       out std_logic;
37         zeralim:                 out std_logic;
38         db_estado:              out std_logic_vector(3 downto
39     0);
40         mensagem0:              out std_logic_vector(6 downto
41     0);
42         mensagem1:              out std_logic_vector(6 downto
43     0)
44     );
45 end entity;
46
47 architecture fsm of unidade_controle is
48     type t_estado is (Aguarda, --estado inicial
49         Prepara, --inicia circuito para
50         condicoes iniciais
51         InicioCiclo, --inicio de novo
52         ciclo, aguardando
53         EsperaJogada, --esperado
54         edge_detector detectar jogada

```

```

55         ArmazenaJogada, --armazena jogada
56         ComparaJogada, --compara jogada
57     com memoria
58         AcertouJogada,
59         CicloCerto, --chegamos no limite
60     atual
61         TerminouCerto,
62         TerminouErrado,
63         AumentaLimite, --aumentar jogadas
64     para proxima rodada
65         EsperaEscrita, --espera escrita
66     da jogada
67         ResetContador,
68         EscreveJogada,
69         PassaJogadaERRADO, --NOVIDADE
70         ExibeJogadaERRADO, --NOVIDADE
71         PassaJogadaCERTO, --NOVIDADE
72         ExibeJogadaCERTO); --NOVIDADE
73     signal Eatual, Eprox: t_estado;
74     signal certo : std_logic;
75
76     begin
77         -- memoria de estado
78         process (clock,reset)
79         begin
80             if reset='1' then
81                 Eatual <= Aguarda;
82                 certo <= '1';
83             elsif clock'event and clock = '1' then
84                 Eatual <= Eprox;
85                 certo <= igual;
86             end if;
87         end process;
88
89         -- logica de proximo estado
90         Eprox <=
91             Aguarda when Eatual=Aguarda and jogar='0' else
92             Prepara when Eatual=Aguarda and jogar='1' else
93             Prepara when (Eatual=TerminouCerto or Eatual=TerminouErrado)
94             and jogar='1' else
95             EsperaEscrita when Eatual=Prepara else
96             InicioCiclo when Eatual=ResetContador else
97
98             EsperaJogada when Eatual=InicioCiclo else
99             EsperaJogada when Eatual=EsperaJogada and jogada='0' and
100     timeout='0' and timeout_total='0' else
101             EsperaJogada when Eatual=AcertouJogada and fim='0' else --nao
102     acabou de jogar ainda
103
104             ArmazenaJogada when Eatual=EsperaJogada and jogada='1' and
105     timeout='0' and timeout_total='0' else

```



```

106         ComparaJogada when Eatual=ArmazenaJogada else
107
108         AcertouJogada when Eatual=ComparaJogada and certo='1' and
109 fim='0' else
110         TerminouErrado when Eatual=ComparaJogada and certo='0' else
111         TerminouErrado when Eatual=EsperaJogada and timeout='1' else
112         TerminouErrado when Eatual=EsperaJogada and timeout_total='1'
113     else
114
115
116         AumentaLimite when Eatual=ComparaJogada and certo='1' and
117 fim='1' else
118         CicloCerto when Eatual=AumentaLimite and limiteMaximo='0'
119     else
120
121         EsperaEscrita when Eatual=CicloCerto else
122         EsperaEscrita when Eatual=EsperaEscrita and jogada='0' and
123 timeout='0' and timeout_total='0' else
124         TerminouErrado when Eatual=EsperaEscrita and timeout='1' else
125         TerminouErrado when Eatual=EsperaEscrita and
126 timeout_total='1' else
127         EscreveJogada when Eatual=EsperaEscrita and jogada='1' and
128 timeout='0' and timeout_total='0' else
129         ResetContador when Eatual=EscreveJogada else
130
131         TerminouCerto when Eatual=AumentaLimite and limiteMaximo='1'
132     else
133         TerminouCerto when Eatual=TerminouCerto and jogar='0' and
134 repetir_edge='0' else
135         TerminouErrado when Eatual=TerminouErrado and jogar='0' and
136 repetir_edge='0' else
137
138         ExibeJogadaERRADO when Eatual=TerminouErrado and jogar='0'
139 and repetir_edge='1' else
140         ExibeJogadaERRADO when Eatual=ExibeJogadaERRADO and
141 timeout_rep='0' else
142         PassaJogadaERRADO when Eatual=ExibeJogadaERRADO and
143 timeout_rep='1' and acabou_repetir='0' else
144         TerminouErrado when Eatual=ExibeJogadaERRADO and
145 timeout_rep='1' and acabou_repetir='1' else
146         ExibeJogadaERRADO when Eatual=PassaJogadaERRADO else
147
148         ExibeJogadaCERTO when Eatual=TerminouCerto and jogar='0' and
149 repetir_edge='1' else
150         ExibeJogadaCERTO when Eatual=ExibeJogadaCERTO and
151 timeout_rep='0' else
152         PassaJogadaCERTO when Eatual=ExibeJogadaCERTO and
153 timeout_rep='1' and acabou_repetir='0' else
154         TerminouCerto when Eatual=ExibeJogadaCERTO and
155 timeout_rep='1' and acabou_repetir='1' else
156         ExibeJogadaCERTO when Eatual=PassaJogadaCERTO else

```

```

157
158         Aguarda;
159
160     -- logica de saída (maquina de Moore)
161     with Eatual select
162         zera <= '1' when Prepara | ResetContador | TerminouErrado |
163 TerminouCerto,
164             '0' when others;
165
166     with Eatual select
167         conta <= '1' when AcertouJogada | CicloCerto | PassaJogadaERRADO |
168 PassaJogadaCERTO,
169             '0' when others;
170
171     with Eatual select
172         pronto <= '1' when TerminouCerto | TerminouErrado | PassaJogadaERRADO
173 | ExibeJogadaERRADO | PassaJogadaCERTO | ExibeJogadaCERTO,
174             '0' when others;
175
176     with Eatual select
177         acertou <= '1' when TerminouCerto | PassaJogadaCERTO |
178 ExibeJogadaCERTO,
179             '0' when others;
180
181     with Eatual select
182         errou <= '1' when TerminouErrado | PassaJogadaERRADO |
183 ExibeJogadaERRADO,
184             '0' when others;
185
186     with Eatual select
187         registra <= '1' when ArmazenaJogada | PassaJogadaERRADO |
188 ExibeJogadaERRADO | PassaJogadaCERTO | ExibeJogadaCERTO,
189             '0' when others;
190
191     with Eatual select
192         incrementalimite <= '1' when AumentaLimite, --no proximo
193 ciclo, quero limite+1
194             '0' when others;
195
196     with Eatual select
197         zeraLim <= '1' when Prepara,
198             '0' when others;
199
200     with Eatual select
201         escreve <= '1' when EscreveJogada,
202             '0' when others;
203
204     with Eatual select
205         treset <= '1' when AcertouJogada | InicioCiclo |
206 TerminouErrado | PassaJogadaERRADO | CicloCerto | TerminouCerto |
207 PassaJogadaCERTO,
208             '0' when others;
209

```

```

210     with Eatual select
211         tenable <= '1' when EsperaJogada | EsperaEscrita,
212         '0' when others;
213
214     with Eatual select
215         espera <= '1' when EsperaEscrita,
216         '0' when others;
217
218     with Eatual select
219         enable_nivel <= '1' when Aguarda | Prepara | TerminouCerto |
220 TerminouErrado,
221         '0' when others;
222
223     with Eatual select
224         escreve_tempo <= '1' when ArmazenaJogada | EscreveJogada,
225         '0' when others;
226
227     with Eatual select
228         zera_tempo <= '1' when Prepara | TerminouCerto |
229 TerminouErrado,
230         '0' when others;
231
232     with Eatual select
233         conta_jogadas <= '1' when ArmazenaJogada | EscreveJogada |
234 PassaJogadaCERTO | PassaJogadaERRADO,
235         '0' when others;
236
237     with Eatual select
238         enable_rep <= '1' when ExibeJogadaERRADO |
239 ExibeJogadaCERTO,
240         '0' when others;
241
242     with Eatual select
243         reset_rep <= '1' when TerminouErrado | TerminouCerto |
244 PassaJogadaCERTO | PassaJogadaERRADO,
245         '0' when others;
246
247     with Eatual select
248         mostra_media <= '1' when TerminouErrado | TerminouCerto,
249         '0' when others;
250
251     with Eatual select --HEX 4
252         mensagem0 <= "1000010" when Aguarda,--G
253         "0001111" when Prepara,--R
254         "1000110" when InicioCiclo,--C
255         "1100000" when EsperaJogada,--J
256         "1100000" when ArmazenaJogada,--J
257         "1100000" when ComparaJogada,--J
258         "1100000" when AcertouJogada,--J
259         "1000110" when CicloCerto,--C
260         "0000110" when EsperaEscrita,--E
261         "1100000" when EscreveJogada,--J
262         "1111111" when

```

```

261 TerminouCerto,--nada
262         "1111111" when
263 TerminouErrado,--nada
264         "1000110" when ResetContador,--C
265         "1000111" when AumentaLimite,--L
266         "1111111" when PassaJogadaERRADO |
267 PassaJogadaCERTO,--nada
268         "1111111" when ExibeJogadaERRADO |
269 ExibeJogadaCERTO;--nada
270
271     with Eatual select --HEX5
272         mensagem1 <= "0001000" when Aguarda,--A
273         "0001100" when Prepara,--P
274         "1111001" when InicioCiclo,--I
275         "0000110" when EsperaJogada,--E
276         "0001000" when ArmazenaJogada,--A
277         "1000110" when ComparaJogada,--C
278         "0001000" when AcertouJogada,--A
279         "1000110" when CicloCerto,--C
280         "0000110" when EsperaEscrita,--E
281         "0000110" when EscreveJogada,--E
282         "1000010" when TerminouCerto,--G
283         "0001100" when TerminouErrado,--P
284         "0001111" when ResetContador,--R
285         "0001000" when AumentaLimite,--A
286         "1111111" when PassaJogadaERRADO |
287 PassaJogadaCERTO, --nada
288         "1111111" when ExibeJogadaERRADO |
289 ExibeJogadaCERTO;--nada
290
291     with Eatual select
292         esta_jogando <= '0' when TerminouCerto | TerminouErrado |
293 PassajogadaERRADO | PassaJogadaCERTO | ExibeJogadaERRADO |
294 ExibeJogadaCERTO,
295         '1' when others;
296
297     -- saida de depuracao (db_estado)
298     with Eatual select
299         db_estado <=
300 x"0" when Aguarda,      -- -> #40
301 x"1" when Prepara,      -- -> #79
302 x"2" when InicioCiclo,  -- -> #24
303 x"3" when EsperaJogada, -- -> #30
304 x"4" when ArmazenaJogada, -- -> #19
305 x"5" when ComparaJogada, -- -> #12
306 x"6" when AcertouJogada, -- -> #02
307 x"7" when CicloCerto,   -- -> #78
308 x"8" when EsperaEscrita, -- -> #00
309 x"9" when EscreveJogada, -- -> #10
310 x"A" when TerminouCerto, -- -> #08
311 x"B" when TerminouErrado, -- -> #03

```

```

312 x"C" when ResetContador, -- -> #46
313 x"D" when AumentaLimite, -- -> #21
314 x"E" when PassaJogadaERRADO, -- -> #06
315 x"F" when ExibeJogadaERRADO, -- -> #0E
316 x"E" when PassaJogadaCERTO, -- -> #06
317 x"F" when ExibeJogadaCERTO; -- -> #0E
318
319
320 end fsm;

```

3.4 circuito final

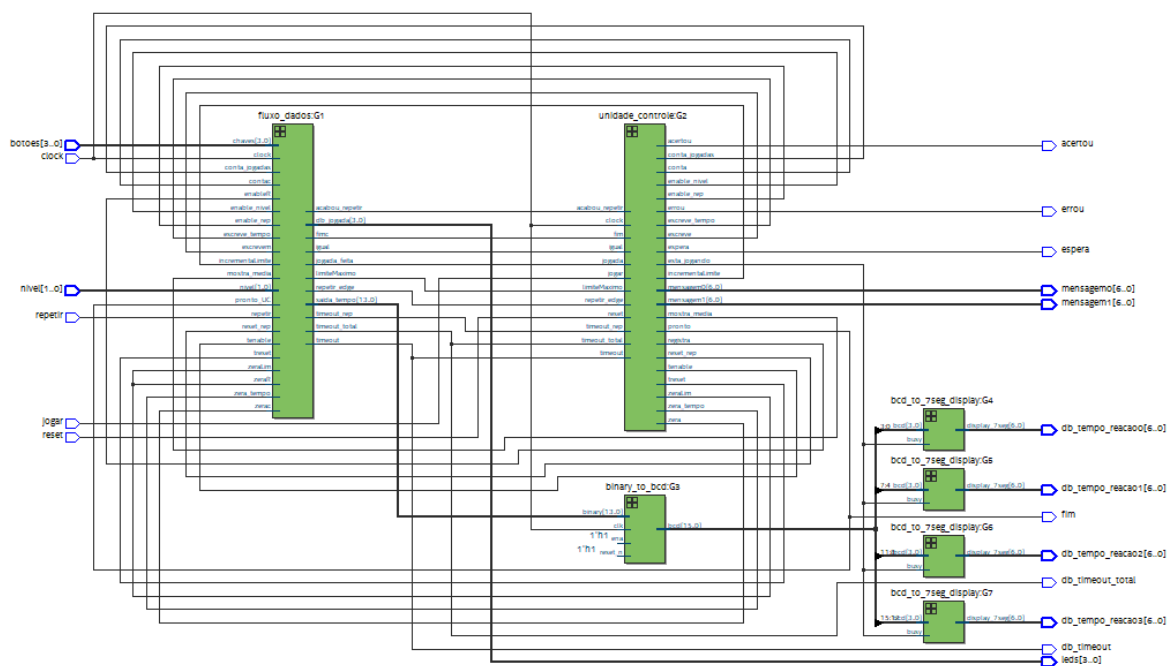


Imagem 4: Diagrama do circuito obtido no RTL Viewer

```

1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity circuito_semana3 is
5  port(
6      clock      : in  std_logic;
7      reset      : in  std_logic;
8      jogar      : in  std_logic;
9      repetir    : in  std_logic; --NOVIDADE
10     botoes : in  std_logic_vector(3 downto 0);

```

```

11     nivel      : in std_logic_vector(1 downto 0); --NOVIDADE
12     leds       : out std_logic_vector(3 downto 0);
13     espera : out std_logic; --NOVIDADE
14     fim        : out std_logic;
15     acertou    : out std_logic;
16     errou      : out std_logic;
17     -- Usamos quatro display HEX para mostrar tempo
18     db_tempo_reacao0: out std_logic_vector(6 downto 0);--NOVIDADE HEX 0
19     db_tempo_reacao1: out std_logic_vector(6 downto 0);--NOVIDADE HEX 1
20     db_tempo_reacao2: out std_logic_vector(6 downto 0);--NOVIDADE HEX 2
21     db_tempo_reacao3: out std_logic_vector(6 downto 0);--NOVIDADE HEX 3
22
23     --LEDS 4 e 5
24     db_timeout: out std_logic;
25     db_timeout_total: out std_logic;
26
27     --HEX4 e HEX5
28     mensagem0:      out std_logic_vector(6 downto 0);
29     mensagem1:      out std_logic_vector(6 downto 0) -- HEX 5
30 );
31 end entity;
32
33
34 architecture arch_semana3 of circuito_semana3 is
35
36     component fluxo_dados
37     port(
38         clock:                in std_logic;
39         zerac:                 in std_logic;
40         zeraLim:               in std_logic;
41         contac:                in std_logic;
42         escrevem:              in std_logic;
43         repetir:               in std_logic;
44         nivel:                 in std_logic_vector(1 downto 0);
45         enable_nivel:          in std_logic;
46         chaves:                in std_logic_vector(3 downto 0);
47         escreve_tempo:         in std_logic;
48         enable_rep:             in std_logic; --NOVIDADE
49         reset_rep:             in std_logic; --NOVIDADE
50         mostra_media:          in std_logic; --NOVIDADE
51         zeraR:                 in std_logic;
52         enableR:               in std_logic;
53         incrementalLimite: in std_logic;
54         treset:                in std_logic;
55         tenable:               in std_logic;
56         pronto_UC:             in std_logic;
57         zera_tempo:             in std_logic;
58         conta_jogadas:         in std_logic;
59         timeout:                out std_logic;
60         timeout_rep:           out std_logic; --NOVIDADE
61         timeout_total:         out std_logic; --NOVIDADE

```

```

62         limiteMaximo:          out std_logic;
63     igual:                      out std_logic;
64     fimc:                      out std_logic;
65         db_tem_jogada:         out std_logic;
66         jogada_feita:          out std_logic;
67         repetir_edge:         out std_logic;
68         acabou_repetir:        out std_logic;
69         saida_tempo:           out std_logic_vector(13 downto 0);
70     db_contagem:                out std_logic_vector(3 downto 0);
71     db_memoria:                out std_logic_vector(3 downto 0);
72         db_limite:              out std_logic_vector(3 downto 0);
73         db_nivel:              out std_logic_vector(3 downto 0);
74         db_jogada:             out std_logic_vector(3 downto 0)
75     );
76 end component;
77
78 component unidade_controle
79     port(
80         clock:                  in  std_logic;
81         reset:                  in  std_logic;
82         jogar:                  in  std_logic;
83         repetir_edge:           in  std_logic;
84         fim:                    in  std_logic;
85         igual:                  in  std_logic;
86         jogada:                 in   std_logic;
87         limiteMaximo:           in  std_logic;
88         timeout:                in  std_logic;
89         acabou_repetir:         in  std_logic;
90         timeout_rep:            in  std_logic;--NOVIDADE
91         timeout_total:          in  std_logic;--NOVIDADE
92         treset:                 out std_logic;
93         tenable:                out std_logic;
94         enable_rep:             out std_logic;--NOVIDADE
95         reset_rep:              out std_logic;--NOVIDADE
96         mostra_media:           out std_logic;--NOVIDADE
97         esta_jogando:           out std_logic;
98         escreve:                out std_logic;
99         zera:                   out std_logic;
100        conta:                  out std_logic;
101        pronto:                 out std_logic;
102        registra:               out std_logic;
103        espera:                 out std_logic;
104        enable_nivel:            out std_logic;
105        escreve_tempo:           out std_logic;
106        zera_tempo:              out std_logic;
107        conta_jogadas:           out std_logic;
108        acertou:                out std_logic;
109        errou:                   out std_logic;
110        incrementalimite:        out std_logic;
111        zeraLim:                 out std_logic;
112        db_estado:               out std_logic_vector(3

```

```

113     downto 0);
114     mensagem0:                                out std_logic_vector(6
115     downto 0);
116     mensagem1:                                out std_logic_vector(6
117     downto 0)
118     );
119 end component;
120
121 component hexa7seg
122     port (
123         hexa : in  std_logic_vector(3 downto 0);
124         sseg : out std_logic_vector(6 downto 0)
125     );
126 end component;
127
128 --NOVIDADES: converter numero para exibir em displays HEX
129 component binary_to_bcd_digit
130     port(
131         clk      : in      std_logic;
132                --system clock
133         reset_n  : in      std_logic;
134                --active low asynchronous reset
135         ena      : in      std_logic;
136                --activate operation
137         binary   : in      std_logic;
138                --bit shifted into digit
139         c_out    : buffer std_logic;
140                --carry out shifted to next larger digit
141         bcd      : buffer std_logic_vector(3 downto 0));
142 --resulting BCD output
143 end component;
144
145 component binary_to_bcd
146     generic(
147         bits      : INTEGER := 10;                --size of the
148         binary input numbers in bits
149         digits    : INTEGER := 3);                --number of BCD
150         digits to convert to
151     port(
152         clk      : in      std_logic;
153                --system clock
154         reset_n  : in      std_logic;
155                --active low
156         asynchrnous reset
157         ena      : in      std_logic;
158                --latches in new
159         binary number and starts conversion
160         binary   : in      std_logic_vector(bits-1 downto 0);
161                --binary number to convert
162         busy      : out     std_logic;
163                --indicates conversion in

```



```

164 progress
165         bcd          :      out      std_logic_vector(digits*4-1 downto
166 0)); --resulting BCD number
167 end component;
168
169 component bcd_to_7seg_display
170     port(
171         bcd          :      in
172 std_logic_vector(3 downto 0));
173         busy          :      in      std_logic;
174         display_7seg :      out      std_logic_vector(6 downto 0));
175 end component;
176
177 signal not_reset          : std_logic;
178 signal fimc_FD            : std_logic;
179 signal conta_UC          : std_logic;
180 signal zera_UC            : std_logic;
181 signal igual              : std_logic;
182 signal tudo_certo        : std_logic;
183 signal jogada_feita_FD    : std_logic;
184 signal enable             : std_logic;
185 signal final_UC           : std_logic;
186 signal registra_UC        : std_logic;
187 signal aux                : std_logic_vector(3 downto 0);
188 signal jogada_FD          : std_logic_vector(3 downto 0);
189 signal contagem           : std_logic_vector(3 downto 0);
190 signal memoria            : std_logic_vector(3 downto 0);
191 signal estado             : std_logic_vector(3 downto 0);
192 signal jogada             : std_logic_vector(3 downto 0);
193
194
195
196 signal limite_FD : std_logic_vector (3 downto 0);
197 signal limMax_FD : std_logic;
198 signal inclim_UC, conta_jogadas : std_logic;
199 signal zeraLim   : std_logic;
200 signal escrevem, acabou_repetir : std_logic;
201 signal errou_UC, fim_UC : std_logic;
202 signal timeout, treset, tenable, enable_nivel, escreve_tempo, zera_tempo :
203 std_logic;
204 signal saida_tempo : std_logic_vector(13 downto 0);
205 signal repetir_edge : std_logic;
206
207 --NOVIDADES
208 --signal display_HEX0, display_HEX1, display_HEX2, display_HEX3:
209 std_logic_vector(3 downto 0);
210 signal saida_BCD : std_logic_vector(15 downto 0);
211 signal enable_rep, reset_rep, timeout_rep, busy_BCD, mostra_media,
212 timeout_total, esta_jogando: std_logic;
213
214 begin
215
216 G1: fluxo_dados port map

```

```

217     (
218         clock                => clock,
219         zerac                 => zera_UC,
220         zeralim               => zeralim,
221         contac                => conta_UC,
222         escrevem              => escrevem,
223         repetir               => repetir,
224         nivel                 => nivel,
225         enable_nivel          => enable_nivel,
226         escreve_tempo         => escreve_tempo,
227         enable_rep             => enable_rep, --NOVIDADE
228         reset_rep             => reset_rep, --NOVIDADE
229         mostra_media          => mostra_media, --NOVIDADE
230         chaves                 => botoes,
231         zeraR                  => zeralim,
232         enableR                => registra_UC,
233         incrementalLimite      => inclim_UC,
234         limiteMaximo           => limMax_FD,
235         treset                 => treset,
236         tenable                => tenable,
237         pronto_UC              => fim_UC,
238         zera_tempo             => zera_tempo,
239         conta_jogadas          => conta_jogadas,
240         timeout                => timeout,
241         timeout_rep            => timeout_rep, --NOVIDADE
242         timeout_total          => timeout_total, --NOVIDADE
243         igual                   => igual,
244         fimc                    => fimc_FD,
245         db_tem_jogada          => open,
246         jogada_feita           => jogada_feita_FD,
247         repetir_edge           => repetir_edge,
248         acabou_repetir         => acabou_repetir,
249         saida_tempo            => saida_tempo,
250         db_contagem            => contagem,
251         db_memoria             => memoria,
252         db_limite              => limite_FD,
253         db_jogada              => jogada_FD
254     );
255
256 G2: unidade_controle port map
257     (
258         clock                => clock,
259         reset                 => reset,
260         jogar                  => jogar,
261         repetir_edge           => repetir_edge,
262         fim                    => fimc_FD,
263         igual                   => igual,
264         jogada                  => jogada_feita_FD,
265         limiteMaximo           => limMax_FD,
266         timeout                 => timeout,
267         acabou_repetir         => acabou_repetir,

```

```

268         timeout_rep           => timeout_rep,--NOVIDADE
269         timeout_total         => timeout_total,
270         treset                 => treset,
271         tenable                => tenable,
272         enable_rep             => enable_rep,--NOVIDADE
273         reset_rep              => reset_rep,--NOVIDADE
274         mostra_media           => mostra_media,--NOVIDADE
275         esta_jogando           => esta_jogando,--NOVIDADE
276         escreve                => escrevem,
277         zera                   => zera_UC,
278         conta                  => conta_UC,
279         pronto                 => fim_UC,
280         registra               => registra_UC,
281         espera                 => espera,
282         acertou                => acertou,
283         errou                  => errou_UC,
284         incrementalLimite      => inclim_UC,
285         enable_nivel           => enable_nivel,
286         escreve_tempo          => escreve_tempo,
287         zera_tempo             => zera_tempo,
288         conta_jogadas          => conta_jogadas,
289         zeraLim                => zeraLim,
290         db_estado              => estado,
291         mensagem0              => mensagem0,
292         mensagem1              => mensagem1
293     );
294
295     --aux <= "00"&saida_tempo(13 downto 12);
296
297     --G4: hexa7seg port map(hexa=>saida_tempo(3 downto 0),
298     sseg=>db_tempo_reacao0);
299     --G5: hexa7seg port map(hexa=>saida_tempo(7 downto 4),
300     sseg=>db_tempo_reacao1);
301     --G6: hexa7seg port map(hexa=>saida_tempo(11 downto 8),
302     sseg=>db_tempo_reacao2);
303     --G7: hexa7seg port map(hexa=>aux,      sseg=>db_tempo_reacao3);
304
305     not_reset <= not reset;
306
307     G3: binary_to_bcd
308         generic map(
309             bits           =>          14,           --size of the binary
310             input numbers in bits
311             digits =>          4)           --number of BCD digits to
312             convert to
313         port map(
314             clk            => clock,           --system clock
315             reset_n        => '1',           --active low asynchronus
316             reset
317             ena            => '1',           --latches in new
318             binary number and starts conversion

```

```

319         binary => saida_tempo, --binary number to convert
320         busy      => busy_BCD, --indicates conversion in progress
321         bcd        => saida_BCD); --resulting BCD number
322
323 G4: bcd_to_7seg_display port map(bcd=>saida_BCD(3 downto 0),
324     busy=>esta_jogando, display_7seg=>db_tempo_reacao0);
325 G5: bcd_to_7seg_display port map(bcd=>saida_BCD(7 downto 4),
326     busy=>esta_jogando, display_7seg=>db_tempo_reacao1);
327 G6: bcd_to_7seg_display port map(bcd=>saida_BCD(11 downto 8),
328     busy=>esta_jogando, display_7seg=>db_tempo_reacao2);
329 G7: bcd_to_7seg_display port map(bcd=>saida_BCD(15 downto 12),
330     busy=>esta_jogando, display_7seg=>db_tempo_reacao3);
331
332 leds <= jogada_FD;
333 errou <= errou_UC;
334 fim <= fim_UC;
335 db_timeout <= timeout;
336 db_timeout_total <= timeout_total;
337
338 end architecture;

```

4. Testes no ModelSim

Foram realizados 3 testes no ModelSim para verificação da função REPETIR.

1. Acionamento da função REPETIR após errar primeira jogada da quarta rodada
2. Acionamento da função REPETIR após timeout na primeira jogada da quarta rodada
3. Acionamento da função REPETIR após ganhar o jogo no nível 01 (oito jogadas)

Vale observar que os testes realizados não cobrem todos os cenários possíveis, mas consideramos que a cobertura deste plano é suficiente num nível lógico para testar as modificações. Afinal, possíveis erros decorrentes da mudança de nível estão associados a problemas físicos da placa e não a aspectos lógicos do circuito.

4.1 Cenário de teste 1

```

1  library IEEE;
2  use IEEE.std_logic_1164.all;
3  use IEEE.numeric_bit.all;
4
5  entity tb1_semana2 is
6  end entity;
7
8  architecture tb of tb1_semana2 is
9      component circuito_semana2 is
10         port(clock      : in  std_logic;
11             reset       : in  std_logic;
12             jogar       : in  std_logic;
13             repetir     : in  std_logic; --NOVIDADE
14             botoes      : in  std_logic_vector(3 downto 0);
15             nivel       : in  std_logic_vector(1 downto 0);

```

```

16  --NOVIDADE
17      leds          : out std_logic_vector(3 downto 0);
18      espera        : out std_logic; --NOVIDADE
19      fim           : out std_logic;
20      acertou       : out std_logic;
21      errou         : out std_logic;
22      -- Usamos quatro display HEX para mostrar tempo
23      db_tempo_reacao0: out std_logic_vector(6 downto
24  0);--NOVIDADE HEX 0
25      db_tempo_reacao1: out std_logic_vector(6 downto
26  0);--NOVIDADE HEX 1
27      db_tempo_reacao2: out std_logic_vector(6 downto
28  0);--NOVIDADE HEX 2
29      db_tempo_reacao3: out std_logic_vector(6 downto
30  0);--NOVIDADE HEX 3
31      db_estado:      out std_logic_vector(6 downto 0));
32  end component;
33
34  type arranjo_memoria is array(0 to 15) of std_logic_vector(3 downto
35  0);
36  signal memoria : arranjo_memoria :=
37  (
38      "0001",
39      "0010",
40      "0100",
41      "1000",
42      "0100",
43      "0010",
44      "0001",
45      "0001",
46      "0010",
47      "0010",
48      "0100",
49      "0100",
50      "1000",
51      "1000",
52      "0001",
53      "0100"
54  );
55
56  constant TbPeriod      : time := 1000 ns;
57  signal TbSimulation     : std_logic := '0';
58  signal TbButtonOnWait  : integer := 10;
59  signal TbButtonOffWait : integer := 103;
60  signal TbZero          : std_logic_vector(3 downto 0) :=
61  "0000";
62
63  signal clock, reset, jogar, fim, acertou, errou : std_logic;
64  signal botoes, leds : std_logic_vector(3 downto 0);
65  signal nivel : std_logic_vector(1 downto 0);
66  signal repetir, espera : std_logic;

```

```

67     signal db_tempo_reacao0, db_tempo_reacao1, db_tempo_reacao2,
68     db_tempo_reacao3, db_estado : std_logic_vector(6 downto 0);
69
70 begin
71     DUT: circuito_semana2 port map
72     (
73         --ENTRADAS
74         clock           => clock,
75         reset           => reset,
76         jogar           => jogar,
77         repetir         => repetir,
78         botoes          => botoes,
79         nivel           => nivel,
80         --SAIDAS
81         leds            => leds,
82         espera          => espera,
83         fim             => fim,
84         acertou         => acertou,
85         errou           => errou,
86         --SINAIS DE DEPURACÃO
87         db_tempo_reacao0 => db_tempo_reacao0,
88         db_tempo_reacao1 => db_tempo_reacao1,
89         db_tempo_reacao2 => db_tempo_reacao2,
90         db_tempo_reacao3 => db_tempo_reacao3,
91         db_estado        => db_estado
92     );
93
94     clock <= not clock after TbPeriod/2 when TbSimulation = '1' else
95     '0';
96     stimuli: process
97     begin
98         TbSimulation <= '1';
99
100         reset <= '0';
101         jogar <= '0';
102         repetir <= '0';
103         botoes <= "0000";
104         nivel <= "00";
105
106         -- Condições iniciais
107         reset <= '1';
108         wait for 1000 ns;
109         reset <= '0';
110         wait for 1000 ns;
111
112         --Início do jogo
113         jogar <= '1';
114         wait for 1000 ns;
115         jogar <= '0';
116         wait for 1000 ns;
117

```

```

118
119         for i in 0 to 3 loop
120             for j in 0 to i loop
121                 wait for (5*i + 5*j + 10*i*j)*TbPeriod;
122                 botoes <= memoria(j);
123                 wait for TbButtonOnWait * TbPeriod;
124                 botoes <= TbZero;
125                 wait for TbButtonOffWait * TbPeriod;
126             end loop;
127         end loop;
128         --ate aqui tudo certo
129         --agora vamos errar
130         botoes <= "1000";
131         wait for TbButtonOnWait * TbPeriod;
132         botoes <= TbZero;
133         wait for TbButtonOffWait * TbPeriod;
134
135         repetir <= '1';
136         wait for TbButtonOnWait * TbPeriod;
137         repetir <= '0';
138         wait for TbButtonOffWait * TbPeriod;
139
140
141         wait for 90000*TbPeriod;
142
143         TbSimulation <= '0';
144         wait;
145     end process;
146
147 end architecture;

```

Observe que os tempos de reação utilizados no testbench dependem do índice da jogada correspondente. Dessa forma, ao observar a forma de onda, é fácil visualizar a mudança entre jogadas na função REPETIR pela alteração do valor do tempo de reação correspondente.

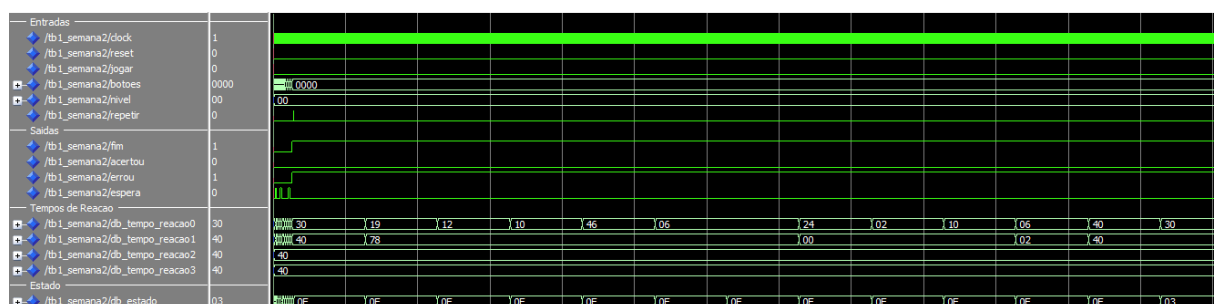


Imagem 5: Cenário de teste 1

4.2 Cenário de teste 2

```
1  library IEEE;
2  use IEEE.std_logic_1164.all;
3  use IEEE.numeric_bit.all;
4
5  entity tb3_semana2 is
6  end entity;
7
8  architecture tb of tb3_semana2 is
9      component circuito_semana2 is
10         port(clock      : in  std_logic;
11              reset      : in  std_logic;
12              jogar      : in  std_logic;
13              repetir : in  std_logic; --NOVIDADE
14              botoes     : in  std_logic_vector(3 downto 0);
15              nivel      : in  std_logic_vector(1 downto 0);
16      --NOVIDADE
17              leds       : out std_logic_vector(3 downto 0);
18              espera     : out std_logic; --NOVIDADE
19              fim         : out std_logic;
20              acertou    : out std_logic;
21              errou      : out std_logic;
22      -- Usamos quatro display HEX para mostrar tempo
23              db_tempo_reacao0: out std_logic_vector(6 downto
24 0);--NOVIDADE HEX 0
25              db_tempo_reacao1: out std_logic_vector(6 downto
26 0);--NOVIDADE HEX 1
27              db_tempo_reacao2: out std_logic_vector(6 downto
28 0);--NOVIDADE HEX 2
29              db_tempo_reacao3: out std_logic_vector(6 downto
30 0);--NOVIDADE HEX 3
31              db_estado:      out std_logic_vector(6 downto 0));
32      end component;
33
34      type arranjo_memoria is array(0 to 15) of std_logic_vector(3 downto
35 0);
36      signal memoria : arranjo_memoria :=
37      (
38          "0001",
39          "0010",
40          "0100",
41          "1000",
42          "0100",
43          "0010",
44          "0001",
45          "0001",
46          "0010",
47          "0010",
48          "0100",
49          "0100",
```



```

50         "1000",
51         "1000",
52         "0001",
53         "0100"
54     );
55
56     constant TbPeriod      : time := 1000 ns;
57     signal TbSimulation     : std_logic := '0';
58     signal TbButtonOnWait   : integer := 10;
59     signal TbButtonOffWait  : integer := 103;
60     signal TbZero           : std_logic_vector(3 downto 0) :=
61 "0000";
62
63     signal clock, reset, jogar, fim, acertou, errou : std_logic;
64     signal botoes, leds : std_logic_vector(3 downto 0);
65     signal nivel : std_logic_vector(1 downto 0);
66     signal repetir, espera : std_logic;
67     signal db_tempo_reacao0, db_tempo_reacao1, db_tempo_reacao2,
68 db_tempo_reacao3, db_estado : std_logic_vector(6 downto 0);
69
70 begin
71     DUT: circuito_semana2 port map
72     (
73         --ENTRADAS
74         clock      => clock,
75         reset      => reset,
76         jogar      => jogar,
77         repetir    => repetir,
78         botoes     => botoes,
79         nivel      => nivel,
80         --SAIDAS
81         leds       => leds,
82         espera     => espera,
83         fim        => fim,
84         acertou    => acertou,
85         errou      => errou,
86         --SINAIS DE DEPURACÃO
87         db_tempo_reacao0 => db_tempo_reacao0,
88         db_tempo_reacao1 => db_tempo_reacao1,
89         db_tempo_reacao2 => db_tempo_reacao2,
90         db_tempo_reacao3 => db_tempo_reacao3,
91         db_estado      => db_estado
92     );
93
94     clock <= not clock after TbPeriod/2 when TbSimulation = '1' else
95 '0';
96     stimuli: process
97     begin
98         TbSimulation <= '1';
99
100         reset <= '0';

```

```

101         jogar <='0';
102         repetir <='0';
103         botoes <= "0000";
104         nivel <= "00";
105
106
107         -- Condições iniciais
108         reset <= '1';
109         wait for 1000 ns;
110         reset <= '0';
111         wait for 1000 ns;
112
113         --Início do jogo
114         jogar <= '1';
115         wait for 1000 ns;
116         jogar <= '0';
117         wait for 1000 ns;
118
119         for i in 0 to 3 loop
120             for j in 0 to i loop
121                 wait for (5*i + 5*j + 10*i*j)*TbPeriod;
122                 botoes <= memoria(j);
123                 wait for TbButtonOnWait * TbPeriod;
124                 botoes <= TbZero;
125                 wait for TbButtonOffWait * TbPeriod;
126             end loop;
127         end loop;
128         --ate aqui tudo certo
129
130         --agora vamos dar timeout
131         wait for 5009 * TbPeriod;
132
133         repetir <='1';
134         wait for TbButtonOnWait * TbPeriod;
135         repetir <='0';
136         wait for TbButtonOffWait * TbPeriod;
137
138
139         wait for 90000*TbPeriod;
140
141         TbSimulation <= '0';
142         wait;
143     end process;
144
145 end architecture;

```

Ao observar a imagem da forma de onda obtida, observe que o estado ExibeJogada da função repetir é visitado 11 vezes. Ou seja, são exibidos 11 tempos de reação enquanto no cenário 1 observamos 12. De fato, o timeout não é armazenado na RAM_tempos.

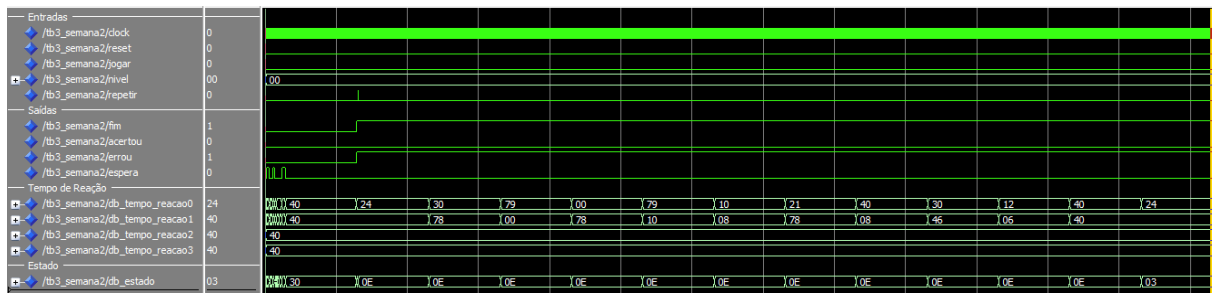


Imagem 6: Cenário de Teste 2

Nessa figura, podemos observar ainda que a ativação dos sinais *fim* e *erro* ocorre muito depois do que no cenário 1, e há uma estadia prolongada no estado *EsperaJogada*, confirmando de fato a ocorrência do timeout.

4.3 Cenário de teste 3

Finalmente, vamos verificar que a extensão da função REPETIR para o caso de vitória funciona conforme esperado.

```

1  library IEEE;
2  use IEEE.std_logic_1164.all;
3  use IEEE.numeric_bit.all;
4
5  entity tb2_semana2 is
6  end entity;
7
8  architecture tb of tb2_semana2 is
9      component circuito_semana2 is
10         port(clock          : in  std_logic;
11              reset          : in  std_logic;
12              jogar          : in  std_logic;
13              repetir       : in  std_logic; --NOVIDADE
14              botoes         : in  std_logic_vector(3 downto 0);
15              nivel          : in  std_logic_vector(1 downto 0);
16      --NOVIDADE
17              leds           : out std_logic_vector(3 downto 0);
18              espera        : out std_logic; --NOVIDADE
19              fim            : out std_logic;
20              acertou       : out std_logic;
21              erro          : out std_logic;
22      -- Usamos quatro display HEX para mostrar tempo
23          db_tempo_reacao0: out std_logic_vector(6 downto
24  0); --NOVIDADE HEX 0
25          db_tempo_reacao1: out std_logic_vector(6 downto
26  0); --NOVIDADE HEX 1
27          db_tempo_reacao2: out std_logic_vector(6 downto
28  0); --NOVIDADE HEX 2
29          db_tempo_reacao3: out std_logic_vector(6 downto
30  0); --NOVIDADE HEX 3
31          db_estado:      out std_logic_vector(6 downto 0));
32      end component;
33

```

```

34     type arranjo_memoria is array(0 to 15) of std_logic_vector(3 downto
35     0);
36     signal memoria : arranjo_memoria :=
37     (
38         "0001",
39         "0010",
40         "0100",
41         "1000",
42         "0100",
43         "0010",
44         "0001",
45         "0001",
46         "0010",
47         "0010",
48         "0100",
49         "0100",
50         "1000",
51         "1000",
52         "0001",
53         "0100"
54     );
55
56     constant TbPeriod          : time := 1000 ns;
57     signal TbSimulation         : std_logic := '0';
58     signal TbButtonOnWait      : integer := 10;
59     signal TbButtonOffWait     : integer := 103;
60     signal TbZero              : std_logic_vector(3 downto 0) :=
61     "0000";
62
63     signal clock, reset, jogar, fim, acertou, erro : std_logic;
64     signal botoes, leds        : std_logic_vector(3 downto 0);
65     signal nivel : std_logic_vector(1 downto 0);
66     signal repetir, espera : std_logic;
67     signal db_tempo_reacao0, db_tempo_reacao1, db_tempo_reacao2,
68     db_tempo_reacao3, db_estado : std_logic_vector(6 downto 0);
69
70     begin
71         DUT: circuito_semana2 port map
72         (
73             --ENTRADAS
74             clock          => clock,
75             reset          => reset,
76             jogar          => jogar,
77             repetir        => repetir,
78             botoes          => botoes,
79             nivel           => nivel,
80             --SAIDAS
81             leds            => leds,
82             espera          => espera,
83             fim             => fim,
84             acertou         => acertou,

```

```

85         erro1 => erro1,
86         --SINAIS DE DEPURACÃO
87         db_tempo_reacao0 => db_tempo_reacao0,
88         db_tempo_reacao1 => db_tempo_reacao1,
89         db_tempo_reacao2 => db_tempo_reacao2,
90         db_tempo_reacao3 => db_tempo_reacao3,
91         db_estado => db_estado
92     );
93
94     clock <= not clock after TbPeriod/2 when TbSimulation = '1' else
95     '0';
96     stimuli: process
97     begin
98         TbSimulation <= '1';
99
100         reset <= '0';
101         jogar <= '0';
102         repetir <= '0';
103         botoes <= "0000";
104         nivel <= "00";
105
106         -- Condições iniciais
107         reset <= '1';
108         wait for 1000 ns;
109         reset <= '0';
110         wait for 1000 ns;
111
112         --Início do jogo
113         jogar <= '1';
114         wait for 1000 ns;
115         jogar <= '0';
116         wait for 1000 ns;
117
118         nivel <= "01"; --Oito Jogadas, vamos ganhar
119         for i in 0 to 7 loop
120             for j in 0 to i loop
121                 wait for (5*i + 5*j + 10*i*j)*TbPeriod;
122                 botoes <= memoria(j);
123                 wait for TbButtonOnWait * TbPeriod;
124                 botoes <= TbZero;
125                 wait for TbButtonOffWait * TbPeriod;
126             end loop;
127         end loop;
128         --agora repete sequencia completa
129         for j in 0 to 7 loop
130             wait for (8*j)*TbPeriod;
131             botoes <= memoria(j);
132             wait for TbButtonOnWait * TbPeriod;
133             botoes <= TbZero;
134             wait for TbButtonOffWait * TbPeriod;
135

```

```

136         end loop;
137
138         repetir <='1';
139         wait for TbButtonOnWait * TbPeriod;
140         repetir <='0';
141         wait for TbButtonOffWait * TbPeriod;
142
143
144         wait for 90000*TbPeriod;
145
146         TbSimulation <= '0';
147         wait;
148     end process;
149
150 end architecture;

```

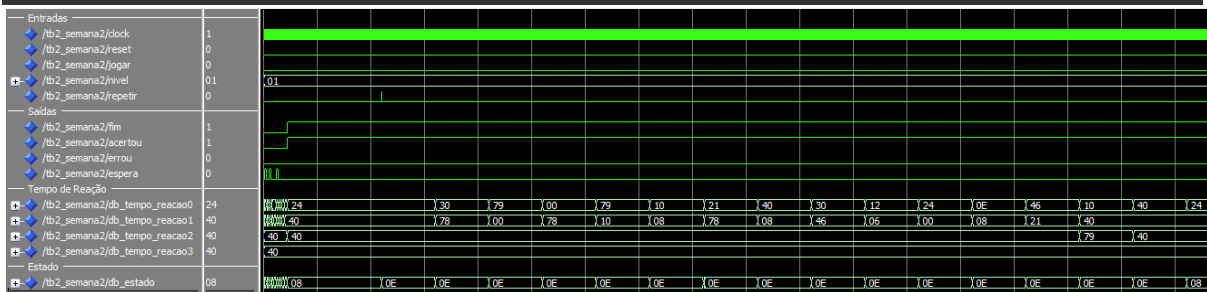


Imagem 7: Cenário de Teste 3

O estado *TerminouCerto*(08) é atingido, confirmando a vitória do jogador. A função REPETIR é executada como esperado. Ao final o circuito retorna ao estado *TerminouCerto*. Portanto, os testes com o ModelSim correram conforme o esperado.

5. Testes realizados na placa

Durante a gravação do vídeo de demonstração do projeto para a feira, obtivemos capturas de tela ilustrando o funcionamento desta versão final do projeto.

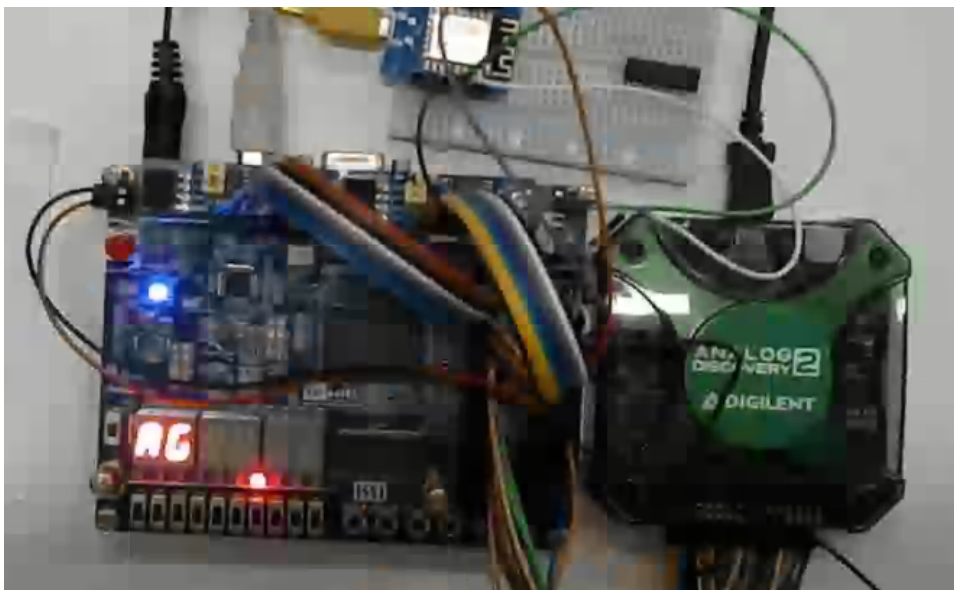


Imagem 8: Mensagem “AG” indica estado inicial *Aguarda*

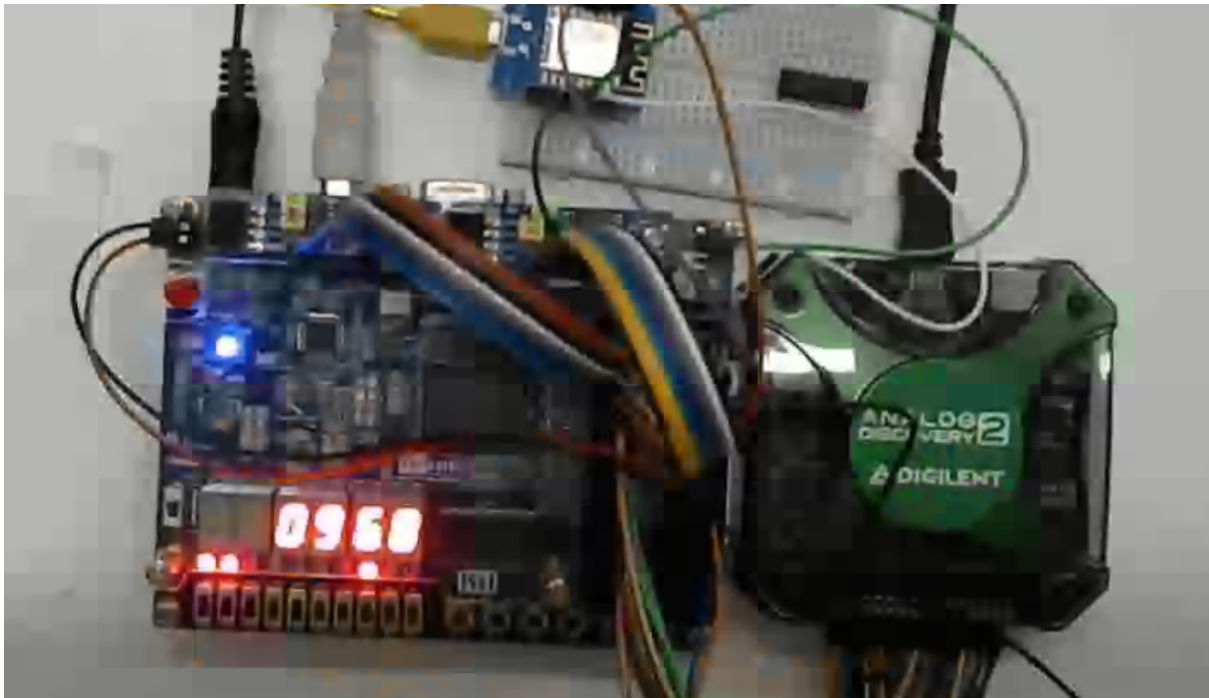


Imagem 9: Durante repetição, tempos de reação são exibidos

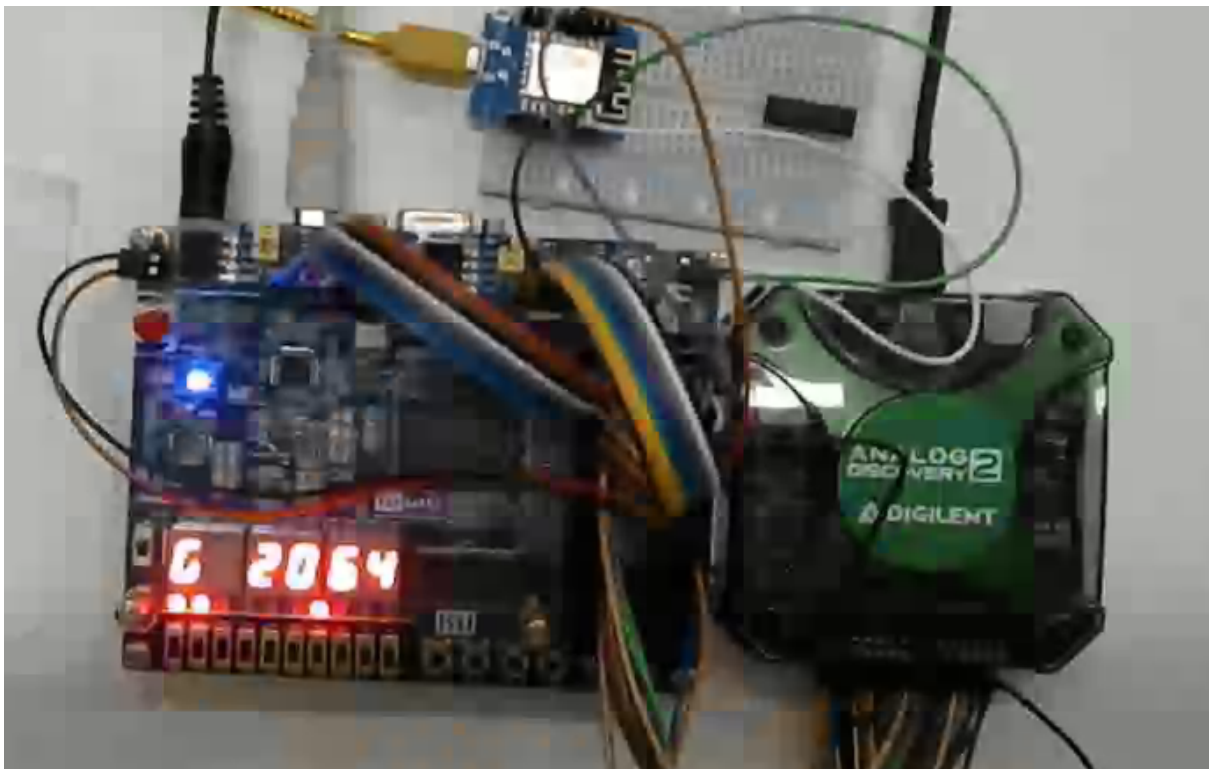


Imagem 10: Ao final do jogo, tempo médio das jogadas realizadas é exibido

Os testes realizados para a banca foram um sucesso. Estamos muito satisfeitos com a performance do projeto.

OBS: Muitos dos testes realizados em sala foram gravados e podem ser vistos neste link <https://www.youtube.com/channel/UCPrUBAnayQFdo-wcMvePkjw>

6. Pinagem do Projeto


















Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	Current Strength	Slew Rate
 acertou	Output	PIN_L2	2A	B2A_N0	2.5 V (default)		12mA (default)	1 (default)
 botoes[3]	Input	PIN_K17	5B	B5B_N0	2.5 V (default)		12mA (default)	
 botoes[2]	Input	PIN_L17	5B	B5B_N0	2.5 V (default)		12mA (default)	
 botoes[1]	Input	PIN_M18	5B	B5B_N0	2.5 V (default)		12mA (default)	
 botoes[0]	Input	PIN_P17	5A	B5A_N0	2.5 V (default)		12mA (default)	
 clock	Input	PIN_T22	5A	B5A_N0	2.5 V (default)		12mA (default)	
 db_estado[6]	Output	PIN_W19	4A	B4A_N0	2.5 V (default)		12mA (default)	1 (default)
 db_estado[5]	Output	PIN_C2	2A	B2A_N0	2.5 V (default)		12mA (default)	1 (default)
 db_estado[4]	Output	PIN_C1	2A	B2A_N0	2.5 V (default)		12mA (default)	1 (default)
 db_estado[3]	Output	PIN_P14	4A	B4A_N0	2.5 V (default)		12mA (default)	1 (default)
 db_estado[2]	Output	PIN_T14	4A	B4A_N0	2.5 V (default)		12mA (default)	1 (default)
 db_estado[1]	Output	PIN_M8	3B	B3B_N0	2.5 V (default)		12mA (default)	1 (default)
 db_estado[0]	Output	PIN_N9	3B	B3B_N0	2.5 V (default)		12mA (default)	1 (default)
 db_tempo_reacao[6]	Output	PIN_AA22	4A	B4A_N0	2.5 V (default)		12mA (default)	1 (default)
 db_tempo_reacao[5]	Output	PIN_Y21	4A	B4A_N0	2.5 V (default)		12mA (default)	1 (default)
 db_tempo_reacao[4]	Output	PIN_Y22	4A	B4A_N0	2.5 V (default)		12mA (default)	1 (default)
 db_tempo_reacao[3]	Output	PIN_W21	4A	B4A_N0	2.5 V (default)		12mA (default)	1 (default)

Imagem 11: Pinagem do projeto

Segue uma tabela para facilitar o uso do Analog Discovery.

Sinal	Analog Discovery
clock	DIO0
reset	DIO1
iniciar	DIO2
botoes	DIO3-6
repetir	DIO7
nivel	DIO8-9
leds	LED0-3
db_timeout	LED5
espera	LED6
errou	LED7
ganhou	LED8
fim	LED9
mensagem	HEX4-5
tempo_reacao	HEX0-3

7. Referências Bibliográficas

[1][Binary to BCD Converter \(VHDL\) - Logic - eewiki](#)