

UNIVERSIDADE DE SÃO PAULO
Escola Politécnica



PCS3635 – Laboratório Digital I

Projeto da Disciplina: Relatório da Semana 4

Professor Dr. Paulo Cugnasca

Bancada B3

João Pedro Arroyo	N° USP: 12550991
Rafael de Almeida Innecco	N° USP: 12550535

SÃO PAULO, 05/04/2023

1. Especificação do Projeto

1.1 Descrição do Projeto

A proposta de projeto para a segunda parte da disciplina consiste na adaptação do circuito do Jogo Desafio da Memória desenvolvido nas experiências de 1 a 6 para a composição de um jogo baseado em ritmo. O comportamento geral do circuito do projeto é baseado na série de jogos *Dance Dance Revolution* (DDR), produzida pela empresa Konami, com adaptações desenvolvidas pelos alunos, principalmente na interação física do usuário com o jogo. As principais modificações ao circuito podem ser descritas de forma sucinta como:

- Próximas jogadas a serem feitas comunicadas ao jogador;
- O jogador recebe pontuação baseada no tempo entre jogadas;
- Possibilidade de dois botões serem acionados para uma mesma jogada;
- Dois modos de jogo: sequência de jogadas já existente e gravação feita pelo jogador;
 - Caso o jogador grave as jogadas, isso será feito separadamente das jogadas que tentam pontuar;
- A comunicação com o jogador será feita através de dispositivos externos, como botões, chaves, *displays* de 7 segmentos, e *buzzers*.
- Botões externos para capturar a jogada.
- Duas dificuldades para os modos de jogo.

Vale ressaltar que a modificação da comunicação com o jogador é feita tanto para propósito de montar uma estrutura externa de apoio ao jogo quanto para melhorar a qualidade da interação do jogador com o jogo.

1.2 Modificações no circuito

Para que essas modificações sejam atingidas, o comportamento de alguns componentes do circuito deverá ser modificado, enquanto outros serão conservados. Alguns elementos que serão mantidos do Jogo Desafio da Memória são os mecanismos que executam a comparação entre a jogada feita e a jogada esperada, além do mecanismo de contagem de jogadas para acesso à memória.

Mudanças de destaque no circuito incluem a criação de mais uma memória e mais um registrador, ambos relativos a informações de temporização. De maneira similar ao que é feito para as jogadas, a memória guarda o tempo esperado entre duas jogadas consecutivas, enquanto o registrador marca quanto tempo o jogador de fato demorou. Para a contagem do tempo, será utilizado o contador já existente que era responsável pelo timeout, já que pelo funcionamento do jogo não ocorre mais essa situação. Similarmente, o jogo não termina se o jogador comete um erro, ao invés disso, ocorre uma penalização em sua pontuação. Por esse motivo, a saída do comparador entre a jogada feita e esperada não é mais direcionada para a unidade de controle, e sim para um novo componente combinatório responsável pela atribuição de pontos para cada jogada por uma métrica a ser determinada.

Outras entradas desse componente são as saídas da memória e registrador de tempos.

Outro ponto é que, como se quer introduzir a possibilidade de dois botões serem utilizados para a mesma jogada, os mecanismos de detecção de jogadas e comparação entre elas deverão ser modificados.

1.4 Especificação de Requisitos

Código: 01	<input checked="" type="checkbox"/> Funcional <input type="checkbox"/> Não Funcional
Requisito: Mostrar jogadas seguintes ao jogador.	Requisitos associados: 03
Descrição: Permitir que o jogador veja as quatro próximas jogadas em uma matriz de leds 4x4. Cada linha corresponde a uma jogada e a linha mais abaixo é a jogada atual. As jogadas mostradas devem ser atualizadas conforme o andar da partida.	
Prioridade: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Média <input type="checkbox"/> Baixa	Estabilidade: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Média <input type="checkbox"/> Baixa
Rationale: Há duas partes no desenvolvimento do requisito. Primeiro, deve-se alterar o código VHDL do jogo base da memória para que jogadas futuras sejam mostradas de forma contínua ao longo da partida. Os testes da alteração no VHDL podem ser feitos pelo software ModelSim. Em seguida, a programação na placa FPGA e junção de outro componente eventualmente relevante ao projeto deve ser feita para análise do real funcionamento do circuito.	

Tabela 1: Requisito para mostrar quatro jogadas seguintes em matriz de leds.

Código: 02	<input checked="" type="checkbox"/> Funcional <input type="checkbox"/> Não Funcional
Requisito: Mostrar pontuação atual da partida.	Requisitos associados: 03
Descrição: O jogador deve poder ver quantos pontos já marcou ao longo da partida. Ao final, a pontuação atingida deve permanecer à mostra até que algum novo jogo se inicie ou seja dado <i>reset</i> . Devido ao valor da pontuação máxima exceder 99 (decimal) são requeridos três <i>displays</i> de sete segmentos.	
Prioridade: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Média <input type="checkbox"/> Baixa	Estabilidade: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Média <input type="checkbox"/> Baixa
Rationale: A cada acerto a pontuação deve ser incrementada do valor correspondente ao tempo levado para clicar no botão desde o início daquela jogada. A análise da implementação do requisito ocorre em duas etapas: alteração do código VHDL e simulação no Modelsim; teste com a placa FPGA, <i>displays</i> requeridos e eventuais componentes externos previamente implementados.	

Tabela 2: Requisito para mostrar pontuação em tempo real.

Código: 03	<input type="checkbox"/> Funcional <input checked="" type="checkbox"/> Não Funcional
Requisito: Tempo limite por jogada.	Requisitos associados: 04
Descrição: Cada jogada é contabilizada como correta se é feita até um tempo limite. Após este tempo, uma nova jogada é proposta ou há uma temporização até a jogada seguinte, a depender da música gravada.	
Prioridade: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Média <input type="checkbox"/> Baixa	Estabilidade: <input type="checkbox"/> Alta <input checked="" type="checkbox"/> Média <input type="checkbox"/> Baixa
Rationale: Junto deste requisito é implementado o sistema que contabiliza uma pontuação diferente a depender do ponto em que houve acerto. O teste do requisito consiste inicialmente da simulação com o Modelsim; em seguida deve-se testar na placa, para chegar se o funcionamento é mantido como o esperado na escala de tempo real (aproximadamente 1 segundo por jogada) do jogo.	

Tabela 3: Requisito para implementar tempo limite por jogada.

Código: 04	<input type="checkbox"/> Funcional <input checked="" type="checkbox"/> Não Funcional
Requisito: Permitir jogada simultânea (dupla).	Requisitos associados: Não há.
Descrição: Deve ser possível fazer até duas jogadas por vez.	
Prioridade: <input type="checkbox"/> Alta <input checked="" type="checkbox"/> Média <input type="checkbox"/> Baixa	Estabilidade: <input type="checkbox"/> Alta <input checked="" type="checkbox"/> Média <input type="checkbox"/> Baixa
Rationale: A implementação deste requisito exige alteração no componente <i>edge detector</i> ou na forma de usá-lo. Portanto, deve-se ter especial cuidado com o funcionamento do circuito quando os botões são de fato pressionados por um humano, visto que o tempo de duração excede em muito o do <i>clock</i> . Testes com a FPGA são, assim, de grande importância.	

Tabela 4: Requisito para permitir jogada com mais de um botão.

Código: 05	<input checked="" type="checkbox"/> Funcional <input type="checkbox"/> Não Funcional
Requisito: Gerar som correspondente à tecla apertada.	Requisitos associados: Não há.
Descrição: Quando o jogador pressiona certa tecla, o som correspondente deve ser gerado. Será usado um <i>buzzer</i> para isso.	

Prioridade: <input type="checkbox"/> Alta <input checked="" type="checkbox"/> Média <input type="checkbox"/> Baixa	Estabilidade: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Média <input type="checkbox"/> Baixa
<p>Rationale: Um dispositivo externo deve ser usado para produzir o som correspondente quando uma tecla é apertada. Cada botão deve emitir um som de frequência diferente quando pressionado. O teste consiste principalmente de simulações com o Modelsim. Com a FPGA, deve-se checar se as frequências escolhidas são de fato agradáveis.</p>	

Tabela 5: Requisito para gerar som correspondente à tecla apertada.

Código: 06	<input checked="" type="checkbox"/> Funcional <input type="checkbox"/> Não Funcional
Requisito: Modo de jogo em que a sequência é criada.	Requisitos associados: Não há.
<p>Descrição: Segundo modo de jogo, em que a sequência pode ser gravada e jogada pelo jogador.</p>	
Prioridade: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Média <input type="checkbox"/> Baixa	Estabilidade: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Média <input type="checkbox"/> Baixa
<p>Rationale: Quando a gravação é feita, esta música fica disponível até que outra seja guardada em seu lugar. É usada uma segunda memória, então o modo <i>default</i> segue sempre disponível e inalterável. O teste consiste, neste caso, principalmente da simulação com o Modelsim, visto que não há componentes internos e que, devido ao tipo de requisito, dificilmente haveria manifestação de algum problema na placa FPGA que não aparecesse no Modelsim.</p>	

Tabela 6: Requisito para criar modo de jogo em que sequência é gravada e pode ser jogada.

Código: 07	<input checked="" type="checkbox"/> Funcional <input type="checkbox"/> Não Funcional
Requisito: Adiciona seletor de velocidade, aplicável a todos os modos de jogo.	Requisitos associados: Não há.
<p>Descrição: Antes da partida, o jogador pode escolher entre dois modos de velocidade: normal e desafiador.</p>	
Prioridade: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Média <input type="checkbox"/> Baixa	Estabilidade: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Média <input type="checkbox"/> Baixa
<p>Rationale: É um requisito de dificuldade de implementação supostamente menor, mas com impacto possivelmente bastante significativo na experiência do usuário. Devido ao caráter do requisito, deve ser testado extensivamente na placa, posto que o sistema de geração de <i>clock</i> não é o mesmo.</p>	

Tabela 7: Requisito para criação de um segundo modo de dificuldade.

Código: 08	<input checked="" type="checkbox"/> Funcional <input type="checkbox"/> Não Funcional
Requisito: Converter resultado do <i>display</i> para decimal.	Requisitos associados: 02
Descrição: O resultado mostrado nos <i>displays</i> para a pontuação deve estar em decimal.	
Prioridade: <input type="checkbox"/> Alta <input type="checkbox"/> Média <input checked="" type="checkbox"/> Baixa	Estabilidade: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Média <input type="checkbox"/> Baixa
Rationale: Requisito importante para interface homem-máquina. Teste feito principalmente com o Modelsim e <i>testbench</i> específico para o componente.	

Tabela 8: Requisito para conversão do placar de hexadecimal para decimal.

Código: 09	<input checked="" type="checkbox"/> Funcional <input type="checkbox"/> Não Funcional
Requisito: Capturar jogadas com botões externos.	Requisitos associados: Não há
Descrição: Implementação de componentes externos para captura de jogadas.	
Estabilidade: <input checked="" type="checkbox"/> Alta <input type="checkbox"/> Média <input type="checkbox"/> Baixa	Prioridade: <input type="checkbox"/> Alta <input checked="" type="checkbox"/> Média <input type="checkbox"/> Baixa
Rationale: Requisito importante para interface homem-máquina. Teste feito inteiramente em laboratório com circuito programado na placa, posto que se trata de conexão a componentes externos.	

Tabela 9: Requisito para capturar jogadas com botões externos.

1.5 Cronograma

Semana do Projeto	Atividades Propostas:
Semana 1	Requisitos: 03 e 06
Semana 2	Requisitos: 02, 01 e 09
Semana 3	Requisitos: 04 e 08
Semana 4	Requisitos: 05 e 07

Tabela 10: Cronograma de requisitos a serem cumpridos por semana.

2. Planejamento da semana 1

2.1 Implementação dos requisitos 3 e 6

2.1.1 Especificação do funcionamento do circuito

Antes de serem implementadas as novas funcionalidades no código VHDL - inicialmente baseado na experiência 6 -, foi especificado qual deveria ser, em detalhe, o comportamento do circuito. Primeiro, determinou-se que após ativação do sinal “jogar” há um segundo para o jogador se preparar para a partida. Em seguida, foi determinado um tamanho adequado de memória. Tal depende de quantas jogadas devem ser armazenadas, ou seja, de quantas teclas o jogador deverá apertar durante a duração de cada música. Por padrão foi adotado um total de 64 jogadas, além de um tempo de funcionamento de 30 segundos por partida. Consequentemente, deve-se ter um intervalo de meio segundo por jogada. Como o *clock* padrão tem frequência de 1 kHz, tal corresponde a 500 ciclos do relógio. Por fim, caso não se queira uma jogada instantaneamente após o fim da anterior, pode-se marcar “0000” como dado correspondente no *memory initialization file*. Desse modo, o jogador não deve pressionar botão algum - será penalizado com um erro caso aperte alguma tecla, mas não recebe ponto algum caso contrário. Tal é possível devido ao sinal “tem_jogada”, que detecta se alguma tecla foi pressionada e não sofre *reset* ao longo do conjunto de estados referentes à comparação e atualização do placar. Nota-se que é preciso fazer uma temporização caso o jogador faça a jogada antes do tempo limite, para que a jogada seguinte possa ser feita apenas a partir do momento em que tal botão de fato pode ser pressionado. Analogamente, na escrita há o mesmo intervalo de tempo (0.5 segundo) para o qual o contador aponta a determinado endereço da memória. Caso nenhuma jogada seja feita é marcado “0000”. Ademais, há um modo específico para jogar a sequência criada pelo usuário, diferente daquele usado para armazenar a escrita na memória. Naturalmente, como há possibilidade de executar o jogo com duas sequências diferentes é necessária uma segunda memória RAM, também com 64 posições. Abaixo, apresentam-se um trecho do arquivo .mif e o diagrama de transição de estados (DTE) elaborado previamente à codificação da unidade de controle (UC).

```

1 % conteúdo da memoria ram_60x4 %
2 WIDTH=4;
3 DEPTH=60;
4 ADDRESS_RADIX=UNS;
5 DATA_RADIX=BIN;
6 CONTENT
7 BEGIN
8 0 : 0001;
9 1 : 0010;
10 2 : 0100;
11 3 : 1000;
12 4 : 0100;
13 5 : 0010;
14 6 : 0001;
15 7 : 0010;
16 8 : 0100;
17 9 : 1000;
18 10 : 0100;
19 11 : 0010;
20 12 : 0001;
21 13 : 0010;
22 14 : 0100;
23 15 : 1000;
24 16 : 0100;
25 17 : 0010;
26 18 : 0001;
27 ...

```

Figura 1: Arquivo de inicialização da memória.

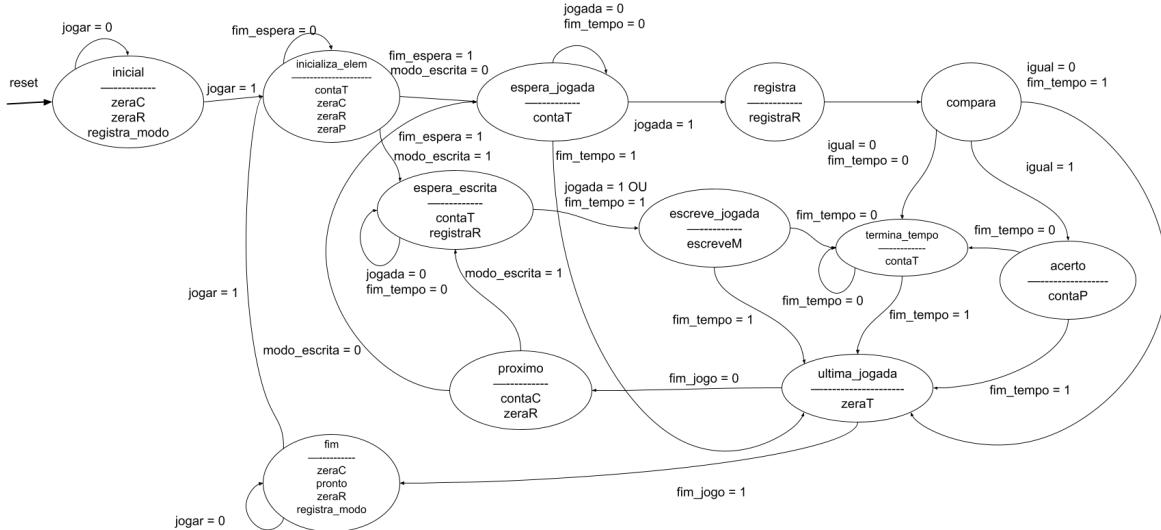


Figura 2: DTE do circuito jogo desafio ritmo.

Os sinais de interface externa do circuito são bastante similares aos da experiência 6, com algumas mudanças. Para os *inputs*, foi adicionado um sinal que representa o modo de jogo escolhido (seleciona_modo). Para os *outputs*, os sinais

“perdeu”, “ganhou”, “db_timeout”, “db_rodada” e “db_enderecolgualRodada” foram removidos, visto que o jogo sempre atinge o mesmo final, havendo variação apenas na pontuação, e que o *loop* externo (rodadas) não existe mais. A figura 3 mostra o conjunto completo de entradas e saídas do circuito.

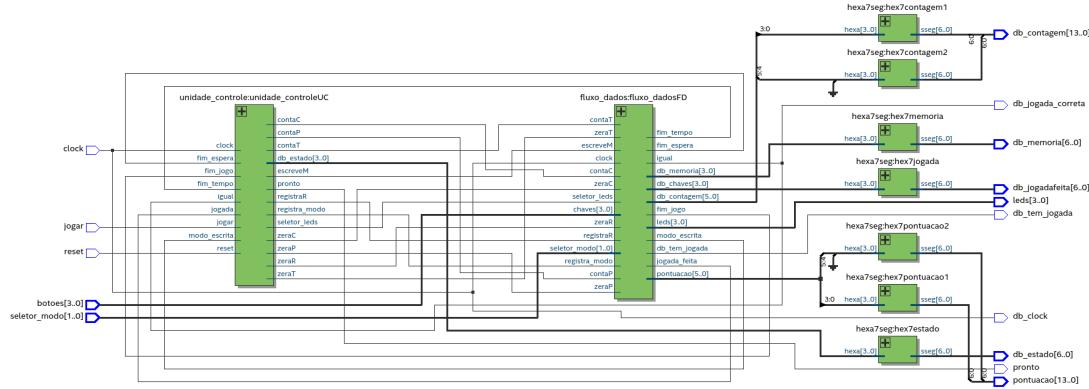


Figura 3: Diagrama de blocos para sinais de interface externa do circuito desafio ritmo.

2.2.2 Alterações no fluxo de dados

As alterações lógicas explicitadas acima tornam necessária a modificação do conjunto de sinais de controle e de condição do circuito. Portanto, alteram-se as entradas e as saídas do fluxo de dados, respectivamente. Ademais, há sinais - de depuração ou saída do circuito - que não são de condição, porém são saídas do fluxo de dados. Por fim, foi necessária a instanciação de novos componentes, bem como alteração de algumas conexões entre componentes pré-existentes.

```

component fluxo_dados
  port (
    clock           : in std_logic;
    chaves          : in std_logic_vector (3 downto 0);
    seletor_modo    : in std_logic_vector (1 downto 0);
    -----
    zeraC           : in std_logic;
    contaC          : in std_logic;
    escreveM        : in std_logic;
    zeraR           : in std_logic;
    registraR       : in std_logic;
    contaT          : in std_logic;
    zeraT           : in std_logic;
    seletor_leds    : in std_logic;
    contaP          : in std_logic;
    zeraP           : in std_logic;
    registra_modo   : in std_logic;
    -----
    igual            : out std_logic;
    fim_jogo         : out std_logic;
    jogada_feita    : out std_logic;
    fim_tempo        : out std_logic;
    fim_espera       : out std_logic;
    modo_escrita     : out std_logic;
    -----
    db_tem_jogada   : out std_logic;
    db_contagem      : out std_logic_vector (5 downto 0);
    db_memoria       : out std_logic_vector (3 downto 0);
    db_chaves         : out std_logic_vector (3 downto 0);
    leds              : out std_logic_vector (3 downto 0);
    pontuacao        : out std_logic_vector (5 downto 0)
  );
end component;

```

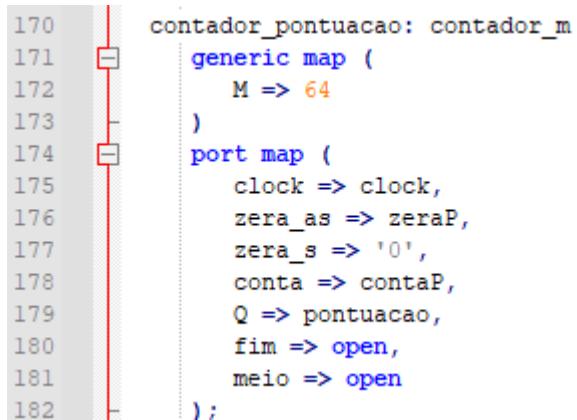
Figura 4: Entidade modificada do fluxo de dados.

A figura 4 mostra a entidade modificada do fluxo de dados. Em primeiro lugar, tem-se a nova entrada “seletor_modo”, que indica se a partida corresponde a um jogo com os dados padrão da música, à gravação de novos dados personalizados ou a uma partida com os dados personalizados. Foram adicionados também “contaP” e “zeraP”, correspondentes ao *enable* e ao *reset* do contador responsável pelo placar da partida. O sinal “registraModo” é um *enable* do registrador responsável por gravar o modo de jogo - não se pode permitir ao jogador mudar o modo durante a partida. A saída “modo_escrita” indica à UC qual deve ser o funcionamento do jogo - segue para o estado correspondente no DTE - enquanto “pontuacao” indica o placar jogada a jogada.

138	signal s_endereco : std_logic_vector (5 downto 0);
139	signal s_dado : std_logic_vector (3 downto 0);
140	signal s_dado_alternativo: std_logic_vector (3 downto 0);
141	signal s_dado_fixo : std_logic_vector (3 downto 0);
142	signal s_not_zera_end : std_logic;
143	signal s_not_escreve : std_logic;
144	signal s_chaves : std_logic_vector(3 downto 0);
145	signal s_chaveacionada : std_logic := '0';
146	
147	signal modo : std_logic_vector(1 downto 0);

Figura 5: Sinais intermediários do fluxo de dados.

A figura 5 mostra os sinais intermediários do fluxo de dados. Primeiro, tem-se “s_endereco”, que corresponde ao endereço da memória correspondente ao modo - nota-se que agora o vetor tem 6 bits para poder cobrir todo o conteúdo necessário. Em seguida, tem-se “s_dado”, que corresponde ao dado da memória em “s_endereco”. Tal é obtido a partir de um multiplexador que tem como entradas “s_dado_fixo” e “s_dado_alternativo” e como seletor “modo”. Os sinais “s_not_zera_end” e “s_not_escreve” são versões negadas dos sinais responsáveis por zerar contador de endereços e por escrever na memória alternativa, visto que esses sinais são ativo baixo dentro de seus componentes e que não se quer trabalhar com a lógica invertida no restante do circuito. Por fim, “s_chaves” corresponde à saída do registrador com os botões.



```
170  contador_pontuacao: contador_m
171  generic map (
172      M => 64
173  )
174  port map (
175      clock => clock,
176      zera_as => zeraP,
177      zera_s => '0',
178      conta => contaP,
179      Q => pontuacao,
180      fim => open,
181      meio => open
182 );
```

Figura 6: Contador da pontuação.

A figura 6 apresenta a instanciação do contador de pontuação. Os sinais usados são os mesmos explicados na figura 4. Nota-se que não há necessidade de usar nenhuma das *flags* relativas ao módulo do contador.

```

225      registrador_jogada: registrador_n
226          generic map(N => 4)
227          port map (
228              clock  => clock,
229              clear  => zeraR,
230              enable  => registraR,
231              D      => chaves,
232              Q      => s_chaves
233          );
234
235      registrador_modo: registrador_n
236          generic map (N => 2)
237          port map (
238              clock  => clock,
239              clear  => '0',
240              enable  => registra_modo,
241              D      => seletor_modo,
242              Q      => modo
243          );

```

Figura 7: Instanciação de registrador de modo.

Como não se quer que o modo seja alterado durante uma partida, adicionou-se um registrador para o modo de jogo. Tal tem como *enable* “*registra_modo*”, que não é ativo durante uma partida.

```

282      with modo(1) select
283          s_dado <= s_dado_fixo when '1'
284          .....           s_dado_alternativo when others;
285
286          db_contagem <= s_endereco;
287          db_memoria  <= s_dado;
288          db_chaves    <= s_chaves;
289          db_tem_jogada <= s_chaveacionada;
290
291          modo_escrita <= modo(0);

```

Figura 8: Sinais de depuração e multiplexador de “*s_dado*”.

Os sinais de depuração seguem idênticos aos da experiência 6, salvo por aqueles que foram removidos. Acima, tem-se o multiplexador responsável pela atribuição do dado da memória correta ao sinal intermediário.

2.2.3 Detalhamento da lógicas de transição e de sinais de controle

O DTE apresentado na figura 2 indica o funcionamento da unidade (UC) do circuito, porém não explicita como foi feita a implementação em VHDL. Essa subseção, portanto, tem essa finalidade, além de detalhar alguns funcionamentos.

```

26  entity unidade_controle is
27  port (
28      clock          : in std_logic;
29      -- Sinais de condicao
30      reset          : in std_logic;
31      jogar          : in std_logic;
32      fim_jogo       : in std_logic;
33      jogada         : in std_logic;
34      igual          : in std_logic;
35      fim_tempo      : in std_logic;
36      fim_espera     : in std_logic;
37      modo_escrita   : in std_logic;
38      -----
39      -- Sinais de controle
40      zeraC          : out std_logic;
41      contaC         : out std_logic;
42      -----
43      zeraR          : out std_logic;
44      registraR      : out std_logic;
45      -----
46      zeraP          : out std_logic;
47      contaP         : out std_logic;
48      -----
49      registra_modo  : out std_logic;
50      -----
51      pronto         : out std_logic;
52      -----
53      contaT         : out std_logic;
54      zeraT          : out std_logic;
55      -----
56      escreveM        : out std_logic;
57      -----
58      seletor_leds    : out std_logic;
59      -- Sinais de depuracao
60      db_estado       : out std_logic_vector(3 downto 0)
61  );
62 end entity;

```

Figura 9: Entidade da UC do projeto.

Em relação ao circuito da experiência 5 alguns sinais foram adicionados e outros foram mantidos, porém passaram a cumprir funções ligeiramente diferentes. Na parte de sinais de condição, o final do jogo depende de “fim_jogo”, que agora é o “rco” de um contador M com módulo 63. O sinal “fim_espera” segue indicando fim do *delay* inicial - tempo de preparação do jogador -, que passou de 2 para 1 segundo. O “modo_escrita” depende diretamente do seletor do modo e vem, intermediado pelo fluxo de dados, do modo escolhido na entrada. O par “zeraP” e “contaP” é responsável pelo *reset* e *enable* do contador da pontuação (módulo 64). O sinal de controle “registra_modo” é um *enable* para escrita do modo de jogo no respectivo registrador, visto que não é permitido alterar o modo durante o jogo.

```

65      type t_estado is (
66          inicial, inicializa_elem,
67          espera_jogada, registra, compara, acerto,
68          espera_escrita, escreve_jogada,
69          termina_tempo, ultima_jogada, proximo, fim
70      ); -- novo estado para escrita
71      signal Eatual, Eprox: t_estado;

```

Figura 10: Conjunto de estados da UC do projeto.

Alguns estados, como “fim_erro” e “fim_timeout”, ou aqueles relacionados a rodadas, foram excluídos. Os estados correspondentes ao antigo *loop* interno não foram modificados.

```

86      Eprox <|
87          inicial      when (
88              Eatual = inicial and jogar='0'
89          ) else
90              inicializa_elem when (
91                  (Eatual = inicial and jogar = '1') or
92                  (Eatual = fim and jogar = '1') or
93                  (Eatual = inicializa_elem and fim_espera = '0')
94          ) else
95              espera_jogada when (
96                  (Eatual = inicializa_elem and (fim_espera = '1' and modo_escrita = '0')) or
97                  (Eatual = proximo and modo_escrita = '0') or
98                  (Eatual = espera_jogada and jogada = '0' and fim_tempo = '0')
99          ) else
100             registra       when Eatual = espera_jogada and jogada='1' else
101             compara        when Eatual = registra else
102             espera_escrita when (
103                 (Eatual = inicializa_elem and (fim_espera = '1' and modo_escrita = '1')) or
104                 (Eatual = proximo and modo_escrita = '1') or
105                 (Eatual = espera_escrita and (jogada = '0' and fim_tempo = '0'))
106         ) else
107             escreve_jogada when Eatual = espera_escrita and (jogada = '1' or fim_tempo = '1') else
108             acerto         when Eatual = compara and igual = '1' else
109             termina_tempo when (
110                 (Eatual = compara and ((igual = '0' or jogada = '0') and fim_tempo = '0')) or
111                 (Eatual = acerto and fim_tempo = '0') or
112                 (Eatual = escreve_jogada and fim_tempo = '0') or
113                 (Eatual = termina_tempo and fim_tempo = '0')
114         ) else
115             ultima_jogada when (
116                 (Eatual = termina_tempo and fim_tempo = '1') or
117                 (Eatual = escreve_jogada and fim_tempo = '1') or
118                 (Eatual = acerto and fim_tempo = '1') or
119                 (Eatual = compara and ((igual = '0' or jogada = '0') and fim_tempo = '1')) or
120                 (Eatual = espera_jogada and fim_tempo = '1')
121         ) else
122             proximo        when Eatual = ultima_jogada and fim_jogo = '0' else
123             fim            when (
124                 (Eatual = ultima_jogada and fim_jogo = '1') or
125                 (Eatual = fim and jogar = '0')
126         ) else
127             inicial;

```

Figura 11: Lógica de transição de estados da UC (Moore).

A partir do estado inicial, atinge-se apenas “inicializa_elem” - o que ocorre quando “jogar”=1. A única outra forma de se atingir esse estado é a partir de “fim”, caso se queira jogar novamente. Em seguida, a depender do modo (escrita ou não) a respectiva espera é atingida. Tem-se então, um *loop* interno para cada caso, havendo estados análogos e compartilhados (esperar pelo fim do tempo e última jogada). O funcionamento de ambos segue estrutura análoga aos *loops* da experiência 5 - *loop* interno para modo normal e externo para modo escrita. Tal segue até que o fim do jogo é atingido. Então, transiciona-se para o estado “fim”, até que algum jogo seja iniciado.

```

129      -- logica de saida (maquina de Moore)
130      with Eatural select
131          zeraC <=      '1' when inicial | inicializa_elem | fim, -- novos estados;
132          ...           '0' when others;
133      with Eatural select
134          contaC <=    '1' when proximo,
135          ...           '0' when others;
136      with Eatural select
137          zeraR <=      '1' when inicial | inicializa_elem | proximo | fim,
138          ...           '0' when others;
139      with Eatural select
140          registraR <= '1' when registra | espera_escrita,
141          ...           '0' when others;
142      with Eatural select
143          contaT <=    '1' when espera_jogada | inicializa_elem | espera_escrita | termina_tempo,
144          ...           '0' when others;
145      with Eatural select
146          zeraT <=      '1' when ultima_jogada,
147          ...           '0' when others;
148      with Eatural select
149          zeraP <=      '1' when inicializa_elem,
150          ...           '0' when others;
151      with Eatural select
152          contaP <=    '1' when acerto,
153          ...           '0' when others;
154      with Eatural select
155          pronto <=    '1' when fim,
156          ...           '0' when others;
157      with Eatural select
158          escreveM <=   '1' when escreve_jogada,
159          ...           '0' when others;
160      with Eatural select
161          registra_modo <= '1' when inicial | fim,
162          ...           '0' when others;
163      with Eatural select
164          seletor_leds <= '1' when inicializa_elem,
165          ...           '0' when others;

```

Figura 12: Lógica da atribuição dos sinais de controle da UC.

O sinal “zeraP” é ativo apenas em “inicializa_elem”, visto que não há razão para zerar o placar em qualquer momento que não seja o início de um novo jogo. Já “contaP” é ativo apenas em “acerto”, que tanto para modo “00” quanto “10” corresponde a uma jogada correta. O sinal “pronto” é ativo apenas em “fim” - agora o fim do jogo é único - e “registra_modo” é alto em “fim” ou “inicial”, visto que são os estados nos quais se pode começar um novo jogo, sendo necessária a opção de escolha de um novo modo de jogo.

```

167      -- saida de depuracao (db_estado)
168      with Eatural select
169          db_estado <= "0000" when inicial,           -- 0
170          ...           "0001" when inicializa_elem,     -- 1
171          ...           "0010" when espera_jogada,       -- 2
172          ...           "0011" when registra,           -- 3
173          ...           "0100" when compara,           -- 4
174          ...           "0101" when acerto,            -- 5
175          ...           "0110" when espera_escrita,     -- 6
176          ...           "0111" when escreve_jogada,     -- 7
177          ...           "1000" when termina_tempo,      -- 8
178          ...           "1001" when ultima_jogada,       -- 9
179          ...           "1010" when proximo,           -- A
180          ...           "1111" when fim,              -- F
181          ...           "1110" when others;          -- E
182      end architecture fsm;

```

Figura 13: Código dos estados da UC para depuração.

A figura 14 apresenta o novo código de estados para o sinal “db_estado”.

2.2.4 Simulação do circuito com o ModelSim

Para checar a correta implementação das funcionalidades foram elaborados *testbenches* que cobrem três cenários que podem ocorrer no jogo: o jogador acerta todas as jogadas, o jogador erra algumas jogadas e o jogador grava algumas jogadas e depois joga com elas. As tabelas de cada um dos casos de teste estão no apêndice A.

Cenário 1 - Modo de jogo base com performance perfeita:



Figura 14: Primeira parte do cenário de teste 1.

Na figura acima, primeiro tem-se a inicialização da partida. O modo de jogo é selecionado como “jogo base” (código “10”) e em seguida é dado início à partida com a ativação do sinal “jogar”. Então dá-se entrada a uma série de ciclos de comparação e incremento da pontuação - caso a jogada seja correta. O sinal caso, que permite uma melhor visualização do resultado do teste, está sincronizado com o término de cada iteração do *loop* interno, ou seja, com o retorno a “espera_jogada”. Em seguida, antes do término do tempo limite (500 ciclos de *clock*), é feita uma jogada - correta - programa para durar 100 ciclos de relógio, como se pode ver em azul. Tal leva a um incremento unitário na pontuação, sendo que o sinal que detecta a corretude da jogada permanece ativo até o final daquele ciclo de comparação. Em seguida, espera-se o término do intervalo de tempo reservado para aquela jogada. Atinge-se o caso seguinte e novamente há um incremento no contador, atualizando o endereço da memória. É importante ressaltar que manter o botão pressionado não realiza jogada alguma, conforme o funcionamento do *edge detector*.



Figura 15: Segunda parte do cenário de teste 1.

Os ciclos descritos continuam ocorrendo até o final da partida. Como o jogador acertou todas as jogadas neste teste, seu placar final, como esperado, é 63. Nota-se que após o fim da partida há *reset* do contador, porém não do placar, visto que o jogador deve poder ver quanto marcou quando sua partida terminou. O sinal pronto também levanta, indicando o fim da partida.

Cenário 2 - Modo de jogo base com erros por jogada incorreta e tempo limite:

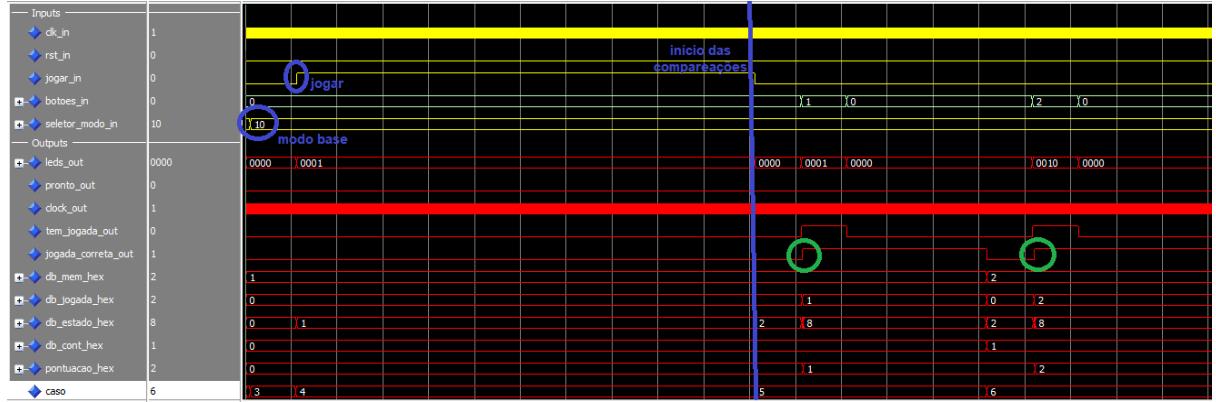


Figura 16: Primeira parte do cenário de teste 2.

A primeira parte do teste é exatamente igual à do cenário 1, visto que o modo é o mesmo e não há erro em nenhuma das duas primeiras comparações.

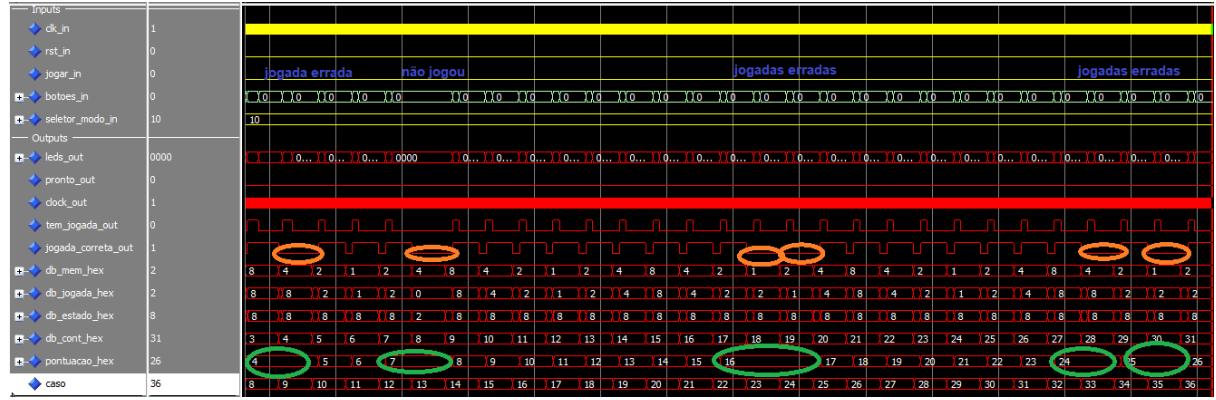


Figura 17: Segunda parte do cenário de teste 2.

A figura acima mostra o trecho em que são feitas jogadas erradas - não há jogada errada algum em outro trecho. Primeiro, no caso 9, é feita uma jogada incorreta, o sinal “jogada_correta” - em laranja - não levanta e o contador do placar não é incrementado. Em seguida, no caso 13, não é feita jogada alguma e o mesmo efeito é observado. Nos casos 23 e 24 é testado o erro de duas jogadas consecutivas e como esperado o contador de placar segue inalterado. Por fim, há jogadas erradas nos casos 33 e 35 - intercaladas com uma correta. Em todos os casos o comportamento observado corresponde ao esperado.

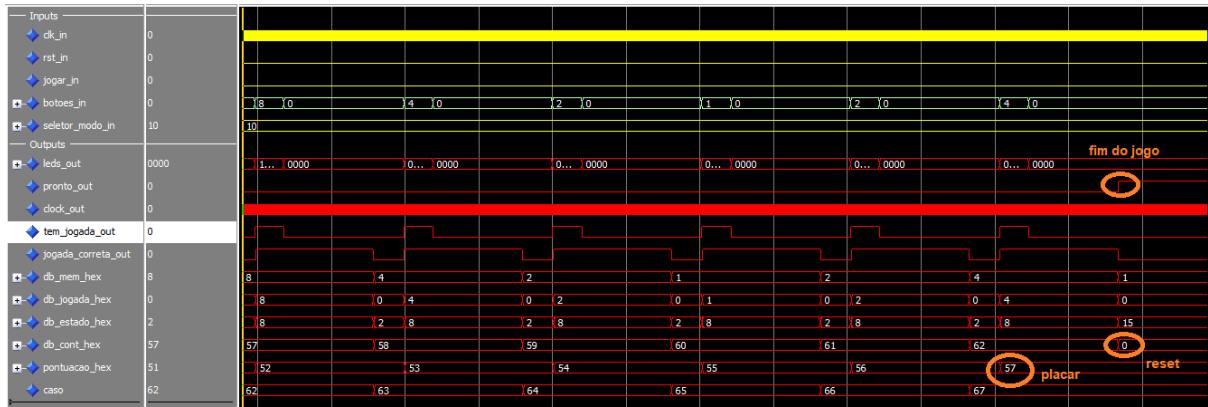


Figura 18: Terceira parte do cenário de teste 2.

Conforme explicado no parágrafo anterior, foram introduzidas 6 jogadas incorretas ao *testbench*. Como a pontuação obtida é 57 em 63, de fato foram contabilizados os 6 erros. Novamente há *reset* do contador de endereços, mas não do placar, e o sinal “pronto” indica o término da partida.

Cenário 3 - Gravação de um jogo e partida no modo personalizado:



Figura 19: Primeira parte do cenário de teste 3.

A inicialização da partida difere apenas no código de gravação, que é “01” para a escrita. Em seguida, tem-se os ciclos de escrita na memória destinada à gravação da sequência do jogador, que é bastante similar ao da partida normal, como se pode ver no DTE. Nota-se que, caso a memória não seja igual ao dado pressionado em certo ciclo, após alguns ciclos de *clock* a saída “db_memoria” passará a mostrar o mesmo dado contido em “db_jogada_hex”. Tal confirma a escrita correta na memória. Por simplicidade, todas as escritas feitas foram “0001”.



Figura 20: Segunda parte do cenário de teste 3.

A figura correspondente à segunda parte da onda mostra o momento em que a operação de escrita termina e é dado início ao jogo personalizado - código “00”. Há destaque para o *reset* do contador, visto que caso contrário não seria possível realizar mais de uma partida em sequência. O retângulo mostra os dados da memória correspondentes aos endereços de “db_contagem”. Como esperado, todos são “0001” - dado escrito na parte anterior. Na linha com a elipse verde tem-se o sinal “jogada_correta”, que levantou em todos os casos.

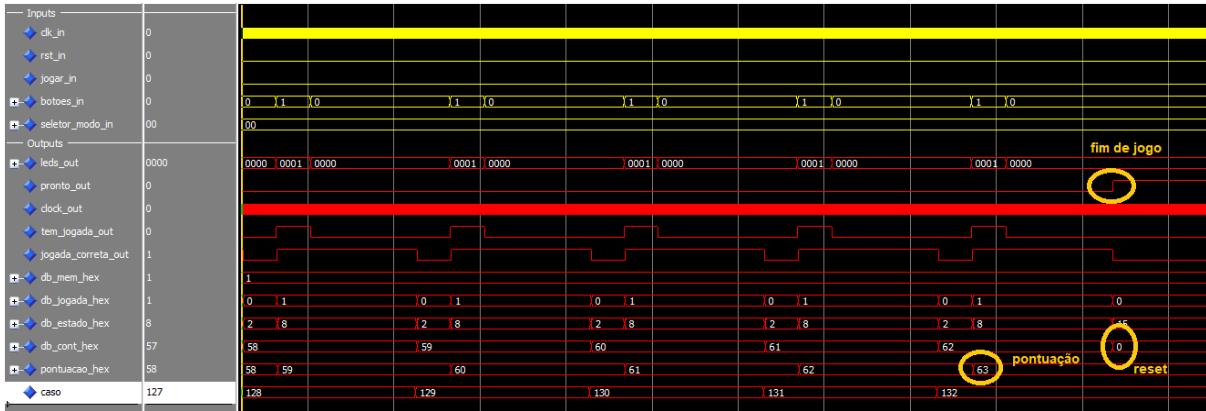


Figura 21: Terceira parte do cenário de teste 3.

Por fim, tem-se o término da partida personalizada. Como configurado no *testbench*, não houve nenhum erro e foi atingido o placar máximo. Novamente há *reset* para o contador e o fim de jogo é indicado pelo sinal “pronto”. Observa-se que o jogador não precisa apertar algum botão necessariamente, sendo marcado “0000”, que corresponde a um período no qual, na partida, botão algum deve ser pressionado. Naturalmente, tal implica que não necessariamente o número de pontos possível de ser obtido é 63, sendo este na verdade o valor máximo - quando nenhum endereço da memória contém o dado “0000”.

2.3 Preparação para atividades práticas

Após a verificação do funcionamento do circuito através das simulações descritas acima, foi criado um projeto no *software* Quartus. A única mudança feita nos arquivos de descrição de *hardware* foi a utilização da arquitetura da memória RAM referente ao uso do arquivo .mif para inicialização. Assim, as duas memórias são instanciadas da forma:

```
memoria_jogada_fixa: entity work.ram_64x4 (ram_mif) -- usar esta linha para Intel Quartus
--memoria_jogada_fixa: entity work.ram_64x4 (ram_modelsim) -- usar arquitetura para ModelSim
port map (
    clk      => clock,
    endereco => s_endereco,
    dado_entrada => s_chaves,
    we      => '1', -- we ativo em baixo, essa memória nunca é sobreescrita
    ce      => '0',
    dado_saida => s_dado_fixo
);
```

Figura 22: Instanciação da memória que não é modificada.

```
memoria_jogada_alternativa: entity work.ram_64x4 (ram_mif) -- usar esta linha para Intel Quartus
--memoria_jogada_alternativa: entity work.ram_64x4 (ram_modelsim) -- usar arquitetura para ModelSim
port map (
    clk      => clock,
    endereco => s_endereco,
    dado_entrada => s_chaves,
    we      => s_not_escreve, -- we ativo em baixo
    ce      => '0',
    dado_saida => s_dado_alternativo
);
```

Figura 23: Instanciação da memória que pode ser modificada pelo jogador.

Em seguida foram geradas as saídas das ferramentas *RTL Viewer* e *State Machine Viewer*, que indicam a arquitetura interna do circuito:

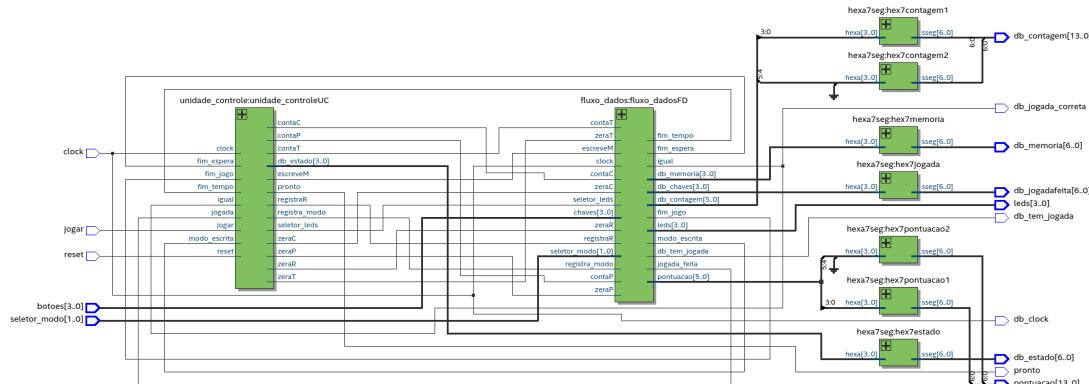


Figura 24: Diagrama RTL do circuito completo.

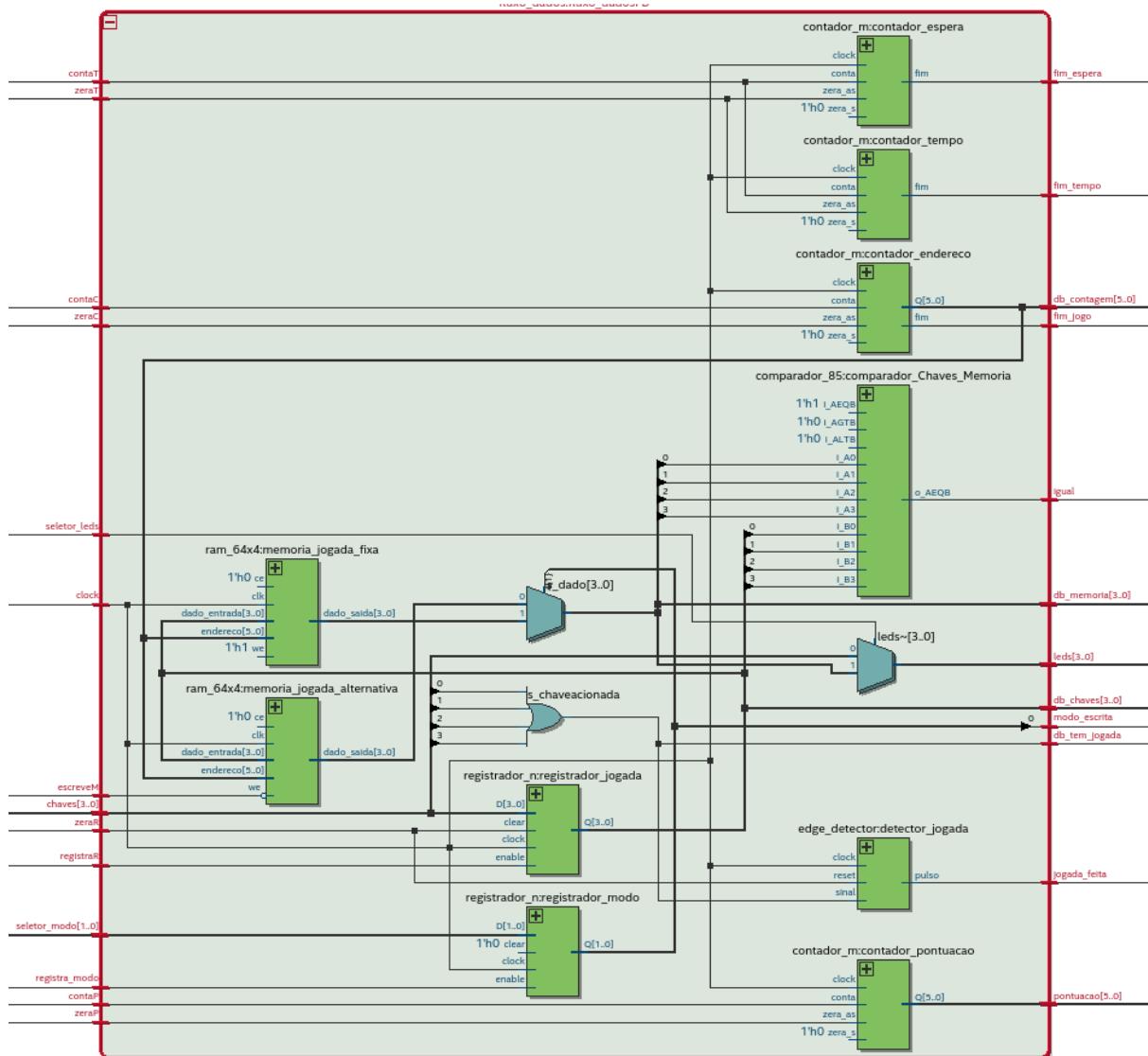


Figura 25: Diagrama RTL do fluxo de dados.

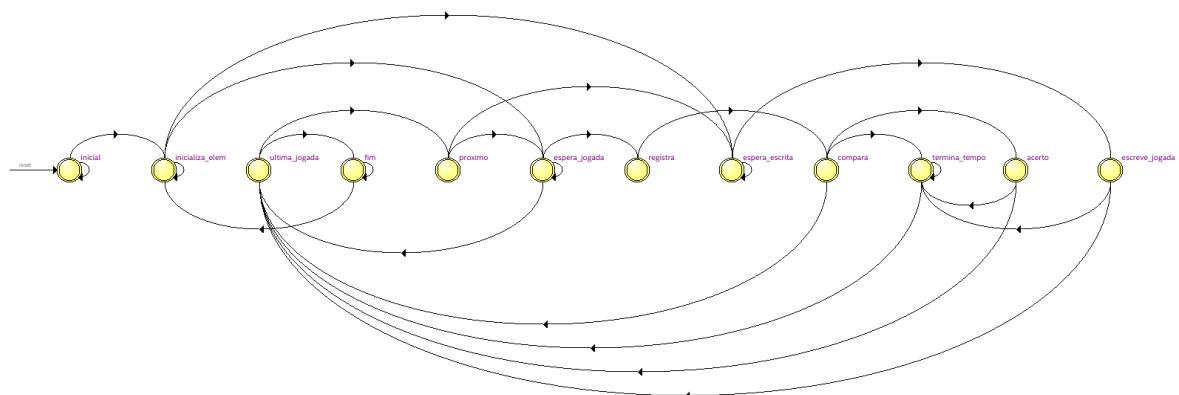


Figura 26: Diagrama de Transição de estados gerado pelo Quartus.

	Source State	Destination State	Condition
1	acerto	termina_tempo	(!fim_tempo)
2	acerto	ultima_jogada	(fim_tempo)
3	compara	acerto	(igual)
4	compara	termina_tempo	(!fim_tempo).(!igual)
5	compara	ultima_jogada	(fim_tempo).(!igual)
6	escreve_jogada	termina_tempo	(!fim_tempo)
7	escreve_jogada	ultima_jogada	(fim_tempo)
8	espera_escrita	escreve_jogada	(!jogada).(fim_tempo) + (jogada)
9	espera_escrita	espera_escrita	(!jogada).(!fim_tempo)
10	espera_jogada	espera_jogada	(!fim_tempo).(!jogada)
11	espera_jogada	registra	(jogada)
12	espera_jogada	ultima_jogada	(fim_tempo).(!jogada)
13	fim	fim	(!jogar)
14	fim	inicializa_elem	(jogar)
15	inicial	inicializa_elem	(jogar)
16	inicial	inicial	(!jogar)
17	inicializa_elem	espera_jogada	(fim_espera).(!modo_escrita)
18	inicializa_elem	inicializa_elem	(!fim_espera)
19	inicializa_elem	espera_escrita	(fim_espera).(modo_escrita)
20	proximo	espera_jogada	(!modo_escrita)
21	proximo	espera_escrita	(modo_escrita)
22	registra	compara	
23	termina_tempo	termina_tempo	(!fim_tempo)
24	termina_tempo	ultima_jogada	(fim_tempo)
25	ultima_jogada	fim	(fim_jogo)
26	ultima_jogada	proximo	(!fim_jogo)

Figura 27: Tabela de transição de estados gerada pelo Quartus.

2.3.1 Planejamento da pinagem

Também utilizando o software Quartus, as entradas e saídas da entidade principal do circuito foram atribuídas a pinos na placa FPGA, seguindo a tabela de pinagem abaixo:

Sinal	Pino na Placa DE0-CV	Pino no FPGA	Analog Discovery
CLOCK	GPIO_0_D0	PIN_N16	Patterns – Clock – 1 KHz – DIO
RESET	GPIO_0_D1	PIN_B16	StaticIO – Button 0/1 – DIO1
INCIAR/JOGAR	GPIO_0_D3	PIN_C16	StaticIO – Button 0/1 – DIO2

BOTOES(0)	GPIO_0_D11	PIN_R22	StaticIO - Button 0/1 - DIO4
BOTOES(1)	GPIO_0_D13	PIN_T22	StaticIO - Button 0/1 - DIO5
BOTOES(2)	GPIO_0_D15	PIN_N19	StaticIO - Button 0/1 - DIO6
BOTOES(3)	GPIO_0_D17	PIN_P19	StaticIO - Button 0/1 - DIO7
SELETOR_MODO(0)	GPIO_1_D11	PIN_J18	StaticIO - Switch 0/1 - DIO8
SELETOR_MODO(1)	GPIO_1_D13	PIN_G11	StaticIO - Switch 0/1 - DIO9
PRONTO	GPIO_1_D15	PIN_J11	StaticIO - LED - DIO10
LEDS(0)	GPIO_1_D17	PIN_A15	StaticIO - LED - DIO12
LEDS(1)	GPIO_1_D19	PIN_L8	StaticIO - LED - DIO13
LEDS(2)	GPIO_1_D21	PIN_B15	StaticIO - LED - DIO14
LEDS(3)	GPIO_1_D23	PIN_E14	StaticIO - LED - DIO15
PONTUACAO(0)	Display HEX3	PIN_Y16	-
PONTUACAO(1)	Display HEX3	PIN_W16	-
PONTUACAO(2)	Display HEX3	PIN_Y17	-
PONTUACAO(3)	Display HEX3	PIN_V16	-
PONTUACAO(4)	Display HEX3	PIN_U17	-
PONTUACAO(5)	Display HEX3	PIN_V18	-
PONTUACAO(6)	Display HEX3	PIN_V19	-
PONTUACAO(7)	Display HEX4	PIN_U20	-
PONTUACAO(8)	Display HEX4	PIN_Y20	-
PONTUACAO(9)	Display HEX4	PIN_V20	-
PONTUACAO(10)	Display HEX4	PIN_U16	-
PONTUACAO(11)	Display HEX4	PIN_U15	-
PONTUACAO(12)	Display HEX4	PIN_Y15	-
PONTUACAO(13)	Display HEX4	PIN_P9	-
db_clock	Led LEDR0	PIN_AA2	-
db_tem_jogada	Led LEDR1	PIN_AA1	-
db_jogada_correta	Led LEDR2	PIN_W2	-
db_estado(0)	Display HEX0	PIN_U21	-
db_estado(1)	Display HEX0	PIN_V21	-
db_estado(2)	Display HEX0	PIN_W22	-
db_estado(3)	Display HEX0	PIN_W21	-
db_estado(4)	Display HEX0	PIN_Y22	-
db_estado(5)	Display HEX0	PIN_Y21	-
db_estado(6)	Display HEX0	PIN_AA22	-
db_memoria(0)	Display HEX1	PIN_AA20	-
db_memoria(1)	Display HEX1	PIN_AB20	-
db_memoria(2)	Display HEX1	PIN_AA19	-
db_memoria(3)	Display HEX1	PIN_AA18	-

db_memoria(4)	Display HEX1	PIN_AB18	-
db_memoria(5)	Display HEX1	PIN_AA17	-
db_memoria(6)	Display HEX1	PIN_U22	-
db_jogada_feita(0)	Display HEX2	PIN_Y19	-
db_jogada_feita(1)	Display HEX2	PIN_AB17	-
db_jogada_feita(2)	Display HEX2	PIN_AA10	-
db_jogada_feita(3)	Display HEX2	PIN_Y14	-
db_jogada_feita(4)	Display HEX2	PIN_V14	-
db_jogada_feita(5)	Display HEX2	PIN_AB22	-
db_jogada_feita(6)	Display HEX2	PIN_AB21	-

Tabela 11: Pinagem sugerida para sinais de interface da entidade externa.

Vale ressaltar que a saída “db_contagem” não foi associado a nenhum pino da placa FPGA, pois não há uma quantidade suficiente de *displays* de sete segmentos. O sinal foi mantido na entidade caso seja necessário para depuração durante as atividades práticas.

2.3.2 Uso do Analog discovery

Como os dispositivos físicos que serão utilizados no projeto final (como leds e botões) ainda não foram adquiridos, será utilizado o dispositivo Analog Discovery para controlar alguns sinais de entrada e saída. Abaixo, tem-se o esquema de números correspondentes a cada fio do Analog Discovery, além do esquema de pinos da placa FPGA.

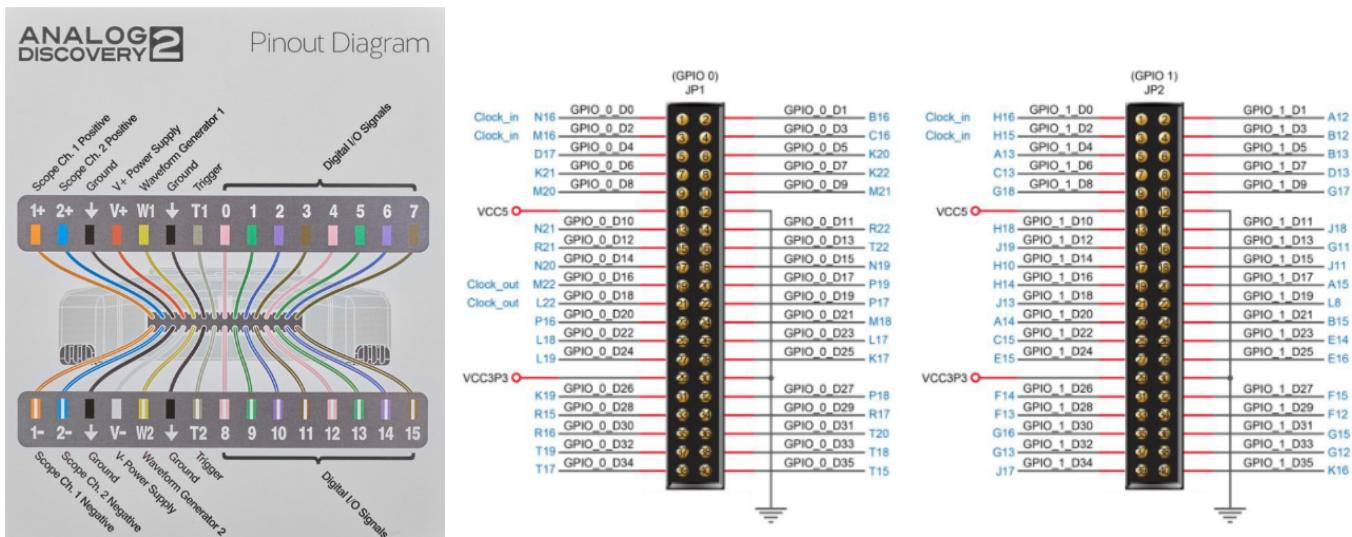


Figura 28: Informações para conexão do Analog Discovery à placa FPGA e pinagem de seus sinais.

Para controlar os sinais de entrada ligados ao Analog Discovery e observar os sinais de saída, é utilizado o *software* Waveforms. A foto abaixo, retirada da apostila “Dicas Analog Discovery” da disciplina, mostra a forma geral da interface do programa e como o sinal de *clock* deve ser definido.

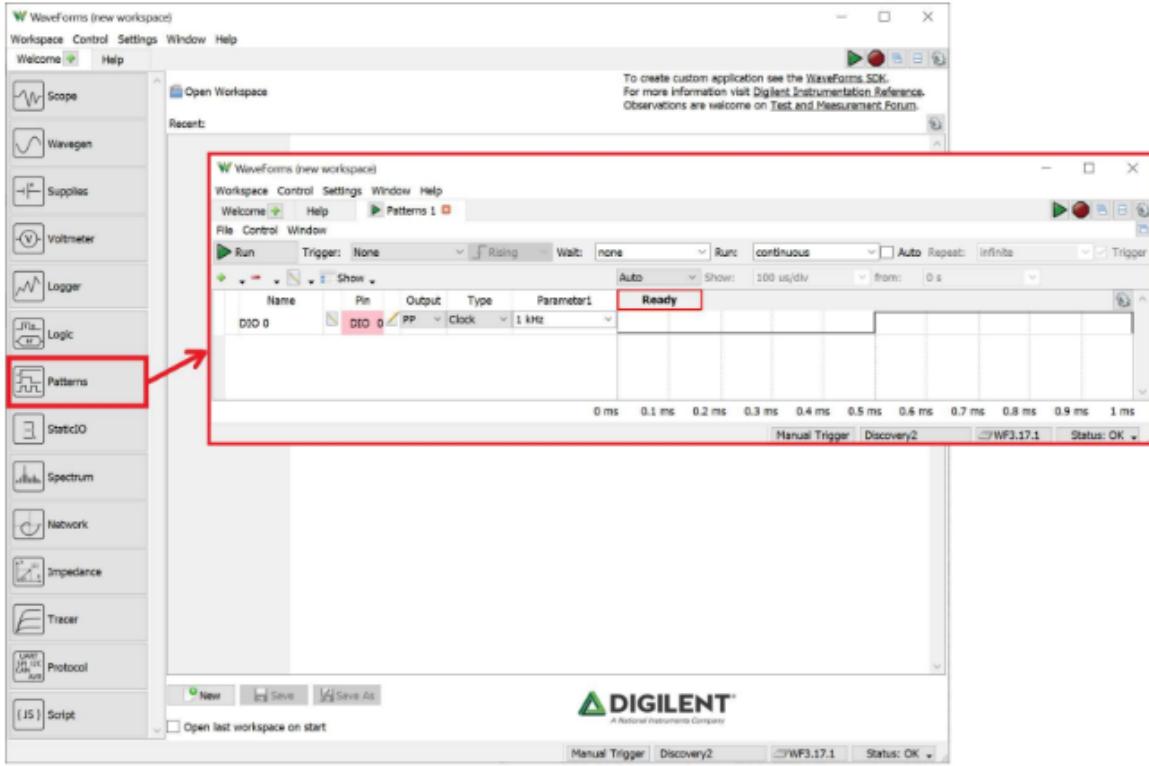


Figura 29: Interface do Waveforms e localização de comandos para configuração do relógio.

Após clicar em “Patterns”, na área à esquerda, em branco, clica-se com o botão direito do *mouse*, abrindo uma caixa de opções na qual é possível adicionar sinais - no caso apenas DIO0 será configurado dessa forma. Posteriormente, define-se o tipo como *clock* e ajusta-se o parâmetro (frequência) para 1kHz. Por fim, destaca-se que para o sinal realmente iniciar sua operação é preciso ativar o *play*, clicando na seta verde que aparece no título do *pattern* criado - no caso *patterns 1*. Quando ativo, a seta verde será substituída por um botão vermelho. Então, é preciso configurar os demais sinais, que correspondem à categoria StaticIO. As demais entradas do circuito devem ser declaradas como botões, enquanto as saídas devem ser declaradas como LEDs. Nota-se que, no canto esquerdo superior, pode-se clicar em *view* e depois em nomes, para renomear cada sinal, inicialmente denominado a partir de seu número de entrada.

3. Atividades experimentais da semana 1

O projeto no *software* Quartus descrito acima foi utilizado para sintetizar o circuito na placa FPGA da bancada. A compilação completa do projeto foi bem sucedida, assim como a programação da placa. Complementarmente, foi feita a ligação da placa com o dispositivo Analog Discovery, seguindo a descrição fornecida no planejamento.

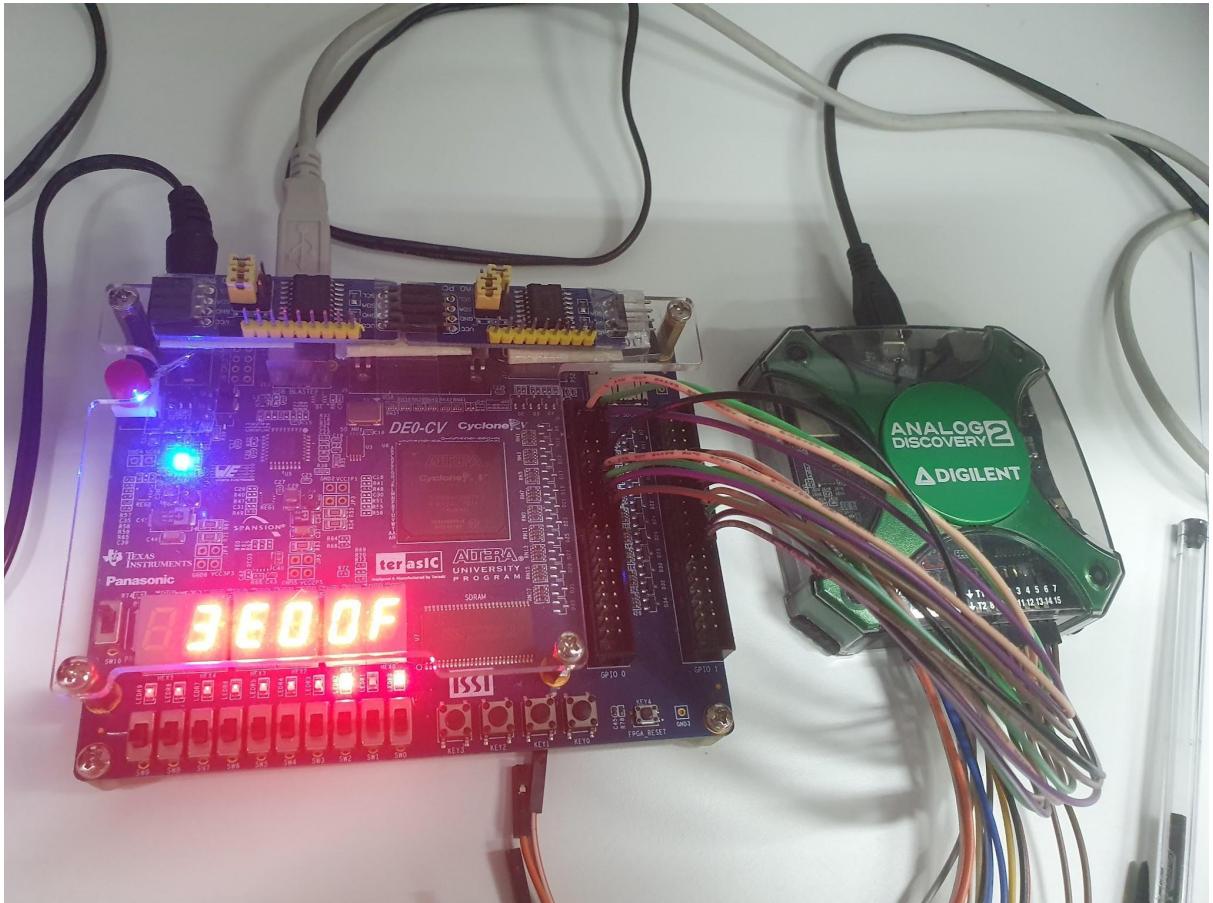


Figura 30: montagem experimental

Com isso, foram executados os testes de cada um dos cenários previstos. Os testes indicaram um pequeno erro na lógica de controle do circuito, pois o contador de pontuação não era zerado com a ativação do sinal “reset”. Para corrigir esse problema, o estado “inicial” foi incluído na lista de sinais que ativam o sinal de controle “zeraP”, da forma:

```
with Eatual select
  zeraP <=
    others      => null;
    '1' when inicializa_elem | inicial,
    '0' when others;
```

Figura 31: modificação no sinal de controle “zeraP”

Além disso, determinou-se que o tempo entre jogadas estava muito curto, e não era possível executar os planos de teste que requerem acertos de várias jogadas em sequência. Para minimizar esse problema, após discussão com o monitor, o tempo entre jogadas foi incrementado de 500 ciclos de *clock* (0.5 segundo) para 1000 ciclos de *clock* (1 segundo), o que foi feito através da modificação direta do módulo de um dos contadores, exemplificado abaixo:

```
contador_tempo: contador_m -- conta passagem de tempo entre jogadas
generic map (
    M => 1000
)
port map (
    clock => clock,
    zera_as => zeraT,
    zera_s => '0',
    conta => contaT,
    Q => open,
    fim => fim_tempo,
    meio => open
);
```

Figura 32: modificação no contador de tempo entre jogadas.

Feitas essas modificações, o projeto foi compilado novamente no Quartus, e a placa FPGA foi reprogramada. Os testes foram feitos com a nova temporização e seus resultados foram satisfatórios.

4. Reavaliação das especificações

Após implementação dos requisitos correspondentes à semana 1, foi feita uma reavaliação do cronograma do projeto, bem como um detalhamento dos componentes de *display* externos utilizados e de funcionalidades que seriam mais ou menos desejáveis para o jogo.

Primeiramente, de forma oposta à definição inicial, o requisito que envolve a possibilidade de jogada com mais de um botão teve sua prioridade reduzida - e deixou de ser pré-requisito para outros requisitos, visto que uma versão que suporta todas as funcionalidades do jogo exceto esta pôde ser desenvolvida. Em lugar disso, uma maior granularidade no sistema de pontuação tornou-se um objetivo de maior relevância, visto que torna a lógica do jogo mais complexa, desafiadora e instigante. Por exemplo, quer-se dar diferentes pontuações dependendo do quanto o jogador demorou para apertar o botão correspondente.

No que tange aos componentes externos, houve uma definição de como o sistema de *displays* deve funcionar. Primeiro, considerou-se a possibilidade de usar a comunicação serial da placa FPGA e montar um programa em alguma linguagem para mostrar, em uma matriz 4 por 4, os 4 dados seguintes da memória ao jogador. No entanto, após avaliação das sugestões dos monitores, optou-se por uma forma mais simples e direta: colocar a matriz de *leds* em uma *protoboard*, ligando a cada linha a respectiva jogada (linha mais abaixo corresponde à atual, a segundo de baixo para cima corresponde à jogada seguinte, e assim por diante). Essa implementação tem como problema uma menor capacidade de adaptação. Por exemplo, seria mais fácil adicionar novos dados caso fosse montada uma interface gráfica no computador. No entanto, considerou-se estável a decisão de quais dados deveriam ser mostrados pelos *leds* - apenas um determinado número de jogadas seguintes. Foi definido que essa implementação deveria ser feita o quanto antes, sendo necessário realocar os requisitos no cronograma - e detalhar o requisito que envolve os *displays*.

Ao se fazer a depuração da implementação na placa FPGA, notou-se que alguém não familiarizado com a lógica usada nos *displays* não conseguiria compreender corretamente o número de acertos. Foram usados dois *displays* em série, sendo o mais à esquerda o bit mais significativo. No entanto, tal está em hexadecimal, e para melhor interface humano-máquina seria mais adequado converter a representação para decimal. Tal deve ser adicionado como um novo requisito.

5. Planejamento da semana 2

5.1 Descrição lógica da implementação dos requisitos 01 e 02

5.1.1 Introdução

Conforme o cronograma, na semana 2 deveriam ser implementados os seguintes requisitos: mostrar a pontuação ao jogador e mostrar as jogadas seguintes ao jogador. Ambos estão relacionados ao uso de componentes externos à placa, posto que as 4 jogadas seguintes devem ser mostradas em uma matriz 4 por 4 de *leds* e o placar deve ser mostrado por um *display* externo à placa - que contenha dois *displays* de sete segmentos, para que seja possível mostrar até a pontuação máxima, que excede 15. Para ambos requisitos há, naturalmente, uma necessidade prévia de alteração do código VHDL. Tal refere-se, respectivamente, à implementação de um sistema de pontuação com maior grau de complexidade e à definição da estrutura da memória RAM. Por fim, visto que não há necessidade de arrumar qualquer erro da semana anterior, caso haja tempo deveriam-se implementar botões externos à placa FPGA para desempenhar o papel atualmente cumprido pelo Analog Discovery.

A análise da dupla quanto à dificuldade e tempo requerido para implementar cada requisito mostrou que seria mais eficiente implementar: alteração na memória e sistema de *leds*, resistores e protoboard - parte técnica e parte física; modificação no sistema de pontuação - apenas parte técnica-; ligação externa aos botões. Desse modo, a ligação com os botões substitui a integração externa com o *display*, que deverá ser realocada para outra semana.

5.1.2 Estrutura da memória RAM

Durante a semana 1, foi utilizada a estrutura de memória RAM definida no *memory initialization file* (mif) mostrada na figura 33. No entanto, considerando a estrutura de matriz 4 por 4 usada para mostrar as jogadas, notou-se que, mesmo com um *delay* inicial, não haveria sentido em, num primeiro momento, já colocar a primeira jogada na linha mais abaixo. Um argumento análogo é feito para os dados finais do jogo. Deste modo, decidiu-se adicionar 4 dados “0000” tanto no início (posições 0,1,2,3) quanto no fim (posições 64,65,66,67) do arquivo .mif. Por conseguinte, deve-se lidar com a mudança do tamanho da memória, alterando o parâmetro “DEPTH” do mif. Além disso, passa a ser necessário usar um bit adicional para cobrir todos os endereços da memória que contém dados e a própria arquitetura do componente ram_64x4.vhd deve ser alterada quanto ao tamanho do *array* de *bit_vectors*. Nota-se que a saída *leds* do circuito também passa a ter tamanho 16, visto que todas as 4 jogadas anteriores precisam ser mostradas. Para isso a memória também passou a ter como saída os 3 dados anteriores e o atual concatenados.

```

1   conteudo da memoria ram_72x4 %
2  WIDTH=4;
3  DEPTH=72;
4  ADDRESS_RADIX=UNS;
5  DATA_RADIX=BIN;
6  CONTENT
7  BEGIN
8    0 : 0000;
9    1 : 0000;
10   2 : 0000;
11   3 : 0000;
12   4 : 0001;
13   5 : 0010;
14   6 : 0100;
15   7 : 1000;
16   8 : 0100;
17   9 : 0010;
18   10 : 0001;
19   11 : 0010;
20   12 : 0100;
21   13 : 1000;
22   14 : 0100;
23   15 : 0010;
24   16 : 0001;
66   58 : 0001;
67   59 : 0010;
68   60 : 0100;
69   61 : 1000;
70   62 : 0100;
71   63 : 0010;
72   64 : 0001;
73   65 : 0010;
74   66 : 0100;
75   67 : 1000;
76   68 : 0000;
77   69 : 0000;
78   70 : 0000;
79   71 : 0000;
80   END;
81

```

Figura 33: Novo arquivo mif

```

32  entity ram_8x4 is
33  generic (
34    constant S : integer := 128
35  );
36  port (
37    clk      : in std_logic;
38    endereco : in std_logic_vector(natural(ceil(log2(real(S)))) - 1 downto 0);
39    dado_entrada : in std_logic_vector(3 downto 0);
40    we       : in std_logic;
41    ce       : in std_logic;
42    dado_saida : out std_logic_vector(3 downto 0);
43    next_data : out std_logic_vector(15 downto 0)
44  );
45 end entity ram_8x4;

```

Figura 34: Nova entidade memória ram

```

75  -- Indica valor das próximas 4 posições de memória
76  next_data <= memoria(to_integer(unsigned(endereco))) & memoria(to_integer(unsigned(endereco) + 1))
77  & memoria(to_integer(unsigned(endereco) + 2)) & memoria(to_integer(unsigned(endereco) + 3));
78

```

Figura 35: Novo trecho na arquitetura da memória ram

Nota-se que a introdução destes novos dados nulos torna irrelevante o *delay* inicial de 1 segundo para os modos que não envolvem escrita - neste o jogo pula os dados nulos iniciais. Portanto, tal foi removido. Por fim, nota-se que os *testbenches* também precisam ser adaptados para o novo funcionamento do jogo, porém as modificações são menores, principalmente relacionadas à temporização. Isso pois,

além da exclusão do *delay* de 1 segundo, o reajuste de *timeout* para um segundo comentado no relatório anterior ainda não havia sido implementado.

5.1.3 Modelo de pontuação com maior granularidade

Na semana 1, o modelo adotado correspondia apenas a acrescer em uma unidade a pontuação caso a jogada fosse feita no intervalo definido e em não fazer nada caso contrário. No entanto, o sistema foi alterado para exigir mais do jogador e proporcionar um jogo mais interessante. Com o *timeout* alterado para um segundo por jogada, foi elaborado um sistema em que a pontuação é uma potência de 2 dependente do intervalo de tal segundo em que o jogador pressiona o botão. Foi definido que caso apertasse em até 0.4 segundo, ganharia 4 pontos; caso apertasse em até 0.75, ganharia 2 pontos; e caso apertasse no restante do tempo ganharia 1 ponto. Dessa descrição, nota-se que a pontuação é sempre uma potência de dois, de modo que um *shift register* é capaz de adequar a pontuação conforme o passar do tempo. No entanto, agora é preciso passar três parâmetros ao contador: tempo total disponível, tempo até 4 pontos e tempo até 2 pontos. Por isso, outro novo componente elaborado foi o contador modificado. Além disso, como agora lida-se com pontuações não unitárias, é necessário um somador para atualizar a pontuação, e não um contador normal. Por fim, destaca-se que os 4 dados finais (“0000”) não interferem no módulo do contador, tendo em vista que servem apenas para não haver erro de acesso a endereços além do tamanho da memória.

Abaixo apresentam-se os novos componentes:

```
34  entity shift_register is
35  generic (
36      constant N           : integer := 8;
37      constant reset_value : integer := 0
38  );
39  port (
40      clock      : in std_logic;
41      clear      : in std_logic;
42      enable_l   : in std_logic;
43      enable_s   : in std_logic;
44      D          : in std_logic_vector (N-1 downto 0);
45      Q          : out std_logic_vector (N-1 downto 0)
46  );
47 end entity shift_register;
```

Figura 36: Entidade do componente *shift register*.

O *shift register* conta com duas constantes genericamente definidas: o tamanho N da entrada e saída e o valor de *reset*. Há uma entrada para *clock* e outra para o *reset* (“*clear*”). A entrada D corresponde a um dado que pode ser carregado enquanto Q é a saída do registrador. Por fim, “enable_s” corresponde ao *enable* para *shift*, e permite um *shift* para a direita (divisão por 2) sempre que for alto. Já “enable_l” permite a carga do dado de entrada na saída.

```

49  architecture comportamental of shift_register is
50    signal IQ: std_logic_vector(N-1 downto 0);
51 begin
52
53   process(clock, clear, enable_l, enable_s, IQ)
54   begin
55     if (clear = '1') then IQ <= std_logic_vector(to_unsigned(reset_value, N));
56     elsif (clock'event and clock='1') then
57       if (enable_l ='1') then IQ <= D;
58       elsif (enable_s = '1') then IQ <= "0" & IQ(N-1 downto 1);
59       else IQ <= IQ;
60       end if;
61     end if;
62     Q <= IQ;
63   end process;
64
65 end architecture comportamental;

```

Figura 37: Arquitetura do componente shift register.

Define-se o sinal intermediário “IQ”. Após o início, tem-se um processo sensível aos *enables*, ao *clock*, ao *reset* e ao sinal intermediário. Primeiro, caso o “clear” esteja ativo a saída recebe o valor de *reset*. Em seguida, na borda de subida do *clock* avalia-se o *enable* de *load*, que se ativo faz com que “IQ” receba “D”. Caso contrário, avalia-se o *enable* de *shift*, que se ativo faz com que “IQ” receba um *shift* para a direita. Por fim, atribui-se o sinal intermediário à saída “Q”.

```

34  entity contador_modificado is
35    generic (
36      constant M : integer := 100; -- módulo do contador
37      constant P1 : integer := 50; -- primeiro ponto de interesse
38      constant P2 : integer := 25 -- segundo ponto de interesse
39    );
40    port (
41      clock      : in  std_logic;
42      zera_as   : in  std_logic;
43      zera_s    : in  std_logic;
44      conta     : in  std_logic;
45      load      : in  std_logic;
46      D         : in  std_logic_vector(natural(ceil(log2(real(M)))) - 1 downto 0);
47      Q         : out std_logic_vector(natural(ceil(log2(real(M))))-1 downto 0);
48      fim       : out std_logic;
49      ponto_1   : out std_logic;
50      ponto_2   : out std_logic
51    );
52  end entity contador_modificado;

```

Figura 38: Entidade do contador modificado.

Nota-se que o contador modificado conta com três constantes genericamente definidas: uma correspondente ao módulo do contador e outras duas correspondentes a pontos de interesse da contagem. Além disso, o componente tem as entradas típicas: *clock*, *reset* síncrono e assíncrono, *enable* de contagem e de *load* e carga de dado. Por fim, as saídas são: a saída Q do contador e as *flags* que mostram se foi atingido o módulo do contador, o primeiro ponto de interesse ou o segundo ponto de interesse.

```

54      architecture comportamental of contador_modificado is
55          signal IQ: integer range 0 to M-1;
56      begin
57
58          process (clock,zera_as,zera_s,conta,IQ, load)
59          begin
60              if zera_as='1' then    IQ <= 0;
61              elsif rising_edge(clock) then
62                  if zera_s='1' then IQ <= 0;
63                  elsif load='1' then IQ <= to_integer(unsigned(D));
64                  elsif conta='1' then
65                      if IQ=M-1 then IQ <= 0;
66                      else           IQ <= IQ + 1;
67                      end if;
68                  else                 IQ <= IQ;
69                  end if;
70              end if;
71          end process;
72
73          -- saida fim
74          fim <= '1' when IQ=M-1 else
75              '0';
76
77          -- saida ponto_1
78          ponto_1 <= '1' when IQ=P1-1 else
79              '0';
80          -- saida ponto_2
81          ponto_2 <= '1' when IQ=P2-1 else
82              '0';
83          -- saida Q
84          Q <= std_logic_vector(to_unsigned(IQ, Q'length));
85
86      end architecture comportamental;

```

Figura 39: Arquitetura do contador modificado.

Primeiro, define-se um sinal intermediário “IQ” análogo ao anterior. Na arquitetura, tem-se um processo sensível ao sinal intermediário, ao *clock* e aos *reset* e *enable*. Caso o *reset* assíncrono esteja ativo, a saída volta a ser zero. Em seguida, apenas na borda de subida do relógio, avalia-se: se o *reset* síncrono for ativo a saída sofre *reset*; então, se o *enable* de *load* estiver ativo, a carga de dado é feita ;caso contrário, se o *enable* estiver ativo a saída é incrementada em uma unidade, até atingir o módulo. Fora do processo, a saída “Q” recebe o sinal intermediário em vetor binário - *casting* desde *integer*. Por fim, as *flags* são configuradas de modo análogo, levantando só se a saída do contador atinge o valor definido menos um - dado que começa em 0.

```

25  entity somador is
26      generic (
27          size : natural := 8
28      );
29
30      port (
31          A, B : in std_logic_vector (size - 1 downto 0); --inputs
32          F : out std_logic_vector (size - 1 downto 0);
33          Z : out std_logic; --zero flag
34          Ov : out std_logic; --overflow flag
35          Co : out std_logic --carry out flag
36      );
37  end entity somador;

```

Figura 40: Entidade do somador.

O somador tem como constante genericamente definida o tamanho dos operandos. As entradas correspondem aos operandos A e B, enquanto as saídas correspondem ao resultado da operação F e às flags Z, Ov e Co - se F é zero, se houve overflow ou se houve carry out.

```

44 begin
45     -- Implementação da soma por Carry-lookahead
46     CA(0) <= '0'; --Carry-in = 0
47     GEN_2: for j in 1 to (size) generate
48         CA(j) <= (A(j - 1) and B(j - 1)) or ((A(j - 1) or B(j - 1)) and CA(j - 1)); --Carry da soma do j-ésimo std_logic
49     end generate GEN_2;
50
51     GEN_3: for j in 0 to (size - 1) generate
52         SAI(j) <= A(j) xor B(j) xor CA(j);
53     end generate GEN_3;
54
55     -- Implementação de flags
56
57     -- Flag de overflow
58     Ov <= CA(size) xor CA(size - 1);
59
60     -- Flag de Carry-out
61     Co <= CA(size);
62
63
64     Z <= '1' when SAI = std_logic_vector(to_unsigned(0, size)) else
65     Z <= '0';
66
67     F <= SAI;
68
69
70
71 end architecture dataflow;

```

Figura 41: Arquitetura do somador no modelo *carry look ahead*.

O primeiro *carry in* é definido como 0, visto que se faz uma soma, e não uma subtração. Em seguida, o *for generate* GEN_2 monta linha a linha a expressão para o j-ésimo *carry* da soma, em função do anterior e dos atuais. Para determinar a saída, primeiro atribuída ao sinal intermediário SAI, faz-se um xor entre A, B e o *carry* da posição j linha a linha. Por fim, implementam-se as *flags* e atribui-se o sinal intermediário à saída.

Nota-se que todos estes componentes são instanciados no fluxo de dados, que teve algumas de suas operações modificadas. No entanto, como a lógica de funcionamento do circuito não foi alterada, a unidade de controle foi muito pouco alterada, não havendo sentido em se ter uma seção dedicada a comentar suas mudanças. Antes de se comentarem as alterações do fluxo de dados, será apresentado o modelo pensado para conexão com os componentes externos.

5.1.4 Leds para mostrar jogadas seguintes ao jogador e botões

Tendo implementado as alterações descritas em 7.1.2, poderia-se conectar o circuito aos correspondentes componentes externos. Para mostrar as jogadas seguintes serão usados 16 *leds* (matriz 4 por 4), 4 de cada cor. A conexão com a placa FPGA deve ser feita usando os pinos GPIO. Portanto, são necessários cabos macho fêmea, com a ponta usada na *protoboard*. Além disso, são necessários resistores para regular a corrente - 1 por *led*. Usando a mesma lógica da semana anterior, pode-se trocar os botões que antes correspondiam ao Analog Discovery para um conjunto de quatro botões externos à placa. Serão necessários cabos para conexão e resistores para regular a corrente.

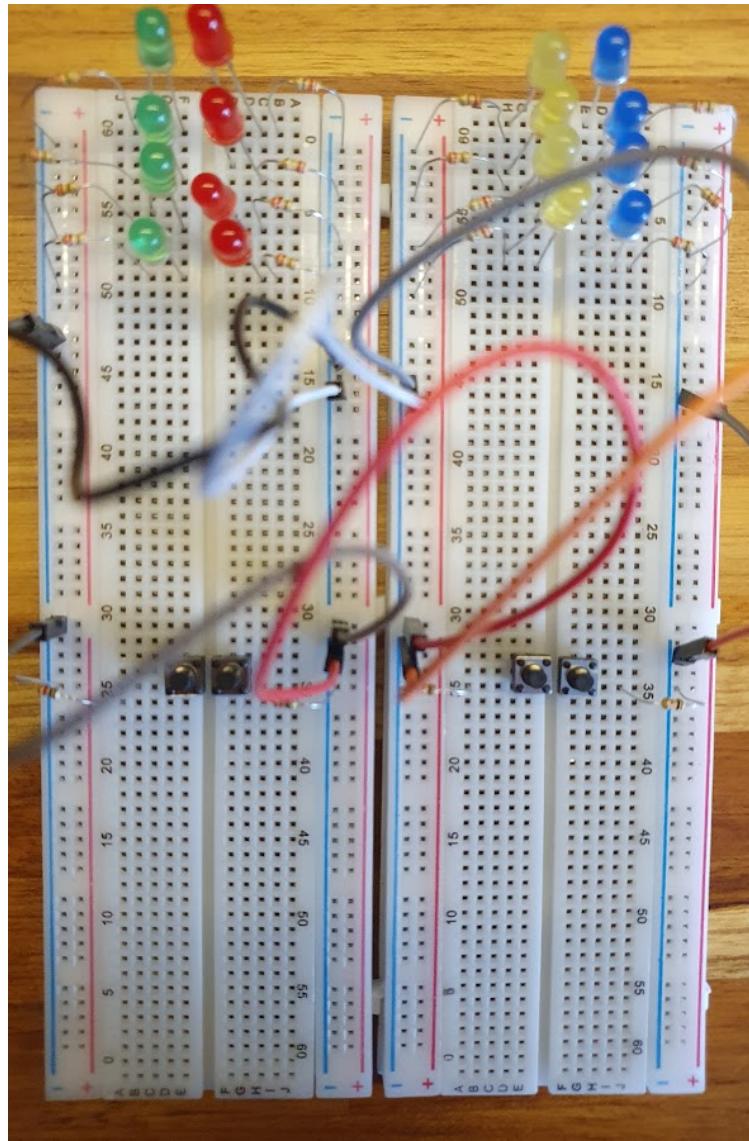


Figura 42: Foto mostrando o circuito na protoboard

5.1.5 Modificações no fluxo de dados do circuito

Primeiro, foram adicionados novos sinais à entidade, relativos à mecânica de granularidade da pontuação. Tais são “diminuiP_jogada” e “resetaP_jogada”, ambos sinais de controle. Além disso, a pontuação foi aumentada para 8 bits. Isso decorre

do fato de que, com a nova regra de pontuação, o máximo passou a ser 256, que exige um bit a mais.

```

22  entity fluxo_dados is
23  port (
24      clock          : in std_logic;
25      chaves         : in std_logic_vector (3 downto 0);
26      seletor_modo   : in std_logic_vector (1 downto 0);
27
28      zeraC          : in std_logic; -- novo nome: zeraC -> zeraC
29      contaC         : in std_logic; -- novo nome: contaC -> contaC
30      carregaC       : in std_logic;
31      escreveM       : in std_logic;
32      zeraR          : in std_logic;
33      registraR      : in std_logic;
34      contaT         : in std_logic;
35      zeraT          : in std_logic;
36      atualizaP       : in std_logic;
37      diminuiP_jogada : in std_logic;
38      resetaP_jogada  : in std_logic;
39      zeraP          : in std_logic;
40      registra_modo   : in std_logic;
41
42      igual           : out std_logic;
43      fim_jogo        : out std_logic; -- novo sinal: saída do contador de rodada
44      jogada_feita    : out std_logic;
45      fim_tempo       : out std_logic;
46      fim_espera      : out std_logic;
47      modo_escrita    : out std_logic;
48
49      db_tem_jogada  : out std_logic;
50      db_contagem     : out std_logic_vector (6 downto 0);
51      db_memoria      : out std_logic_vector (3 downto 0);
52      db_chaves        : out std_logic_vector (3 downto 0);
53      leds             : out std_logic_vector (15 downto 0);
54      pontuacao       : out std_logic_vector (8 downto 0)
55  );
56 end entity;

```

Figura 43: Entidade do fluxo de dados.

Assim como mencionado no trecho que discute as alterações da memória RAM, uma nova saída foi criada. Portanto, a declaração deste componente também foi modificada para contemplar isso. As demais alterações provém da adição dos novos componentes: somador, *shift register* e contador modificado. Como a declaração é quase idêntica à entidade de tais componentes, cujas figuras 4, 6 e 8 já mostram, tal não será mostrado em nova figura.

Para atender à nova mecânica de pontuação e da saída leds os sinais intermediários abaixo foram adicionados.

```

187  signal diminui_pontuacao  : std_logic;
188  signal p_increment         : std_logic_vector(3 downto 0);
189
190  signal p_entrada, p_saida  : std_logic_vector(8 downto 0);
191  signal p_inc_expand        : std_logic_vector(8 downto 0);
192
193  signal led_intermediario1 : std_logic_vector(15 downto 0);
194  signal led_intermediario2 : std_logic_vector(15 downto 0);

```

Figura 44: Novos sinais intermediários do fluxo de dados.

O contador responsável pelo endereço da memória do respectivo modo de jogo foi trocado para um contador modificado, com interesse apenas na *flag*

correspondente ao módulo do contador. A modificação, de fato, entra na possibilidade de se poder dar *load*, sendo usada uma entrada específica para isso, bem como um sinal de *enable* correspondente. O que se quer carregar é a constante 4, em binário, com extensão para 7 bits, que corresponde ao tamanho usado para o endereço. O interesse está em fazer esse carregamento no início do modo de gravação, pois tal impossibilita o jogador de gravar nos 4 primeiros dados da RAM não fixa, mantendo o *delay* inicial da primeira jogada.

```

205      contador_endereco: contador_modificado
206      generic map (
207          M => 68
208      )
209      port map (
210          clock => clock,
211          zera_as => zeraC,
212          zera_s => '0',
213          conta => contaC,
214          load => carregaC,
215          D => std_logic_vector(to_unsigned(4, 7)),
216          Q => s_endereco,
217          fim => fim_jogo,
218          ponto_1 => open,
219          ponto_2 => open
220      );

```

Figura 45: Instanciação do contador modificado para mapear endereços.

A figura 46 apresenta a implementação da lógica de pontuação. Nota-se que, primeiro, instancia-se um *shift register* capaz apenas de realizar *shift* - opção de *load* bloqueada. A saída 'Q' corresponde ao incremento da pontuação. Nota-se que o *shift* ocorre apenas quando o sinal de controle “diminuiP_jogada” for alto e contador modificado tiver sua saída igual a algum dos extremos de pontuação - 400 períodos de *clock* ou 750 períodos de *clock*. Nota-se que o valor para o qual o *reset* ocorre é 4, que corresponde à pontuação máxima por jogada, atingida caso não tenha havido *shift* algum. Como a pontuação, no entanto, tem mais de 4 bits, faz-se um processo de *sign extend* antes, concatenando-se com zeros mais significativos. Em seguida, a saída atual da pontuação é somada ao incremento da correspondente jogada, e então tal deve ser colocado na entrada da pontuação. Não se faz uso de qualquer *flag*.

```

235      diminui_pontuacao <= (primeiro_ponto or segundo_ponto) and diminuiP_jogada;
236      incremento_pontuacao: shift_register
237      generic map (
238          N          => 4,
239          reset_value => 4
240      )
241      port map (
242          clock      => clock,
243          clear      => resetaP_jogada,
244          enable_l   => '0',
245          enable_s   => diminui_pontuacao,
246          D          => "0000",
247          Q          => p_increment
248      );
249
250      p_inc_expand <= "00000" & p_increment;
251      calcula_pontuacao: somador
252      generic map (
253          size => 9
254      )
255      port map (
256          A  => p_saida,
257          B  => p_inc_expand,
258          F  => p_entrada,
259          Z  => open,
260          Ov => open,
261          Co => open
262      );

```

Figura 46: Lógica de pontuação.

Outra instância do contador modificado foi usada para rastrear o tempo levado para executar cada jogada. Como comentado anteriormente, os tempos definidos como limite para as pontuações 4, 2 e 1 são, respectivamente: 400 ciclos de *clock*, 750 ciclos de *clock* e 1000 ciclos de *clock*.

```

341      contador_tempo: contador_modificado -- conta passagem de tempo entre jogadas
342      generic map (
343          M  => 1000,
344          P1 => 750,
345          P2 => 400
346      )
347      port map (
348          clock => clock,
349          zera_as => zeraT,
350          zera_s => '0',
351          conta => contaT,
352          load  => '0',
353          D    => std_logic_vector(to_unsigned(0, 10)),
354          Q    => open,
355          fim  => fim_tempo,
356          ponto_1 => primeiro_ponto,
357          ponto_2 => segundo_ponto
358      );

```

Figura 47: Contador de tempo por jogada modificado.

A saída “leds” corresponde aos últimos 4 dados da respectiva memória durante o jogo. Desse modo, é necessário fazer um multiplexador com o modo como seletor.

```

380      with modo select
381          leds <= led_intermediario1 when "10",
382              led_intermediario2 when "01",
383              std_logic_vector(to_unsigned(0, 16)) when others;

```

Figura 48: Multiplexador para saída *leds*.

5.2 Simulações com o ModelSim

Para checar a correta implementação das funcionalidades foram elaborados *testbenches* que cobrem três cenários que podem ocorrer no jogo: o jogador acerta todas as jogadas, havendo alguns acertos parciais, o jogador erra algumas jogadas e o jogador grava algumas jogadas e depois joga com elas. As tabelas de cada um dos casos de teste estão no apêndice A.

Cenário 1 - Modo de jogo base com três 2 pontos e dois 1 ponto:

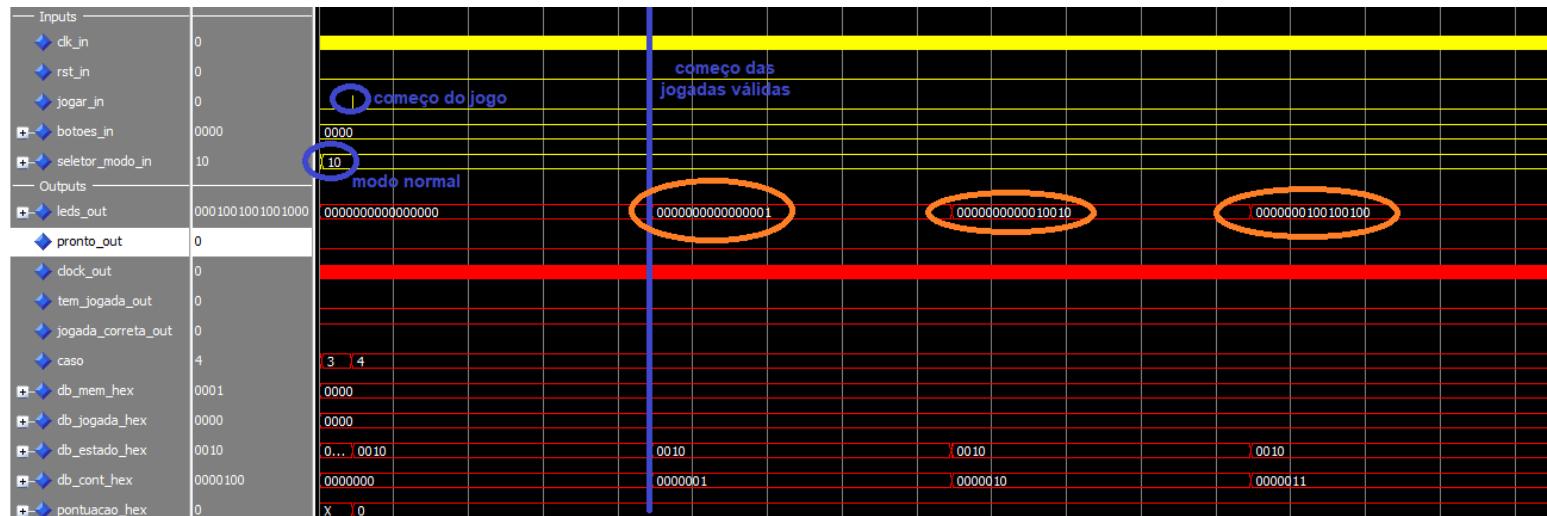


Figura 49: Primeira parte do cenário de teste 1.

Na figura acima, primeiro tem-se a inicialização da partida. O modo de jogo é selecionado como “jogo base” (código “10”) e em seguida é dado início à partida com a ativação do sinal “jogar”. Então dá-se entrada a uma série de ciclos de comparação e incremento da pontuação - caso a jogada seja correta. O sinal caso, que permite uma melhor visualização do resultado do teste, está sincronizado com o término de cada iteração do *loop* interno, ou seja, com o retorno a “espera_jogada”. Em seguida, antes do término do tempo limite (1000 ciclos de *clock*), é feita uma jogada - correta - programa para durar 100 ciclos de relógio. Tal leva a um incremento unitário na pontuação, sendo que o sinal que detecta a corretude da jogada permanece ativo até o final daquele ciclo de comparação. Em seguida, espera-se o término do intervalo de tempo reservado para aquela jogada. Atinge-se o caso seguinte e novamente há um incremento no contador, atualizando o endereço da memória. É importante ressaltar que manter o botão pressionado não realiza jogada alguma, conforme o funcionamento do *edge detector*. O sinal *leds* armazena cada novo dado da memória em seus 4 bits menos significativos e dá um *shift left* de 4 bits para os demais, na prática - não é essa a implementação. A figura

mostra várias jogadas, sendo possível ver o funcionamento do sinal nas elipses laranjas.



Figura 50: Segunda parte do cenário de teste 1.

A figura 50 destaca como funcionam na prática jogadas não perfeitas. Primeiro, em verde, tem-se uma jogada feita até 400 ciclos de *clock* da rodada, que resulta em incremento de 4 pontos na pontuação, como se pode ver em decimal mais abaixo. Em seguida, destacam-se em amarelo as jogadas com valor de 2 pontos e em laranja as jogadas com valor de 1 ponto. O incremento da pontuação, novamente, pode ser visualizado na linha do respectivo sinal. Nota-se que a pontuação menor é atribuída a jogadas que foram feitas de forma atrasada, o que é visualmente perceptível pelo momento em que o sinal que detecta a jogada correta levanta em relação ao começo da jogada (momento da transição de “caso”).



Figura 51: Terceira parte do cenário de teste 1.

Os ciclos descritos continuam ocorrendo até o final da partida. Como o jogador acertou todas as jogadas neste teste, perdendo 2 pontos em três e três pontos em duas, seu placar final é $4*64 - 2*3 - 3*2 = 244$. Nota-se que após o fim da partida há *reset* do contador, porém não do placar, visto que o jogador deve poder

ver quanto marcou quando sua partida terminou. O sinal pronto também levanta, indicando o fim da partida.

Cenário 2 - Modo de jogo base com erros por jogada incorreta e tempo limite:

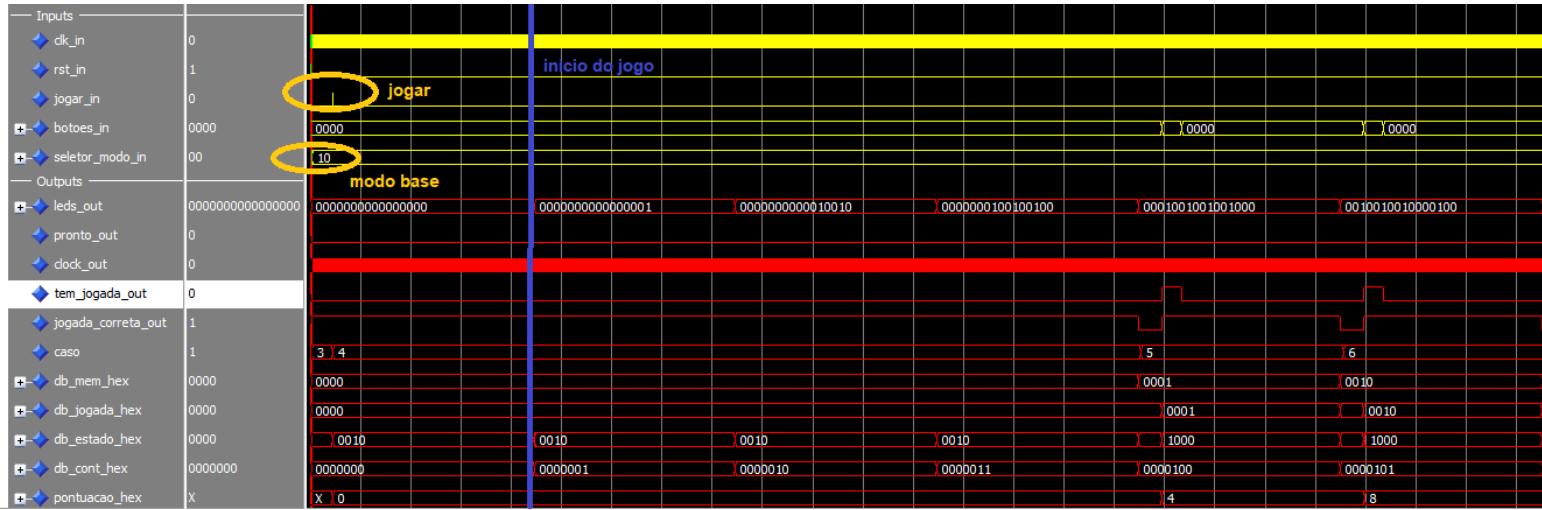


Figura 52: Primeira parte do cenário de teste 2.

A primeira parte do teste é exatamente igual à do cenário 1, visto que o modo é o mesmo e não há erro em nenhuma das duas primeiras comparações.

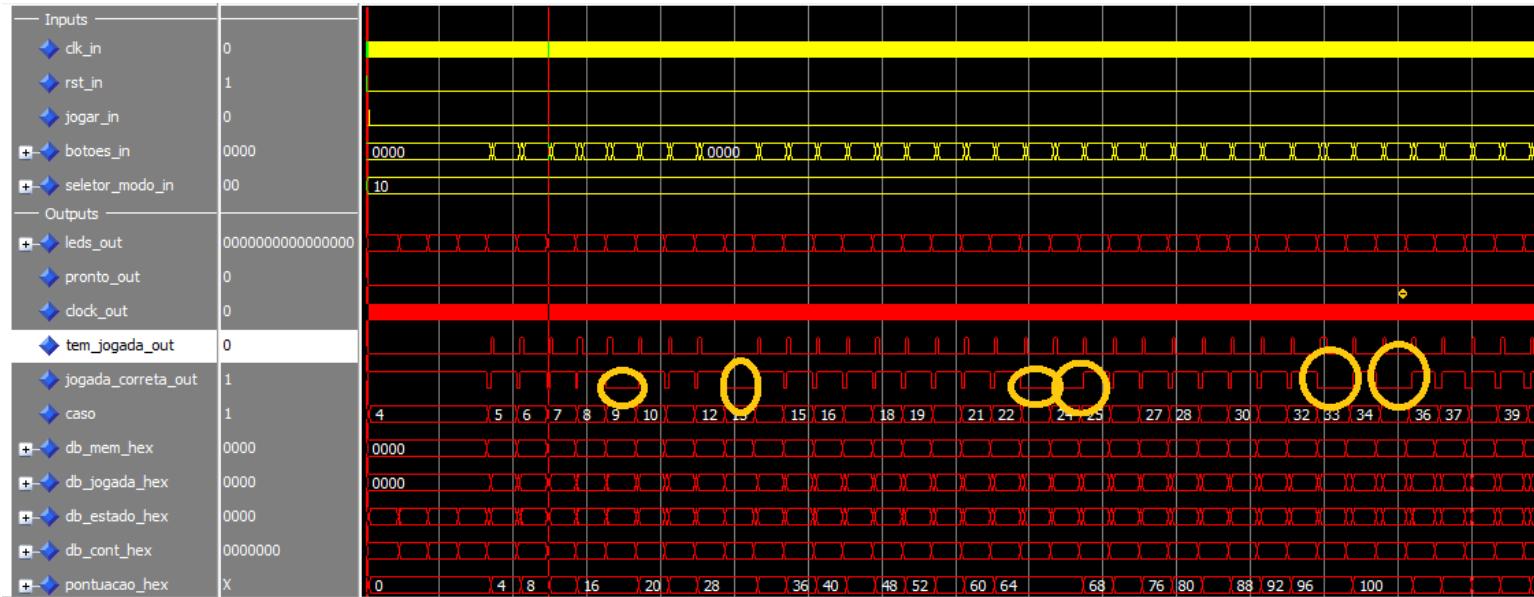


Figura 53: Segunda parte do cenário de teste 2.

A figura acima mostra o trecho em que são feitas jogadas erradas - não há jogada errada algum em outro trecho. Primeiro, no caso 9, é feita uma jogada incorreta, o sinal “jogada_correta” - em laranja - não levanta e o contador do placar não é incrementado. Em seguida, no caso 13, não é feita jogada alguma e o mesmo efeito é observado. Nos casos 23 e 24 é testado o erro de duas jogadas consecutivas e como esperado o contador de placar segue inalterado. Por fim, há

jogadas erradas nos casos 33 e 35 - intercaladas com uma correta. Em todos os casos o comportamento observado corresponde ao esperado.

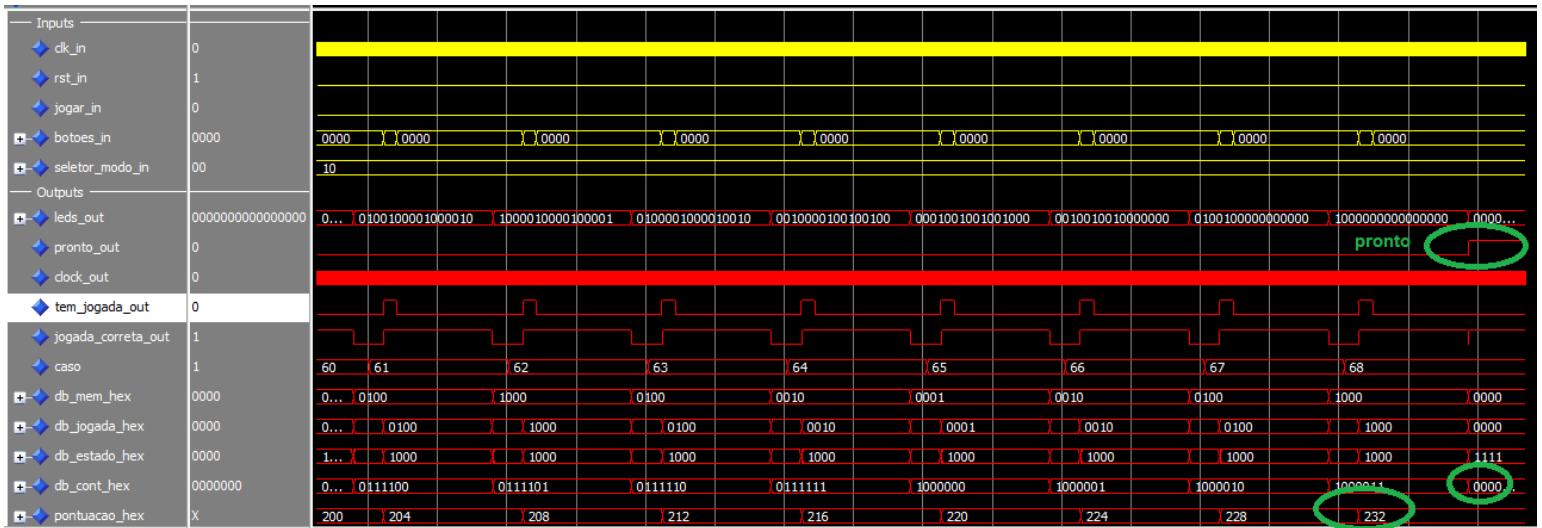


Figura 54: Terceira parte do cenário de teste 2.

Conforme explicado no parágrafo anterior, foram introduzidas 6 jogadas incorretas ao *testbench*. Como a pontuação máxima é 256, de fato foram contabilizados os 6 erros: $256 - 6 \times 4 = 232$. Novamente há *reset* do contador de endereços, mas não do placar, e o sinal “pronto” indica o término da partida.

Cenário 3 - Gravação de um jogo e partida no modo personalizado:

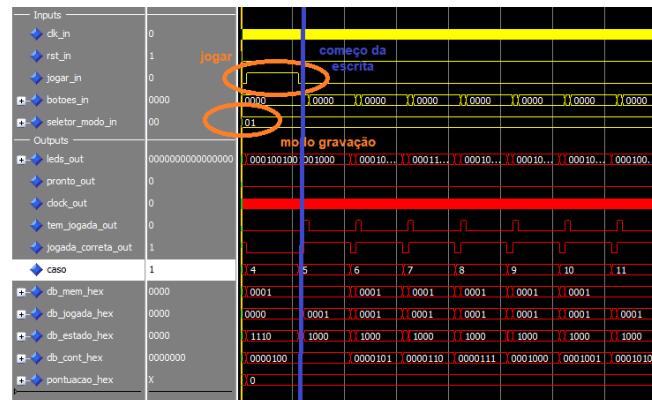


Figura 55: Primeira parte do cenário de teste 3.

A inicialização da partida difere apenas no código de gravação, que é “01” para a escrita. Em seguida, tem-se os ciclos de escrita na memória destinada à gravação da sequência do jogador, que é bastante similar ao da partida normal, como se pode ver no DTE. Nota-se que, caso a memória não seja igual ao dado pressionado em certo ciclo, após alguns ciclos de *clock* a saída “db_memoria” passará a mostrar o mesmo dado contido em “db_jogada_hex”. Tal confirma a escrita correta na memória. Por simplicidade, todas as escritas feitas foram “0001”.

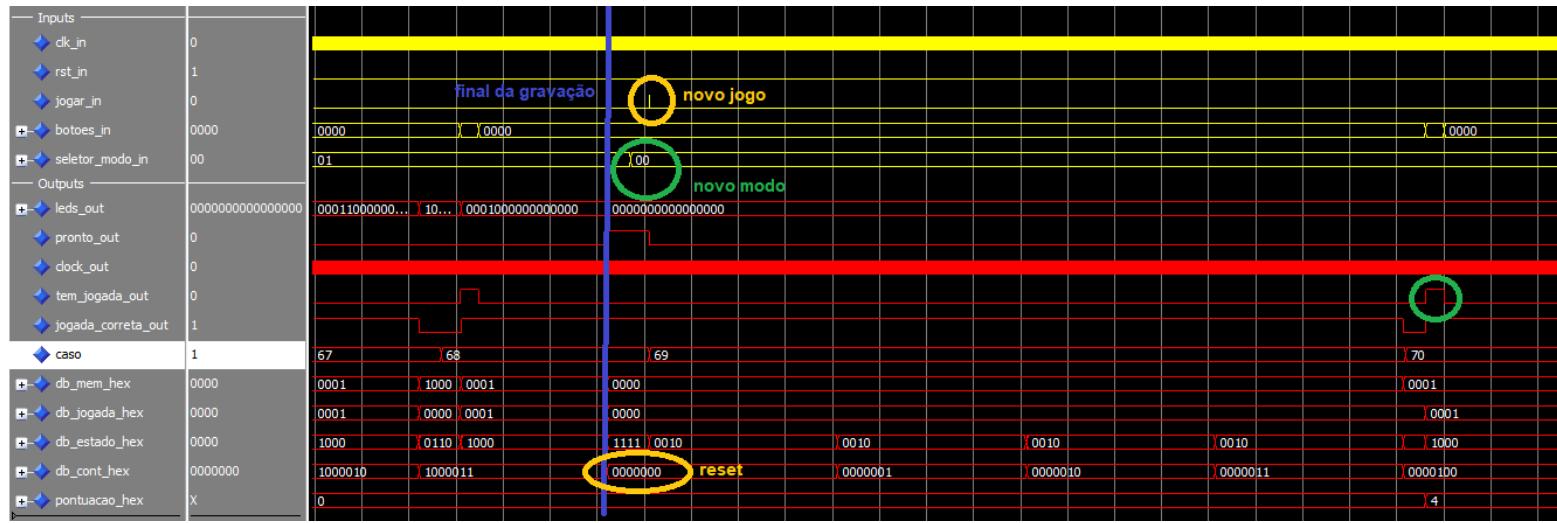


Figura 56: Segunda parte do cenário de teste 3.

A figura correspondente à segunda parte da onda mostra o momento em que a operação de escrita termina e é dado início ao jogo personalizado - código “00”. Há detaque para o *reset* do contador, visto que caso contrário não seria possível realizar mais de uma partida em sequência. Como esperado, o primeiro dado é “0001” - dado escrito na parte anterior. Na linha com a elipse verde tem-se a primeira jogada, que incrementa a pontuação.

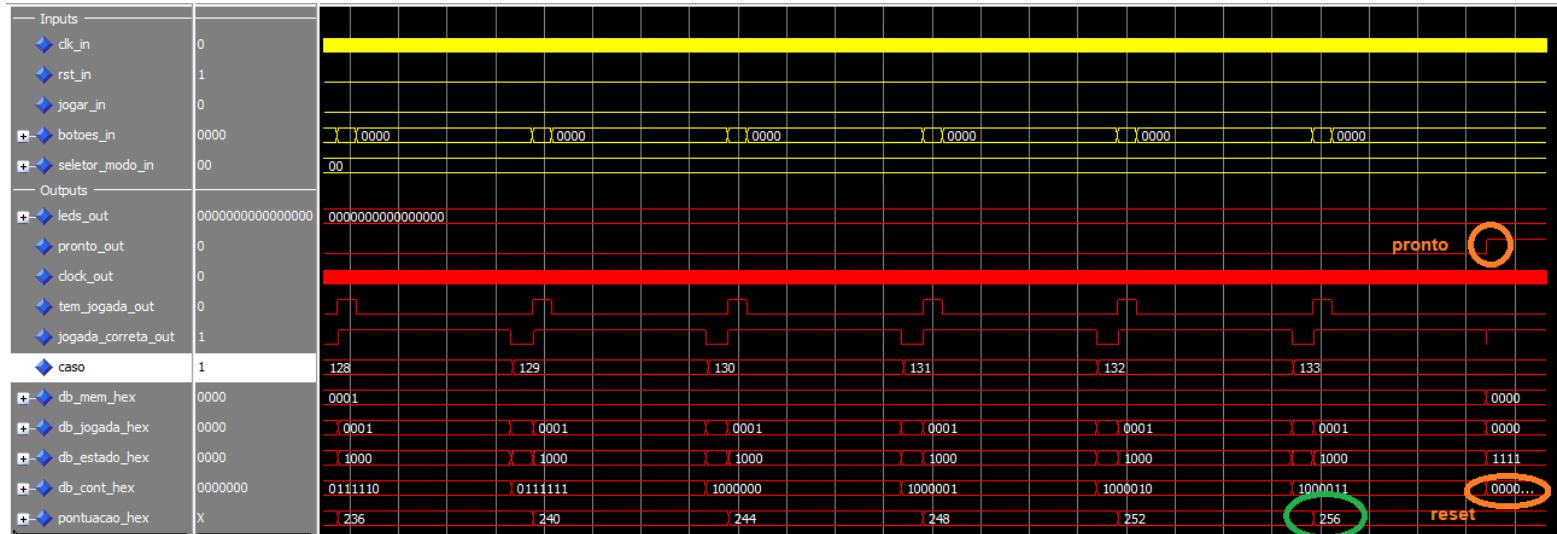


Figura 57: Terceira parte do cenário de teste 3.

Por fim, tem-se o término da partida personalizada. Como configurado no *testbench*, não houve nenhum erro e foi atingido o placar máximo. Novamente há reset para o contador e o fim de jogo é indicado pelo sinal “pronto”. Observa-se que o jogador não precisa apertar algum botão necessariamente, sendo marcado “0000”, que corresponde a um período no qual, na partida, botão algum deve ser pressionado. Naturalmente, tal implica que não necessariamente o número de pontos possível de ser obtido é 4×64 , sendo este na verdade o valor máximo - quando nenhum endereço da memória contém o dado “0000”.

5.3 Preparação para atividades práticas

Após a verificação da corretude da lógica do circuito, foi criado um novo projeto no software Quartus. Dentro do projeto, a arquitetura utilizada para a instanciação das memórias RAM foi modificada para suportar a utilização do arquivo .mif preparado para inicialização do conteúdo.

Em seguida, foram geradas as saídas das ferramentas *RTL Viewer* e *State Machine Viewer*, que mostram a composição interna do circuito.

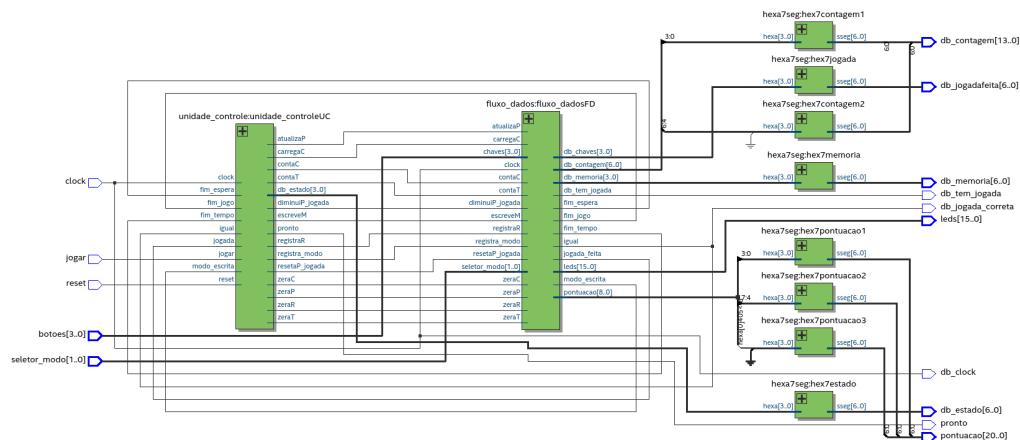


Figura 58: Diagrama RTL do circuito completo

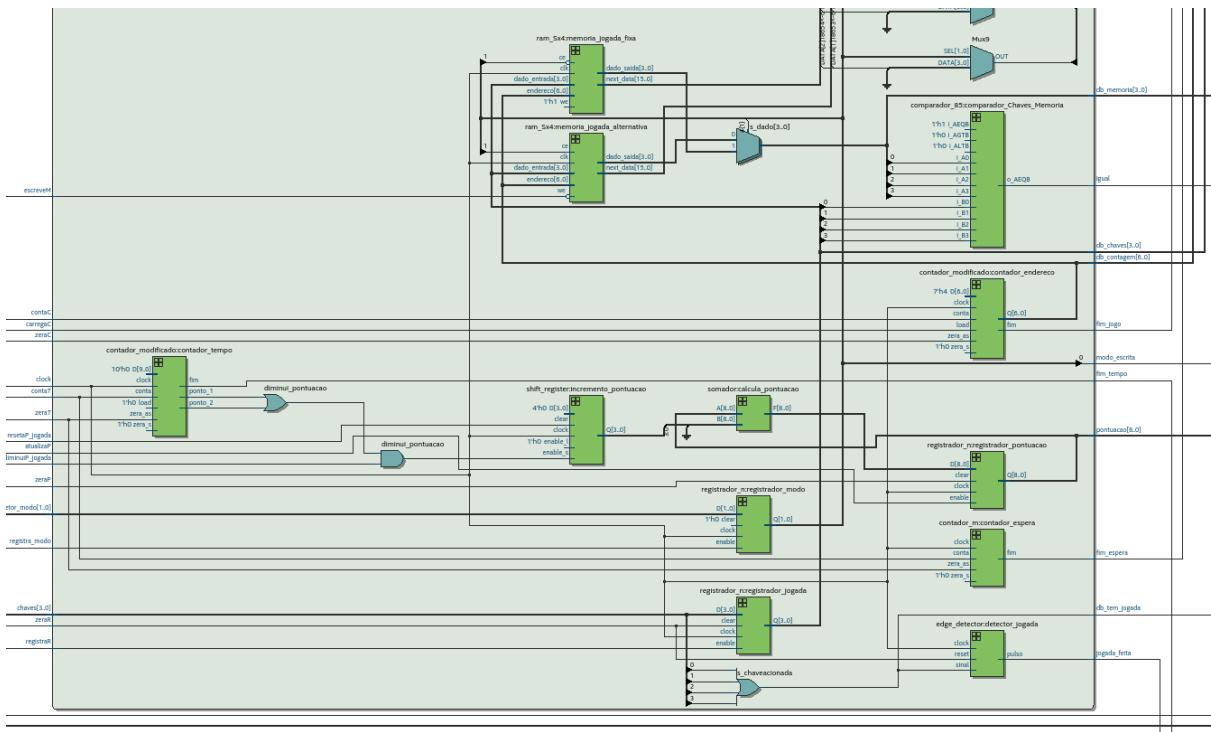


Figura 59: Parte do diagrama RTL do fluxo de dados

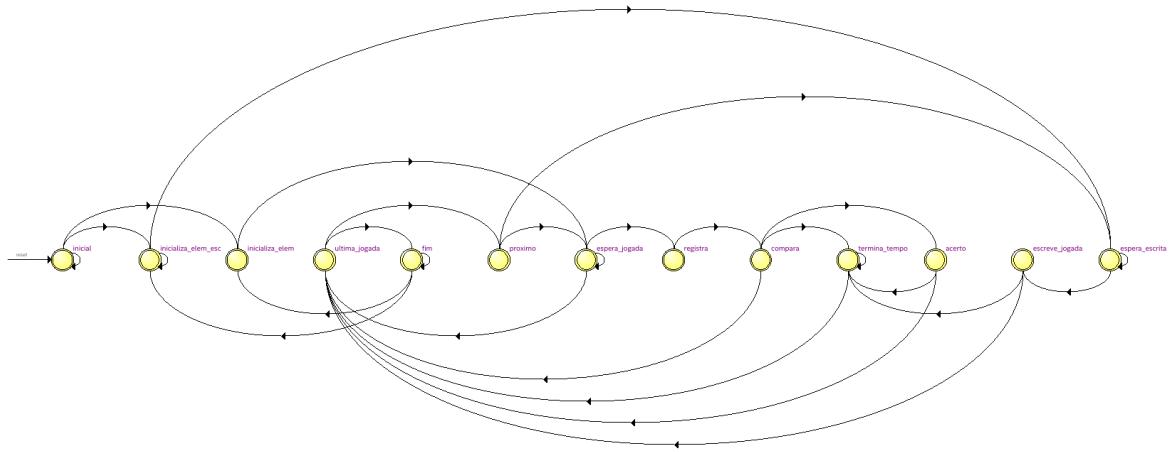


Figura 60: Diagrama de Transição de Estados gerado pelo Quartus

	Source State	Destination State	Condition
1	acerto	ultima_jogada	(fim_tempo)
2	acerto	termina_tempo	(!fim_tempo)
3	compara	ultima_jogada	(fim_tempo).(!=igual)
4	compara	termina_tempo	(!fim_tempo).(!=igual)
5	compara	acerto	(igual)
6	escreve_jogada	ultima_jogada	(fim_tempo)
7	escreve_jogada	termina_tempo	(!fim_tempo)
8	espera_escrita	escreve_jogada	(!jogada).(fim_tempo) + (jogada)
9	espera_escrita	espera_escrita	(!jogada).(!fim_tempo)
10	espera_jogada	espera_jogada	(!fim_tempo).(!=jogada)
11	espera_jogada	registra	(jogada)
12	espera_jogada	ultima_jogada	(fim_tempo).(!=jogada)
13	fim	inicializa_elem_esc	(jogar).(modo_escrita)
14	fim	inicializa_elem	(jogar).(!=modo_escrita)
15	fim	fim	(!jogar)
16	inicial	inicializa_elem_esc	(jogar).(modo_escrita)
17	inicial	inicial	(!jogar)
18	inicial	inicializa_elem	(jogar).(!=modo_escrita)
19	inicializa_elem	espera_jogada	
20	inicializa_elem_esc	inicializa_elem_esc	(!fim_espera)
21	inicializa_elem_esc	espera_escrita	(fim_espera)
22	proximo	espera_jogada	(!modo_escrita)
23	proximo	espera_escrita	(modo_escrita)
24	registra	compara	
25	termina_tempo	ultima_jogada	(fim_tempo)
26	termina_tempo	termina_tempo	(!fim_tempo)
27	ultima_jogada	fim	(fim_jogo)
28	ultima_jogada	proximo	(!fim_jogo)

Figura 61: Tabela de transição de estados gerada pelo Quartus

5.3.1 Planejamento da pinagem

No mesmo projeto do Quartus, as entradas e saídas da entidade principal do circuito foram atribuídas a pinos na placa FPGA, seguindo a tabela de pinagem abaixo.

Sinal	Pino na Placa DE0-CV	Pino no FPGA	Analog Discovery	Protoboard
CLOCK	GPIO_0_D0	PIN_N16	Patterns - Clock - 1 KHz - DIO	-
RESET	GPIO_0_D1	PIN_B16	StaticIO - Button 0/1 - DIO1	-
INCIAR/JOGAR	GPIO_0_D3	PIN_C16	StaticIO - Button 0/1 - DIO2	-
BOTOES(0)	GPIO_1_D27	PIN_F15	-	botão coluna azul
BOTOES(1)	GPIO_D29	PIN_F12	-	botão coluna amarela
BOTOES(2)	GPIO_1_D31	PIN_G15	-	botão coluna vermelha
BOTOES(3)	GPIO_1_D33	PIN_G12	-	botão coluna verde
SELETOR_MODO(0)	GPIO_0_D4	PIN_D17	StaticIO - Switch 0/1 - DIO8	-
SELETOR_MODO(1)	GPIO_0_D5	PIN_K20	StaticIO - Switch 0/1 - DIO9	-
PRONTO	GPIO_0_D2	PIN_M16	StaticIO - LED - DIO10	-
LEDS(0)	GPIO_1_D0	PIN_H16	-	led azul, linha 1
LEDS(1)	GPIO_1_D1	PIN_A12	-	led amarelo, linha 1
LEDS(2)	GPIO_1_D10	PIN_H18	-	led vermelho, linha 1
LEDS(3)	GPIO_1_D11	PIN_J18	-	led verde, linha 1
LEDS(4)	GPIO_1_D2	PIN_H15	-	led azul, linha 3
LEDS(5)	GPIO_1_D3	PIN_B12	-	led amarelo, linha 3
LEDS(6)	GPIO_1_D12	PIN_J19	-	led vermelho, linha 3
LEDS(7)	GPIO_1_D13	PIN_G11	-	led verde, linha 3
LEDS(8)	GPIO_1_D4	PIN_A13	-	led azul, linha 2
LEDS(9)	GPIO_1_D5	PIN_B13	-	led amarelo, linha 2
LEDS(10)	GPIO_1_D14	PIN_H10	-	led vermelho, linha 2
LEDS(11)	GPIO_1_D15	PIN_J11	-	led verde, linha 2
LEDS(12)	GPIO_1_D6	PIN_C13	-	led azul, linha 4
LEDS(13)	GPIO_1_D7	PIN_D13	-	led amarelo, linha 4
LEDS(14)	GPIO_1_D16	PIN_H14	-	led vermelho, linha 4
LEDS(15)	GPIO_1_D17	PIN_A15	-	led verde, linha 4
PONTUACAO(0)	Display HEX2	PIN_Y19	-	-
PONTUACAO(1)	Display HEX2	PIN_AB17	-	-
PONTUACAO(2)	Display HEX2	PIN_AA10	-	-
PONTUACAO(3)	Display HEX2	PIN_Y14	-	-
PONTUACAO(4)	Display HEX2	PIN_V14	-	-
PONTUACAO(5)	Display HEX2	PIN_AB22	-	-

PONTUACAO(6)	Display HEX2	PIN_AB21	-	-
PONTUACAO(7)	Display HEX3	PIN_Y16	-	-
PONTUACAO(8)	Display HEX3	PIN_W16	-	-
PONTUACAO(9)	Display HEX3	PIN_Y17	-	-
PONTUACAO(10)	Display HEX3	PIN_V16	-	-
PONTUACAO(11)	Display HEX3	PIN_U17	-	-
PONTUACAO(12)	Display HEX3	PIN_V18	-	-
PONTUACAO(13)	Display HEX3	PIN_V19	-	-
PONTUACAO(14)	Display HEX4	PIN_U20	-	-
PONTUACAO(15)	Display HEX4	PIN_Y20	-	-
PONTUACAO(16)	Display HEX4	PIN_V20	-	-
PONTUACAO(17)	Display HEX4	PIN_U16	-	-
PONTUACAO(18)	Display HEX4	PIN_U15	-	-
PONTUACAO(19)	Display HEX4	PIN_Y15	-	-
PONTUACAO(20)	Display HEX4	PIN_P9	-	-
db_clock	Led LEDR0	PIN_AA2	-	-
db_tem_jogada	Led LEDR1	PIN_AA1	-	-
db_jogada_correta	Led LEDR2	PIN_W2	-	-
db_estado(0)	Display HEX0	PIN_U21	-	-
db_estado(1)	Display HEX0	PIN_V21	-	-
db_estado(2)	Display HEX0	PIN_W22	-	-
db_estado(3)	Display HEX0	PIN_W21	-	-
db_estado(4)	Display HEX0	PIN_Y22	-	-
db_estado(5)	Display HEX0	PIN_Y21	-	-
db_estado(6)	Display HEX0	PIN_AA22	-	-
db_jogada_feita(0)	Display HEX1	PIN_AA20	-	-
db_jogada_feita(1)	Display HEX1	PIN_AB20	-	-
db_jogada_feita(2)	Display HEX1	PIN_AA19	-	-
db_jogada_feita(3)	Display HEX1	PIN_AA18	-	-
db_jogada_feita(4)	Display HEX1	PIN_AB18	-	-
db_jogada_feita(5)	Display HEX1	PIN_AA17	-	-
db_jogada_feita(6)	Display HEX1	PIN_U22	-	-

Tabela 12: Tabela de pinagem planejada para o circuito construído.

5.3.2 Uso do Analog discovery

O dispositivo Analog Discovery para controlar alguns sinais de entrada e saída. Abaixo, tem-se o esquema de números correspondentes a cada fio do Analog Discovery, além do esquema de pinos da placa FPGA.

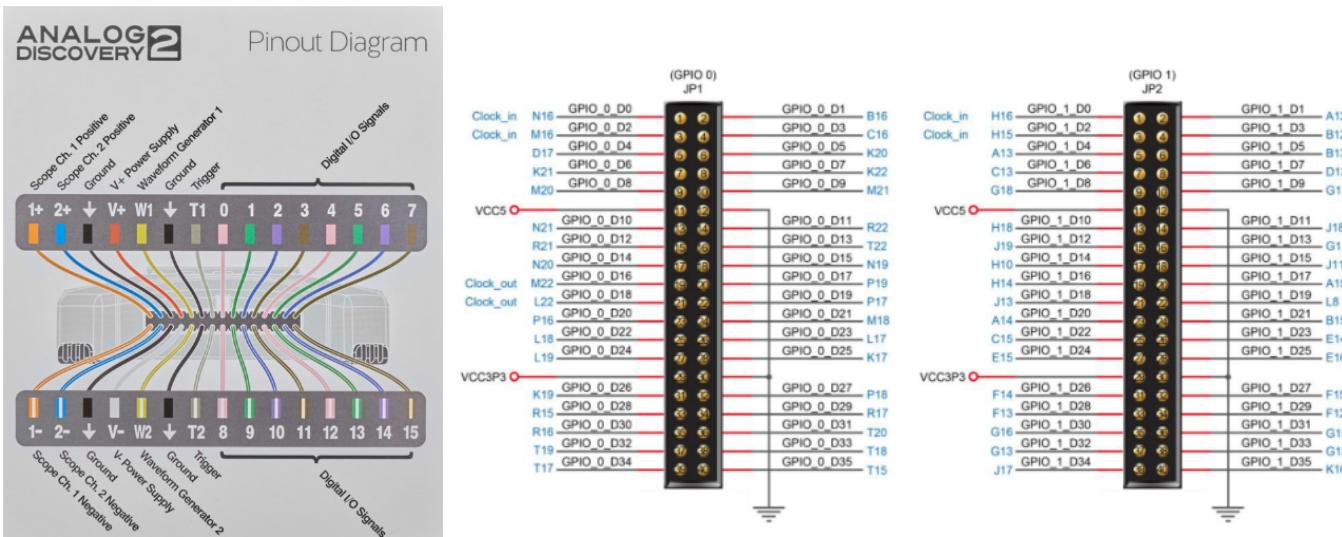


Figura 62: Informações para conexão do Analog Discovery à placa FPGA e pinagem de seus sinais.

Para controlar os sinais de entrada ligados ao Analog Discovery e observar os sinais de saída, é utilizado o software Waveforms. A foto abaixo, retirada da apostila “Dicas Analog Discovery” da disciplina, mostra a forma geral da interface do programa e como o sinal de *clock* deve ser definido.

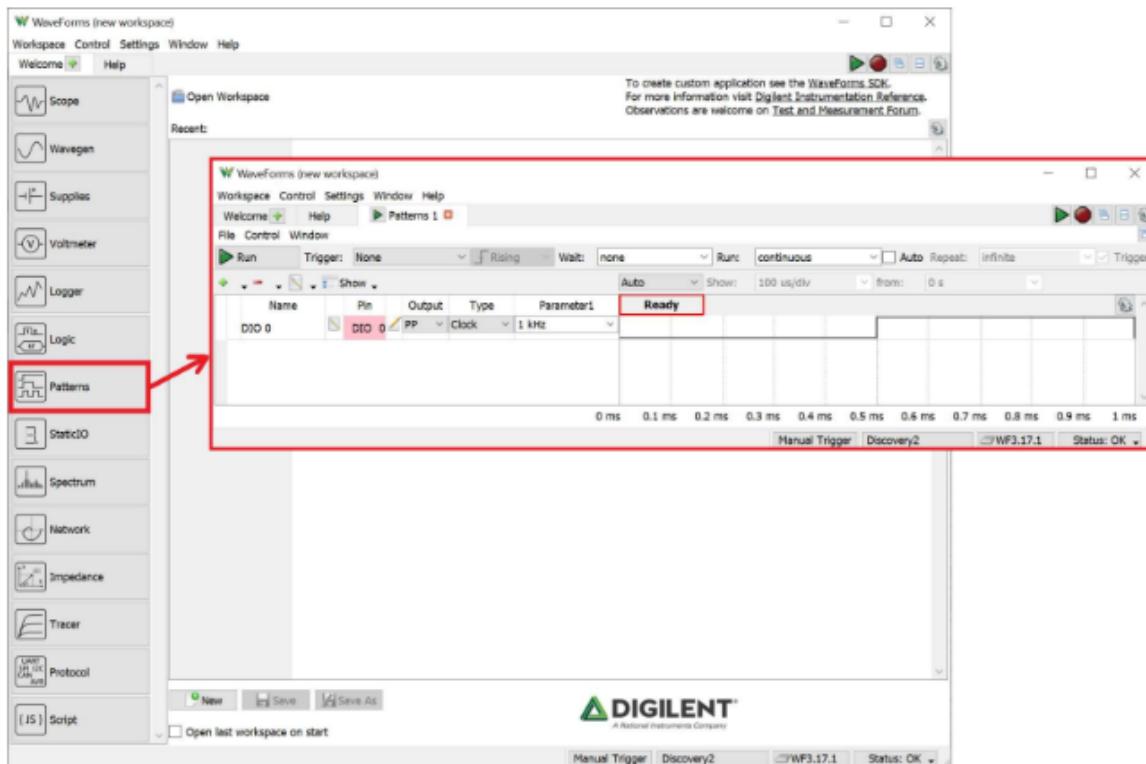


Figura 63: Interface do Waveforms e localização de comandos para configuração do relógio.

Após clicar em “Patterns”, na área à esquerda, em branco, clica-se com o botão direito do mouse, abrindo uma caixa de opções na qual é possível adicionar

sinais - no caso apenas DIO0 será configurado dessa forma. Posteriormente, define-se o tipo como *clock* e ajusta-se o parâmetro (frequência) para 1kHz. Por fim, destaca-se que para o sinal realmente iniciar sua operação é preciso ativar o *play*, clicando na seta verde que aparece no título do *pattern* criado - no caso *patterns 1*. Quando ativo, a seta verde será substituída por um botão vermelho. Então, é preciso configurar os demais sinais, que correspondem à categoria StaticIO. As demais entradas do circuito devem ser declaradas como botões, enquanto as saídas devem ser declaradas como LEDs. Nota-se que, no canto esquerdo superior, pode-se clicar em *view* e depois em nomes, para renomear cada sinal, inicialmente denominado a partir de seu número de entrada.

6. Atividades experimentais

Primeiro, fez-se a montagem do circuito, conectando os leds, os botões e seus respectivos resistores na protoboard. Em seguida, foram feitas as ligações com os pinos GPIO da placa - 1 por componente. Por fim, fez-se um *ground* comum para os botões e cada linha de *leds*, que foi conectado ao *ground* da placa. Em seguida, foram feitas as conexões relativas ao Analog Discovery, todas alocadas no GPIO_0. Com isso, programou-se o circuito na FPGA. No entanto, foi detectado um erro: os leds nunca acendiam e o sinal de depuração de jogada correta levantava quando não se fazia jogada alguma. Os botões funcionaram como esperado. Disso, após se checar que não havia grave erro de montagem com os leds, fez-se a hipótese de que a memória estaria apenas com dados nulos. De fato, após se colocar o sinal de depuração “db_memoria” na FPGA tal foi confirmado. O sinal “db_contagem” também foi colocado em um dos *displays* e notou-se que a contagem estava correta. Após análise do código VHDL notou-se que o componente da memória RAM não estava corretamente instanciado, posto que ainda constava o anterior - ram_64x4 - em vez do modificado - ram_Sx4. Não se sabe a razão de não ter sido obtido algum erro correspondente na simulação. Em seguida, fez a modificação do código VHDL para resolver tal pendência. Aproveitou-se a necessidade de recompilação para se fazer mais duas modificações. Primeiro, a inserção da saída “last_data” da memória, usada para mostrar as 4 últimas jogadas registradas pelo jogador no modo de gravação. Em segundo lugar, adicionou-se um sinal intermediário, com lógica negada, para os botões. Isso foi feito para que fosse utilizada a configuração pull-up do circuito com cada botão, o que minimiza interferência.

Após implementar essas modificações, percebeu-se que dois leds estavam ligados de maneira incorreta na placa FPGA, sendo este um problema somente na montagem do circuito, o problema foi detectado e a montagem corrigida para a execução dos testes.

Para melhor controle dos testes, primeiro diminuiu-se a velocidade das jogadas. Com isso, fizeram-se os três testes planejados. O resultado obtido mostra correta implementação do sistema de pontuação, com incremento de 4, 2 ou 1 conforme os tempos especificados. Além disso, jogadas erradas ou com *timeout* não alteraram a pontuação. O teste relacionado à escrita foi bem sucedido, posto que, ao jogar-se o modo personalizado, a sequência de *leds* foi idêntica à gravada. Além disso, a pontuação funcionou normalmente para esse caso. Por fim, nota-se que os sinais de *reset* e início do jogo tiveram funcionamento adequado.

Em relação aos requisitos, notou-se que classificar o resultado do usuário seria menos relevante que mostrar a pontuação máxima, para comparação. Isso pois, devido ao modo de gravação, que pode ter um número variável de espaços nulos, a pontuação máxima não é fixa. Desse modo, considera-se que é mais relevante que seja mostrado o quanto o jogador acertou do total possível até o momento, ou seja, a comparação entre sua performance e a performance perfeita. Tal foi adicionado como novo requisito.

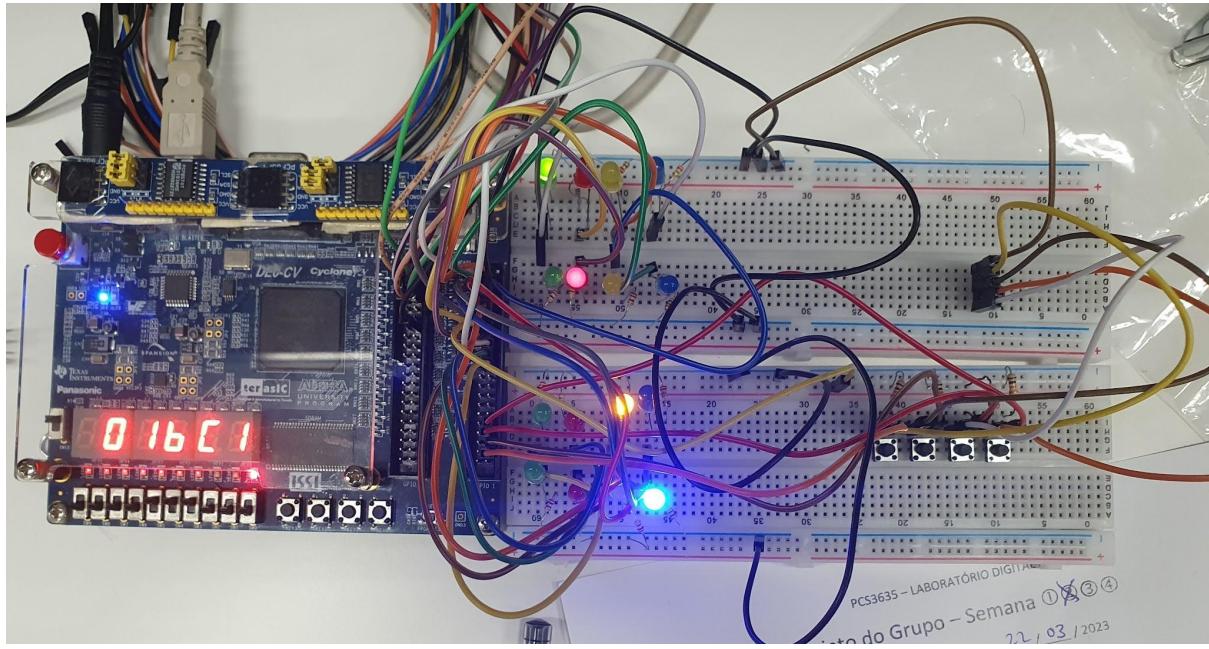


Figura 64: Montagem final do circuito.

7. Conclusão da semana 2

Foram observados comportamentos incorretos na implementação do circuito, inicialmente. Tais não foram identificados na simulação com o Modelsim. Por isso, primeiro checou-se a pinagem e conexões com a protoboard. Como não foi resolvido o problema, foi feita análise a partir dos sinais de depuração. Com isso, identificou-se o problema, correspondente a uma instanciação com componente errado - versão antiga da memória. Após correção deste erro, o circuito funcionou do modo esperado para os três casos de teste. Assim, considera-se satisfatório o resultado.

8. Planejamento da semana 3

8.1 Descrição das modificações do circuito

8.1.1 Descrição lógica das modificações planejadas para a semana 3

Para a semana três tem-se a seguinte programação: integração do componente *display* externo ao circuito, para mostrar o placar em tempo real; Adição da possibilidade de jogada dupla; Mudança do *radix* do placar de hexadecimal para decimal.

O primeiro requisito corresponde a uma mudança nas conexões entre placa FPGA e componentes externos, ou seja, não são supostas alterações significativas no código VHDL, havendo apenas o cuidado com a integração do terceiro requisito da lista. Nota-se que o número de *displays* de sete segmentos a serem usados é um fator de extrema importância, sendo definido, sempre, a partir da pontuação máxima possível de ser obtida. Deve-se ver quantos algarismos possui tal valor - em decimal, não em hexadecimal. Abaixo tem-se uma foto do componente.

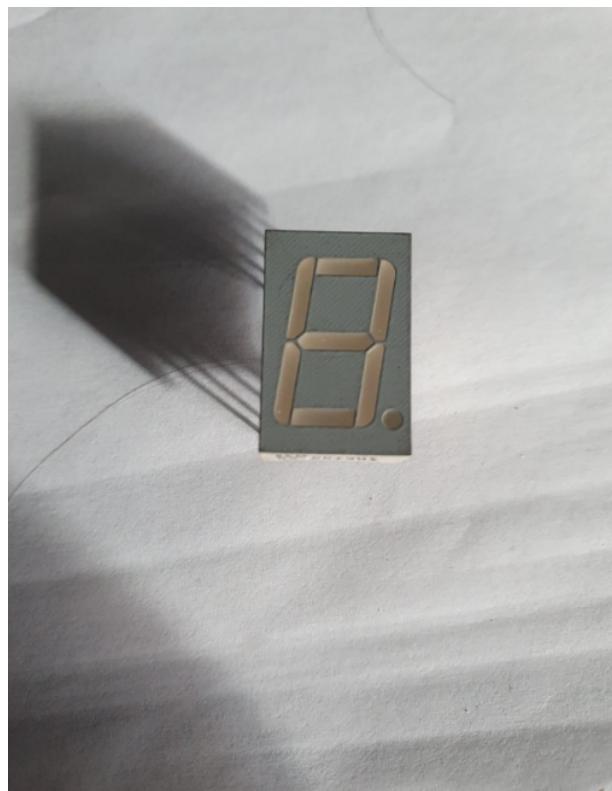


Figura 65: Componente externo *display* de sete segmentos utilizado.

O segundo requisito corresponde a alterações apenas no código VHDL, sendo usados novos componentes para atender ao proposto. Há mudança na lógica de estados, visto que cada jogada agora pode ser composta pelo pressionamento de mais de um botão. Consequentemente, tanto na gravação quanto nos jogos há necessidade de um novo estado - correspondente à espera pela segunda jogada -

atingido quando a primeira é feita. A seguinte figura descreve a lógica de transições por meio de um diagrama de transição de estados (DTE).

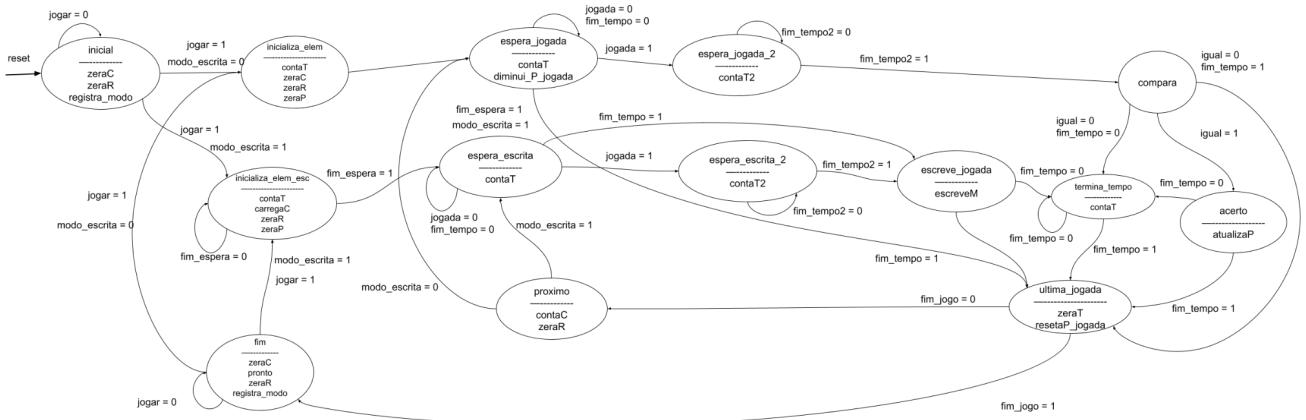


Figura 66: DTE do circuito da semana 3.

Além disso, o sistema de detecção de jogadas, antes resolvido por meio de um componente *edge detector*, tornou-se mais complexo, sendo preciso registrar o primeiro botão pressionado da jogada, em vez de se gerar um pulso.

O terceiro requisito é principalmente resolvido de forma não integrada ao circuito. Primeiro, criou-se um novo componente conversor de hexadecimal para decimal, com um *testbench* correspondente. A relação com o circuito ocorre apenas na entidade principal, na qual há instanciação dos *displays* de sete segmentos, de modo que não são modificados nem a unidade de controle (UC) nem o fluxo de dados (FD) para isso.

8.1.2 Alterações em componentes de menor grau de abstração

a) Componente “somador”:

Devido à conversão entre hexadecimal e decimal, há necessidade de se realizar a subtração em binário. Por conta disso, o componente denominado somador passou a executar as duas operações aritméticas de uma unidade lógico-aritmética (ULA). Assim, faz-se necessário adicionar um sinal seletor de operação como nova entrada do componente, denominado “S”. A lógica da subtração é a mesma, porém, como se usa complemento de dois, é preciso modificar o *carry in* inicial e o operando B. Nega-se B e faz-se *carry in* = ‘1’ caso seja escolhida a subtração.

```

25      port (
26          A, B      : in std_logic_vector (size - 1 downto 0); --inputs
27          S        : in std_logic; -- Seleciona se será feita subtração

```

Figura 67: Sinal seletor para operação.

```

42      B_int <= not B when S = '1' else B;
43      -- Implementação da soma por Carry-lookahead
44      CA(0) <= S;

```

Figura 68: Lógica para preparação da subtração.

b) Componente *edge holder*:

A necessidade de se registrar um botão apertado em vez de gerar um pulso, oriunda das jogadas duplas, é sanada pelo componente *edge holder*. Tal possui uma entrada de dados, uma saída, um relógio e um *reset* - para zero, sempre. O tamanho dos sinais de entrada e saída é genérico. Para a arquitetura, são definidos sinais intermediários para instanciar um componente *edge detector*, porém são vetores. A técnica consiste em usar um *for generate* com a instanciação de um *edge detector* para cada bit. Assim, é possível rastrear cada botão em vez do sinal completo. Por fim, como se quer que a saída seja mantida após ser identificado o pulso, é incluído o próprio sinal como opção para que o bit seja alto, além da possibilidade de haver um pulso. Naturalmente, o *reset*, assíncrono, leva todos os bits a zero.

```

16  entity edge_holder is
17  generic (
18      constant size : natural := 8
19  );
20  port (
21      clock    : in std_logic;
22      reset    : in std_logic;
23      entrada : in std_logic_vector(size-1 downto 0);
24      saida   : out std_logic_vector(size-1 downto 0)
25  );
26 end entity edge_holder;

```

Figura 69: Entidade do componente *edge holder*.

```

28  architecture estrutural of edge_holder is
29    component edge_detector is
30      port (
31        clock    : in std_logic;
32        reset    : in std_logic;
33        sinal    : in std_logic;
34        pulso   : out std_logic
35      );
36    end component;
37    signal pulsos  : std_logic_vector (size-1 downto 0);
38    signal saida_int: std_logic_vector (size-1 downto 0);
39 begin
40   ED: for j in 0 to (size - 1) generate
41     detect_j: edge_detector port map (
42       clock    => clock,
43       reset    => reset,
44       sinal    => entrada(j),
45       pulso   => pulsos(j)
46     );
47   end generate ED;
48
49   GENS: for j in 0 to (size-1) generate
50     saida_int(j) <= (pulsos(j) or saida_int(j)) and (not reset);
51   end generate GENS;
52
53   saida <= saida_int;
54 end architecture estrutural;

```

Figura 70: Arquitetura do componente *edge holder*.

c) hexDecimal:

Um componente hexDecimal é dotado de entrada e saída de quatro bits, além de *carry in* e *carry out*. A conversão é baseada em dois casos: enquanto não é atingido 10, o respectivo dígito é incrementado. Se excedido, deve ser decrementado e o seguinte dígito incrementado. Para tal são instanciados dois componentes somadores, fixados um em soma e outro em subtração. Os devidos casos são abordados e quando necessário há *sign extend*.

```

16  entity hexDecimal is
17    port (
18      hexa    : in std_logic_vector (3 downto 0);
19      Cin     : in std_logic;
20      dec     : out std_logic_vector (3 downto 0);
21      Cout    : out std_logic
22    );
23  end entity hexDecimal;

```

Figura 71: Entidade do componente hexDecimal.

```

45      begin
46          hexa_ext <= "00" & hexa;
47          Cin_ext <= "00000" & Cin;
48
49          incrementa_hexa: somador
50              generic map (
51                  size => 6
52              )
53              port map (
54                  A => hexa_ext,
55                  B => Cin_ext,
56                  S => '0',
57                  F => hexa_int,
58                  Z => open,
59                  N => open,
60                  Ov => Ov,
61                  Co => open
62          );

```

Figura 72: Incremento de cada dígito.

```

66      subtrai_dez: somador
67          generic map (
68              size => 6
69          )
70          port map (
71              A => hexa_2,
72              B => dez,
73              S => '1',
74              F => dec_int,
75              Z => open,
76              N => lss,
77              Ov => open,
78              Co => open
79          );
80
81      dec_ext <= hexa_2 when lss = '1' else dec_int;
82
83      Cout <= Ov or (not lss);
84
85      dec <= dec_ext(3 downto 0);

```

Figura 73: Decremento de cada dígito.

d) Memória ram_Sx4:

Como no começo da escrita tem-se endereço zero, mas teoricamente já seriam mostrados os três dados anteriores, foi preciso aumentar o conjunto de dados da memória até suprir todos os endereços possíveis dado o número de bits destinados ao endereçamento. Assim, desde 71 até 127 fez-se todos os elementos “0000”. Nota-se que, quando feito endereçamento com zero, nas linhas envolvendo os leds, que contém a saída do contador menos um, na verdade estava-se acessando 127.

8.1.3 Alterações no fluxo de dados

Nesta subseção são comentadas as alterações feitas no arquivo fluxo_dados.vhd, além de se mostrar o resultado por meio da descrição *register transfer level* (RTL). Primeiro, foram adicionados um sinal de entrada e um de saída: “contaT2” e “fim_tempo_2”, respectivamente. O primeiro está associado à contagem para *timeout*, ao passo que o segundo é uma *flag* associada ao mesmo contador.

```
52      zeraR      : in  std_logic;
53      registraR   : in  std_logic;
54      contaT     : in  std_logic;
55      contaT2    : in  std_logic;
56      zeraT      : in  std_logic;
57      atualizaP   : in  std_logic;
58      diminuiP_jogada : in  std_logic;
59      resetaP_jogada : in  std_logic;
60      zeraP      : in  std_logic;
61      registra_modo : in  std_logic;
62
63      igual       : out std_logic;
64      fim_jogo    : out std_logic; -- novo sinal: saida do contador de rodada
65      jogada_feita: out std_logic;
66      fim_tempo   : out std_logic;
67      fim_tempo_2 : out std_logic;
68      fim_espera  : out std_logic;
69      modo_escrita: out std_logic;
```

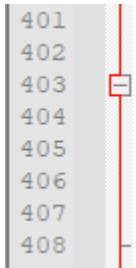
Figura 74: Novos sinais de interface externa do fluxo de dados.

Na arquitetura, a primeira modificação maior advém da instanciação de três componentes conversores de hexadecimal para decimal. A saída é um vetor comum cujas partes são divididas entre cada componente. O primeiro *carry in* é nulo, o último é *carry out* é deixado em *open* e para os demais o *carry out* de uma instância se torna *carry in* da seguinte.

```
300      hexParaDecl: hexDecimal
301      port map (
302          hexa    => pont_expand (3 downto 0),
303          Cin     => '0',
304          dec     => pontuacao_dec(3 downto 0),
305          Cout    => pont_conv_cout_1
306      );
307      hexaParaDec2: hexDecimal
308      port map (
309          hexa    => pont_expand (7 downto 4),
310          Cin     => pont_conv_cout_1,
311          dec     => pontuacao_dec (7 downto 4),
312          Cout    => pont_conv_cout_2
313      );
314      hexaParaDec3: hexDecimal
315      port map (
316          hexa    => pont_expand (11 downto 8),
317          Cin     => pont_conv_cout_2,
318          dec     => pontuacao_dec (11 downto 8),
319          Cout    => open
320      );
```

Figura 75: Componentes hexDecimal instanciados no FD.

O FD passou a conter, também, uma instância do *edge holder*. A chave é a entrada, posto que se quer mapear seus bits. O *clock* é universal, e seu *reset* é o mesmo do registrador de jogadas. A saída é um intermediário “*s_chaves*”.



```
401 detecta_jogada: edge_holder
402 generic map (size => 4)
403 port map (
404     clock      => clock,
405     reset      => zeraR,
406     entrada    => chaves,
407     saida      => s_chaves
408 );
```

Figura 76: Componente *edge holder* instanciado no FD.

Demais alterações menores feitas no componente podem ser observadas no arquivo fonte, anexo.

8.1.4 Alterações na unidade de controle

A unidade de controle foi modificada para dar suporte à funcionalidade de jogada dupla. Tal consiste, principalmente, na adição de um novo estado, tanto na gravação quanto na partida, para espera e detecção da segunda jogada. Como consequência, toda jogada, mesmo que o dado na memória não contenha dois bits ‘1’, pode receber dois botões. No entanto, o circuito obviamente retorna a jogada como um erro.

Primeiramente, comenta-se a nova entidade da UC. Foram adicionados dois sinais, um de entrada e outro de saída. O sinal “*fim_tempo_2*”, entrada, corresponde a um sinal de condição, ao passo que “*contaT2*”, saída, é sinal de controle. Tal configuração é dual à declaração destes sinais no FD.

```

36      fim_tempo      : in std_logic;
37      fim_tempo_2    : in std_logic;
38      fim_espera     : in std_logic;
39      modo_escrita   : in std_logic;
40
41      -- Sinais de controle
42      zeraC          : out std_logic;
43      contaC         : out std_logic;
44      carregaC       : out std_logic;
45
46      zeraR          : out std_logic;
47      registraR      : out std_logic;
48
49      zeraP          : out std_logic;
50      atualizaP      : out std_logic;
51      diminuiP_jogada: out std_logic; -- utilizado para diminuir pontuação ao longo do tempo
52      resetaP_jogada : out std_logic; -- utilizado para colocar a pontuação da rodada em 0100
53
54      registra_modo  : out std_logic;
55
56      pronto         : out std_logic;
57
58      contaT         : out std_logic;
59      contaT2        : out std_logic;
60      zeraT          : out std_logic;
61

```

Figura 77: Nova entidade da UC.

O conjunto de estados da UC foi aumentado em dois. Agora tem-se: “espera_jogada_2” e “espera_escrita_2”.

```

69  architecture fsm of unidade_controle is
70  type t_estado is (
71      inicial, inicializa_elem, inicializa_elem_esc,
72      espera_jogada, compara, acerto,
73      espera_escrita, escreve_jogada,
74      termina_tempo, ultima_jogada, proximo, fim,
75      espera_jogada_2, espera_escrita_2
76  ); -- novo estado para escrita
77  signal Eatual, Eprox: t_estado;

```

Figura 78: Conjunto de estados da UC.

O estado “espera_escrita_2” é atingido quando se faz uma jogada e não havia sido feita uma jogada antes - parte-se de “espera_escrita”. Caso já esteja no estado, não há transição enquanto não der *timeout*. O funcionamento para se atingir “espera_jogada_2” é análogo. Nota-se que as transições a partir desses estados são incondicionais.

```

121      espera_escrita_2 when (
122          (Eatual = espera_escrita and jogada = '1') or
123          (Eatual = espera_escrita_2 and fim_tempo_2 = '0')
124      )

```

Figura 79: Lógica de transição para “espera_escrita_2”.

```

110      |
111      |      espera_jogada_2 when (
112      |          (Eatual = espera_jogada and jogada = '1') or
113      |          (Eatual = espera_jogada_2 and fim_tempo_2 = '0')

```

Figura 80: Lógica de transição para “espera_jogada_2”.

Ressalta-se que o único sinal de controle acionado em qualquer uma das segundas esperas é o “contaT2”. Por fim, abaixo tem-se o novo código de estados.

```

195      with Eatual select
196          db_estado <= "0000" when inicial,           -- 0
197              "0001" when inicializa_elem,        -- 1
198              "0010" when espera_jogada,         -- 2
199              "0011" when espera_jogada_2,       -- 3
200              "0100" when compara,             -- 4
201              "0101" when acerto,              -- 5
202              "0110" when espera_escrita,       -- 6
203              "0111" when escreve_jogada,       -- 7
204              "1000" when termina_tempo,        -- 8
205              "1001" when ultima_jogada,        -- 9
206              "1010" when proximo,            -- A
207              "1011" when espera_escrita_2,     -- B
208              "1111" when fim,                -- F
209              "1110" when others;            -- E
210      end architecture fsm;

```

Figura 81: Código para estado para depuração.

8.2 Simulações com o ModelSim

São aplicados três *testbenches* para avaliação do circuito, baseados nos usados na semana anterior. Foram feitas correções na temporização das jogadas, devido ao acréscimo da segunda jogada, e incluídos casos em que é exigida uma jogada dupla, havendo acerto e erro.

a) Cenário de teste 1 - Acerto integral:

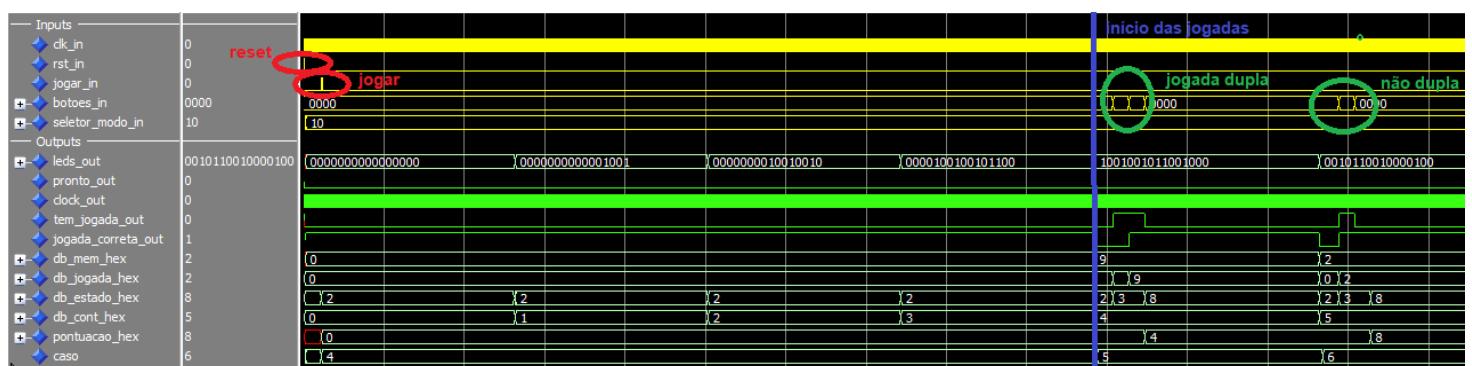


Figura 82: Primeira parte do primeiro cenário de teste.

Nota-se que a primeira parte do cenário é extremamente similar à da semana anterior. Ressalta-se a mudança da inclusão de jogada dupla. Tal ocorre, por exemplo, no caso 5. O caso seguinte mostra a diferença para uma jogada simples. Nela, há apenas uma transição de valor da entrada botões. Caso a jogada seja dupla, tal chega ao estado de comparação assim que o *edge detector* nota a segunda jogada.

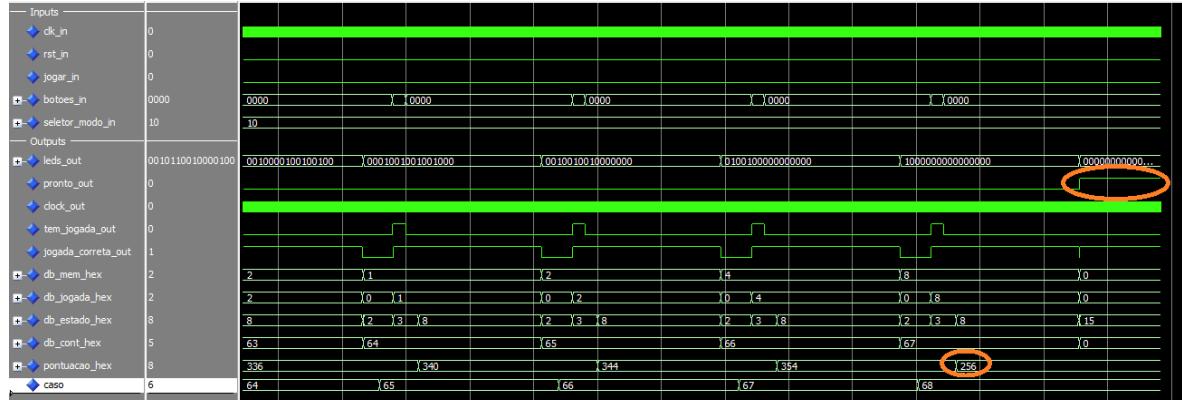


Figura 83: Segunda parte do segundo cenário de teste.

A imagem acima destaca o término do jogo na jogada adequada bem como a correta execução do teste, posto que é atingida a máxima pontuação.

b) Cenário de teste 2 - Acerto parcial:

São introduzidos erros nas jogadas 1, 5, 19, 29, 31 e timeout na 9 - elipses vermelhas. Além disso, no caso 14 são obtidos apenas dois pontos - elipse verde. Tal se confirma na imagem abaixo.



Figura 84: Erros no cenário 2.

Como esperado, foram obtidos 230 pontos de 256. O sinal pronto levantou de maneira esperada.

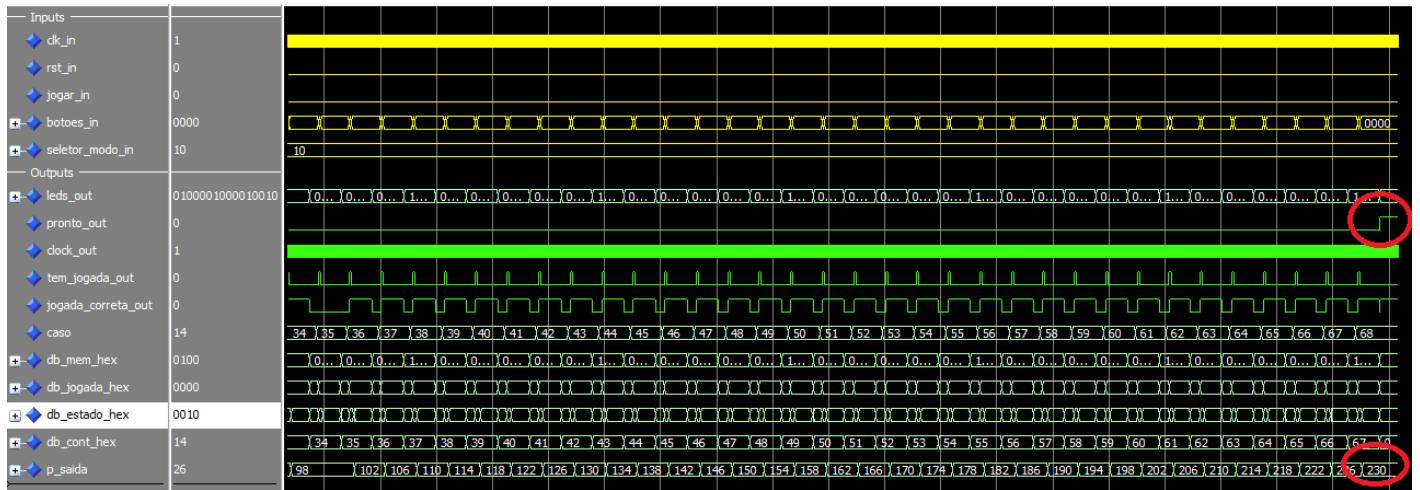


Figura 85: Parte final do cenário 2.

- c) Cenário de teste 3 - Gravação e partida no modo personalizado com performance perfeita.

Uma sequência de uns é gravada na memória num primeiro momento. Em seguida, joga-se a sequência gravada. Nesse primeiro trecho são gravados todos os dados “0001”.

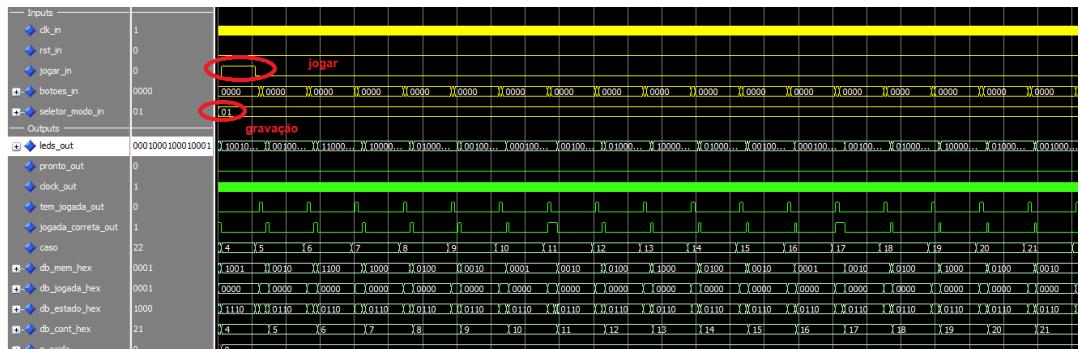


Figura 86: Parte inicial da gravação.

Em seguida, é jogada uma partida com a sequência gravada, com performance 100%. Nota-se o seletor “00”, o sinal de pronto ao final, e a pontuação final 256, obtida pelo cursor amarelo à direita é mostrada no canto inferior esquerdo.

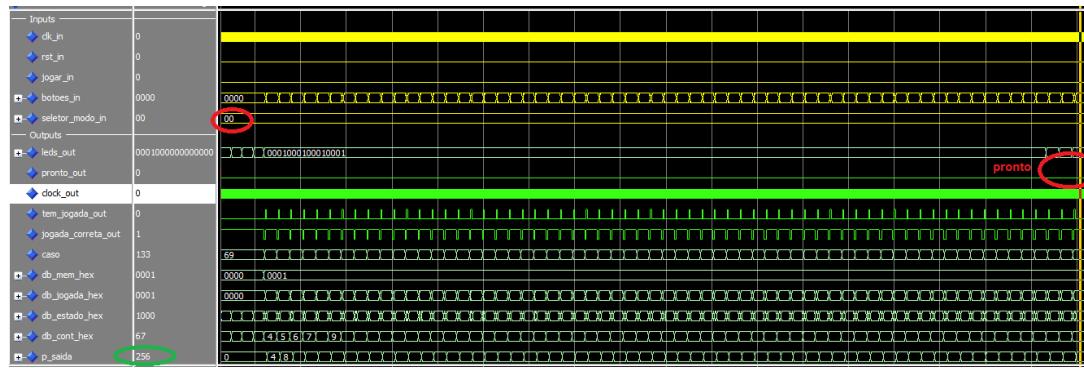


Figura 87: Partida com dados gravados.

A versão final do circuito para o planejamento apresentou o funcionamento esperado em todos os testes. Deste modo, espera-se que as falhas identificadas ao longo do planejamento tenham sido corrigidas e que a execução da implementação possa ocorrer sem demais empecilhos.

8.3 Preparação para atividades práticas

Após a verificação da corretude da lógica do circuito, foi criado um novo projeto no software Quartus. Dentro do projeto, a arquitetura utilizada para a instanciação das memórias RAM foi modificada para suportar a utilização do arquivo .mif preparado para inicialização do conteúdo.

Em seguida, foram geradas as saídas das ferramentas *RTL Viewer* e *State Machine Viewer*, que mostram a composição interna do circuito.

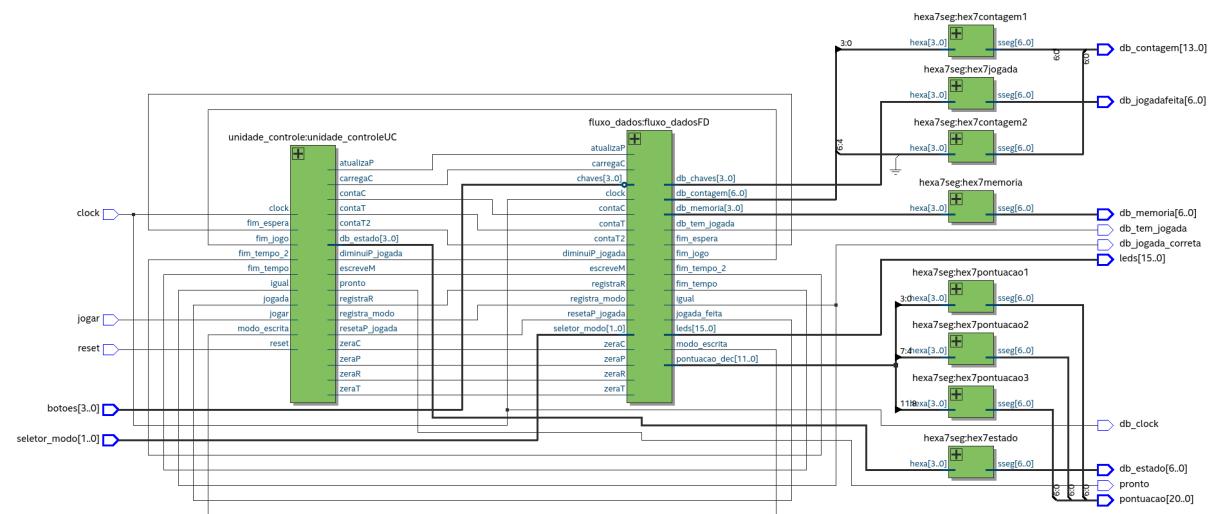


Figura 88: Diagrama RTL do circuito completo

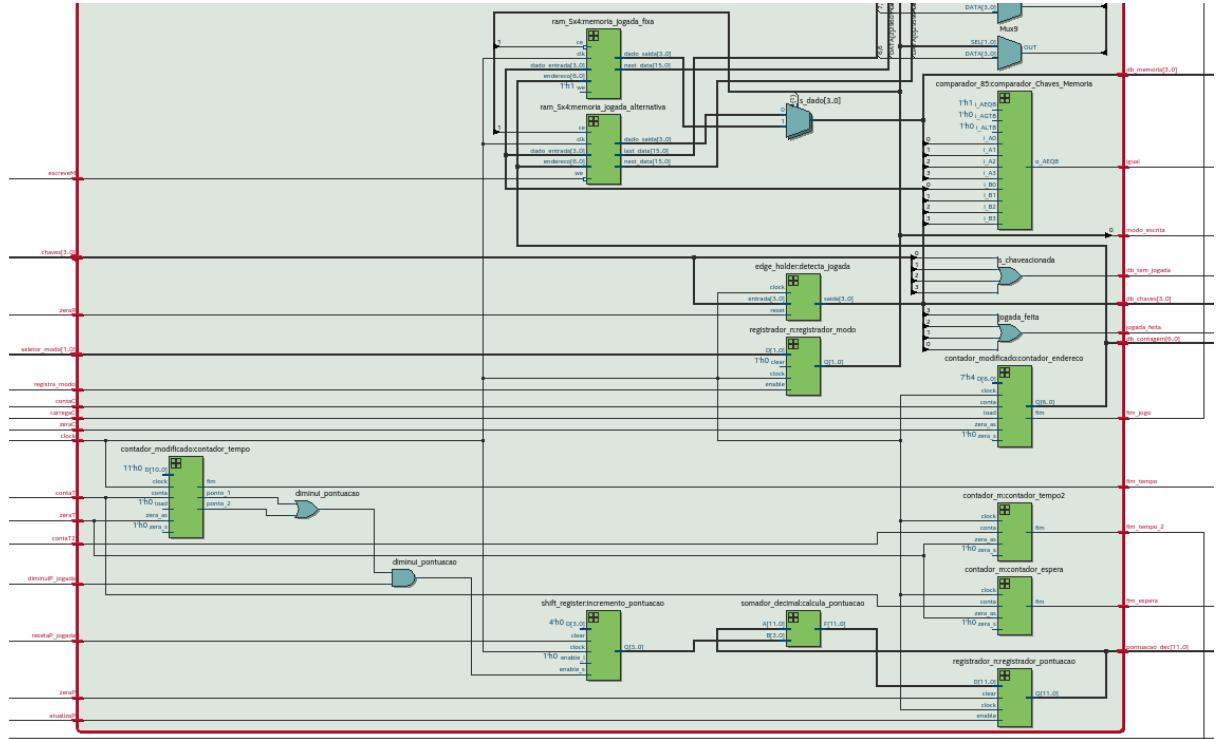


Figura 89: Parte do diagrama RTL do fluxo de dados

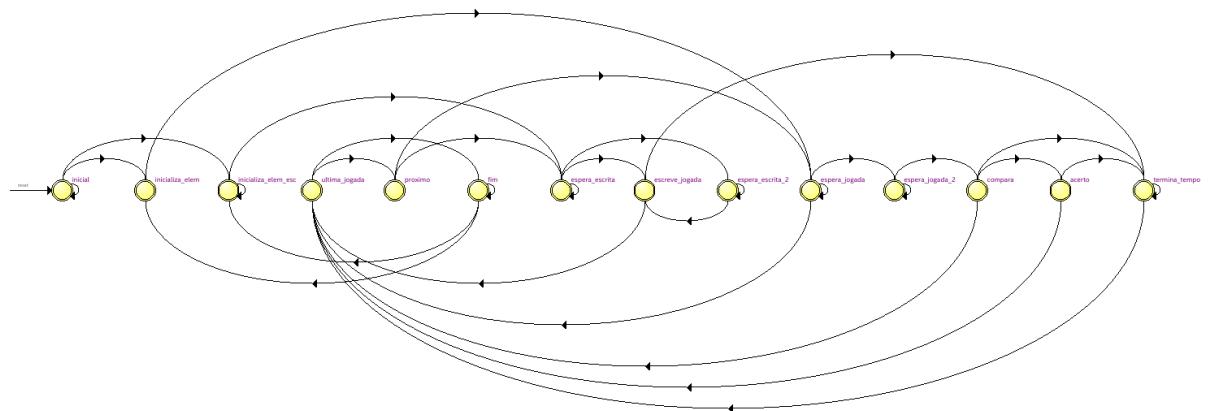


Figura 90: Diagrama de Transição de Estados gerado pelo Quartus

Source State	Destination State	Condition
1 acerto	ultima_jogada	(fim_tempo)
2 acerto	termina_tempo	(!fim_tempo)
3 compara	ultima_jogada	(fim_tempo).(!=igual)
4 compara	termina_tempo	(!fim_tempo).(!=igual)
5 compara	acerto	(igual)
6 escreve_jogada	ultima_jogada	(fim_tempo)
7 escreve_jogada	termina_tempo	(!fim_tempo)
8 espera_escrita	escreve_jogada	(fim_tempo).(!=jogada)
9 espera_escrita	espera_escrita_2	(jogada)
10 espera_escrita	espera_escrita	(!=jogada).(!fim_tempo)
11 espera_escrita_2	escreve_jogada	(fim_tempo_2)
12 espera_escrita_2	espera_escrita_2	(!fim_tempo_2)
13 espera_jogada	espera_jogada	(!fim_tempo).(!=jogada)
14 espera_jogada	ultima_jogada	(fim_tempo).(!=jogada)
15 espera_jogada	espera_jogada_2	(jogada)
16 espera_jogada_2	espera_jogada_2	(!fim_tempo_2)
17 espera_jogada_2	compara	(fim_tempo_2)
18 fim	inicializa_elem_esc	(jogar).(modo_escrita)
19 fim	inicializa_elem	(jogar).(!modo_escrita)
20 fim	fim	(!=jogar)
21 inicial	inicializa_elem_esc	(jogar).(modo_escrita)
22 inicial	inicial	(!=jogar)
23 inicial	inicializa_elem	(jogar).(!modo_escrita)
24 inicializa_elem	espera_jogada	
25 inicializa_elem_esc	inicializa_elem_esc	(!fim_espera)
26 inicializa_elem_esc	espera_escrita	(fim_espera)
27 proximo	espera_jogada	(!modo_escrita)
28 proximo	espera_escrita	(modo_escrita)
29 termina_tempo	ultima_jogada	(fim_tempo)
30 termina_tempo	termina_tempo	(!fim_tempo)
31 ultima_jogada	proximo	(!fim_jogo)
32 ultima_jogada	fim	(fim_jogo)

Figura 91: Tabela de transição de estados gerada pelo Quartus

8.3.1 Planejamento da pinagem

No mesmo projeto do Quartus, as entradas e saídas da entidade principal do circuito foram atribuídas a pinos na placa FPGA, seguindo a tabela de pinagem abaixo.

Sinal	Pino na Placa DE0-CV	Pino no FPGA	Analog Discovery	Protoboard
CLOCK	GPIO_0_D0	PIN_N16	Patterns – Clock – 1 KHz – DIO	-
RESET	GPIO_0_D1	PIN_B16	StaticIO – Button 0/1 – DIO1	-
INCIAR/JOGAR	GPIO_0_D3	PIN_C16	StaticIO – Button 0/1 – DIO2	-
BOTOES(0)	GPIO_1_D27	PIN_F15	-	botão coluna azul
BOTOES(1)	GPIO_1_D29	PIN_F12	-	botão coluna amarela
BOTOES(2)	GPIO_1_D31	PIN_G15	-	botão coluna vermelha
BOTOES(3)	GPIO_1_D33	PIN_G12	-	botão coluna verde
SELETOR_MODO(0)	GPIO_0_D4	PIN_D17	StaticIO – Switch 0/1 – DIO8	-
SELETOR_MODO(1)	GPIO_0_D5	PIN_K20	StaticIO – Switch 0/1 – DIO9	-
PRONTO	GPIO_0_D2	PIN_M16	StaticIO – LED – DIO10	-

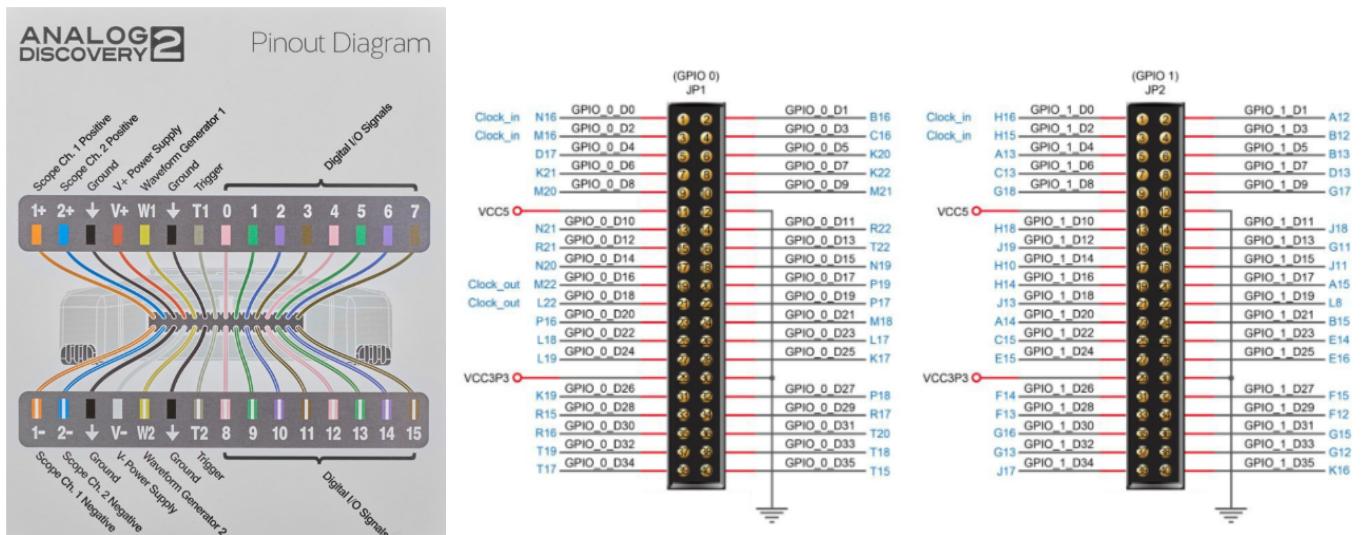
LEDS(0)	GPIO_1_D0	PIN_H16	-	led azul, linha 1
LEDS(1)	GPIO_1_D2	PIN_H15	-	led amarelo, linha 1
LEDS(2)	GPIO_1_D4	PIN_A13	-	led vermelho, linha 1
LEDS(3)	GPIO_1_D6	PIN_C13	-	led verde, linha 1
LEDS(4)	GPIO_1_D1	PIN_A12	-	led azul, linha 2
LEDS(5)	GPIO_1_D3	PIN_B12	-	led amarelo, linha 2
LEDS(6)	GPIO_1_D5	PIN_B13	-	led vermelho, linha 2
LEDS(7)	GPIO_1_D7	PIN_D13	-	led verde, linha 2
LEDS(8)	GPIO_1_D10	PIN_H18	-	led azul, linha 3
LEDS(9)	GPIO_1_D12	PIN_J19	-	led amarelo, linha 3
LEDS(10)	GPIO_1_D14	PIN_H10	-	led vermelho, linha 3
LEDS(11)	GPIO_1_D16	PIN_H14	-	led verde, linha 3
LEDS(12)	GPIO_1_D11	PIN_J18	-	led azul, linha 4
LEDS(13)	GPIO_1_D13	PIN_G11	-	led amarelo, linha 4
LEDS(14)	GPIO_1_D15	PIN_J11	-	led vermelho, linha 4
LEDS(15)	GPIO_1_D17	PIN_A15	-	led verde, linha 4
PONTUACAO(0)	GPIO_0_D10	PIN_N21	-	Primeiro display
PONTUACAO(1)	GPIO_0_D12	PIN_R21	-	Primeiro display
PONTUACAO(2)	GPIO_0_D14	PIN_N20	-	Primeiro display
PONTUACAO(3)	GPIO_0_D16	PIN_M22	-	Primeiro display
PONTUACAO(4)	GPIO_0_D18	PIN_L22	-	Primeiro display
PONTUACAO(5)	GPIO_0_D20	PIN_P16	-	Primeiro display
PONTUACAO(6)	GPIO_0_D22	PIN_P18	-	Primeiro display
PONTUACAO(7)	GPIO_0_D11	PIN_R22	-	Segundo display
PONTUACAO(8)	GPIO_0_D13	PIN_T22	-	Segundo display
PONTUACAO(9)	GPIO_0_D15	PIN_N19	-	Segundo display
PONTUACAO(10)	GPIO_0_D17	PIN_P19	-	Segundo display
PONTUACAO(11)	GPIO_0_D19	PIN_P17	-	Segundo display
PONTUACAO(12)	GPIO_0_D21	PIN_M18	-	Segundo display
PONTUACAO(13)	GPIO_0_D23	PIN_L17	-	Segundo display
PONTUACAO(14)	GPIO_0_D26	PIN_K19	-	Terceiro display
PONTUACAO(15)	GPIO_0_D28	PIN_R15	-	Terceiro display
PONTUACAO(16)	GPIO_0_D30	PIN_R16	-	Terceiro display
PONTUACAO(17)	GPIO_0_D32	PIN_T19	-	Terceiro display
PONTUACAO(18)	GPIO_0_D34	PIN_T17	-	Terceiro display
PONTUACAO(19)	GPIO_0_D27	PIN_P18	-	Terceiro display
PONTUACAO(20)	GPIO_0_D29	PIN_R17	-	Terceiro display
db_clock	Led LEDR0	PIN_AA2	-	-
db_tem_jogada	Led LEDR1	PIN_AA1	-	-
db_jogada_correta	Led LEDR2	PIN_W2	-	-
db_estado(0)	Display HEX0	PIN_U21	-	-
db_estado(1)	Display HEX0	PIN_V21	-	-

db_estado(2)	Display HEX0	PIN_W22	-	-
db_estado(3)	Display HEX0	PIN_W21	-	-
db_estado(4)	Display HEX0	PIN_Y22	-	-
db_estado(5)	Display HEX0	PIN_Y21	-	-
db_estado(6)	Display HEX0	PIN_AA22	-	-
db_jogada_feita(0)	Display HEX1	PIN_AA20	-	-
db_jogada_feita(1)	Display HEX1	PIN_AB20	-	-
db_jogada_feita(2)	Display HEX1	PIN_AA19	-	-
db_jogada_feita(3)	Display HEX1	PIN_AA18	-	-
db_jogada_feita(4)	Display HEX1	PIN_AB18	-	-
db_jogada_feita(5)	Display HEX1	PIN_AA17	-	-
db_jogada_feita(6)	Display HEX1	PIN_U22	-	-

Tabela 13: Tabela de pinagem planejada para o circuito construído.

8.3.2 Uso do Analog discovery

O dispositivo Analog Discovery para controlar alguns sinais de entrada e saída. Abaixo, tem-se o esquema de números correspondentes a cada fio do Analog Discovery, além do esquema de pinos da placa FPGA.



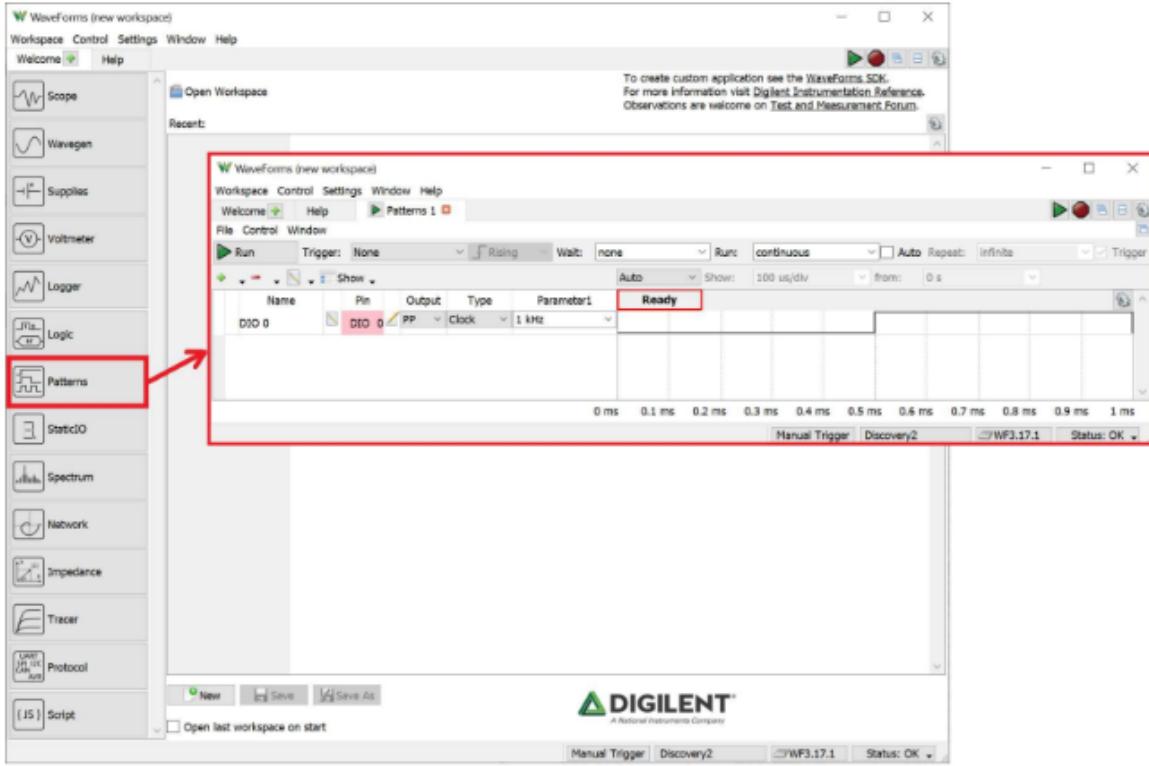


Figura 93: Interface do Waveforms e localização de comandos para configuração do relógio.

Após clicar em “Patterns”, na área à esquerda, em branco, clica-se com o botão direito do *mouse*, abrindo uma caixa de opções na qual é possível adicionar sinais - no caso apenas DIO0 será configurado dessa forma. Posteriormente, define-se o tipo como *clock* e ajusta-se o parâmetro (frequência) para 1kHz. Por fim, destaca-se que para o sinal realmente iniciar sua operação é preciso ativar o *play*, clicando na seta verde que aparece no título do *pattern* criado - no caso *patterns 1*. Quando ativo, a seta verde será substituída por um botão vermelho. Então, é preciso configurar os demais sinais, que correspondem à categoria StaticIO. As demais entradas do circuito devem ser declaradas como botões, enquanto as saídas devem ser declaradas como LEDs. Nota-se que, no canto esquerdo superior, pode-se clicar em *view* e depois em nomes, para renomear cada sinal, inicialmente denominado a partir de seu número de entrada.

9. Atividades experimentais da semana 3

Primeiro, fez-se a montagem do circuito, conectando leds, botões e *displays* na *protoboard*. Em seguida as devidas conexões entre os componentes e a FPGA foram feitas. Por fim, os sinais ainda usados no Analog Discovery foram conectados. Com isso, o circuito foi programado na FPGA e o waveforms foi ligado com os correspondentes sinais do Analog Discovery. Abaixo, tem-se uma foto da montagem:

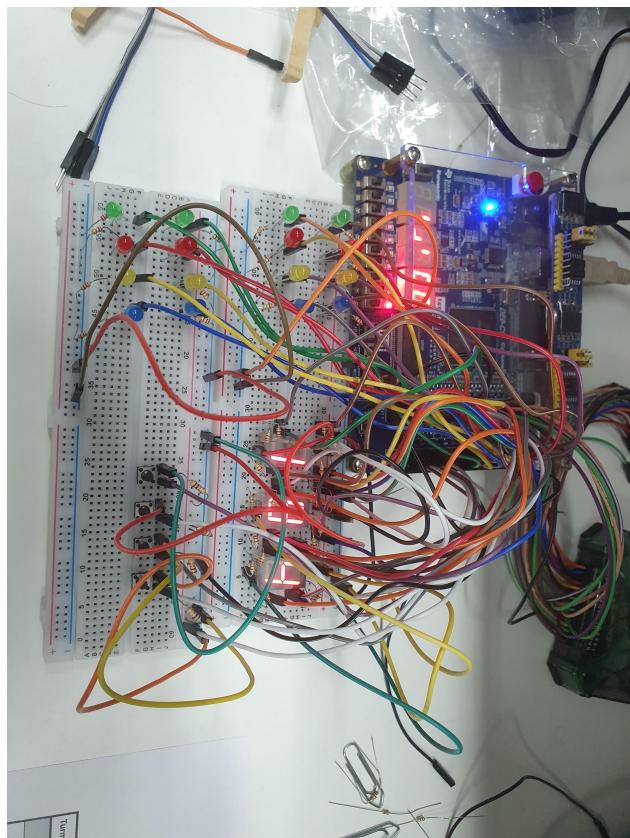


Figura 93: Montagem final do circuito.

Notou-se que os *displays* estavam todos apagados, bem como os leds. No entanto, os botões registravam jogadas. Testaram-se os três modos de jogo, sendo que tanto para o jogo gravado quanto para o pré-gravado os *leds* seguiam apagados. Porém, na gravação de jogadas, acendiam. Em seguida, testou-se novamente o jogo gravado e dessa vez os leds indicaram as jogadas correspondentes. Disso concluiu-se que, novamente, poderia-se ter um problema de inicialização de memória. Como tal não ocorreu no Modelsim, uma rápida solução foi substituir a arquitetura que usa o *.mif* pela gravada de forma rígida no VHDL. Efetuada essa mudança, o funcionamento voltou aos conformes, faltando apenas corrigir os *displays*. Após análise das conexões, percebeu-se que, como são do tipo anodo, faltava a conexão com o Vcc. Corrigindo-se isso, os *displays* apresentaram funcionamento quase perfeito. Alguns dos segmentos não ligavam nunca ou ligavam quando não deveriam e não ligavam quando deveriam. Para os que não

ligavam quando não deveriam, notou-se que um contato físico com resistores vizinhos era o problema. Para os demais, avaliou-se o que o *display* mostrava para cada algarismo. Então, comparando quando cada um dos segmentos que apresentava funcionamento inadequado ligava, foi fácil notar que havia alguma inversão das conexões entre nos GPIOs, que foi devidamente corrigida.

Após as correções, os testes de funcionamento propostos foram realizados, e foi possível verificar que o comportamento esperado foi obtido pela montagem experimental.

10. Conclusão da semana 3

Foram implementados com sucesso os requisitos agendados para a semana 3. Como pendência, tem-se descobrir o porquê de a arquitetura .mif não ter funcionado - equivale a uma memória nula. Para a semana seguinte, como não houve atraso, continua-se com o planejamento: implementar os efeitos sonoros com o *buzzer*. Além disso, uma sugestão dada pelo professor foi a implementação de modos de dificuldade diferentes, variando a velocidade. Isso não era planejado, mas uma avaliação mostra que no balanço dificuldade versus valor agregado é algo bastante desejável. Pode-se simplesmente instanciar um novo contador modificado com módulo para *timeout* diferente e usar alguns multiplexadores, ou seja, é uma alteração simples. Ao mesmo tempo, é algo que tem impacto direto na satisfação dos jogadores.

11. Planejamento da semana 4

11.1 Explicação dos requisitos a serem implementados

Inicialmente, a programação indicava a implementação do buzzer e de um display que mostrasse a pontuação máxima do jogo. No entanto, após sugestão do professor na semana anterior, notou-se que um requisito de mais simples execução, porém que agregaria maior valor ao jogo seria a introdução de diferentes modos de jogo em termos de dificuldade. Por isso, o requisito de mostrar a pontuação máxima em um *display* externo foi excluído da semana. Inicialmente, pensou-se em implementar 2 dificuldades diferentes, variando-se os tempos de *timeout* do contador modificado - tanto o total quanto as demais *flags*. Todavia, uma outra maneira de atingir esse objetivo, e que ainda por cima resolve outra questão, é usar o *clock* interno da FPGA - a figura 94 mostra a tabela de pinos.

Table 3-6 Pin Assignment of Clock Inputs

Signal Name	FPGA Pin No.	Description
CLOCK_50	PIN_M9	50 MHz clock input(Bank 3B)
CLOCK2_50	PIN_H13	50 MHz clock input(Bank 7A)
CLOCK3_50	PIN_E10	50 MHz clock input(Bank 8A)
CLOCK4_50	PIN_V15	50 MHz clock input(Bank 4A)

Figura 94: Tabela de pinos com *clocks* internos da FPGA.

Esse sinal de *clock* de alta frequência será utilizado como base para não só gerar sinais de *clock* com diferentes frequências como também as frequências necessárias para os sinais do buzzer. Para isso, será utilizada uma versão modificada do componente “contador_m”, que envia um sinal quando a contagem está na segunda metade de seu ciclo. Vale ressaltar que ambos os requisitos não requerem modificações na unidade de controle, que continua igual à da semana anterior.

11.2 Modificações circuito

11.2.1 Componente “contador_m”

```
33  entity contador_m is
34      generic (
35          constant M: integer := 100 -- modulo do contador
36      );
37      port (
38          clock      : in  std_logic;
39          zera_as   : in  std_logic;
40          zera_s    : in  std_logic;
41          conta     : in  std_logic;
42          0         : out std_logic_vector(natural(ceil(log2(real(M))))-1 downto 0);
43          fim       : out std_logic;
44          meio      : out std_logic
45      );
46  end entity contador_m;
47
```

Figura 95: Entidade do contador_m.

A entidade desse componente foi mantida, mas o significado da saída “meio” foi modificado. Anteriormente, esse sinal indicava se a contagem estava em seu

ponto médio, o que era utilizado para referência por componentes anteriores. Após a modificação, o sinal se mantém ativado a partir do ponto médio até que a contagem seja reiniciada. As funcionalidades que faziam uso desse sinal do contador já haviam sido transferidas para o componente “contador_modificado”, logo modificações na ligação com outros componentes não foram necessárias.

```
-- saída meio
meio <= '1' when IQ>=M/2-1 else
| |
| '0';
-- saída Q
Q <= std_logic_vector(to_unsigned(IQ, Q'length));
```

Figura 96: Ativação modificada da saída “meio”.

11.2.2 Componente seletor_som

Para selecionar a frequência do som gerado pelo *buzzer*, foi criado um novo componente. Este recebe como entradas os botões pressionados pelo jogador, além do *clock* do circuito, a única saída é uma onda quadrada de frequência que depende das entradas.

```
entity seletor_som is
  port (
    clock      : in  std_logic;
    toca       : in  std_logic_vector(3 downto 0);
    saída      : out std_logic
  );
end entity;
```

Figura 97: Entidade do componente.

Para a seleção de frequência, é utilizado um decodificador simples, que mapeia 10 possíveis entradas para saídas individuais (4 botões apertados individualmente e 6 possíveis pares de botões).

```

entity decoder_a is
    port (
        entrada : in std_logic_vector(3 downto 0);
        saida   : out std_logic_vector(9 downto 0)
    );
end entity;

architecture behavioral of decoder_a is
begin
    with entrada select saida <=
        "0000000000" when "0000",
        "0000000001" when "0001",
        "0000000010" when "0010",
        "0000000100" when "0100",
        "0000001000" when "1000",
        "0000010000" when "0011",
        "0000100000" when "0101",
        "0001000000" when "1001",
        "0010000000" when "0110",
        "0100000000" when "1010",
        "1000000000" when "1100",
        "0000000000" when others;
end architecture;

```

Figura 98: Entidade e arquitetura do decoder_a.

```

begin
    escolhe_nota : decoder_a
        port map(
            entrada    => toca,
            saida      => s_notas
        );

```

Figura 99: Instanciação do decoder_a.

A arquitetura do componente também inclui 10 diferentes instâncias do “contador_m”, cada uma responsável por gerar uma nota diferente, sendo as notas escolhidas F_4 , G_4 , A_4 , B_4 , C_5 , D_5 , E_5 , F_5 , $A^{\#}_4$ e $D^{\#}_5$. Cada um dos contadores é instanciado da forma:

```

A4 : contador_m
generic map(
    M => 113636
)
port map(
    clock    => clock,
    zera_as => '0',
    zera_s  => not_s_notas(2),
    conta   => s_notas(2),
    Q        => open,
    fim     => open,
    meio   => tom(2)
);

```

Figura 100: Contador que gera a frequência da nota A_4 .

E a saída é ativada através de uma operação “or” com todos os bits de “tom”.

```
saída <= tom(9) or tom(8) or tom(7) or tom(6) or tom(5) or tom(4) or tom(3) or tom(2) or tom(1) or tom(0);
```

Figura 101: Ativação da saída do componente.

11.2.3 Modificações no fluxo de dados

Para a implementação das novas funcionalidades, foram adicionados dois sinais à entidade do fluxo de dados, um responsável para enviar o sinal de *clock* que deve ser utilizado pela unidade de controle, e outro que é diretamente ligado ao *buzzer*.

```
22  entity fluxo_dados is
23  port (
24      clock          : in std_logic;
25      chaves         : in std_logic_vector (3 downto 0);
26      seletor_modo   : in std_logic_vector (2 downto 0);
27      -----
28      zeraC          : in std_logic; -- novo nome: zeraC -> zeraC
29      contaC         : in std_logic; -- novo nome: contaC -> contaC
30      carregaC       : in std_logic;
31      escreveM       : in std_logic;
32      zeraR          : in std_logic;
33      registraR      : in std_logic;
34      contaT         : in std_logic;
35      contaT2        : in std_logic;
36      zeraT          : in std_logic;
37      atualizaP      : in std_logic;
38      diminuiP_jogada: in std_logic;
39      resetaP_jogada : in std_logic;
40      zeraP          : in std_logic;
41      registra_modo  : in std_logic;
42      -----
43      igual           : out std_logic;
44      fim_jogo        : out std_logic; -- novo sinal: saída do contador de rodada
45      jogada_feita    : out std_logic;
46      fim_tempo       : out std_logic;
47      fim_tempo_2     : out std_logic;
48      fim_espera      : out std_logic;
49      modo_escrita    : out std_logic;
50      -----
51      db_tem_jogada  : out std_logic;
52      db_contagem     : out std_logic_vector (6 downto 0);
53      db_memoria      : out std_logic_vector (3 downto 0);
54      db_chaves        : out std_logic_vector (3 downto 0);
55      leds            : out std_logic_vector (15 downto 0);
56      pontuacao_dec   : out std_logic_vector (11 downto 0);
57      -----
58      clock_interno   : out std_logic;
59      buzzer          : out std_logic
60  );
61 end entity;
62 ...
```

Figura 102: Nova entidade do fluxo de dados.

A geração do sinal de *clock* compatível com a velocidade escolhida é feita a partir de duas instâncias do componente “contador_m”, que tem seus módulos ajustados para que, ao receberem um sinal de *clock* de 50 MHz, suas saídas “meio” oscilem com frequências 1 kHz e 1,6 kHz respectivamente. Essas saídas são então selecionadas por um multiplexador a partir do modo selecionado, e a saída do multiplexador é adotada como *clock* pelo circuito.

```

clock_generator_1: contador_m
generic map (M => 50000)
port map (
    clock    => clock,
    zera_as => '0',
    zera_s  => '0',
    conta   => '1',
    Q        => open,
    fim     => open,
    meio    => clk_1
);
clock_generator_2: contador_m
generic map (M => 31250)
port map (
    clock    => clock,
    zera_as => '0',
    zera_s  => '0',
    conta   => '1',
    Q        => open,
    fim     => open,
    meio    => clk_2
);

with modo(2) select clock_int <=
    clk_2 when '1',
    clk_1 when others;

clock_interno <= clock_int;

```

Figura 103: Geração e seleção dos sinais de *clock*.

Para que o circuito funcione corretamente, alguns componentes devem se manter ligados no sinal de *clock* original, sendo eles o registrador do modo, os contadores que geram o *clock*, e o componente “seletor_som” descrito acima.

```

seletor_nota: seletor_som
port map (
    clock    => clock,
    toca    => chaves,
    saida   => buzzer
);

```

Figura 104: Instanciação do componente *seletor_som*.

11.2.4. Modificações na arquitetura do circuito

A entidade externa do circuito do jogo foi modificada para apresentar a saída de ativação do *buzzer*. Além disso, as instâncias do fluxo de dados e da unidade de controle foram modificadas para que a saída “*clock_interno*” daquele esteja ligada na entrada de *clock* desta.

```

19  entity jogo_desafio ritmo is -- novo nome de entidade
20  port (
21      clock          : in std_logic;
22      reset          : in std_logic;
23      jogar          : in std_logic; -- novo nome: iniciar -> jogar
24      botoes         : in std_logic_vector (3 downto 0); -- novo nome: chaves -> botoes
25      seletor_modo   : in std_logic_vector (2 downto 0); -- seletor modo = XY => X = seletor de memória; Y = seletor de escrita
26  -----
27      leds           : out std_logic_vector (15 downto 0);
28      pronto         : out std_logic;
29      pontuacao     : out std_logic_vector (20 downto 0); -- ocupa três displays de sete segmentos
30      buzzer         : out std_logic;
31  -----
32      db_clock       : out std_logic;
33      db_tem_jogada : out std_logic;
34      db_jogada_correta : out std_logic;
35      db_contagem    : out std_logic_vector (13 downto 0); -- Ocupa dois displays de sete segmentos
36      db_memoria     : out std_logic_vector (6 downto 0);
37      db_jogadafeita : out std_logic_vector (6 downto 0);
38      db_estado       : out std_logic_vector (6 downto 0)
39  );
40 end entity;
41

```

Figura 105: Nova entidade do circuito.

```

fluxo_dadosFD: fluxo_dados -- Instanciacao modificada
port map (
    clock          => clock,
    chaves         => not_botoes,
    seletor_modo   => seletor_modo,
    -----
    zeraC          => zeraC,
    contaC        => contaC,
    carregaC      => carregaC,
    escreveM      => escreveM,
    zeraR          => zeraR,
    registraR     => registraR,
    contaT        => contaT,
    contaT2       => contaT2,
    zeraT          => zeraT,
    atualizaP     => atualizaP,
    diminuiP_jogada => diminuiP_jogada,
    resetaP_jogada => resetaP_jogada,
    zeraP          => zeraP,
    registra_modo  => registra_modo,
    -----
    igual          => igual,
    fim_jogo       => fim_jogo,
    jogada_feita  => jogada_feita,
    fim_tempo      => fim_tempo,
    fim_tempo_2    => fim_tempo_2,
    fim_espera     => fim_espera,
    modo_escrita   => modo_escrita,
    -----
    db_tem_jogada  => db_tem_jogada,
    db_contagem     => db_cont_hex,
    db_memoria     => db_mem_hex,
    db_chaves       => db_jogada_hex,
    leds            => leds,
    pontuacao_dec  => pontuacao_hex,
    -----
    clock_interno  => clock_interno,
    buzzer         => buzzer
);

unidade_controleUC: unidade_controle --Instanciacao modificada
port map (
    clock          => clock_interno,
    reset          => reset,
    jogar          => jogar,
    fim_jogo       => fim_jogo,
    jogada         => jogada_feita,
    igual          => igual,
    fim_tempo      => fim_tempo,
    fim_tempo_2    => fim_tempo_2,
    fim_espera     => fim_espera,
    modo_escrita   => modo_escrita,
    -----
    zeraC          => zeraC,
    contaC        => contaC,
    carregaC      => carregaC,
    zeraR          => zeraR,
    registraR     => registraR,
    zeraP          => zeraP,
    atualizaP     => atualizaP,
    diminuiP_jogada => diminuiP_jogada,
    resetaP_jogada => resetaP_jogada,
    registra_modo  => registra_modo,
    pronto         => pronto,
    contaT        => contaT,
    contaT2       => contaT2,
    zeraT          => zeraT,
    escreveM      => escreveM,
    db_estado       => db_estado_hex
);

```

Figuras 106 e 107: Instâncias do fluxo de dados e da unidade de controle

11.3 Simulações no Modelsim

Para checagem da correta implementação do sistema relativo ao *buzzer*, bem como do conversor de *clock*, que a depender do modo altera o relógio para um certo fator inteiro da composição da frequência fornecida ao circuito - 50MHz no caso da placa. Como não foram feitas modificações na unidade de controle e o sistema do *buzzer* não afeta o funcionamento das demais funcionalidades, apenas o conversor de *clock* poderia gerar algum mal funcionamento no já feito até este

ponto. Ocorre que, caso esta conversão funcione em um caso, funcionaria nos demais também, posto que independe de parâmetros como o “seletor_modo”. Por conta disso, decidiu-se por modificar a abordagem para o *testbench*. Em vez de usar “tb_jogo_base”, “tb_erro” e “tb_escrita”, foi montado um *testbench*, aplicado em condições diferentes com relação ao modo de jogo - porém mesmo *clock* real (não a entrada do circuito, mas sim a saída do componente conversor de *clock*). Nele, foram inseridos *timeouts*, erros e acertos. A única mudança nas entradas é o bit mais significativo do seletor, relacionado à dificuldade, posto que o mesmo *testbench* pode checar o correto funcionamento dos dois modos. Abaixo tem-se as formas de onda do teste, com destaque sempre no *buzzer* e na pontuação final. O sinal de saída do *buzzer* é suficiente para checar o sistema inteiro dessa função, enquanto a pontuação final, se correta, indica sincronização integral entre as jogadas do *testbench* e as que ocorreram na simulação, mostrando que o funcionamento foi o esperado.

Antes das formas de onda, nota-se uma adaptação que se teve que fazer para os testes com o Modelsim. Na prática, como o *clock* interno da FPGA é de 50MHz, mas o usado no jogo é de 1kHz, há um fator de 50 mil entre eles, ou seja, há cada 50 mil *clocks* internos há contabilização de um ciclo de relógio. No entanto, isso não é possível de ser testado com o Modelsim, visto que excede o limite do *software*. Por isso, fez-se uma conveniente mudança no fluxo de dados: trocou-se o módulo do contador para ‘4’. Isto não muda em nada a lógica de simulação, não invalidando-a, porém torna possível o teste com o Modelsim.

```

253      clock_generator_1: contador_m
254          generic map (M => 4)
255          port map (
256              clock    => clock,
257              zera_as => '0',
258              zera_s  => '0',
259              conta   => '1',
260              Q        => open,
261              fim      => open,
262              meio    => clk_1

```

Figura 108: Instanciação modificada do *clock_generator*.

Na figura 110 tem-se a visão geral da forma de onda obtida com as condições mostradas na figura 109.

```

126      -- gera pulso de reset (1 periodo de clock)
127      caso <= 1;
128      rst_in <= '1';
129      wait for clockPeriod;
130      rst_in <= '0';

131
132      -- espera para inicio dos testes
133      caso <= 2;
134      wait for 10 * clockPeriod;

135
136      -- pulso do sinal de Iniciar (muda na borda de descida do clock)
137      caso <= 3;
138      seletor_modo_in <= "010"; -- Codigo do jogo normal
139      wait for 100*clockPeriod;

140
141      -- pulso do sinal de Iniciar (muda na borda de descida do clock)
142      caso <= 4;
143      wait until falling_edge(clk_in);
144      jogar_in <= '1';
145      wait for 2*clockPeriod; -- chega ao espera jogada
146      -- wait for 1000*clockPeriod; -- 1 segundo: tempo para começar o jogo - FOI REMOVIDO
147      jogar_in <= '0';
148

```

Figura 109: Condições de simulação.

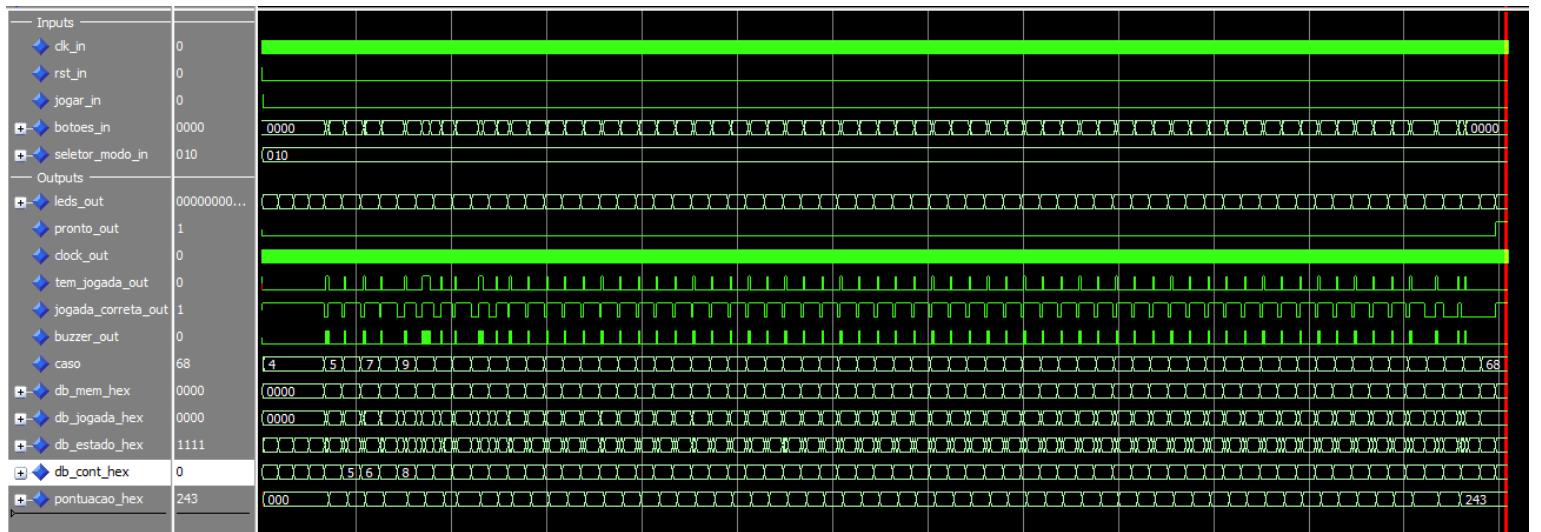


Figura 110: Visão geral da forma de onda obtida.

A visão geral mostra a periodicidade, em geral, durante a execução. Para todas as jogadas, exceto as últimas 4, há pontuação integral. Para as demais, tem-se, nessa ordem: pontuação parcial de 2 pontos, pontuação parcial de 1 ponto, erro e *timeout*. Perdem-se $2*4+(4-2)+(4-1) = 13$ pontos no total. Sendo o máximo 256, o resultado obtido, mostrado no canto inferior esquerdo, está de acordo. Tal atesta a corretude do conversor de *clock*. Com relação ao *buzzer* é possível observar seu acionamento sempre que uma jogada é feita. Para melhor entendimento da operação do circuito, abaixo tem-se a parte final detalhada:

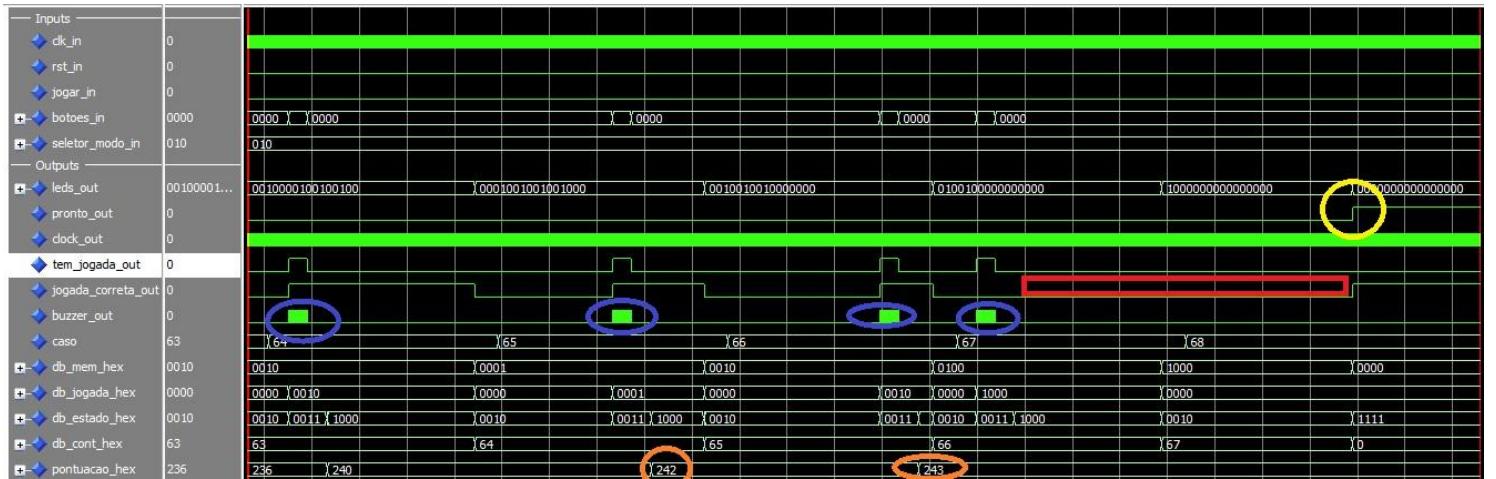


Figura 111: Detalhamento da parte final do teste.

No detalhamento, nas elipses azuis, podem-se ver os momentos nos quais o sinal que identifica a ativação do *buzzer* levanta. Estes correspondem aos momentos nos quais algum dos botões é ativado. Em laranja, tem-se as pontuações após determinadas jogadas. No caso 65 observa-se um acréscimo de 2 pontos, enquanto no seguinte de apenas 1 ponto. Posteriormente, não há mais qualquer acerto, o que indicado pelo retângulo vermelho, região na qual não há ativação do sinal “*jogada_certa*”. Por fim, em amarelo, o sinal “*pronto*” o término da partida.

Para checar o correto funcionamento do segundo modo de jogo, fez-se uma modificação no fluxo de dados, trocando-se o valor do módulo de cada *clock generator*. Desse modo, o *testbench* pôde ser integralmente reaproveitado. O resultado, naturalmente, deveria ser exatamente igual ao da visão geral, apresentada na figura 110. Como tal foi obtido, considera-se correta a implementação do segundo modo de dificuldade.

11.4 Preparação para atividades práticas

Após a verificação da correta implementação tanto da lógica do *buzzer* quanto do conversor de *clock*, foi criado um novo projeto no *software Quartus*, alterando-se os módulos dos *clock generators* para o adequado à atividade experimental.

Abaixo tem-se o RTL modificado do fluxo de dados, excetuando-se os leds, que agora contém novos componentes. Não se mostra a *State machine viewer*, pois a FSM não foi modificada.

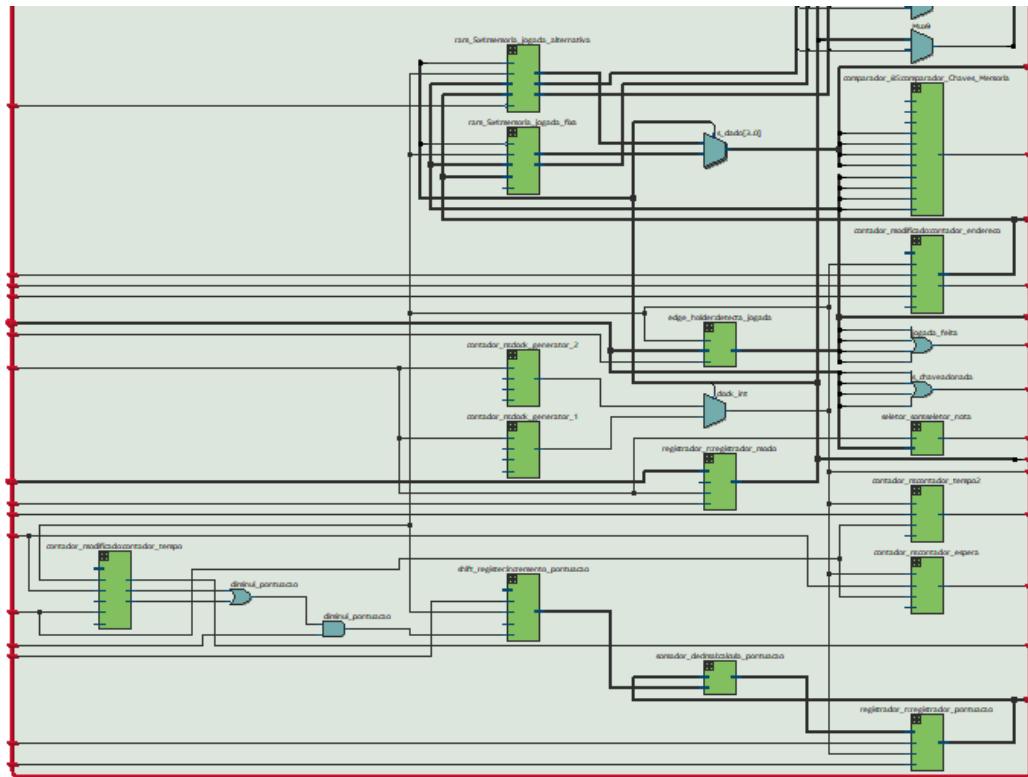


Figura 112: RTL do fluxo de dados..

11.3.1 Planejamento da pinagem

No mesmo projeto do Quartus, as entradas e saídas da entidade principal do circuito foram atribuídas a pinos na placa FPGA, seguindo a tabela de pinagem abaixo. Ressalta-se que o *clock* foi trocado para o interno da FPGA. Por conta disso, pensou-se ser um bom momento para cortar as demais dependências com o Analog Discovery.

Sinal	Pino na Placa DE0-CV	Pino no FPGA	Analog Discovery	Protoboard
CLOCK	--	PIN_M9	-	-
RESET	Push-Button(0)	PIN_U7	-	-
INCIAR/JOGAR	Push-Button(1)	PIN_W9	-	-
BOTOES(0)	GPIO_1_D27	PIN_F15	-	botão coluna azul
BOTOES(1)	GPIO_1_D29	PIN_F12	-	botão coluna amarela
BOTOES(2)	GPIO_1_D31	PIN_G15	-	botão coluna vermelha
BOTOES(3)	GPIO_1_D33	PIN_G12	-	botão coluna verde
BUZZER	GPIO_0_D35	PIN_T15	-	buzzer
SELETOR_MODO(0)	sw1	PIN_U13	-	-

SELETOR_MODO(1)	sw2	PIN_V13	-	-
SELETOR_MODO(2)	sw3	PIN_T13	-	-
PRONTO	LED8	PIN_L2	-	-
LEDS(0)	GPIO_1_D0	PIN_H16	-	led azul, linha 1
LEDS(1)	GPIO_1_D2	PIN_H15	-	led amarelo, linha 1
LEDS(2)	GPIO_1_D4	PIN_A13	-	led vermelho, linha 1
LEDS(3)	GPIO_1_D6	PIN_C13	-	led verde, linha 1
LEDS(4)	GPIO_1_D1	PIN_A12	-	led azul, linha 2
LEDS(5)	GPIO_1_D3	PIN_B12	-	led amarelo, linha 2
LEDS(6)	GPIO_1_D5	PIN_B13	-	led vermelho, linha 2
LEDS(7)	GPIO_1_D7	PIN_D13	-	led verde, linha 2
LEDS(8)	GPIO_1_D10	PIN_H18	-	led azul, linha 3
LEDS(9)	GPIO_1_D12	PIN_J19	-	led amarelo, linha 3
LEDS(10)	GPIO_1_D14	PIN_H10	-	led vermelho, linha 3
LEDS(11)	GPIO_1_D16	PIN_H14	-	led verde, linha 3
LEDS(12)	GPIO_1_D11	PIN_J18	-	led azul, linha 4
LEDS(13)	GPIO_1_D13	PIN_G11	-	led amarelo, linha 4
LEDS(14)	GPIO_1_D15	PIN_J11	-	led vermelho, linha 4
LEDS(15)	GPIO_1_D17	PIN_A15	-	led verde, linha 4
PONTUACAO(0)	GPIO_0_D10	PIN_N21	-	Primeiro display
PONTUACAO(1)	GPIO_0_D12	PIN_R21	-	Primeiro display
PONTUACAO(2)	GPIO_0_D14	PIN_N20	-	Primeiro display
PONTUACAO(3)	GPIO_0_D16	PIN_M22	-	Primeiro display
PONTUACAO(4)	GPIO_0_D18	PIN_L22	-	Primeiro display
PONTUACAO(5)	GPIO_0_D20	PIN_P16	-	Primeiro display
PONTUACAO(6)	GPIO_0_D22	PIN_L18	-	Primeiro display
PONTUACAO(7)	GPIO_0_D11	PIN_R22	-	Segundo display
PONTUACAO(8)	GPIO_0_D13	PIN_T22	-	Segundo display
PONTUACAO(9)	GPIO_0_D15	PIN_N19	-	Segundo display
PONTUACAO(10)	GPIO_0_D17	PIN_P19	-	Segundo display
PONTUACAO(11)	GPIO_0_D19	PIN_P17	-	Segundo display
PONTUACAO(12)	GPIO_0_D21	PIN_M18	-	Segundo display
PONTUACAO(13)	GPIO_0_D23	PIN_L17	-	Segundo display
PONTUACAO(14)	GPIO_0_D26	PIN_K19	-	Terceiro display
PONTUACAO(15)	GPIO_0_D28	PIN_R15	-	Terceiro display
PONTUACAO(16)	GPIO_0_D30	PIN_R16	-	Terceiro display
PONTUACAO(17)	GPIO_0_D32	PIN_T19	-	Terceiro display
PONTUACAO(18)	GPIO_0_D34	PIN_T17	-	Terceiro display
PONTUACAO(19)	GPIO_0_D27	PIN_P18	-	Terceiro display
PONTUACAO(20)	GPIO_0_D29	PIN_R17	-	Terceiro display
db_clock	Led LEDR0	PIN_AA2	-	-

db_tem_jogada	Led LEDR1	PIN_AA1	-	-
db_jogada_correta	Led LEDR2	PIN_W2	-	-
db_memoria(0)	Display HEX0	PIN_U21	-	-
db_memoria(1)	Display HEX0	PIN_V21	-	-
db_memoria(2)	Display HEX0	PIN_W22	-	-
db_memoria(3)	Display HEX0	PIN_W21	-	-
db_memoria(4)	Display HEX0	PIN_Y22	-	-
db_memoria(5)	Display HEX0	PIN_Y21	-	-
db_memoria(6)	Display HEX0	PIN_AA22	-	-
db_jogada_feita(0)	Display HEX1	PIN_AA20	-	-
db_jogada_feita(1)	Display HEX1	PIN_AB20	-	-
db_jogada_feita(2)	Display HEX1	PIN_AA19	-	-
db_jogada_feita(3)	Display HEX1	PIN_AA18	-	-
db_jogada_feita(4)	Display HEX1	PIN_AB18	-	-
db_jogada_feita(5)	Display HEX1	PIN_AA17	-	-
db_jogada_feita(6)	Display HEX1	PIN_U22	-	-

Tabela 14: Tabela de pinagem planejada para o circuito construído.

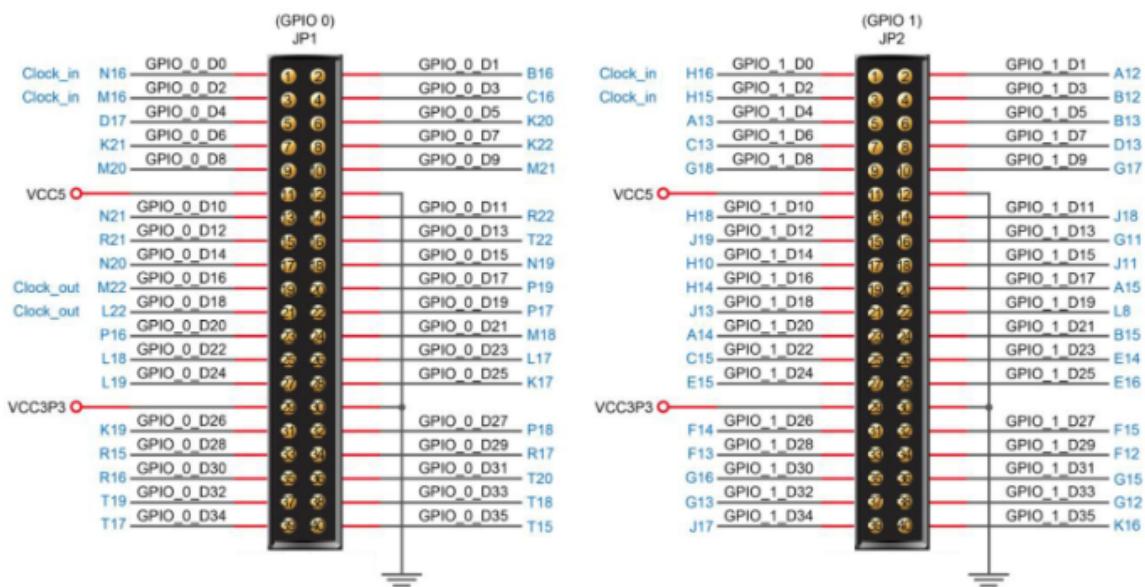


Figura 113:Numeração dos pinos das GPIOs.

12. Atividades experimentais

Após montagem do circuito conforme especificado, fez-se a programação na placa FPGA. No entanto, para o *clock* mais lento, alguns problemas foram identificados. De forma anômala, após qualquer botão ser apertado o jogo terminava. Em seguida, notou-se estranho comportamento do *clock* interno da FPGA. Por isso, optou-se por, como segundo recurso, usar o *clock* do Analog Discovery. Além disso, trocou o *reset* para uma chave, posto que o comportamento anômalo poderia vir, também, do próprio botão, como esse passo não corrigiu o problema, pode-se concluir que esse não era o caso. Nota-se que a lógica invertida foi corretamente usada para as entradas passadas a botões da placa FPGA. Com estas alterações, fizeram-se diversos testes com *clock* do Analog Discovery. Novamente, de forma aparentemente aleatória, identificaram-se términos prematuros das partidas quando algum botão era apertado. O problema era mais acentuado para o modo de jogo mais lento. Decidiu-se, por não se saber a origem do problema, excluir o arquivo modificado .qar e usar o base, enviado no planejamento. Novamente, foram feitas as mudanças necessárias para colocar o programa na placa, adicionando-se a lógica invertida, bem como as frequências adequadas para se escutar no *buzzer*. Com isto, o circuito teve funcionamento mais próximo do esperado, não havendo interrupções durante o jogo por conta dos botões. Todavia, identificou-se, que os modos estavam funcionando de forma oposta ao esperado quanto à velocidade.

```
clock_generator_1: contador_m
generic map (M => 50000)
port map (
    clock  => clock,
    zera_as => '0',
    zera_s => modo(2),
    conta   => not_modo_2,
    Q       => open,
    fim     => open,
    meio    => clk_1
);
clock_generator_2: contador_m
generic map (M => 30000)
port map (
    clock  => clock,
    zera_as => '0',
    zera_s => not_modo_2,
    conta   => modo(2),
    Q       => open,
    fim     => open,
    meio    => clk_2
);
clock_int <= (clk_1 and not_modo_2) or (clk_2 and modo(2));
```

Figura 114: Lógica de seleção de clocks final.

Disto, procedeu-se a análise do multiplexador que seleciona os *clocks*. Modificando o multiplexador para uma expressão em álgebra booleana equivalente, o circuito apresentou funcionamento adequado. Não se sabe exatamente a razão de isto ter arrumado a falha, porém o novo .qar foi salvo e será usado para a apresentação final. Abaixo tem-se uma foto do jogo em operação.

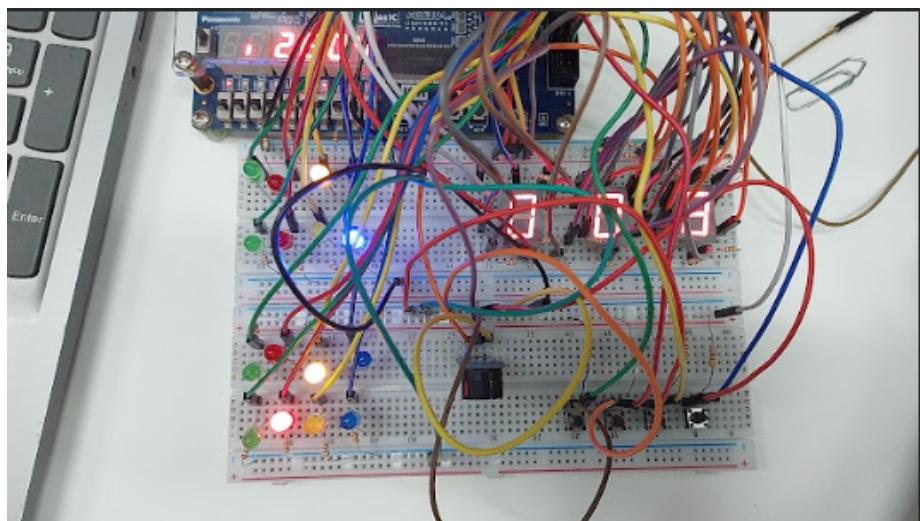


Figura 115: Jogo em operação.

13. Conclusão da semana 4

Nesta semana, a montagem foi concluída com êxito sem necessidade de correção. No entanto, o sistema de conversão de *clock* apresentou funcionamento anômalo, demandando todo o tempo do laboratório para ser corrigido. Nesse sentido, pode-se considerar que as atividades experimentais foram majoritariamente bem sucedidas. Devido à demora para solucionar o problema, não houve tempo para se pensar, por exemplo, em algum *design* para a apresentação do projeto, que a princípio será feita utilizando a montagem de protoboards.

14. Modificações para a feira de projetos

Após a semana 4, durante um *open-lab*, foram realizados novos testes, visando garantir o correto funcionamento do projeto. Observaram-se certos comportamentos anômalos, em ocasiões aparentemente aleatórias. Tais consistiam no encerramento prematuro das partidas. Foi realizada uma sequência de procedimentos de depuração para encontrar o erro: colocaram-se nos *displays* a contagem, o estado e a saída da memória. Com isto, observou-se que a contagem não era de fato incrementada e que o jogo terminava imediatamente, voltando ao estado “0” - código para “inicial”. Com isto, fez-se a seguinte hipótese: tal estaria acontecendo por uma combinação de sinais de condição não ocorrer durante a execução do jogo. Para confirmar isto, fez-se um teste. Dentro da FSM da unidade de controle, trocou-se o estado atingido após cláusula “else” de “inicial” para “fim”. Com isto, observou-se, após reprogramação na placa, que agora o estado “fim” que era atingido prematuramente. Portanto, a hipótese foi confirmada. Ocorre que, desde a semana anterior, não foram feitas alterações na unidade de controle, nem alterações no fluxo de dados que resultariam em mudanças nos sinais de controle. Desse modo, há uma aparente contradição. Após consulta com outros colegas de turma, descobriu-se que alguns tiveram problema ao usar o *clock* interno da FPGA com o divisor de *clock* para o circuito. Tal seria proveniente do fato de o tempo de subida e de descida do sinal obtido com o divisor de *clock* ser maior que o próprio ciclo de *clock* original - entrada do divisor -, o que gera uma zona na qual não se tem certeza do que está acontecendo. Com essa possibilidade contabilizada, fizeram-se as seguintes modificações: colocou-se o *clock* do Analog Discovery de volta na placa - fazendo-se a modificação correspondente na pinagem - e criou-se um sinal de *clock* separado para ser usado no *buzzer*. Para gerar a segunda dificuldade, criou-se uma nova instância do contado e fez-se um multiplexador para as saídas que envolviam a escolha da dificuldade, com este *bit* como seletor. Com isto, foi observado um comportamento adequado para o circuito na placa. Não se documentam simulações com o Modelsim para esta seção, tendo em vista seu caráter prático. De fato, o Modelsim não indicou erro algum, dado que o *clock* por ele usado, que é criado (artificialmente) no *testbench*, não apresenta esse problema. Abaixo tem-se as figuras correspondentes às alterações do fluxo de dados mencionadas no parágrafo.

```

22  entity fluxo_dados is
23  port (
24      clock          : in  std_logic;
25      | clock_busser   : in  std_logic;
26      chaves         : in  std_logic_vector (3 downto 0);
27      seletor_modo    : in  std_logic_vector (2 downto 0);
28
29      seraC           : in  std_logic; -- novo nome: seraC -> seraC
30      contaC          : in  std_logic; -- novo nome: contaC -> contaC
31      carregaC        : in  std_logic;
32      escreveM        : in  std_logic;
33      seraR           : in  std_logic;
34      registraR       : in  std_logic;
35      contaT          : in  std_logic;
36      contaT2         : in  std_logic;
37      seraT            : in  std_logic;
38      atualizaP       : in  std_logic;
39      diminuiP_jogada : in  std_logic;
40      resetaP_jogada  : in  std_logic;
41      seraP            : in  std_logic;
42      registra_modo   : in  std_logic;
43
44      igual            : out std_logic;
45      fim_jogo         : out std_logic; -- novo sinal: saída do contador de rodada
46      jogada_feita     : out std_logic;
47      fim_tempo        : out std_logic;
48      fim_tempo_2      : out std_logic;
49      fim_espera       : out std_logic;
50      modo_escrita     : out std_logic;
51
52      db_tem_jogada   : out std_logic;
53      db_contagem      : out std_logic_vector (6 downto 0);
54      db_memoria       : out std_logic_vector (3 downto 0);
55      db_chaves         : out std_logic_vector (3 downto 0);
56      leds              : out std_logic_vector (15 downto 0);
57      pontuacao_dec    : out std_logic_vector (11 downto 0);
58
59      clock_interno    : out std_logic;
60      busser            : out std_logic;
61      | seletor_out      : out std_logic_vector(2 downto 0)
62  );
63 end entity;

```

Figura 116: Nova entidade do fluxo de dados, como *clock* específico do Buzzer.

```

422  port map (
423      clock => clk_1,
424      zera_as => zeraT,
425      zera_s => '0',
426      conta => contaT,
427      load  => '0',
428      D     => std_logic_vector(to_unsigned(0, 10)),
429      Q     => open,
430      fim   => fim_tempo_int2,
431      ponto_1 => primeiro_ponto_2,
432      ponto_2 => segundo_ponto_2
433  );
434
435  --
436  fim_tempo <= (fim_tempo_int1 and not modo(2)) or (fim_tempo_int2 and modo(2));
437  primeiro_ponto <= (primeiro_ponto_1 and not modo(2)) or (primeiro_ponto_2 and modo(2));
438  segundo_ponto <= (segundo_ponto_1 and not modo(2)) or (segundo_ponto_2 and modo(2));
439

```

Figura 117: Nova instância de contador e expressões booleanas que implementam multiplexadores.

```

486      selector_nota: selector_som
487      port map (
488          clock    => clock_buzzer,
489          toca     => chaves,
490          saida    => buzzer
491      );

```

Figura 118: Instanciação do seletor_som modificada com novo *clock*.

Por fim, nota-se que a pinagem foi modificada, principalmente para tornar a montagem mais simples, diminuindo o cruzamento de fios na ligação entre *protoboard* e GPIOs. Outra medida adotada, dado que já se usaria o Analog Discovery para o *clock*, foi colocar alguns sinais de volta nele, com interação do usuário pelo *software Waveforms*. Tais sinais correspondem a: *reset* (botão), todos os bits do modo de jogo (chave), pronto (led) e jogar (botão). O esquema utilizado está disponível na pasta “Entregas” do repositório no Github - com link compartilhado em pdf à parte enviado junto à documentação final - e corresponde ao arquivo: waveformFeira.dwf3wor. Observa-se que os nomes da figura abaixo, quando usado o arquivo na feira, foram trocados para serem mais amigáveis ao público geral, não referenciando nada relativo a sistemas digitais, circuitos, etc.

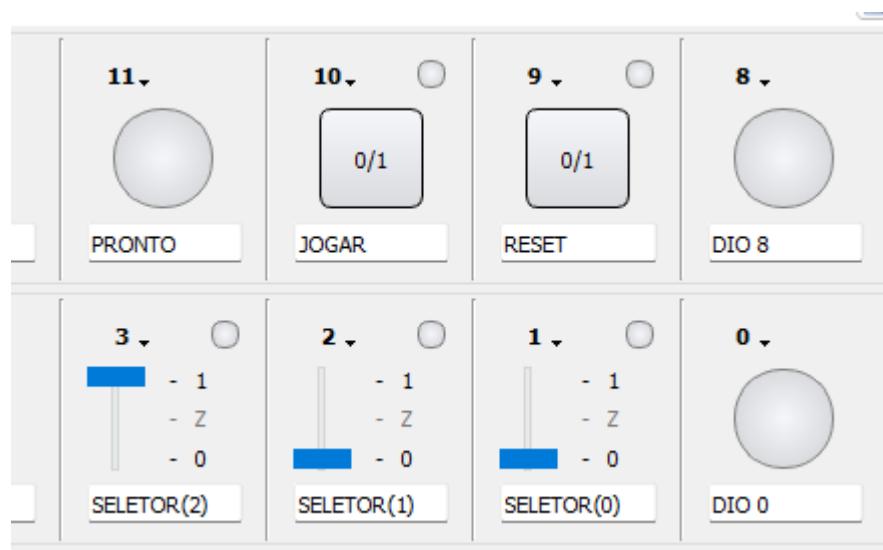


Figura 119: Configuração da interface do Waveforms com sinais do Analog Discovery.

Abaixo tem-se a tabela de pinos atualizada com as mudanças citadas acima:

Sinal	Pino na Placa DE0-CV	Pino no FPGA	Analog Discovery	Protoboard
CLOCK	GPIO_1_D0	PIN_H16	Patterns – Clock – 1 KHz – DIO	-
RESET	GPIO_1_D4	PIN_A13	StaticIO – Button 0/1 – DIO9	-
INCIAR/JOGAR	GPIO_1_D6	PIN_C13	StaticIO – Button 0/1 – DIO10	-
BOTOES(0)	GPIO_0_D27	PIN_P18	-	botão coluna azul
BOTOES(1)	GPIO_0_D29	PIN_R17	-	botão coluna amarela
BOTOES(2)	GPIO_0_D31	PIN_T20	-	botão coluna vermelha
BOTOES(3)	GPIO_0_D33	PIN_T18	-	botão coluna verde
BUZZER	GPIO_0_D35	PIN_T15	-	buzzer
SELETOR_MODO(0)	GPIO_1_D1	PIN_A12	StaticIO – Switch 0/1 – DIO1	-
SELETOR_MODO(1)	GPIO_1_D3	PIN_B12	StaticIO – Switch 0/1 – DIO2	-
SELETOR_MODO(2)	GPIO_1_D5	PIN_B13	StaticIO – Switch 0/1 – DIO3	-
PRONTO	GPIO_1_D2	PIN_H15	StaticIO – LED – DIO11	-
LEDS(0)	GPIO_0_D0	PIN_N16	-	led azul, linha 1
LEDS(1)	GPIO_0_D2	PIN_M16	-	led amarelo, linha 1
LEDS(2)	GPIO_0_D4	PIN_D17	-	led vermelho, linha 1
LEDS(3)	GPIO_0_D6	PIN_K21	-	led verde, linha 1
LEDS(4)	GPIO_0_D1	PIN_B16	-	led azul, linha 2
LEDS(5)	GPIO_0_D3	PIN_C16	-	led amarelo, linha 2
LEDS(6)	GPIO_0_D5	PIN_K20	-	led vermelho, linha 2
LEDS(7)	GPIO_0_D7	PIN_K22	-	led verde, linha 2
LEDS(8)	GPIO_0_D10	PIN_N21	-	led azul, linha 3
LEDS(9)	GPIO_0_D12	PIN_R21	-	led amarelo, linha 3
LEDS(10)	GPIO_0_D14	PIN_N20	-	led vermelho, linha 3
LEDS(11)	GPIO_0_D16	PIN_M22	-	led verde, linha 3
LEDS(12)	GPIO_0_D11	PIN_R22	-	led azul, linha 4
LEDS(13)	GPIO_0_D13	PIN_T22	-	led amarelo, linha 4
LEDS(14)	GPIO_0_D15	PIN_N19	-	led vermelho, linha 4
LEDS(15)	GPIO_0_D17	PIN_P19	-	led verde, linha 4
PONTUACAO(0)	GPIO_1_D11	PIN_J18	-	Primeiro display
PONTUACAO(1)	GPIO_1_D13	PIN_G11	-	Primeiro display
PONTUACAO(2)	GPIO_1_D15	PIN_J11	-	Primeiro display
PONTUACAO(3)	GPIO_1_D17	PIN_A15	-	Primeiro display
PONTUACAO(4)	GPIO_1_D19	PIN_L8	-	Primeiro display
PONTUACAO(5)	GPIO_1_D21	PIN_B15	-	Primeiro display
PONTUACAO(6)	GPIO_1_D23	PIN_E14	-	Primeiro display
PONTUACAO(7)	GPIO_1_D10	PIN_H18	-	Segundo display

PONTUACAO(8)	GPIO_1_D12	PIN_J19	-	Segundo display
PONTUACAO(9)	GPIO_1_D14	PIN_H10	-	Segundo display
PONTUACAO(10)	GPIO_1_D16	PIN_H14	-	Segundo display
PONTUACAO(11)	GPIO_1_D18	PIN_J13	-	Segundo display
PONTUACAO(12)	GPIO_1_D20	PIN_A14	-	Segundo display
PONTUACAO(13)	GPIO_1_D22	PIN_C15	-	Segundo display
PONTUACAO(14)	GPIO_1_D26	PIN_F14	-	Terceiro display
PONTUACAO(15)	GPIO_1_D28	PIN_F13	-	Terceiro display
PONTUACAO(16)	GPIO_1_D30	PIN_G16	-	Terceiro display
PONTUACAO(17)	GPIO_1_D32	PIN_G13	-	Terceiro display
PONTUACAO(18)	GPIO_1_D34	PIN_J17	-	Terceiro display
PONTUACAO(19)	GPIO_1_D27	PIN_F15	-	Terceiro display
PONTUACAO(20)	GPIO_1_D29	PIN_F12	-	Terceiro display
db_clock	Led LEDR0	PIN_AA2	-	-
db_tem_jogada	Led LEDR1	PIN_AA1	-	-
db_jogada_correta	Led LEDR2	PIN_W2	-	-
db_memoria(0)	Display HEX0	PIN_U21	-	-
db_memoria(1)	Display HEX0	PIN_V21	-	-
db_memoria(2)	Display HEX0	PIN_W22	-	-
db_memoria(3)	Display HEX0	PIN_W21	-	-
db_memoria(4)	Display HEX0	PIN_Y22	-	-
db_memoria(5)	Display HEX0	PIN_Y21	-	-
db_memoria(6)	Display HEX0	PIN_AA22	-	-
db_jogada_feita(0)	Display HEX1	PIN_AA20	-	-
db_jogada_feita(1)	Display HEX1	PIN_AB20	-	-
db_jogada_feita(2)	Display HEX1	PIN_AA19	-	-
db_jogada_feita(3)	Display HEX1	PIN_AA18	-	-
db_jogada_feita(4)	Display HEX1	PIN_AB18	-	-
db_jogada_feita(5)	Display HEX1	PIN_AA17	-	-
db_jogada_feita(6)	Display HEX1	PIN_U22	-	-

Tabela 15: Pinagem usada na feira de projetos.

15. Conclusão final

Com a implementação das modificações descritas na seção 14, o projeto apresentou funcionamento adequado em todas as demonstrações da feira. Notou-se que o elevado número de fios conectando a *protoboard* e GPIOs tornou a visualização dos *displays* um pouco mais difícil, bem como da primeira linha de leds. Por outro lado, nesse sentido a decisão de voltar certos sinais de interface ao Analog Discovery foi acertada, na medida em que não há dificuldade em clicar no botão ou fechar a chave desejada, diferentemente do que seria observado caso estivessem na placa FPGA. Desse modo, pode-se concluir que o projeto foi concluído de maneira satisfatória, tendo-se usado os conceitos teóricos e as ferramentas tratadas na disciplina Laboratório Digital I. Como perspectiva a uma melhora no que foi desenvolvido, há a possibilidade de construção de uma maquete e/ou integração com uma interface estilo jogo no computador, com comunicação serial. Estas modificações poderiam dar ao projeto maior caráter de produto, além de resolver por completo o problema gerado pelo grande número de fios.

Apêndice A: Tabelas dos casos de teste

Observação: O cenário 1 encontra-se na versão usada para a experiência da semana 4. O cenário 2 não foi usado essa semana - o cenário 1 passou a ter jogadas erradas -, mas foi atualizado para a semana 3, posto que ainda constava a versão da semana 2, sem jogada dupla. Na semana 3 não houve modificação do cenário 3, então este segue igual da semana 2.

Cenário #1 – Acerto de todas as jogadas				
#	Operação	Sinais de entrada	Resultado esperado	Resultado observado
c.i.	Condições Iniciais		pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 00 db_memoria = 1 db_estado = 0 db_jogada = 0 pontuacao = 00	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 00 db_memoria = 1 db_estado = 0 db_jogada = 0 pontuacao = 00
1	“Resetar” circuito	acionar reset	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 01 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 00	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 01 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 00
2	Ativar modo "10"	seletor_modo = "010"/"110" - ambos testados.	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 00 db_memoria = 1 db_estado = 0 db_jogada = 0 pontuacao = 00	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 00 db_memoria = 1 db_estado = 0 db_jogada = 0 pontuacao = 00
3	Acionar sinal iniciar	acionar iniciar	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 00 db_memoria = 1 db_estado = 0 db_jogada = 0 pontuacao = 00	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 00 db_memoria = 1 db_estado = 0 db_jogada = 0 pontuacao = 00
4	Aguarda 4 jogadas nulas		pronto = 0 leds = 0001 jogada_correta = 0 db_contagem = 00 db_memoria = 1 db_estado = 1 db_jogada = 0 pontuacao = 00	pronto = 0 leds = 0001 jogada_correta = 0 db_contagem = 00 db_memoria = 1 db_estado = 1 db_jogada = 0 pontuacao = 00
5	Aciona jogada 1: dupla	acionar chave(0)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 00 db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 4	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 00 db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 4

6	Aciona jogada 2	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 01 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 8	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 01 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 8
7	Aciona jogada 3	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 02 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 12	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 02 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 12
8	Aciona jogada 4	acionar chave(3)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 03 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 16	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 03 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 16
9	Aciona jogada 5	acionar chave(2) após 400 ciclos	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 04 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 20	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 04 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 20
10	Aciona jogada 6	acionar chave(1) após 750 ciclos	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 05 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 24	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 05 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 24
11	Aciona jogada 7	acionar chave(0) após 400 ciclos	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 04 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 28	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 04 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 28
12	Aciona jogada 8	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 07 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 32	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 07 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 32
13	Aciona jogada 9	acionar chave(2) após 400 ciclos	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 08 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 36	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 08 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 36

14	Aciona jogada 10	acionar chave(3) após 750 ciclos	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 09 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 40	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 09 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 40
15	Aciona jogada 11	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 0A db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 44	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 0A db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 44
16	Aciona jogada 12	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 0B db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 48	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 0B db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 48
17	Aciona jogada 13	acionar chave(0)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 0C db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 52	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 0C db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 52
18	Aciona jogada 14	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 0D db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 56	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 0D db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 56
19	Aciona jogada 15	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 0E db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 60	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 0E db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 60
20	Aciona jogada 16	acionar chave(3)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 0F db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 64	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 0F db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 64
21	Aciona jogada 17	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 10 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 68	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 10 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 68

22	Aciona jogada 18	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 11 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 72	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 11 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 72
23	Aciona jogada 19	acionar chave(0)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 12 db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 76	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 12 db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 76
24	Aciona jogada 20	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 13 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 80	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 13 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 80
25	Aciona jogada 21	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 14 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 84	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 14 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 84
26	Aciona jogada 22	acionar chave(3)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 15 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 88	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 15 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 88
27	Aciona jogada 23	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 16 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 92	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 16 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 92
28	Aciona jogada 24	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 17 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 96	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 17 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 96
29	Aciona jogada 25	acionar chave(0)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 18 db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 100	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 18 db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 100

30	Aciona jogada 26	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 19 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 104	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 19 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 104
31	Aciona jogada 27	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 1A db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 108	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 1A db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 108
32	Aciona jogada 28	acionar chave(3)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 1B db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 112	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 1B db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 112
33	Aciona jogada 29	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 1C db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 116	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 1C db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 116
34	Aciona jogada 30	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 1D db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 120	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 1D db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 120
35	Aciona jogada 31	acionar chave(0)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 1E db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 124	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 1E db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 124
36	Aciona jogada 32	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 1F db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 128	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 1F db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 128
37	Aciona jogada 33	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 20 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 132	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 20 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 132

38	Aciona jogada 34	acionar chave(3)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 21 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 136	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 21 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 136
39	Aciona jogada 35	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 22 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 140	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 22 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 140
40	Aciona jogada 36	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 23 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 144	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 23 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 144
41	Aciona jogada 37	acionar chave(0)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 24 db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 148	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 24 db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 148
42	Aciona jogada 38	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 25 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 152	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 25 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 152
43	Aciona jogada 39	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 26 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 156	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 26 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 156
44	Aciona jogada 40	acionar chave(3)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 27 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 160	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 27 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 160
45	Aciona jogada 41	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 28 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 164	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 28 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 164

46	Aciona jogada 42	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 29 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 168	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 29 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 168
47	Aciona jogada 43	acionar chave(0)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 2A db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 172	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 2A db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 172
48	Aciona jogada 44	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 2B db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 176	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 2B db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 176
49	Aciona jogada 45	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 2C db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 180	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 2C db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 180
50	Aciona jogada 46	acionar chave(3)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 2D db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 184	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 2D db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 184
51	Aciona jogada 47	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 2E db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 188	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 2E db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 188
52	Aciona jogada 48	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 2F db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 192	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 2F db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 192
53	Aciona jogada 49	acionar chave(0)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 30 db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 196	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 30 db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 196

54	Aciona jogada 50	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 31 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 200	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 31 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 200
55	Aciona jogada 51	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 32 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 204	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 32 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 204
56	Aciona jogada 52	acionar chave(3)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 33 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 208	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 33 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 208
57	Aciona jogada 53	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 34 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 212	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 34 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 212
58	Aciona jogada 54	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 35 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 216	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 35 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 216
59	Aciona jogada 55	acionar chave(0)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 36 db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 220	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 36 db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 220
60	Aciona jogada 56	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 37 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 224	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 37 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 224
61	Aciona jogada 57	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 38 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 228	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 38 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 228

62	Aciona jogada 58	acionar chave(3)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 39 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 232	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 39 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 232
63	Aciona jogada 59	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 3A db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 236	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 3A db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 236
64	Aciona jogada 60	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 3B db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 240	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 3B db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 240
65	Aciona jogada 61	acionar chave(0) entre 500 e 750 ciclos de <i>clock</i>	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 3C db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 242	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 3C db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 242
66	Aciona jogada 62	acionar chave(1) entre 750 e 1000 ciclos de <i>clock</i>	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 3D db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 243	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 3D db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 243
67	Aciona jogada 63	acionar chave(3): erro	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 3E db_memoria = 4 db_estado = 8 db_jogada = 8 pontuacao = 243	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 3E db_memoria = 4 db_estado = 8 db_jogada = 8 pontuacao = 243
68	Aciona jogada 64	esperar término do jogo - não acionar chave, esperar timeout	pronto = 1 leds = 0000 jogada_correta = 0 db_contagem = 3E db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 243	pronto = 1 leds = 0000 jogada_correta = 0 db_contagem = 3E db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 243

Cenário #2 – Erro em algumas jogadas				
#	Operação	Sinais de entrada	Resultado esperado	Resultado observado
c.i.	Condições Iniciais		pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 00 db_memoria = 1 db_estado = 0 db_jogada = 0 pontuacao = 00	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 00 db_memoria = 1 db_estado = 0 db_jogada = 0 pontuacao = 00
1	“Resetar” circuito	acionar reset	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 01 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 00	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 01 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 00
2	Ativar modo "10"	seletor_modo = "10"	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 00 db_memoria = 1 db_estado = 0 db_jogada = 0 pontuacao = 00	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 00 db_memoria = 1 db_estado = 0 db_jogada = 0 pontuacao = 00
3	Acionar sinal iniciar	acionar iniciar	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 00 db_memoria = 1 db_estado = 0 db_jogada = 0 pontuacao = 00	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 00 db_memoria = 1 db_estado = 0 db_jogada = 0 pontuacao = 00
4	Aguarda 1 segundo para começo do jogo		pronto = 0 leds = 0001 jogada_correta = 0 db_contagem = 00 db_memoria = 1 db_estado = 1 db_jogada = 0 pontuacao = 00	pronto = 0 leds = 0001 jogada_correta = 0 db_contagem = 00 db_memoria = 1 db_estado = 1 db_jogada = 0 pontuacao = 00
5	Aciona jogada 1 - jogada dupla	acionar chave(0)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 00 db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 01	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 00 db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 01
6	Aciona jogada 2	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 01 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 02	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 01 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 02
7	Aciona jogada 3	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 02 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 03	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 02 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 03

8	Aciona jogada 4	acionar chave(3)	<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 03 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 04 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 03 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 04 </pre>
9	Aciona jogada 5 (erro)	acionar chave(3)	<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 04 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 04 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 04 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 04 </pre>
10	Aciona jogada 6	acionar chave(1)	<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 05 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 05 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 05 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 05 </pre>
11	Aciona jogada 7	acionar chave(0)	<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 04 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 06 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 04 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 06 </pre>
12	Aciona jogada 8	acionar chave(1)	<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 07 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 07 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 07 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 07 </pre>
13	Aciona jogada 9 (erro)		<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 08 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 07 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 08 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 07 </pre>
14	Aciona jogada 10	acionar chave(3)	<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 09 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 08 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 09 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 08 </pre>
15	Aciona jogada 11	acionar chave(2)	<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 0A db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 09 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 0A db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 09 </pre>

16	Aciona jogada 12	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 0B db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 0A	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 0B db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 0A
17	Aciona jogada 13	acionar chave(0)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 0C db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 0B	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 0C db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 0B
18	Aciona jogada 14	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 0D db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 0C	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 0D db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 0C
19	Aciona jogada 15	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 0E db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 0D	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 0E db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 0D
20	Aciona jogada 16	acionar chave(3)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 0F db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 0E	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 0F db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 0E
21	Aciona jogada 17	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 10 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 0F	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 10 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 0F
22	Aciona jogada 18	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 11 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 10	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 11 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 10
23	Aciona jogada 19 (erro)	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 12 db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 10	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 12 db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 10

24	Aciona jogada 20 (erro)	acionar chave(0)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 13 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 11	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 13 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 11
25	Aciona jogada 21	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 14 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 12	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 14 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 12
26	Aciona jogada 22	acionar chave(3)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 15 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 13	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 15 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 13
27	Aciona jogada 23	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 16 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 14	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 16 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 14
28	Aciona jogada 24	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 17 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 15	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 17 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 15
29	Aciona jogada 25	acionar chave(0)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 18 db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 16	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 18 db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 16
30	Aciona jogada 26	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 19 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 17	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 19 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 17
31	Aciona jogada 27	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 1A db_contagem = 1A db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 18	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 1A db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 18

32	Aciona jogada 28	acionar chave(3)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 1B db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 19	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 1B db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 19
33	Aciona jogada 29 (erro)	acionar chave(3)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 1C db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 19	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 1C db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 19
34	Aciona jogada 30	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 1D db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 1A	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 1D db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 1A
35	Aciona jogada 31 (erro)	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 1E db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 1A	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 1E db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 1A
36	Aciona jogada 32	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 1F db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 1B	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 1F db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 1B
37	Aciona jogada 33	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 20 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 1C	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 20 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 1C
38	Aciona jogada 34	acionar chave(3)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 21 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 1D	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 21 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 1D
39	Aciona jogada 35	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 22 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 1E	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 22 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 1E

40	Aciona jogada 36	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 23 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 1F	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 23 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 1F
41	Aciona jogada 37	acionar chave(0)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 24 db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 20	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 24 db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 20
42	Aciona jogada 38	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 25 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 21	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 25 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 21
43	Aciona jogada 39	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 26 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 22	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 26 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 22
44	Aciona jogada 40	acionar chave(3)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 27 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 23	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 27 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 23
45	Aciona jogada 41	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 28 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 24	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 28 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 24
46	Aciona jogada 42	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 29 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 25	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 29 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 25
47	Aciona jogada 43	acionar chave(0)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 2A db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 26	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 2A db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 26

48	Aciona jogada 44	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 2B db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 27	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 2B db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 27
49	Aciona jogada 45	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 2C db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 28	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 2C db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 28
50	Aciona jogada 46	acionar chave(3)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 2D db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 29	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 2D db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 29
51	Aciona jogada 47	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 2E db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 2A	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 2E db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 2A
52	Aciona jogada 48	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 2F db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 2B	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 2F db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 2B
53	Aciona jogada 49	acionar chave(0)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 30 db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 2C	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 30 db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 2C
54	Aciona jogada 50	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 31 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 2D	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 31 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 2D
55	Aciona jogada 51	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 32 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 2E	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 32 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 2E

56	Aciona jogada 52	acionar chave(3)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 33 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 2F	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 33 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 2F
57	Aciona jogada 53	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 34 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 30	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 34 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 30
58	Aciona jogada 54	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 35 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 31	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 35 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 31
59	Aciona jogada 55	acionar chave(0)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 36 db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 37	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 36 db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 37
60	Aciona jogada 56	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 37 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 33	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 37 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 33
61	Aciona jogada 57	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 36 db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 34	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 36 db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 34
62	Aciona jogada 58	acionar chave(3)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 39 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 35	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 39 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 35
63	Aciona jogada 59	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 3A db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 36	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 3A db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 36

64	Aciona jogada 60	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 3B db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 37	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 3B db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 37
65	Aciona jogada 61	acionar chave(0)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 3C db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 38	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 3C db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 38
66	Aciona jogada 62	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 3D db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 39	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 3D db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 39
67	Aciona jogada 63	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 3E db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 3A	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 3E db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 3A

Cenário #3 – Modo de Escrita				
#	Operação	Sinais de entrada	Resultado esperado	Resultado observado
c.i.	Condições Iniciais		pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 00 db_memoria = 1 db_estado = 0 db_jogada = 0 pontuacao = 00	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 00 db_memoria = 1 db_estado = 0 db_jogada = 0 pontuacao = 00
1	“Resetar” circuito	acionar reset	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 01 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 00	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 01 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 00
2	Ativar modo "01"	seletor_modo = "01"	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 00 db_memoria = 1 db_estado = 0 db_jogada = 0 pontuacao = 00	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 00 db_memoria = 1 db_estado = 0 db_jogada = 0 pontuacao = 00

3	Acionar sinal iniciar	acionar iniciar	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 00 db_memoria = 1 db_estado = 0 db_jogada = 0 pontuacao = 00	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 00 db_memoria = 1 db_estado = 0 db_jogada = 0 pontuacao = 00
4	Aguarda 1 segundo para começo do jogo		pronto = 0 leds = 0001 jogada_correta = 0 db_contagem = 00 db_memoria = 1 db_estado = 1 db_jogada = 0 pontuacao = 00	pronto = 0 leds = 0001 jogada_correta = 0 db_contagem = 00 db_memoria = 1 db_estado = 1 db_jogada = 0 pontuacao = 00
5	Grava jogada 1	acionar chave(0)	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 00 db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 00	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 00 db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 00
6	Grava jogada 2	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 01 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 00	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 01 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 00
7	Grava jogada 3	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 02 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 00	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 02 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 00
8	Grava jogada 4	acionar chave(3)	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 03 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 00	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 03 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 00
9	Grava jogada 5	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 04 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 00	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 04 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 00
10	Grava jogada 6	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 05 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 00	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 05 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 00

11	Grava jogada 7	acionar chave(0)	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 04 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 00	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 04 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 00
12	Grava jogada 8	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 07 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 00	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 07 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 00
13	Grava jogada 9	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 08 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 00	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 08 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 00
14	Grava jogada 10	acionar chave(3)	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 09 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 00	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 09 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 00
15	Grava jogada 11	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 0A db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 00	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 0A db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 00
16	Grava jogada 12	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 0B db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 00	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 0B db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 00
17	Grava jogada 13	acionar chave(0)	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 0C db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 00	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 0C db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 00
18	Grava jogada 14	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 0D db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 00	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 0D db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 00

19	Grava jogada 15	acionar chave(2)	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 0E db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 00 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 0E db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 00 </pre>
20	Grava jogada 16	acionar chave(3)	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 0F db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 00 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 0F db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 00 </pre>
21	Grava jogada 17	acionar chave(2)	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 10 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 00 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 10 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 00 </pre>
22	Grava jogada 18	acionar chave(1)	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 11 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 00 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 11 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 00 </pre>
23	Grava jogada 19	acionar chave(0)	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 12 db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 00 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 12 db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 00 </pre>
24	Grava jogada 20	acionar chave(1)	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 13 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 00 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 13 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 00 </pre>
25	Grava jogada 21	acionar chave(2)	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 14 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 00 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 14 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 00 </pre>
26	Grava jogada 22	acionar chave(3)	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 15 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 00 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 15 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 00 </pre>

27	Grava jogada 23	acionar chave(2)	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 16 db_memoria = 4 db_estado = 4 db_jogada = 4 pontuacao = 00 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 16 db_memoria = 4 db_estado = 4 db_jogada = 4 pontuacao = 00 </pre>
28	Grava jogada 24	acionar chave(1)	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 17 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 00 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 17 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 00 </pre>
29	Grava jogada 25	acionar chave(0)	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 18 db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 00 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 18 db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 00 </pre>
30	Grava jogada 26	acionar chave(1)	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 19 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 00 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 19 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 00 </pre>
31	Grava jogada 27	acionar chave(2)	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 1A db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 00 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 1A db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 00 </pre>
32	Grava jogada 28	acionar chave(3)	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 1B db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 00 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 1B db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 00 </pre>
33	Grava jogada 29	acionar chave(2)	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 1C db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 00 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 1C db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 00 </pre>
34	Grava jogada 30	acionar chave(1)	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 1D db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 00 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 1D db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 00 </pre>

35	Grava jogada 31	acionar chave(0)	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 1E db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 00	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 1E db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 00
36	Grava jogada 32	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 1F db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 00	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 1F db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 00
37	Grava jogada 33	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 20 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 00	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 20 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 00
38	Grava jogada 34	acionar chave(3)	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 21 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 00	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 21 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 00
39	Grava jogada 35	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 22 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 00	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 22 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 00
40	Grava jogada 36	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 23 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 00	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 23 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 00
41	Grava jogada 37	acionar chave(0)	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 24 db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 00	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 24 db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 00
42	Grava jogada 38	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 25 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 00	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 25 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 00

43	Grava jogada 39	acionar chave(2)	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 26 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 00 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 26 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 00 </pre>
44	Grava jogada 40	acionar chave(3)	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 27 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 00 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 27 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 00 </pre>
45	Grava jogada 41	acionar chave(2)	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 28 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 00 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 28 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 00 </pre>
46	Grava jogada 42	acionar chave(1)	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 29 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 00 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 29 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 00 </pre>
47	Grava jogada 43	acionar chave(0)	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 2A db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 00 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 2A db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 00 </pre>
48	Grava jogada 44	acionar chave(1)	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 2B db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 00 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 2B db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 00 </pre>
49	Grava jogada 45	acionar chave(2)	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 2C db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 00 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 2C db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 00 </pre>
50	Grava jogada 46	acionar chave(3)	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 2D db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 00 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 2D db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 00 </pre>

51	Grava jogada 47	acionar chave(2)	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 2E db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 00 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 2E db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 00 </pre>
52	Grava jogada 48	acionar chave(1)	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 2F db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 00 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 2F db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 00 </pre>
53	Grava jogada 49	acionar chave(0)	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 30 db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 00 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 30 db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 00 </pre>
54	Grava jogada 50	acionar chave(1)	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 31 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 00 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 31 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 00 </pre>
55	Grava jogada 51	acionar chave(2)	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 32 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 00 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 32 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 00 </pre>
56	Grava jogada 52	acionar chave(3)	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 33 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 00 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 33 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 00 </pre>
57	Grava jogada 53	acionar chave(2)	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 34 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 00 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 34 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 00 </pre>
58	Grava jogada 54	acionar chave(1)	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 35 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 00 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 35 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 00 </pre>

59	Grava jogada 55	acionar chave(0)	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 36 db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 00 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 36 db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 00 </pre>
60	Grava jogada 56	acionar chave(1)	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 37 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 00 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 37 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 00 </pre>
61	Grava jogada 57	acionar chave(2)	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 38 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 00 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 38 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 00 </pre>
62	Grava jogada 58	acionar chave(3)	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 39 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 00 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 39 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 00 </pre>
63	Grava jogada 59	acionar chave(2)	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 3A db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 00 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 3A db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 00 </pre>
64	Grava jogada 60	acionar chave(1)	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 3B db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 00 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 3B db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 00 </pre>
65	Grava jogada 61	acionar chave(0)	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 3C db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 00 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 3C db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 00 </pre>
66	Grava jogada 62	acionar chave(1)	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 3D db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 00 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 3D db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 00 </pre>

67	Grava jogada 63	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 3E db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 00	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 3E db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 00
68	Ativar modo "00"	seletor_modo = "00"	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 3E db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 00	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 3E db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 00
69	Acionar sinal iniciar	acionar iniciar	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 00 db_memoria = 1 db_estado = 0 db_jogada = 0 pontuacao = 00	pronto = 0 leds = 0000 jogada_correta = 0 db_contagem = 00 db_memoria = 1 db_estado = 0 db_jogada = 0 pontuacao = 00
70	Aguarda 1 segundo para começo do jogo		pronto = 0 leds = 0001 jogada_correta = 0 db_contagem = 00 db_memoria = 1 db_estado = 1 db_jogada = 0 pontuacao = 00	pronto = 0 leds = 0001 jogada_correta = 0 db_contagem = 00 db_memoria = 1 db_estado = 1 db_jogada = 0 pontuacao = 00
71	Grava jogada 1	acionar chave(0)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 00 db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 01	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 00 db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 01
72	Grava jogada 2	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 01 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 02	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 01 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 02
73	Grava jogada 3	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 02 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 03	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 02 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 03
74	Grava jogada 4	acionar chave(3)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 03 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 04	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 03 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 04

75	Grava jogada 5	acionar chave(2)	<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 04 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 05 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 04 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 05 </pre>
76	Grava jogada 6	acionar chave(1)	<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 05 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 06 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 05 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 06 </pre>
77	Grava jogada 7	acionar chave(0)	<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 04 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 07 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 04 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 07 </pre>
78	Grava jogada 8	acionar chave(1)	<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 07 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 08 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 07 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 08 </pre>
79	Grava jogada 9	acionar chave(2)	<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 08 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 09 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 08 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 09 </pre>
80	Grava jogada 10	acionar chave(3)	<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 09 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 0A </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 09 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 0A </pre>
81	Grava jogada 11	acionar chave(2)	<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 0A db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 0B </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 0A db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 0B </pre>
82	Grava jogada 12	acionar chave(1)	<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 0B db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 0C </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 0B db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 0C </pre>

83	Grava jogada 13	acionar chave(0)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 0C db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 0D	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 0C db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 0D
84	Grava jogada 14	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 0D db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 0E	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 0D db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 0E
85	Grava jogada 15	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 0E db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 0F	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 0E db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 0F
86	Grava jogada 16	acionar chave(3)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 0F db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 10	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 0F db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 10
87	Grava jogada 17	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 10 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 11	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 10 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 11
88	Grava jogada 18	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 11 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 12	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 11 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 12
89	Grava jogada 19	acionar chave(0)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 12 db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 13	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 12 db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 13
90	Grava jogada 20	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 13 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 14	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 13 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 14

91	Grava jogada 21	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 14 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 15	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 14 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 15
92	Grava jogada 22	acionar chave(3)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 15 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 16	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 15 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 16
93	Grava jogada 23	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 16 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 17	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 16 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 17
94	Grava jogada 24	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 17 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 18	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 17 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 18
95	Grava jogada 25	acionar chave(0)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 18 db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 19	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 18 db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 19
96	Grava jogada 26	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 19 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 1A	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 19 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 1A
97	Grava jogada 27	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 1A db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 1B	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 1A db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 1B
98	Grava jogada 28	acionar chave(3)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 1B db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 1C	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 1B db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 1C

99	Grava jogada 29	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 1C db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 1D	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 1C db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 1D
100	Grava jogada 30	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 1D db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 1E	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 1D db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 1E
101	Grava jogada 31	acionar chave(0)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 1E db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 1F	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 1E db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 1F
102	Grava jogada 32	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 1F db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 20	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 1F db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 20
103	Grava jogada 33	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 20 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 21	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 20 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 21
104	Grava jogada 34	acionar chave(3)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 21 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 22	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 21 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 22
105	Grava jogada 35	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 22 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 23	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 22 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 23
106	Grava jogada 36	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 23 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 24	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 23 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 24

107	Grava jogada 37	acionar chave(0)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 24 db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 25	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 24 db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 25
108	Grava jogada 38	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 25 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 26	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 25 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 26
109	Grava jogada 39	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 26 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 27	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 26 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 27
110	Grava jogada 40	acionar chave(3)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 27 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 28	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 27 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 28
111	Grava jogada 41	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 28 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 29	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 28 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 29
112	Grava jogada 42	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 29 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 2A	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 29 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 2A
113	Grava jogada 43	acionar chave(0)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 2A db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 2B	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 2A db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 2B
114	Grava jogada 44	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 2B db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 2C	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 2B db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 2C

115	Grava jogada 45	acionar chave(2)	<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 2C db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 2D </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 2C db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 2D </pre>
116	Grava jogada 46	acionar chave(3)	<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 2D db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 2E </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 2D db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 2E </pre>
117	Grava jogada 47	acionar chave(2)	<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 2E db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 2F </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 2E db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 2F </pre>
118	Grava jogada 48	acionar chave(1)	<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 2F db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 30 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 2F db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 30 </pre>
119	Grava jogada 49	acionar chave(0)	<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 30 db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 31 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 30 db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 31 </pre>
120	Grava jogada 50	acionar chave(1)	<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 31 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 32 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 31 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 32 </pre>
121	Grava jogada 51	acionar chave(2)	<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 32 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 33 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 32 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 33 </pre>
122	Grava jogada 52	acionar chave(3)	<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 33 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 34 </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 33 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 34 </pre>

123	Grava jogada 53	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 34 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 35	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 34 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 35
124	Grava jogada 54	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 35 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 36	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 35 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 36
125	Grava jogada 55	acionar chave(0)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 36 db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 37	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 36 db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 37
126	Grava jogada 56	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 37 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 38	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 37 db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 38
127	Grava jogada 57	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 38 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 39	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 38 db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 39
128	Grava jogada 58	acionar chave(3)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 39 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 3A	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 39 db_memoria = 8 db_estado = 8 db_jogada = 8 pontuacao = 3A
129	Grava jogada 59	acionar chave(2)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 3A db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 3B	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 3A db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 3B
130	Grava jogada 60	acionar chave(1)	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 3B db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 3C	pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 3B db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 3C

131	Grava jogada 61	acionar chave(0)	<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 3C db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 3D </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 3C db_memoria = 1 db_estado = 8 db_jogada = 1 pontuacao = 3D </pre>
132	Grava jogada 62	acionar chave(1)	<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 3D db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 3E </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 3D db_memoria = 2 db_estado = 8 db_jogada = 2 pontuacao = 3E </pre>
133	Grava jogada 63	acionar chave(2)	<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 3E db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 3F </pre>	<pre> pronto = 0 leds = 0000 jogada_correta = 1 db_contagem = 3E db_memoria = 4 db_estado = 8 db_jogada = 4 pontuacao = 3F </pre>