



ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO  
Departamento de Engenharia de Computação e Sistemas Digitais

## PCS3635 – LABORATÓRIO DIGITAL I

### EXPERIÊNCIA 4 – Desenvolvimento de Projeto de Circuitos Digitais em FPGA

Planejamento da Bancada B1 – Turma 3 – Prof. Antonio

Data de Emissão: 26 de Janeiro de 2025.

<b>Nome:</b> Ana Vitória Abreu Murad	<b>Número USP:</b> 14613160
<b>Nome:</b> Heitor Gama Ribeiro	<b>Número USP:</b> 14577494
<b>Nome:</b> Yasmin Francisquetti Barnes	<b>Número USP:</b> 13828230

#### 1 INTRODUÇÃO

A presente experiência tem como objetivo familiarizar os alunos com aspectos de circuitos digitais, utilizando um sinal periódico como entrada de *clock* e a entrada de dados através de elementos externos. Esse experimento possui dependência no anterior e conta com algumas atualizações e mudanças no que já foi implementado. A experiência em si inclui a modificação do circuito para incluir uma nova lógica da máquina de estados e componentes adicionais no fluxo de dados (FD), adicionando um estado de espera pelo acionamento das entradas de chaves. No final, é necessário fazer a adaptação do módulo que une os elementos (o fluxo de dados, a unidade de controle e os quatro displays hexadecimais) para incluir novos sinais de condição e de depuração. Os módulos serão projetados em Verilog, testados no software *ModelSim* e em seguida compilados no *Intel Quartus Prime*, onde também será feita a pinagem da placa e, depois, o circuito será

sintetizado na placa FPGA DE0-CV. Na parte lógica, o que muda nessa versão do circuito é a sinalização dos dados das chaves nos LEDs de saída e a espera pelos dados inseridos como um estado novo da máquina, além de estados separados no fim para o sucesso ou a falha da jogada.

## 2 DESCRIÇÃO DO PROJETO

O projeto a ser implementado consiste em adaptar um sistema digital simples, que compara dados de uma memória ROM com os dados de entrada (chaves), para um circuito que simula um jogo de memória semelhante ao *Genius*. Nesta etapa, será desenvolvido um sistema capaz de detectar o acionamento das chaves pelo operador (simulado através de botões no software *WaveForms*), permitindo que a comparação ocorra apenas após cada novo acionamento. Um sinal de clock de 1 kHz será utilizado, tornando impossível detectar visualmente os sinais de *acertou* ou *errou* durante os testes; esses sinais permanecerão ativos até o reinício do jogo, permitindo sua observação após a execução. Novos componentes serão adicionados à UC e ao FD, juntamente com sinais adicionais que integrarão esses componentes ao circuito principal.

O circuito do sistema digital armazena 16 dados de 4 bits em uma memória interna ROM, percorrendo seus endereços com um contador interno. Após o *reset*, o circuito aguarda o acionamento do sinal *iniciar* e, em seguida, a ativação de uma chave de entrada, indicada pela saída *db\_tem\_jogada*. O dado inserido é armazenado, exibido nos LEDs e na saída *db\_jogada*. O circuito compara o dado inserido com o correspondente da memória, indicando o resultado em *db\_igual*, e avança o contador para o próximo dado. As saídas *db\_contagem*, *db\_memoria* e *db\_estado* exibem, respectivamente, o endereço, o dado atual da memória e o estado da Unidade de Controle em displays de sete segmentos. O ciclo continua enquanto o jogador acerta e todos os 16 dados são verificados; em caso de sucesso, ativa-se o sinal *acertou*, e em caso de erro, o ciclo é interrompido e o sinal *errou* é ativado. Ao final, o sinal *pronto* indica o término, permanecendo ativo até o reinício do circuito pelo sinal *iniciar*, assim como os sinais *acertou* e *errou*.

## 3 DETALHAMENTO DO PROJETO LÓGICO

O objetivo principal desta seção é apresentar em detalhes o projeto lógico do fluxo de dados (FD), da unidade de controle (UC) e do sistema digital como um todo.

### 3.1 PROJETO DO FLUXO DE DADOS

O fluxo de dados projetado na experiência anterior era composto por quatro componentes internos integrados: o contador (“contador\_163”), a memória (“sync\_rom\_16x4”), o registrador (“registrador\_4”) e o comparador (“comparador\_85”). A memória, pré-programada com 16 valores de 4 bits, era responsável por armazenar dados que seriam comparados às chaves inseridas no circuito. O contador determinava o endereço de leitura (*s\_endereço*), o registrador armazenava os valores inseridos pelas chaves, e o comparador identificava se o valor inserido correspondia ao armazenado na memória na posição indicada pelo contador, gerando o sinal de igualdade.

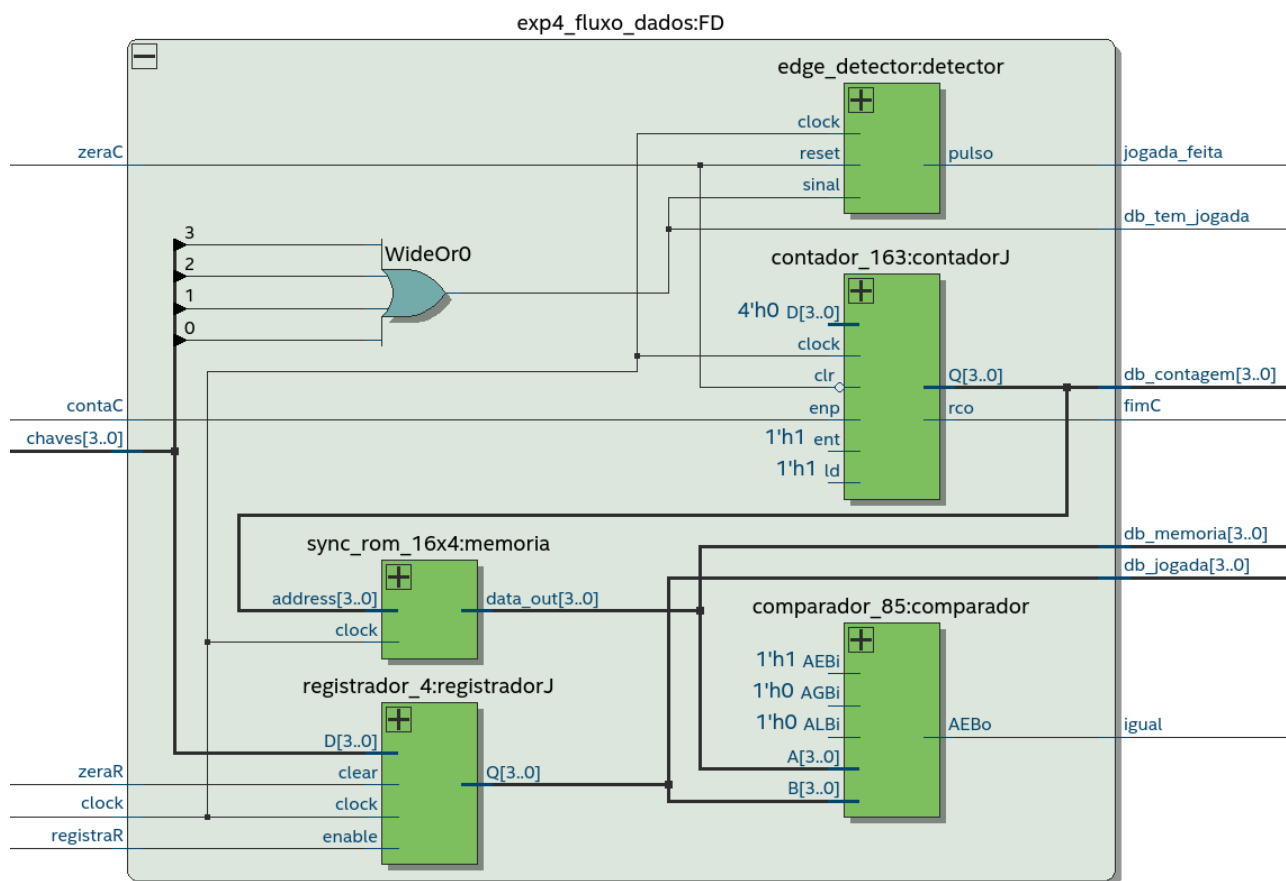
Na nova implementação, o FD inclui cinco componentes, com a adição do módulo “edge\_detector”. Esse módulo possui três entradas e uma saída, descritas a seguir:

- *clock*: Recebe o sinal de clock do sistema digital.
- *reset*: Conectado ao sinal *zeraC*, escolhido pelos alunos para garantir o funcionamento contínuo do detector de borda ao longo do circuito sem necessidade de reinicialização durante um ciclo. Após o término de uma rodada, o detector é resetado para preparar o próximo ciclo de jogadas.
- *senal*: Recebe o resultado da operação lógica OR entre todos os bits das chaves de entrada ( $\text{assign } \text{senal} = | \text{chaves}$ ), também chamada de *WideOr*. O sinal é ativado (1) sempre que algum bit das chaves estiver ativo, como nas combinações válidas de entrada (0001, 0010, 0100 e 1000). O sinal é desativado (0) apenas quando a entrada for 0000. O operador deve desligar uma chave antes de ativar outra.
- *pulso*: A saída do módulo gera um pulso a partir da borda de subida do sinal, indicando à unidade de controle (UC) que um novo valor de chave foi inserido e que uma nova jogada deve ser processada. O funcionamento do módulo *edge\_detector* baseia-se em dois registradores:
  - *reg0*: Atualizado com o valor de sinal quando ele é ativado (1).
  - *reg1*: Sincronizado com o valor de *reg0* na próxima subida do clock.

A operação ‘ $\text{assign } \text{pulso} = \text{reg0} \& \sim \text{reg1}$ ’ garante que o pulso seja ativado apenas durante o intervalo entre duas subidas de clock, permitindo detectar a borda de subida do sinal uma única vez. Isso evita que o sinal permaneça ativo por um período prolongado, como ocorre com *senal*, que continua em nível alto até que o operador altere o valor das chaves.

Utilizando a ferramenta RTLViewer, foi elaborado o diagrama de blocos representando o fluxo de dados projetado e implementado em Verilog, conforme ilustrado na Figura 1:

**Figura 1 - RTLViewer do Fluxo de dados do Circuito**



### 3.2 PROJETO DA UNIDADE DE CONTROLE

Para implementar o projeto lógico descrito, foram adicionados três novos estados: “espera”, “fim\_acerto” e “fim\_erro”, sendo que os dois últimos substituem o antigo estado “fim”.

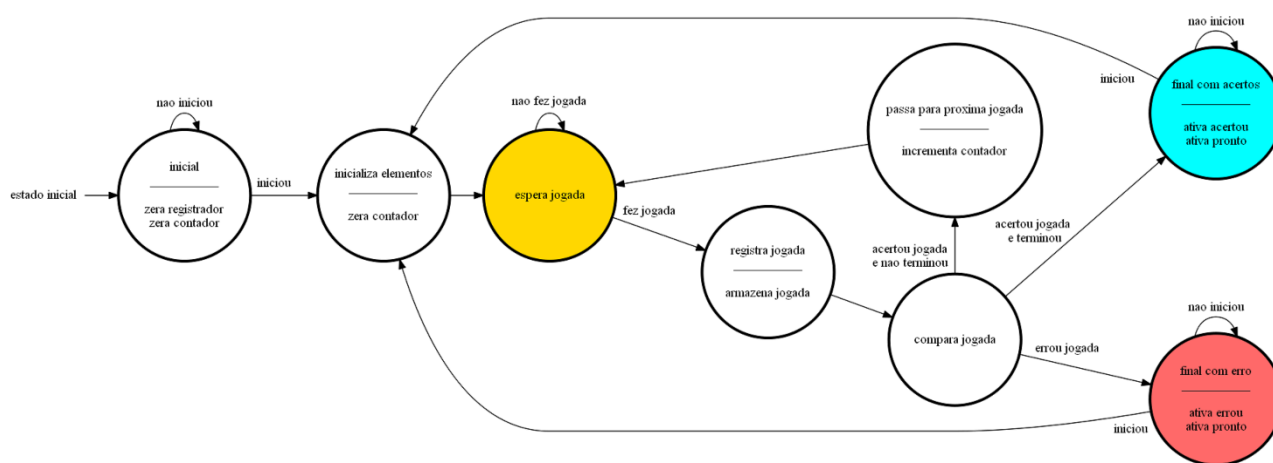
- **espera:** Nesse estado, o sistema aguarda a jogada do jogador, monitorando o sinal *jogada*. Caso uma jogada seja detectada, ocorre a transição para o estado “registra”. Caso contrário, o sistema permanece no estado “espera”.
- **fim\_acerto:** Esse estado é alcançado quando o jogador acerta todas as combinações armazenadas na memória. Nesse caso, os sinais *pronto* e *acertou* são acionados, indicando que o jogo terminou com vitória. O estado permanece ativo até que o sinal *iniciar* seja acionado novamente, o que reinicia o sistema e retorna ao estado de inicialização. Esse intervalo permite que os LEDs de

encerramento fiquem visíveis por tempo suficiente para serem apreciados visualmente.

- fim\_erro: Semelhante ao estado “fim\_acerto”, porém é ativado quando o jogador comete um erro. Nesse caso, os sinais *pronto* e *errou* são acionados.

A Figura 2 ilustra a máquina de estados completa, destacando os novos estados com cores diferenciadas, enquanto os estados do experimento anterior permanecem em branco. A Tabela 1 detalha as funções de cada estado, suas condições de transição e justificativas.

**Figura 2 - Máquina de Estados de Alto Nível**



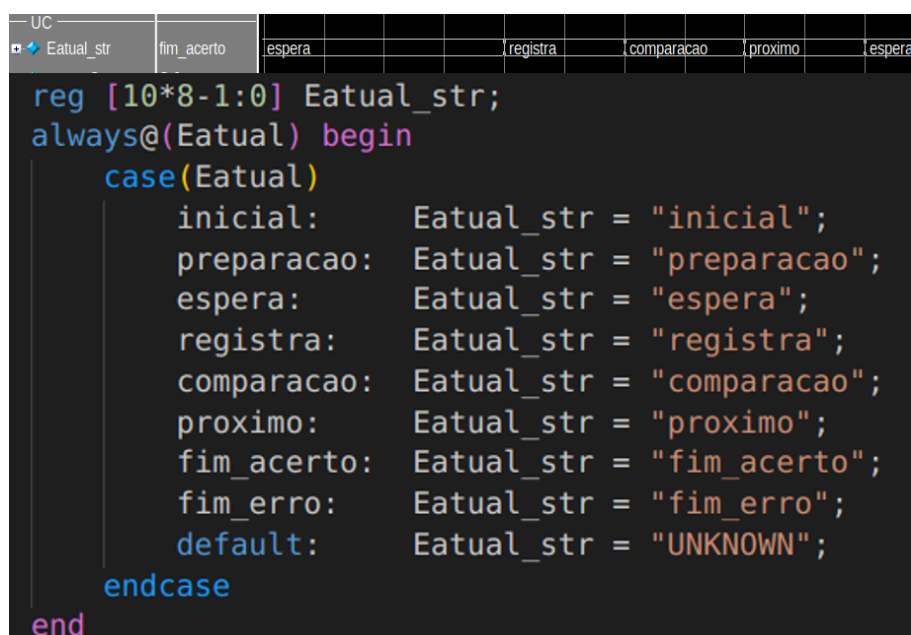
**Tabela 1 – Descrição da Unidade de Controle do Sistema**

Nome do Estado	Descrição do Estado	Próximo Estado	Condições e Justificativas para a Transição entre Estados
inicial	Estado inicial. Zera o contador (zeraC) e o registrador (zeraR).	preparacao	Transita para preparacao quando o sinal iniciar está ativo. Mantém-se em inicial se iniciar não estiver ativo.
preparacao	Prepara o sistema para aguardar uma jogada. Zera o contador (zeraC).	espera	Transita para espera automaticamente após executar as ações do estado preparacao.
espera	Aguardando uma jogada do operador.	registra	Transita para registra quando o sinal jogada está ativo. Mantém-se em espera enquanto jogada não estiver ativo.
registra	Registra o valor das chaves no registrador (registraR).	comparacao	Transita para comparacao automaticamente após executar as ações do estado registra.
comparacao	Compara o valor das chaves com o esperado na memória.	fim_acerto ou proximo ou fim_erro	- Vai para fim_acerto se igual e fim estiverem ativos. - Vai para proximo se apenas igual estiver ativo. - Vai para fim_erro se igual estiver inativo.

proximo	Avança o contador (contaC) para processar a próxima chave.	espera	Transita para espera automaticamente após executar as ações do estado proximo.
fim_acerto	Indica que todas as comparações foram bem-sucedidas. Sinaliza pronto e acertou.	preparacao	Transita para preparacao se o sinal iniciar estiver ativo. Mantém-se em fim_acerto enquanto iniciar não estiver ativo.
fim_erro	Indica que houve um erro em alguma comparação. Sinaliza pronto e errou.	preparacao	Transita para preparacao se o sinal iniciar estiver ativo. Mantém-se em fim_erro enquanto iniciar não estiver ativo.

Para facilitar a visualização e depuração na simulação, foi adicionada ao código da unidade de controle uma variável que converte o número binário do estado atual em uma string, permitindo exibir diretamente o nome do estado no *ModelSim*. Basta monitorar a variável modificando seu radical para formato ASCII. A Figura 3 apresenta a visualização aprimorada e o código que possibilita essa funcionalidade.

**Figura 3 - Conversão de Número de Estado para Nome**



```

reg [10*8-1:0] Eatual_str;
always@(Eatual) begin
  case(Eatual)
    inicial:    Eatual_str = "inicial";
    preparacao: Eatual_str = "preparacao";
    espera:     Eatual_str = "espera";
    registra:   Eatual_str = "registra";
    comparacao: Eatual_str = "comparacao";
    proximo:    Eatual_str = "proximo";
    fim_acerto: Eatual_str = "fim_acerto";
    fim_erro:   Eatual_str = "fim_erro";
    default:    Eatual_str = "UNKNOWN";
  endcase
end

```

### 3.3 PROJETO DO SISTEMA DIGITAL

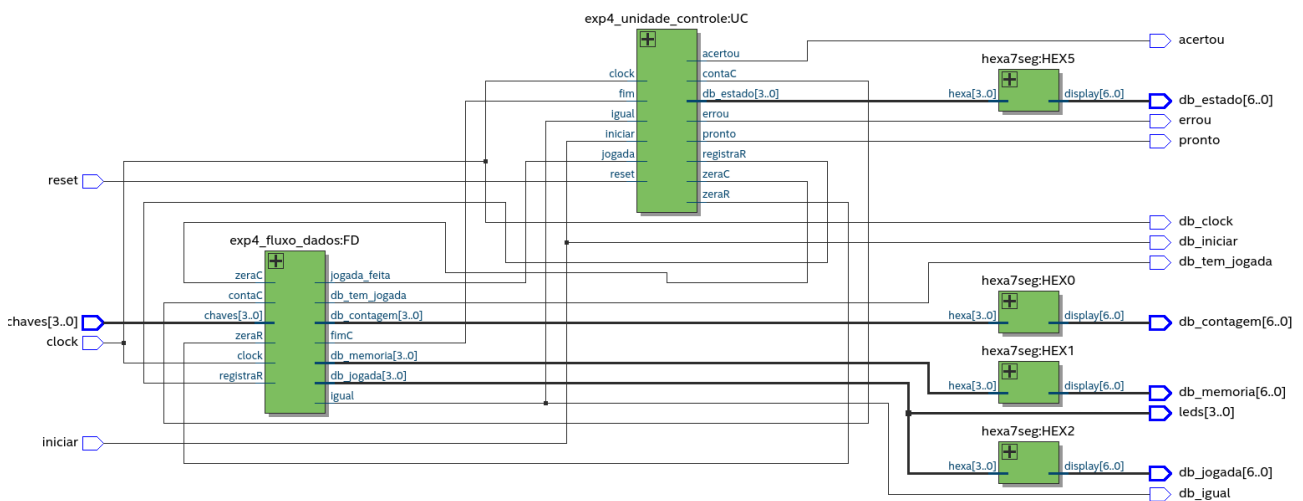
A integração entre o fluxo de dados (FD) e a unidade de controle (UC) é realizada por meio de sinais que sincronizam as operações e determinam o fluxo das etapas do sistema. A UC controla os estados do circuito, gerenciando os sinais de controle, que ativam ou desativam componentes específicos no FD, como o contador, o registrador e o

comparador. Por outro lado, o FD processa as entradas e gera os sinais de condição necessários para a UC, como o fim do ciclo de contagem (*s\_fimC*), a igualdade de dados comparados (*s\_igual*) e a ocorrência de uma jogada (*s\_jogada\_feita*). Essa interação permite que o sistema realize operações sequenciais de maneira ordenada e eficiente.

Para facilitar a depuração, foram projetados sinais adicionais exibidos em displays de sete segmentos. Por exemplo, o *db\_estado* indica o estado atual da UC, enquanto o *db\_memoria* e o *db\_jogada* mostram os valores em uso na memória e nas chaves, respectivamente. Além disso, o *db\_igual* sinaliza a comparação entre o valor inserido e o esperado, e o *db\_contagem* exibe o endereço atual da memória. Esses sinais, junto com o *db\_tem\_jogada*, que detecta a realização de uma jogada, permitem um monitoramento preciso durante a simulação e a implementação prática do circuito.

A Figura 4 apresenta o diagrama de blocos da integração entre o FD e a UC. Nela, observa-se a conexão direta entre os sinais gerados e consumidos pelos dois módulos, destacando as interfaces de controle, processamento e depuração. Em resumo, essa estrutura assegura o funcionamento correto do sistema e possibilita identificar rapidamente possíveis falhas durante os testes.

**Figura 4 - RTLViewer do Circuito Completo**



#### 4 PLANO DE TESTES DO SISTEMA E SIMULAÇÕES

Para validar integralmente os novos estados do circuito, foram elaborados dois casos de teste: um para verificar o estado “fim\_acerto” e outro para avaliar o estado “fim\_erro”. Ambos os casos também incluem a validação do estado “espera”.

#### 4.1 CENÁRIO DE TESTE 1 – SIMULAÇÃO DE ACERTO DO CIRCUITO COMPLETO

Validação do resultado de acerto dos 16 dados da memória, utilizando a testbench “circuito\_exp4\_tb1.v” e o software *ModelSim* para simulação.

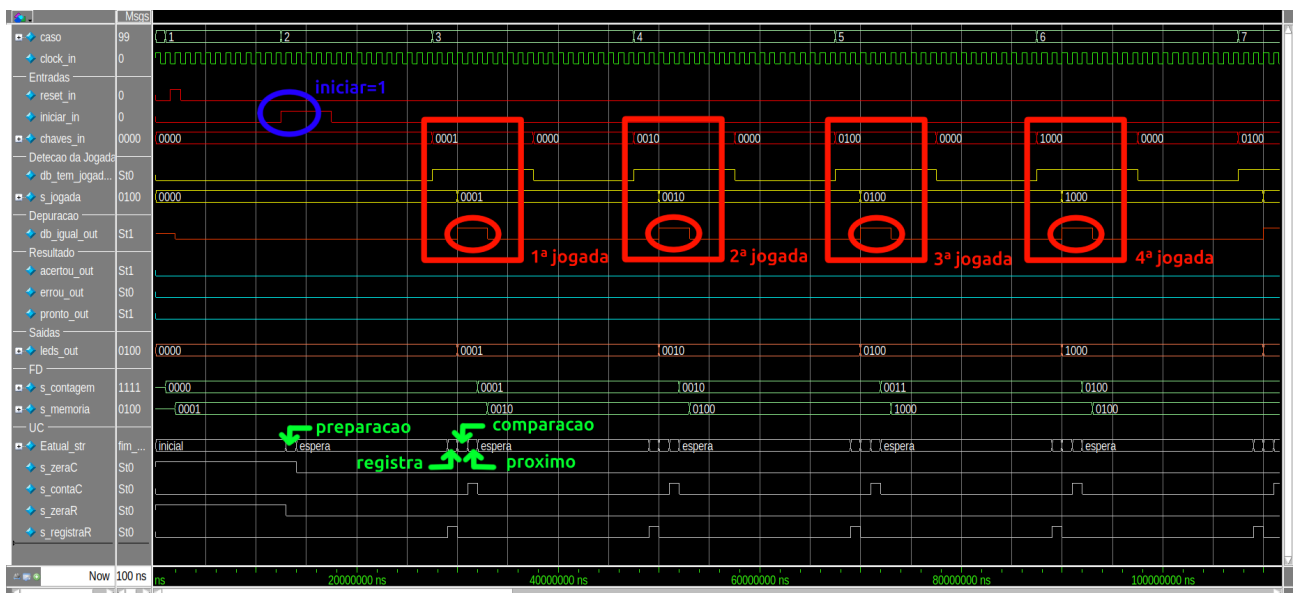
**Tabela 2 – Descrição e Resultados Simulados do Cenário de Teste 1**

#	Operação	Sinais de Entrada	Resultado Esperado	Resultado simulado ok?
c.i.	Condições iniciais	-	db_jogada=0	Sim
1	“Resetar” circuito e aguardar alguns segundos	acionar reset	db_contagem=0, db_memoria=1, db_estado=iniciar(0), db_jogada=1, db_tem_jogada_out=0	Sim
2	Acionar sinal iniciar	acionar iniciar	db_contagem=0, db_memoria=1, db_estado=preparacao(3), db_jogada=1, db_tem_jogada_out=0	Sim
3	Acionar primeira entrada (jogada 1)	acionar chave (0)	db_contagem=0, db_memoria=1, db_estado= ordem 4, 5, 6 e 3, db_jogada=1, db_tem_jogada_out=1, db_igual=1	Sim
4	Acionar segunda entrada (jogada 2)	acionar chave (1)	db_contagem=1, db_memoria=2, db_estado= ordem 4, 5, 6 e 3, db_jogada=2, db_igual=1	Sim
5	Acionar terceira entrada (jogada 3)	acionar chave (2)	db_contagem=2, db_memoria=4, db_estado= ordem 4, 5, 6 e 3, db_jogada=4, db_igual=1	Sim
6	Acionar quarta entrada (jogada 4)	acionar chave (3)	db_contagem=3, db_memoria=8, db_estado= ordem 4, 5, 6 e 3, db_jogada=8, db_igual=1	Sim
7	Acionar quinta entrada (jogada 5)	acionar chave (2)	db_contagem=4, db_memoria=4, db_estado= ordem 4, 5, 6 e 3, db_jogada=4, db_igual=1	Sim
8	Acionar sexta entrada (jogada 6)	acionar chave (1)	db_contagem=5, db_memoria=2, db_estado= ordem 4, 5, 6 e 3, db_jogada=2, db_igual=1	Sim
9	Acionar sétima entrada (jogada 7)	acionar chave (0)	db_contagem=6, db_memoria=1, db_estado= ordem 4, 5, 6 e 3, db_jogada=1, db_igual=1	Sim
10	Acionar oitava entrada (jogada 8)	acionar chave (0)	db_contagem=7, db_memoria=1, db_estado= ordem 4, 5, 6 e 3, db_jogada=1, db_igual=1	Sim
11	Acionar nona entrada (jogada 9)	acionar chave (1)	db_contagem=8, db_memoria=2, db_estado= ordem 4, 5, 6 e 3, db_jogada=2, db_igual=1	Sim

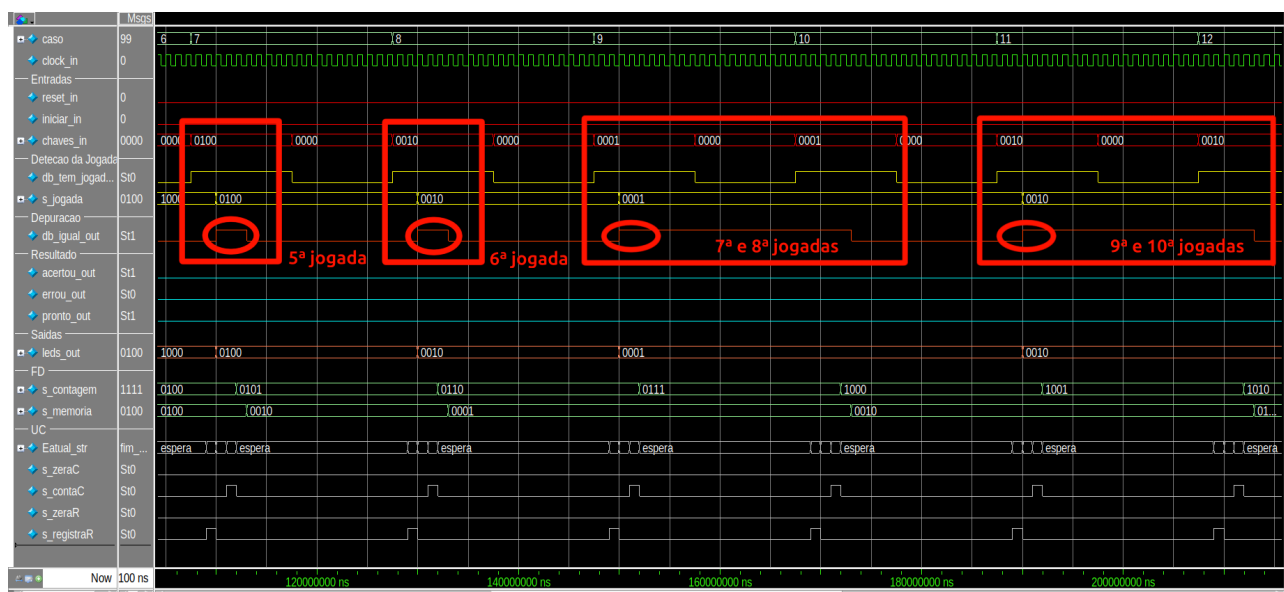


12	Acionar décima entrada (jogada 10)	acionar chave (1)	db_contagem=9, db_memoria=2, db_estado= ordem 4, 5, 6 e 3, db_jogada=2, db_igual=1	Sim
13	Acionar décima primeira entrada (jogada 11)	acionar chave (2)	db_contagem=10, db_memoria=4, db_estado= ordem 4, 5, 6 e 3, db_jogada=4, db_igual=1	Sim
14	Acionar décima segunda entrada (jogada 12)	acionar chave (2)	db_contagem=11, db_memoria=4, db_estado= ordem 4, 5, 6 e 3, db_jogada=4, db_igual=1	Sim
15	Acionar décima terceira entrada (jogada 13)	acionar chave (3)	db_contagem=12, db_memoria=8, db_estado= ordem 4, 5, 6 e 3, db_jogada=8, db_igual=1	Sim
16	Acionar décima quarta entrada (jogada 14)	acionar chave (3)	db_contagem=13, db_memoria=8, db_estado= ordem 4, 5, 6 e 3, db_jogada=8, db_igual=1	Sim
17	Acionar décima quinta entrada (jogada 15)	acionar chave (0)	db_contagem=14, db_memoria=1, db_estado= ordem 4, 5, 6 e 3, db_jogada=1, db_igual=1	Sim
18	Acionar décima sexta entrada (jogada 16)	acionar chave (2)	db_contagem=15, db_memoria=4, db_estado= ordem 4, 5 e A, db_jogada=4, db_igual=1, pronto=1, acertou=1	Sim

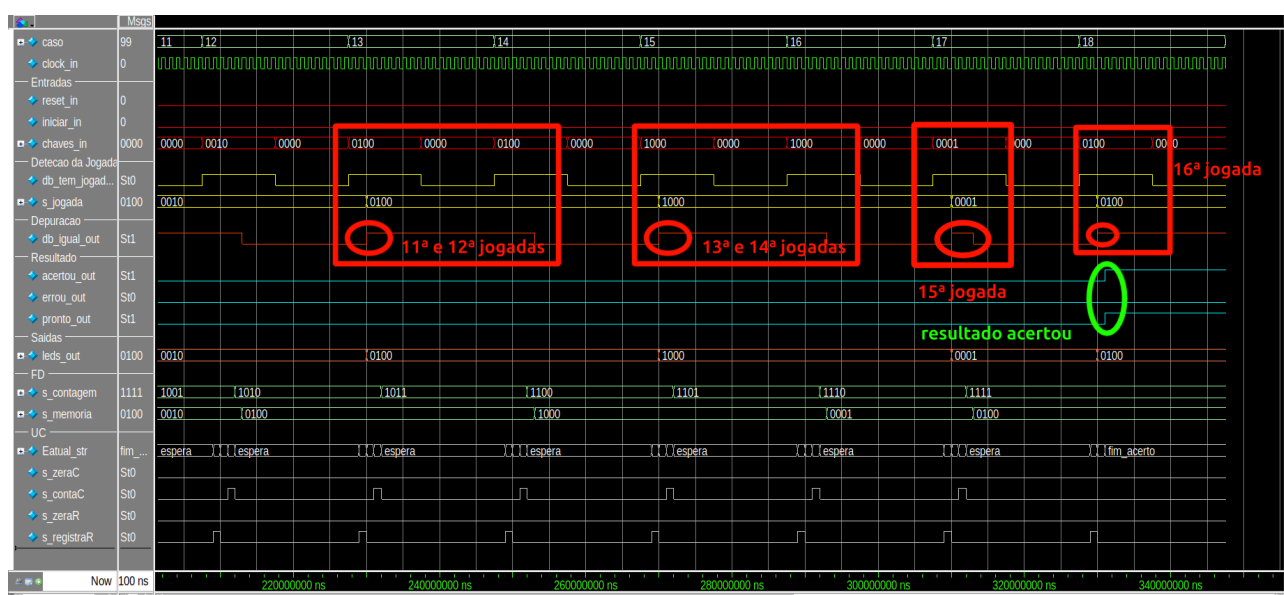
- Condição Inicial e Testes 1 a 6



### • Testes 7 a 11



### • Testes 12 a 18



Nota-se que o pulso do sinal “db\_igual\_out” é mais longo a partir da 7ª jogada e volta a ser curto na 15ª jogada. Isso acontece porque, nas jogadas de 7 a 14, a memória a ser jogada se repete em pares. Em toda jogada repetida consecutiva, o registrador já tem a resposta correta da jogada anterior. Como o registrador só tem o seu *enable* ativado durante o estado “registra”, então mesmo que as chaves transitem momentaneamente para 0000, o sinal de igual continua ativado até que a memória mude.

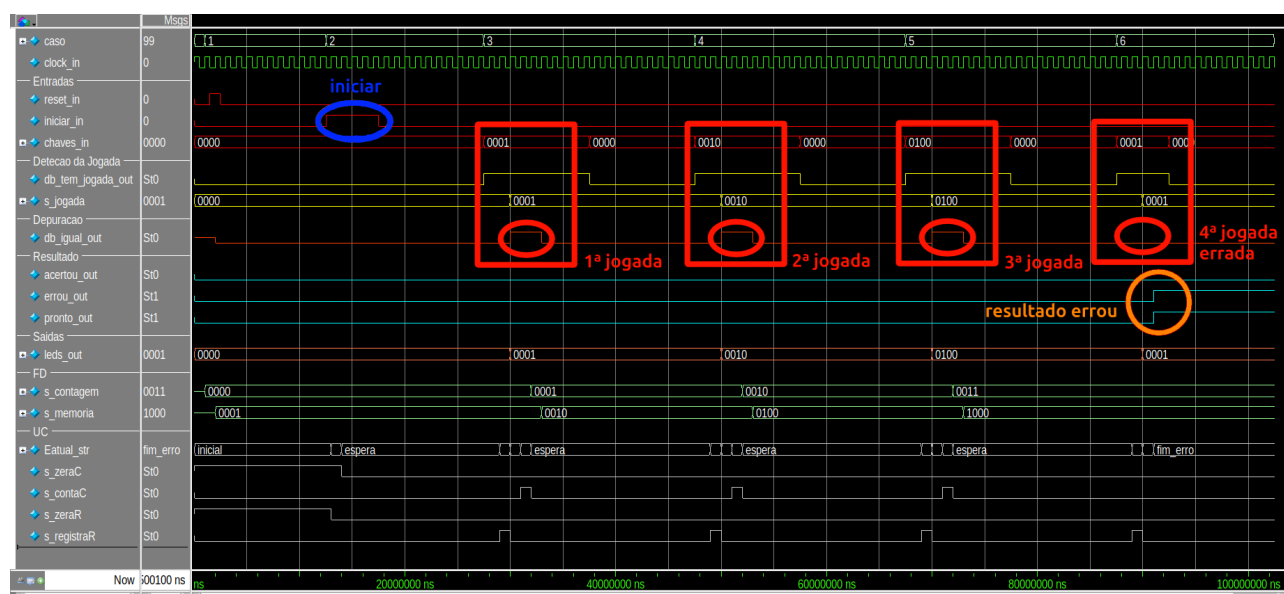
## 4.2 CENÁRIO DE TESTE 2 – SIMULAÇÃO DE ERRO DO CIRCUITO COMPLETO

Validação do cenário de erro do 4º dado, após o acerto dos três primeiros, utilizando a testbench “circuito\_exp4\_tb2.v” e o software *ModelSim* para simulação.

Tabela 3 – Descrição e Resultados Simulados do Cenário de Teste 2

#	Operação	Sinais de Controle	Resultado Esperado	Resultado simulado ok?
c.i.	Condições iniciais	-	db_jogada=0	Sim
1	“Resetar” circuito e aguardar alguns segundos	acionar reset	db_contagem=0, db_memoria=1, db_estado=iniciar(0), db_jogada=1, db_tem_jogada_out=0	Sim
2	Acionar sinal iniciar	acionar iniciar	db_contagem=0, db_memoria=1, db_estado=preparacao(3), db_jogada=1, db_tem_jogada_out=0	Sim
3	Acionar primeira entrada (jogada 1)	acionar chave (0)	db_contagem=0, db_memoria=1, db_estado= ordem 4, 5, 6 e 3, db_jogada=1, db_tem_jogada_out=1, db_igual=1	Sim
4	Acionar segunda entrada (jogada 2)	acionar chave (1)	db_contagem=1, db_memoria=2, db_estado= ordem 4, 5, 6 e 3, db_jogada=2, db_igual=1	Sim
5	Acionar terceira entrada (jogada 3)	acionar chave (2)	db_contagem=2, db_memoria=4, db_estado= ordem 4, 5, 6 e 3, db_jogada=4, db_igual=1	Sim
6	Acionar quarta entrada errada (jogada 4)	acionar chave (0)	db_contagem=3, db_memoria=8, db_estado= ordem 4, 5 e E, db_jogada=1, db_igual=0, pronto=1, errou=1	Sim

- Condição Inicial e Testes



## 5 IMPLANTAÇÃO DO PROJETO

O projeto foi compilado e projetado e as imagens geradas pela ferramenta RTLViewer foram incluídas nas Figuras 1 e 4 deste relatório.

### 5.1 PINAGEM DA PLACA FPGA

O planejamento da pinagem é uma etapa obrigatória no desenvolvimento de projetos com o FPGA Cyclone V, realizado por meio do "Pin Planner" do *Intel Quartus*. As configurações detalham a alocação dos pinos em relação aos nós do circuito. Embora a escolha de LEDs, botões e displays para cada sinal tenha sido previamente definida, houve uma exceção para o sinal *db\_iniciar*, que não foi especificado no enunciado. Foi decidido realizar a pinagem deste sinal durante a aula, pois todos os LEDs já foram alocados.

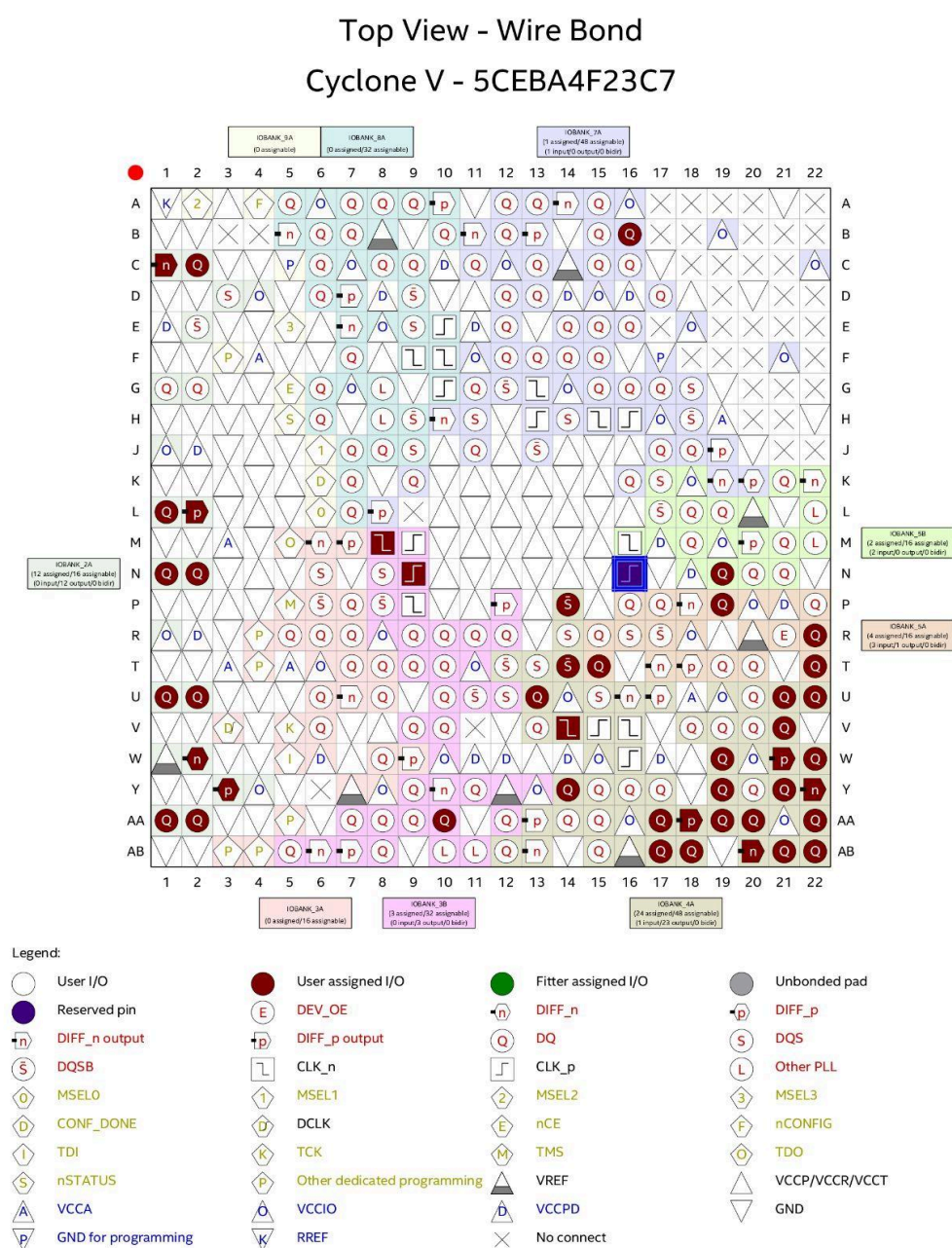
**Tabela 4 - Pinagem da placa FPGA**

Sinal	Pino na Placa DE0-CV	Pino no FPGA	Analog Discovery
ACERTOU	Led LEDR5	PIN_N1	-
CHAVES(0)	GPIO_0_D11	PIN_R22	StaticIO – Button 0/1 – DIO2
CHAVES(1)	GPIO_0_D13	PIN_T22	StaticIO – Button 0/1 – DIO3
CHAVES(2)	GPIO_0_D15	PIN_N19	StaticIO – Button 0/1 – DIO4
CHAVES(3)	GPIO_0_D17	PIN_P19	StaticIO – Button 0/1 – DIO5
CLOCK	GPIO_0_D0	PIN_N16	StaticIO – LED – DIO0 e Patterns – Clock – 1 kHz
DB_CLOCK	Led LEDR8	PIN_L2	
DB_CONTAGEM(0)	Display HEX00	PIN_U21	-
DB_CONTAGEM(1)	Display HEX01	PIN_V21	-
DB_CONTAGEM(2)	Display HEX02	PIN_W22	-
DB_CONTAGEM(3)	Display HEX03	PIN_W21	-
DB_CONTAGEM(4)	Display HEX04	PIN_Y22	-
DB_CONTAGEM(5)	Display HEX05	PIN_Y21	-
DB_CONTAGEM(6)	Display HEX06	PIN_AA22	-
DB_ESTADO(0)	Display HEX50	PIN_N9	-
DB_ESTADO(1)	Display HEX51	PIN_M8	-
DB_ESTADO(2)	Display HEX52	PIN_T14	-
DB_ESTADO(3)	Display HEX53	PIN_P14	-
DB_ESTADO(4)	Display HEX54	PIN_C1	-

DB_ESTADO(5)	Display HEX55	PIN_C2	-
DB_ESTADO(6)	Display HEX56	PIN_W19	-
DB_IGUAL	Led LEDR7	PIN_U1	-
DB_INICIAR			-
DB_JOGADA(0)	Display HEX20	PIN_Y19	-
DB_JOGADA(1)	Display HEX21	PIN_AB17	-
DB_JOGADA(2)	Display HEX22	PIN_AA10	-
DB_JOGADA(3)	Display HEX23	PIN_Y14	-
DB_JOGADA(4)	Display HEX24	PIN_V14	-
DB_JOGADA(5)	Display HEX25	PIN_AB22	-
DB_JOGADA(6)	Display HEX26	PIN_AB21	-
DB_MEMORIA(0)	Display HEX10	PIN_AA20	-
DB_MEMORIA(1)	Display HEX11	PIN_AB20	-
DB_MEMORIA(2)	Display HEX12	PIN_AA19	-
DB_MEMORIA(3)	Display HEX13	PIN_AA18	-
DB_MEMORIA(4)	Display HEX14	PIN_AB18	-
DB_MEMORIA(5)	Display HEX15	PIN_AA17	-
DB_MEMORIA(6)	Display HEX16	PIN_U22	-
DB_TEM_JOGADA	Led LEDR9	PIN_L1	-
ERROU	Led LEDR6	PIN_U2	-
INICIAR	SW0	PIN_U13	-
LEDS(0)	Led LEDR0	PIN_AA2	-
LEDS(1)	Led LEDR1	PIN_AA1	-
LEDS(2)	Led LEDR2	PIN_W2	-
LEDS(3)	Led LEDR3	PIN_Y3	-
PRONTO	Led LEDR4	PIN_N2	-
RESET	GPIO_0_D1	PIN_B16	StaticIO – Button 0/1 – DIO1

Na Figura 04, está a *Top View* da placa FPGA, que permite a visão superior da pinagem após a definição da localização de cada sinal.

Figura 5 - Top view da placa FPGA



## 5.2 ESTRATÉGIA DE MONTAGEM

Os alunos iniciarão a montagem experimental configurando previamente o “Pin Planner” do *Intel Quartus Prime*, associando as entradas e saídas do circuito aos pinos correspondentes na FPGA DE0-CV. Chegando na bancada, verificarão se a placa está corretamente conectada à fonte de alimentação e ao computador via cabo USB.

Após essa etapa, o projeto será importado no *Intel Quartus Prime* por meio do arquivo .qar previamente gerado. O grupo compilará o projeto novamente para validar possíveis ajustes e iniciará a programação da FPGA. Depois de gravar o circuito na placa,

será feita uma verificação inicial das conexões físicas e do funcionamento dos sinais de saída por meio dos LEDs e displays integrados à placa.

Para monitoramento, serão observados os seguintes sinais:

- Displays HEX: “db\_contagem” (HEX0), “db\_memoria” (HEX1), “db\_estado” (HEX2), e “db\_jogadafeita” (HEX5).
- LEDs: Sinais binários como “db\_tem\_jogada”, “pronto”, “acertou” e “errou” serão associados aos LEDs da placa.
- Sinal de Clock: Será verificado o sincronismo do sinal “db\_clock” como referência para o funcionamento correto do circuito.

Durante os testes, o grupo também utilizará o software *ModelSim* para análise das formas de onda e confirmação do comportamento esperado do circuito antes da execução física na FPGA.

### 5.3 ESTRATÉGIA DE DEPURAÇÃO

A depuração será realizada em etapas progressivas para identificar e corrigir falhas no projeto. O objetivo principal é assegurar que o sistema funcione conforme esperado, garantindo a coerência entre os sinais monitorados e o comportamento lógico do circuito. As etapas incluem:

#### 1. Monitoramento dos sinais principais de depuração:

- “db\_contagem” e “db\_memoria”: Verificar se o endereço do contador e o dado da memória estão sendo corretamente atualizados.
- “db\_igual”: Confirmar se o resultado da comparação entre as chaves e os dados da memória é consistente.
- “db\_estado”: Garantir que a máquina de estados transite adequadamente entre os estados definidos no projeto.
- “pronto”, “acertou” e “errou”: Analisar os LEDs correspondentes para validar os resultados finais do circuito.

#### 2. Utilização do Analog Discovery:

- Conectar o *Analog Discovery* ao circuito para capturar e observar os sinais internos, como bordas de clock e transições de estado.
- Empregar o *WaveForms* para verificar a frequência e estabilidade do sinal de clock, além de registrar o comportamento dinâmico dos sinais de entrada e saída.

### 3. Correção de erros:

- Durante o desenvolvimento do projeto lógico, foram identificadas algumas falhas não críticas que comprometeram sinais de depuração, como o sinal `db_jogada`, que não estava devidamente conectado. Embora essas falhas não afetassem o comportamento interno do circuito, elas dificultaram a análise durante os testes. Apesar de os sinais planejados terem sido cuidadosamente revisados durante o Planejamento, é importante verificar cada LED e display, garantindo que não existam conexões soltas ou erros na montagem.
- Se durante o funcionamento do circuito os resultados esperados (como os sinais “acertou” ou “errou”) não forem apresentados ao final dos planos de teste, a primeira estratégia recomendada é ajustar o sinal de clock para operação manual. Isso permitirá que os casos de teste sejam executados em uma velocidade perceptível ao olhar humano, facilitando a observação detalhada dos sinais de depuração e a identificação da origem do erro.
- Se o problema identificado não for decorrente do funcionamento interno do circuito, o que é provável, considerando que o design já foi previamente simulado, a próxima etapa será verificar as configurações no Pin Planner, garantindo que os pinos estão devidamente mapeados, as conexões físicas, incluindo possíveis mau contatos nos cabos ligados ao Analog Discovery, a atribuição e estabilidade do sinal de clock periódico implementado.

### 4. Validação após modificações:

- Após cada ajuste, repetir os testes sistemáticos para assegurar que o erro foi corrigido e que nenhuma funcionalidade já confirmada foi comprometida.

## 5.4 EXECUÇÃO PRÁTICA DO CENÁRIO DE TESTE 1 – TESTE DE ACERTO NA FPGA

Validação do resultado de acerto dos 16 dados da memória, utilizando o clock de 1KHz no circuito sintetizado na placa FPGA.

**Tabela 4 – Descrição e Resultados Práticos do Cenário de Teste 1**

#	Operação	Sinais de Entrada	Resultado Esperado	Resultado prático ok?
c.i.	Condições iniciais	-	<code>db_jogada=0</code>	
1	“Resetar” circuito e aguardar alguns segundos	acionar reset	<code>db_contagem=0</code> , <code>db_memoria=1</code> , <code>db_estado=iniciar(0)</code> , <code>db_jogada=1</code> , <code>db_tem_jogada_out=0</code>	



2	Acionar sinal iniciar	acionar iniciar	db_contagem=0, db_memoria=1, db_estado=preparacao(3), db_jogada=1, db_tem_jogada_out=0	
3	Acionar primeira entrada (jogada 1)	acionar chave (0)	db_contagem=0, db_memoria=1, db_estado= ordem 4, 5, 6 e 3, db_jogada=1, db_tem_jogada_out=1, db_igual=1	
4	Acionar segunda entrada (jogada 2)	acionar chave (1)	db_contagem=1, db_memoria=2, db_estado= ordem 4, 5, 6 e 3, db_jogada=2, db_igual=1	
5	Acionar terceira entrada (jogada 3)	acionar chave (2)	db_contagem=2, db_memoria=4, db_estado= ordem 4, 5, 6 e 3, db_jogada=4, db_igual=1	
6	Acionar quarta entrada (jogada 4)	acionar chave (3)	db_contagem=3, db_memoria=8, db_estado= ordem 4, 5, 6 e 3, db_jogada=8, db_igual=1	
7	Acionar quinta entrada (jogada 5)	acionar chave (2)	db_contagem=4, db_memoria=4, db_estado= ordem 4, 5, 6 e 3, db_jogada=4, db_igual=1	
8	Acionar sexta entrada (jogada 6)	acionar chave (1)	db_contagem=5, db_memoria=2, db_estado= ordem 4, 5, 6 e 3, db_jogada=2, db_igual=1	
9	Acionar sétima entrada (jogada 7)	acionar chave (0)	db_contagem=6, db_memoria=1, db_estado= ordem 4, 5, 6 e 3, db_jogada=1, db_igual=1	
10	Acionar oitava entrada (jogada 8)	acionar chave (0)	db_contagem=7, db_memoria=1, db_estado= ordem 4, 5, 6 e 3, db_jogada=1, db_igual=1	
11	Acionar nona entrada (jogada 9)	acionar chave (1)	db_contagem=8, db_memoria=2, db_estado= ordem 4, 5, 6 e 3, db_jogada=2, db_igual=1	
12	Acionar décima entrada (jogada 10)	acionar chave (1)	db_contagem=9, db_memoria=2, db_estado= ordem 4, 5, 6 e 3, db_jogada=2, db_igual=1	
13	Acionar décima primeira entrada (jogada 11)	acionar chave (2)	db_contagem=10, db_memoria=4, db_estado= ordem 4, 5, 6 e 3, db_jogada=4, db_igual=1	
14	Acionar décima segunda entrada (jogada 12)	acionar chave (2)	db_contagem=11, db_memoria=4, db_estado= ordem 4, 5, 6 e 3, db_jogada=4, db_igual=1	
15	Acionar décima terceira entrada (jogada 13)	acionar chave (3)	db_contagem=12, db_memoria=8, db_estado= ordem 4, 5, 6 e 3, db_jogada=8, db_igual=1	

16	Acionar décima quarta entrada (jogada 14)	acionar chave (3)	db_contagem=13, db_memoria=8, db_estado= ordem 4, 5, 6 e 3, db_jogada=8, db_igual=1	
17	Acionar décima quinta entrada (jogada 15)	acionar chave (0)	db_contagem=14, db_memoria=1, db_estado= ordem 4, 5, 6 e 3, db_jogada=1, db_igual=1	
18	Acionar décima sexta entrada (jogada 16)	acionar chave (2)	db_contagem=15, db_memoria=4, db_estado= ordem 4, 5 e A, db_jogada=4, db_igual=1, pronto=1, acertou=1	

### 5.5 EXECUÇÃO PRÁTICA DO CENÁRIO DE TESTE 2 - TESTE DE ERRO NA FPGA

Validação do cenário de erro do 4º dado, após o acerto dos três primeiros, utilizando o clock de 1KHz no circuito sintetizado na placa FPGA.

**Tabela 5 – Descrição e Resultados Práticos do Cenário de Teste 2**

#	Operação	Sinais de Controle	Resultado Esperado	Resultado prático ok?
c.i.	Condições iniciais	-	db_jogada=0	
1	“Resetar” circuito e aguardar alguns segundos	acionar reset	db_contagem=0, db_memoria=1, db_estado=iniciar(0), db_jogada=1, db_tem_jogada_out=0	
2	Acionar sinal iniciar	acionar iniciar	db_contagem=0, db_memoria=1, db_estado=preparacao(3), db_jogada=1, db_tem_jogada_out=0	
3	Acionar primeira entrada (jogada 1)	acionar chave (0)	db_contagem=0, db_memoria=1, db_estado= ordem 4, 5, 6 e 3, db_jogada=1, db_tem_jogada_out=1, db_igual=1	
4	Acionar segunda entrada (jogada 2)	acionar chave (1)	db_contagem=1, db_memoria=2, db_estado= ordem 4, 5, 6 e 3, db_jogada=2, db_igual=1	
5	Acionar terceira entrada (jogada 3)	acionar chave (2)	db_contagem=2, db_memoria=4, db_estado= ordem 4, 5, 6 e 3, db_jogada=4, db_igual=1	
6	Acionar quarta entrada errada (jogada 4)	acionar chave (0)	db_contagem=3, db_memoria=8, db_estado= ordem 4, 5 e E, db_jogada=1, db_igual=0, pronto=1, errou=1	

## **6 PROJETO DO DESAFIO DA EXPERIÊNCIA**

O enunciado para o projeto do Desafio da Experiência 4 será disponibilizado durante a aula do Laboratório Digital. Nessas condições, as próximas seções serão preenchidas somente no Relato Final.

### **6.1 DESCRIÇÃO DO DESAFIO**

### **6.2 DESCRIÇÃO DO PROJETO LÓGICO**

#### **6.2.1 Alterações do Fluxo de Dados**

#### **6.2.2 Alterações da Unidade de Controle**

#### **6.2.3 Alterações do Sistema Digital**

### **6.3 VERIFICAÇÃO E VALIDAÇÃO DO DESAFIO**

#### **6.3.1 Cenário de Teste 1**

#### **6.3.2 Cenário de Teste 2**

## **7 CONCLUSÕES**

Até o momento do planejamento, todos os testes realizados cumpriram os objetivos definidos para a experiência. Foram executados testes que simularam diferentes cenários de funcionamento do circuito, abrangendo diversos casos possíveis para garantir que o sistema opere conforme o esperado.

Os testes permitiram verificar a integração entre o fluxo de dados e a unidade de controle, bem como a funcionalidade das saídas de depuração, como *db\_contagem*, *db\_memoria*, *db\_estado* e *db\_jogada*. Os resultados obtidos validaram o comportamento do sistema em situações representativas, incluindo as transições entre estados e os sinais de finalização, como *pronto*, *acertou* e *errou*.

A abordagem adotada assegurou a identificação e correção de possíveis inconsistências, deixando o grupo preparado para a etapa de implementação prática durante a aula.