



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

PCS3635 – LABORATÓRIO DIGITAL I

EXPERIÊNCIA 3 – PROJETO DE UMA UNIDADE DE CONTROLE

Relato da Bancada B1 – Turma 3 – Prof. Antonio

Data de Emissão: 20 de Janeiro de 2025.

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

1 INTRODUÇÃO

A presente experiência tem como objetivo familiarizar os alunos com o componente de unidade de controle em um circuito digital, integrar o fluxo de dados desenvolvido na experiência anterior, com algumas modificações, e, por fim, combinar os componentes em um sistema digital completo. A experiência em si inclui a projeção hierárquica dos elementos do circuito, a codificação do fluxo de dados e do sistema digital que une a unidade de controle (UC) e o fluxo de dados (FD) em Verilog, a simulação do projeto no software *ModelSim* e, ao final, a síntese do projeto em uma placa FPGA DE0-CV. Até o final da experiência, os alunos terão compreendido o funcionamento da unidade de controle por meio do estudo do diagrama de estados, de um pseudocódigo e do código em Verilog fornecido, executado os testes indicados após a codificação do fluxo de dados e do circuito principal, simulado o projeto no software *Intel Quartus Prime* e o testado integralmente na FPGA.

2 DESCRIÇÃO DO PROJETO

O projeto tem como objetivo principal comparar os números inseridos pelas chaves com os valores armazenados em uma memória ROM previamente projetada, contendo 16 dados de 4 bits. Para isso, o fluxo de dados do circuito inclui não apenas os elementos já utilizados, como o contador_163 e o comparador_85, mas também novos componentes, como uma ROM e um registrador de 4 bits, responsável por armazenar temporariamente o valor das chaves.

Além disso, o projeto utiliza uma unidade de controle, previamente fornecida aos alunos, que implementa a descrição lógica do circuito digital. Essa lógica é baseada na divisão de tarefas entre diferentes estados da máquina, enviando sinais de controle ao fluxo de dados para completar a tarefa principal. A sequência de funcionamento inclui a inicialização do circuito, a transição para um estado de preparação, o registro do valor das chaves, a comparação dos valores lidos da ROM e a repetição do processo com a próxima posição de memória e um novo valor inserido pelas chaves, até que todas as comparações sejam concluídas.

Embora o funcionamento lógico dos componentes do circuito seja descrito inicialmente, a primeira tarefa dos alunos consiste em integrar os componentes indicados e adicionar displays de sete segmentos para monitorar sinais de depuração. Isso permite que os alunos compreendam o papel dos sinais de controle e condição no funcionamento do circuito antes de explorar detalhadamente cada componente. Após a codificação de todos os elementos, os alunos devem realizar os testes indicados, analisar os resultados e projetar alguns valores para completar as tabelas do Plano de Teste.

Na etapa final, os alunos devem simular o circuito no software *Intel Quartus Prime*, realizar a configuração do Pin Planner para a placa FPGA, e, no dia reservado para o laboratório, testar o circuito diretamente na placa DE0-CV.

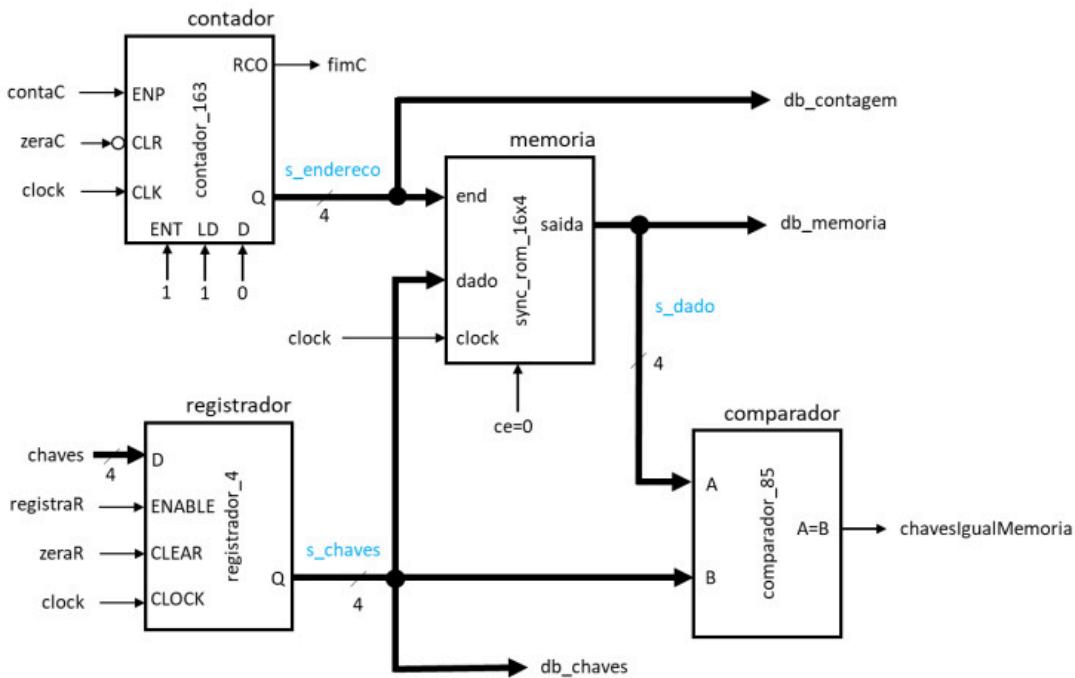
3 DETALHAMENTO DO PROJETO LÓGICO

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

A figura a seguir apresenta o diagrama de blocos do fluxo de dados implementado, contendo os módulos contador_163, comparador_85, sync_rom_16x4 e registrador_4:

Figura 1 – Diagrama de Blocos da Estrutura Interna do Fluxo de Dados



- Contador: O módulo `contador_163` gera os endereços para acessar os dados armazenados na memória ROM. Ele é ativado pelo sinal de controle “`contaC`” e pode ser zerado pelo sinal “`zeraC`”. O valor atual do contador é disponibilizado na saída “`db_contagem`”, que também é usada como endereço de entrada na memória (“`s_endereco`”). Além disso, o sinal “`fimC`” é ativado quando o contador alcança seu valor máximo, indicando o término do ciclo de contagem.
- Memória ROM: A memória ROM (`sync_rom_16x4`) é uma memória síncrona pré-programada com 16 posições com dados de 4 bits (16 endereços). O endereço fornecido pelo contador é utilizado para acessar um dado, que é enviado para o sinal “`db_memoria`” no próximo clock para leitura e disponibilizado ao comparador.
- Registrador: O registrador `_4` é responsável por armazenar temporariamente o valor inserido pelas chaves (“`chaves`”). Esse valor é enviado ao sinal de depuração “`db_chaves`” e também disponibilizado para o comparador (“`s_chaves`”). O registrador pode ser ativado para escrita pelo sinal “`registrarR`” e resetado por “`zeraR`”.
- Comparador: O comparador `_85` compara os valores fornecidos pela memória ROM e pelo registrador (vindos pelos sinais “`s_dado`” e “`s_chaves`”, respectivamente). Se os valores forem iguais, o sinal de saída “`chavesIgualMemoria`” é ativado. Esse componente desempenha um papel crucial no sistema, pois verifica se o valor inserido pelas chaves corresponde a algum dado armazenado na memória.

O fluxo de dados funciona de maneira sincronizada, utilizando o clock do sistema para garantir a operação em diferentes estados definidos pela unidade de controle. A sequência de operação inclui:

1. Inicialização do circuito;
2. Registro do valor das chaves no registrador;
3. Comparaçao entre o valor registrado e o dado atual da memória;
4. Iteração para o próximo endereço da memória até que todas as comparações sejam realizadas.

Com essa estrutura, o circuito digital consegue realizar a tarefa proposta de forma eficiente, com os sinais de depuração exibindo informações relevantes nos displays de sete segmentos durante os testes no laboratório.

3.2 PROJETO DA UNIDADE DE CONTROLE

A unidade de controle do sistema digital sequencial gerencia o processo de comparação dos dados armazenados na memória interna com as chaves de entrada, controlando a operação de leitura e comparação desses dados. A unidade de controle é responsável por percorrer todos os 16 dados armazenados na ROM, realizar a comparação e indicar o resultado da operação, enquanto controla o contador e os endereços da memória.

A máquina de estados tem como objetivo gerenciar a sequência de operações, desde o início da operação até a conclusão da verificação de todos os dados armazenados. Os estados da unidade de controle são representados na Figura 2 pelo diagrama de transição de estados e descritos na Tabela 1, que detalha o funcionamento da UC.

Figura 2 – Diagrama de Transição de Estados da Unidade de Controle

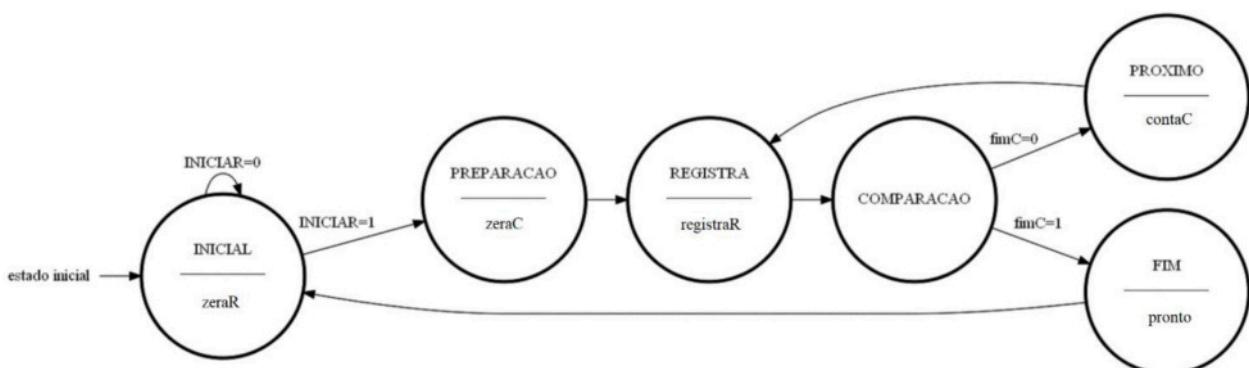


Tabela 1 – Descrição da Unidade de Controle do Sistema

Nome do Estado	Descrição do Estado	Próximo Estado	Condições e Justificativas para a Transição entre Estados
inicial (0000)	Estado inicial. Zera os contadores ("zeraC") e registradores ("zeraR"). Sistema aguarda.	preparacao (0001)	Transição ocorre quando o sinal "iniciar" é ativado. Justificativa: Garantir que o sistema comece em um estado limpo e pronto para execução.
preparacao (0001)	Prepara o sistema, incluindo zerar registradores.	registra (0100)	Transição ocorre automaticamente após preparação. Justificativa: Encaminhar para registrar os dados.
registra (0100)	Realiza o registro do dado contido em "chaves" no registrador.	comparacao (0101)	Transição ocorre automaticamente após registrar. Justificativa: Encaminhar para etapa de comparação.
comparacao (0101)	Compara os valores da ROM ("s_dado") e do registrador ("s_chaves").	proximo (0110) ou fim (1111)	Transição para fim se o sinal "fimC" é ativado. Caso contrário, transição para proximo. Justificativa: Finalizar o ciclo ou continuar conforme necessário.
proximo (0110)	Incrementa o contador para acessar o próximo dado na memória ROM.	registra (0100)	Transição ocorre automaticamente após contar. Justificativa: Processar o próximo dado armazenado.
fim (1111)	Estado final. Sinaliza conclusão (pronto).	inicial (0000)	Transição ocorre automaticamente para reiniciar o sistema.

Ao analisar o diagrama de transição de estados, observa-se que, no estado INICIAL, o sinal "zeraR" é ativado, enquanto, no estado PREPARACAO, o sinal "zeraC" é ativado. No entanto, na tradução para Verilog, ocorreu uma inconsistência: no estado "inicial", ambos os sinais "zeraC" e "zeraR" são ativados simultaneamente, e o mesmo acontece no estado "preparacao", onde os dois sinais também permanecem ativos. Isso é mostrado na Figura 3, com um trecho do módulo "exp3_unidade_controle".

Figura 3 – Código em Verilog da Inconsistência na Unidade de Controle

```
// Logica de saida (maquina Moore)
always @* begin
    zeraC      = (Eatual == inicial || Eatual == preparacao) ? 1'b1 : 1'b0;
    zeraR      = (Eatual == inicial || Eatual == preparacao) ? 1'b1 : 1'b0;
```

3.3 PROJETO DO SISTEMA DIGITAL

No sistema digital sequencial proposto, a integração entre a Unidade de Controle (UC) e o Fluxo de Dados (FD) é fundamental para a operação correta e coordenada do sistema. Eles se comunicam por meio de dois tipos de sinais: sinais de controle e sinais de condição.

A UC emite sinais de controle para o FD com a finalidade de gerenciar e orquestrar as operações. Esses sinais são:

- zeraC (“s_zeraC”): Aciona a lógica que zera o contador_163 (interno do sistema), reiniciando o processo de endereçamento de memória.
- contaC (“s_contaC”): Incrementa o contador_163, avançando para o próximo dado na memória.
- zeraR (“s_zeraR”): Aciona o reset do registrador, preparando-o para
- registraR (“s_registraR”): Permite o armazenamento dos valores lidos nas chaves (entrada), registrando-os para posterior comparação.

Esses sinais controlam diretamente as operações dentro do FD, assegurando que o sistema siga a sequência lógica desejada, conforme definido pelos estados da UC. O FD, por sua vez, envia sinais de condição para a UC, fornecendo informações sobre o andamento do processo e os resultados das operações realizadas. O único sinal de condição enviado nesse sistema digital é o sinal fimC (“s_fimC”), que indica se a contagem foi concluída, informando à UC que todos os 16 dados foram percorridos.

Figura 4 – Código em Verilog da Integração entre FD e UC

```
// Fluxo de Dados
exp3_fluxo_dados FD [
    .clock      ( clock      ),
    .chaves     ( chaves     ),
    .zeraR      ( s_zeraR    ), // zera registradores
    .registraR  ( s_registraR ), // registra nos registradores
    .contaC     ( s_contaC   ), // incrementa contagem
    .zeraC      ( s_zeraC    ), // zera contagem
    .chavesIgualMemoria ( db_igual ),
    .fimC       ( s_fimC    ),
    .db_contagem ( s_contagem ),
    .db_chaves   ( s_chaves   ),
    .db_memoria  ( s_memoria  )
];

// Unidade de Controle
exp3_unidade_controle UC (
    .clock      ( clock      ),
    .reset      ( reset      ),
    .iniciar    ( iniciar    ),
    .fimC       ( s_fimC    ), // fim da contagem
    .zeraC      ( s_zeraC    ),
    .contaC     ( s_contaC   ),
    .zeraR      ( s_zeraR    ),
    .registraR  ( s_registraR ),
    .pronto     ( pronto     ),
    .db_estado   ( s_estado   )
);
```

Essa troca de sinais entre a UC e o FD garante que o sistema funcione de maneira coordenada e eficiente, controlando o fluxo de dados, os registradores e a memória, enquanto verifica e registra os dados conforme a sequência de operações.

Por fim, para depuração, serão utilizados os displays de sete segmentos HEX0, HEX1, HEX2 e HEX5 para acompanhar os sinais de depuração db_contagem, db_memoria, db_chaves e db_estados, respectivamente.

4 PLANO DE TESTES DO SISTEMA E SIMULAÇÕES

4.1 CENÁRIO DE TESTE 1 – SIMULAÇÃO DO FUNCIONAMENTO DO MÓDULO FLUXO DE DADOS

Simulação do funcionamento do módulo exp3_fluxo_dados no *Model/Sim* descrito na linguagem Verilog, utilizando a *TestBench* exp3_fluxo_dados_tb.

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

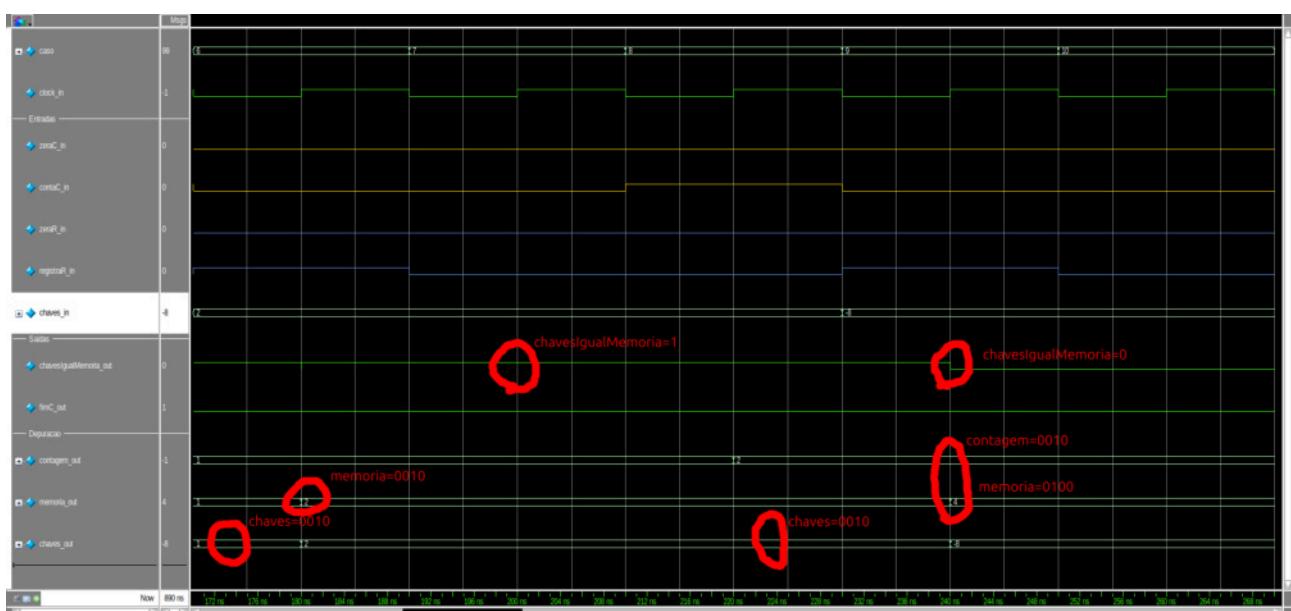
#	Operação	Sinais de Controle	Resultado Esperado	Resultado simulado ok?
c.i.	Condições iniciais (todas as entradas desativadas)	clock=0, zeraC=0, contaC=0, zeraR=0, registaR=0, chaves=0000	chavesIgualMemoria=0, fimC=0, db_contagem=0000, db_memoria=0001, db_chaves=0000	Sim
1	Zerar contador e registrador (manter outras entradas desativadas)	zeraC=1, zeraR=1, clock ↑	chavesIgualMemoria=0, fimC=0, db_contagem=0000, db_memoria=0001, db_chaves=0000	Sim
2	Verificar saídas com chaves=0001 (manter outras entradas desativadas)	zeraC=0, contaC=0, zeraR=0, registaR=0, chaves=0001, clock ↑	chavesIgualMemoria=0, fimC=0, db_contagem=0000, db_memoria=0001, db_chaves=0000	Sim
3	Registrar chaves com chaves=0001 (manter outras entradas desativadas)	chaves=0001, registaR=1, clock ↑	chavesIgualMemoria=1, fimC=0, db_contagem=0000, db_memoria=0001, db_chaves=0001	Sim
4	Verificar saídas com chaves=0001 (manter outras entradas desativadas)	chaves=0001, clock ↑	chavesIgualMemoria=1, fimC=0, db_contagem=0000, db_memoria=0001, db_chaves=0001	Sim
5	Incrementar contador (manter outras entradas desativadas)	contaC=1, clock ↑	chavesIgualMemoria=0, fimC=0, db_contagem=0001, db_memoria=0010, db_chaves=0001	Sim
6	Registrar chaves com chaves=0010 (manter outras entradas desativadas)	chaves=0010, clock ↑	chavesIgualMemoria=1, fimC=0, db_contagem=0001, db_memoria=0010, db_chaves=0010	Sim
7	Verificar saídas com chaves=0010 (manter outras entradas desativadas)	chaves=0010, clock ↑	chavesIgualMemoria=1, fimC=0, db_contagem=0001, db_memoria=0010, db_chaves=0010	Sim
8	Incrementar contador (manter outras entradas desativadas)	contaC=1, clock ↑	chavesIgualMemoria=0, fimC=0, db_contagem=0010, db_memoria=0100, db_chaves=0010	Sim
9	Registrar chaves com chaves=1000 (manter outras entradas desativadas)	chaves=1000, clock ↑	chavesIgualMemoria=0, fimC=0, db_contagem=0010, db_memoria=0100, db_chaves=1000	Sim

10	Verificar saídas com chaves=1000 (manter outras entradas desativadas)	chaves=1000, clock ↑	chavesIgualMemoria=0, fimC=0, db_contagem=0010, db_memoria=0100, db_chaves=1000	Sim
11	Incrementar contador até final da contagem	contaC=1, clock ↑ (13x)	chavesIgualMemoria=0, fimC=1, db_contagem=1111, db_memoria=0100, db_chaves=1000	Sim

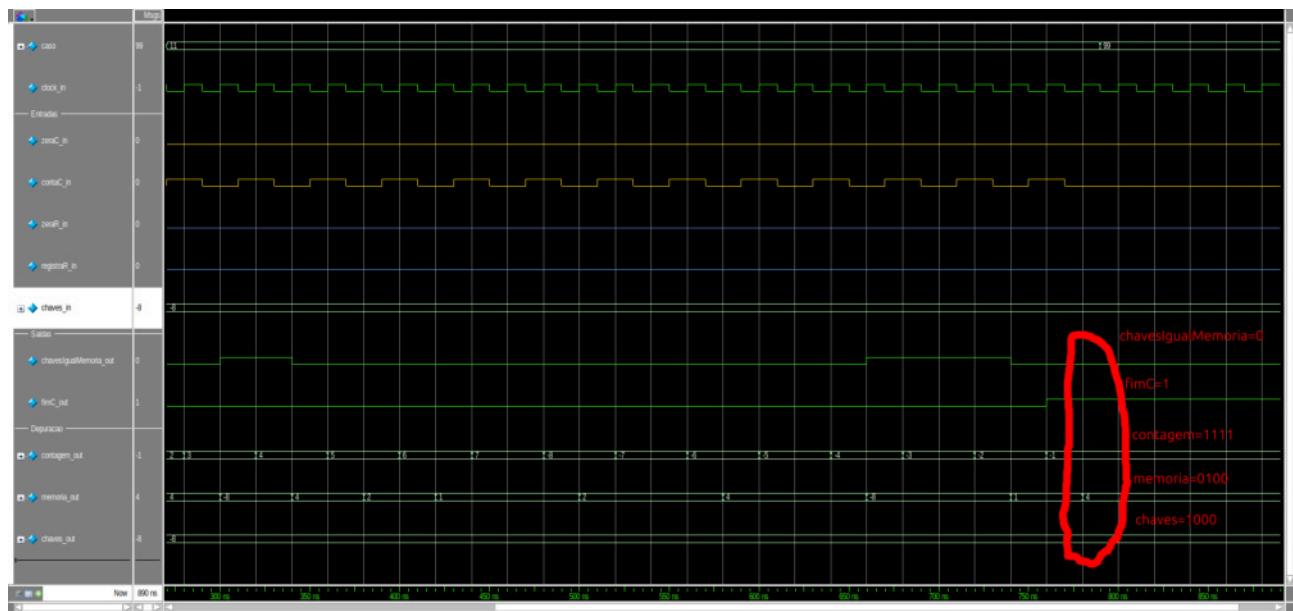
- Testes de 1 a 5



- Testes de 6 a 10



- Teste 11 e finalização



4.2 CENÁRIO DE TESTE 2 – SIMULAÇÃO DO FUNCIONAMENTO DO CIRCUITO COMPLETO

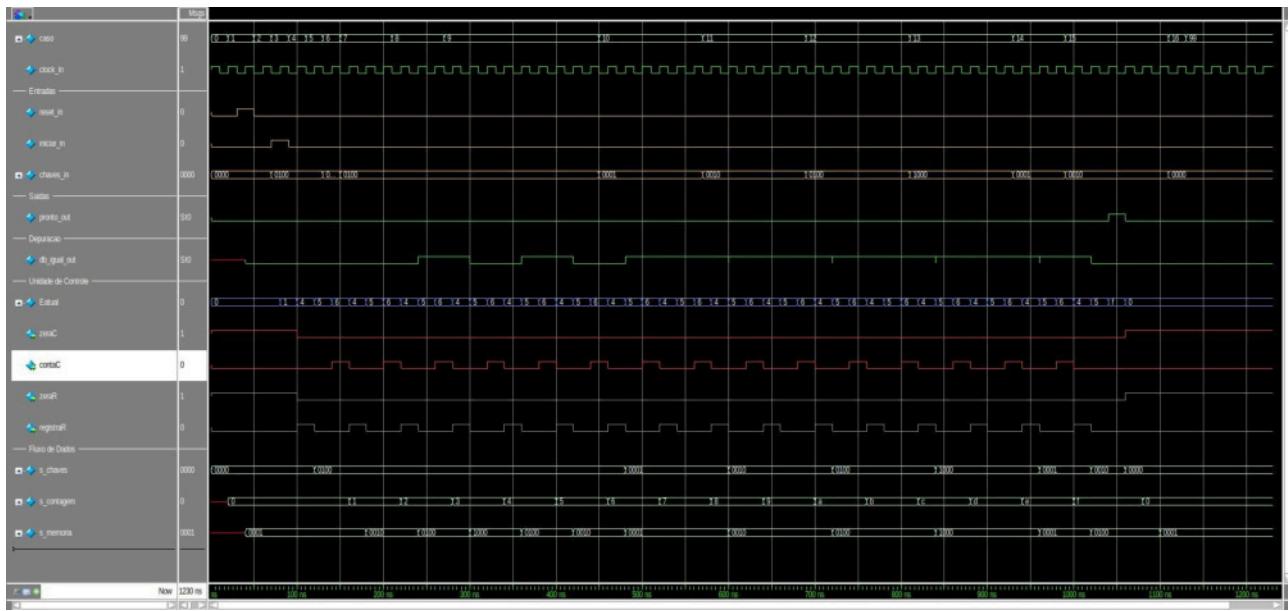
Simulação do funcionamento do módulo circuito_exp3 no *ModelSim* descrito na linguagem Verilog, utilizando a *TestBench* circuito_exp3_tb.

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

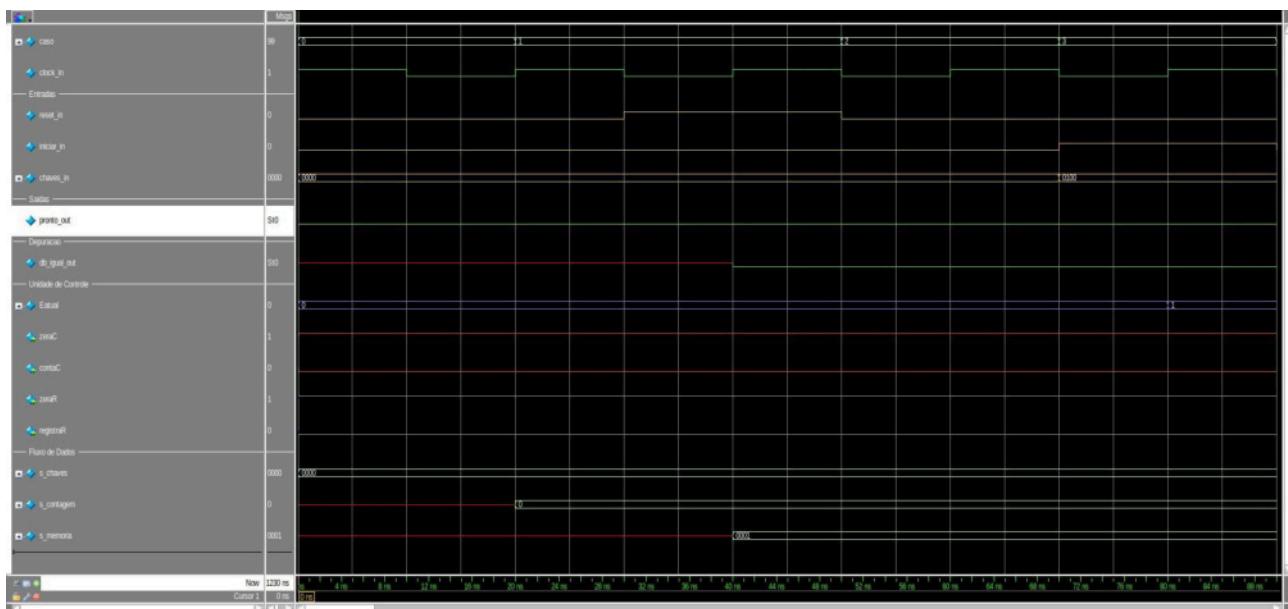
#	Operação	Sinais de Controle	Resultado Esperado	Resultado simulado ok?
c.i.	Condições iniciais	clock=0, reset=0, iniciar=0, chaves=0000	pronto=0, db_igual=0, db_contagem=0000, db_memoria=0000, db_chaves=0000, db_estado=0000	Sim
1	Resetar circuito e observar a saída da memória	reset=1, clock↑	pronto=0, db_igual=0, db_contagem=0000, db_memoria=0001, db_chaves=0000, db_estado=0000	Sim
2	Acionar sinal de clock 5 vezes com iniciar=0	reset=0, iniciar=0, clock↑ (5x)	pronto=0, db_igual=0, db_contagem=0000, db_memoria=0001, db_chaves=0000, db_estado=0000	Sim
3	Ajustar chaves para 0100, ativar iniciar=1 e acionar clock 1x	chaves=0100, iniciar=1, clock↑	pronto=0, db_igual=0, db_contagem=0000, db_memoria=0001, db_chaves=0000, db_estado=0001	Sim
4	Mantém chaves em 0100 e acionar clock 1x	chaves=0100, iniciar=0, clock↑	pronto=0, db_igual=0, db_contagem=0000, db_memoria=0001, db_chaves=0000, db_estado=0100	Sim
5	Mantém chaves em	chaves=0100,	pronto=0, db_igual=0,	Sim

	0100 e acionar clock 1x	iniciar=0, clock↑	db_contagem=0000, db_memoria=0001, db_chaves=0100, db_estado=0101	
6	Mantém chaves em 0100 e acionar clock 1x	chaves=0100, clock↑	pronto=0, db_igual=0, db_contagem=0000, db_memoria=0001, db_chaves=0100, db_estado=0110	Sim
7	Mantém chaves em 0100 e acionar clock 3x	chaves=0100, clock↑ (3x)	pronto=0, db_igual=0, db_contagem=0001, db_memoria=0010, db_chaves=0100, db_estado passa por 4, 5 e 6	Sim
8	Mantém chaves em 0100 e acionar clock 3x	chaves=0100, clock↑ (3x)	pronto=0, db_igual=1, db_contagem=0010, db_memoria=0100, db_chaves=0100, db_estado passa por 4, 5 e 6	Sim
9	Mantém chaves em 0100 e acionar clock 9x	chaves=0100, clock↑ (9x)	pronto=0, db_igual=0, db_contagem=(3, 4, 5), db_memoria=(8, 4, 2), db_chaves=0001, db_estado passa 3x por 4, 5 e 6	Sim
10	Ajustar chaves para 0001 e acionar clock 6x	chaves=0001, clock↑ (6x)	pronto=0, db_igual=1, db_contagem=(6, 7), db_memoria=0001, db_chaves=0001, db_estado passa 2x por 4, 5 e 6	Sim
11	Ajustar chaves para 0010 e acionar clock 6x	chaves=0010, clock↑ (6x)	pronto=0, db_igual=1, db_contagem=(8, 9), db_memoria=0010, db_chaves=0010, db_estado passa 2x por 4, 5 e 6	Sim
12	Ajustar chaves para 0100 e acionar clock 6x	chaves=0100, clock↑ (6x)	pronto=0, db_igual=1, db_contagem=(10, 11), db_memoria=0100, db_chaves=0100, db_estado passa 2x por 4, 5 e 6	Sim
13	Ajustar chaves para 1000 e acionar clock 6x	chaves=1000, clock↑ (6x)	pronto=0, db_igual=1, db_contagem=(12, 13), db_memoria=1000, db_chaves=1000, db_estado passa 2x por 4, 5 e 6	Sim
14	Ajustar chaves para 0000 e acionar clock 3x	chaves=0000, clock↑ (3x)	pronto=0, db_igual=0, db_contagem=1110, db_memoria=0001, db_chaves=0000, db_estado passa por 4, 5 e 6	Sim
15	Mantém chaves em 0000 e acionar clock 3x	chaves=0000, clock↑ (3x)	pronto=1, db_igual=0, db_contagem=1111, db_memoria=0100, db_chaves=0000, db_estado passa por 4, 5 e F	Sim
16	Mantém chaves em 0000 e acionar clock	chaves=0000, clock↑	pronto=0, db_igual=0, db_contagem=1111, db_memoria=0100, db_chaves=0000, db_estado=0000	Sim

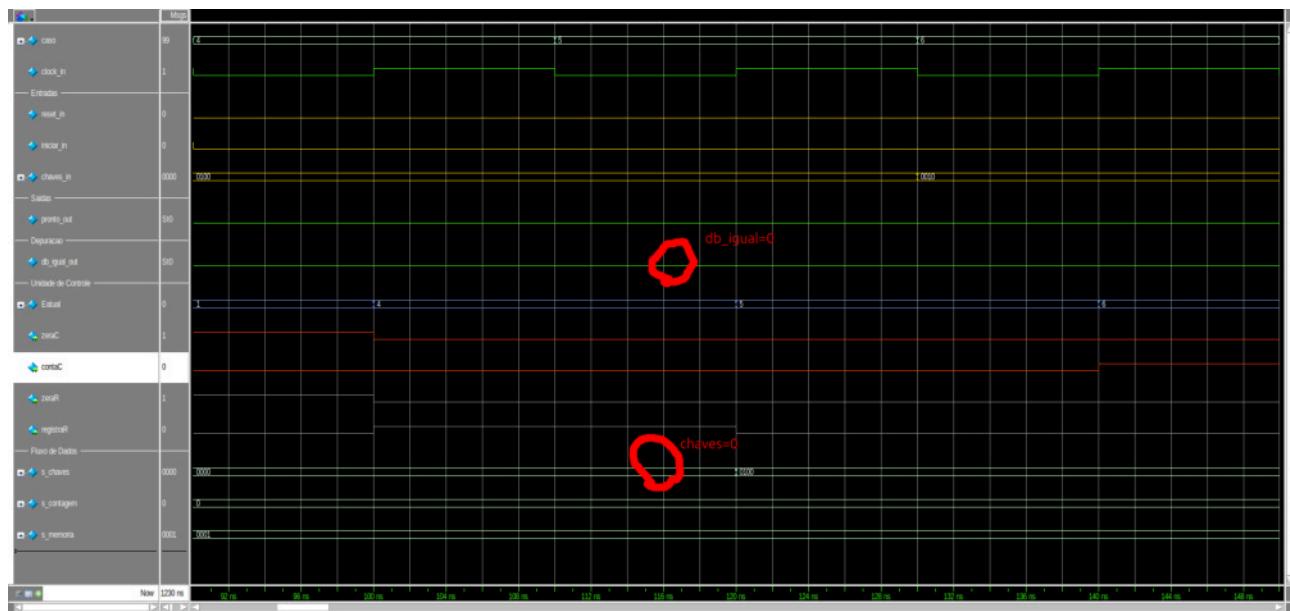
- Visão geral de todos os testes



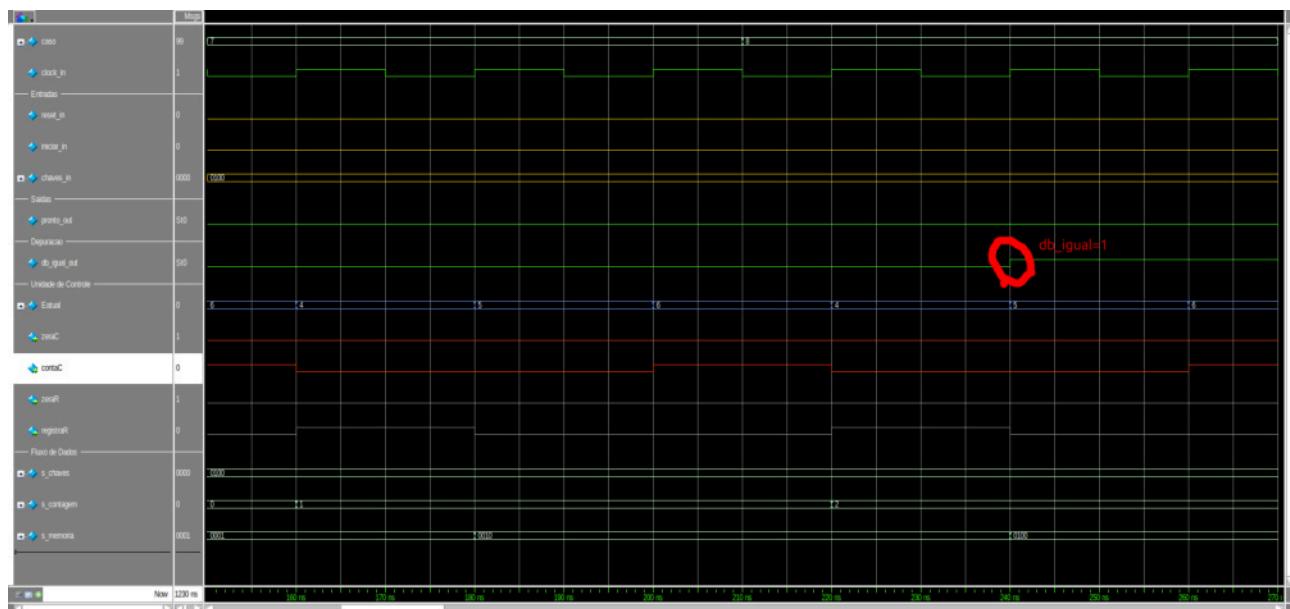
- Testes de 1 a 3



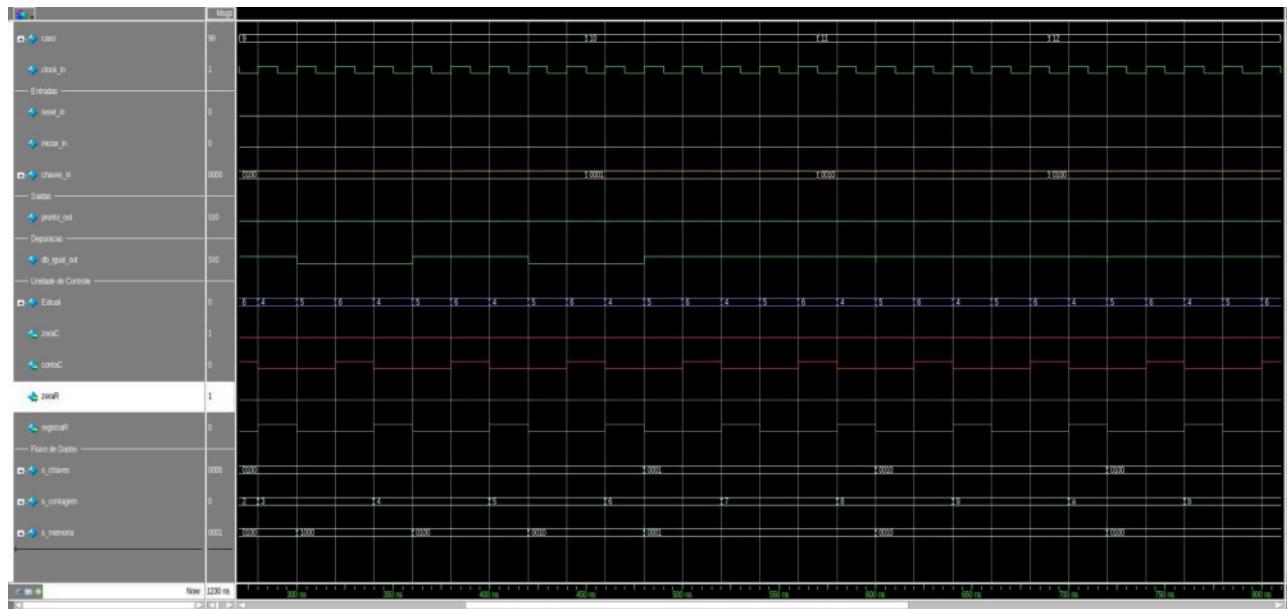
- Testes de 4 a 6



- Testes 7 e 8



- Testes de 9 a 12



- Testes de 13 ao 16 e finalização



5 IMPLANTAÇÃO DO PROJETO

5.1 PINAGEM DA PLACA FPGA

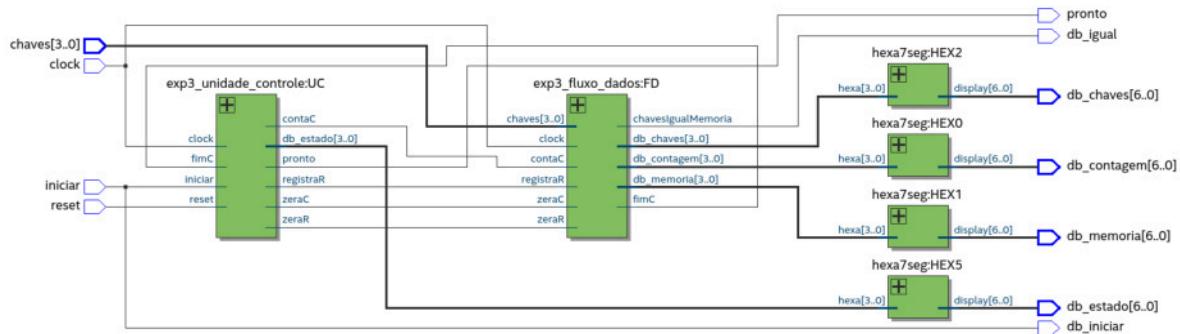
O planejamento da pinagem é um passo essencial no desenvolvimento de projetos utilizando o FPGA Cyclone V, realizado por meio do "pin planner" do Intel Quartus. As configurações evidenciam detalhadamente as alocações dos pinos em relação aos nós do circuito. Além disso, segue também a imagem da RTLViewer do funcionamento do circuito, importante para conferir o funcionamento dos sinais e sua consistência.

Tabela 4 – Pinagem da Placa FPGA

Sinal	Pino na Placa DE0-CV	Pino no FPGA	Analog Discovery
CHAVES(0)	chave SW1	PIN_V13	-
CHAVES(1)	chave SW2	PIN_T13	-
CHAVES(2)	chave SW3	PIN_T12	-
CHAVES(3)	chave SW4	PIN_AA15	-
CLOCK	GPIO_0_D0	PIN_N16	StaticIO - Button 0/1
DB_CHAVES(0)	display HEX20	PIN_Y19	-
DB_CHAVES(1)	display HEX21	PIN_AB17	-
DB_CHAVES(2)	display HEX22	PIN_AA10	-
DB_CHAVES(3)	display HEX23	PIN_Y14	-
DB_CHAVES(4)	display HEX24	PIN_V14	-
DB_CHAVES(5)	display HEX25	PIN_AB22	-
DB_CHAVES(6)	display HEX26	PIN_AB21	-
DB_CONTAC	led LEDR5	PIN_N1	-
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_FIMC	led LEDR6	PIN_U2	-
DB_IGUAL	led LEDR1	PIN_AA1	-
DB_INICIAR	led LEDR2	PIN_W2	-
DB_MEMORIA(0)	display HEX10	PIN_AA20	-
DB_MEMORIA(1)	display HEX11	PIN_AB20	-
DB_MEMORIA(2)	display HEX12	PIN_AA19	-

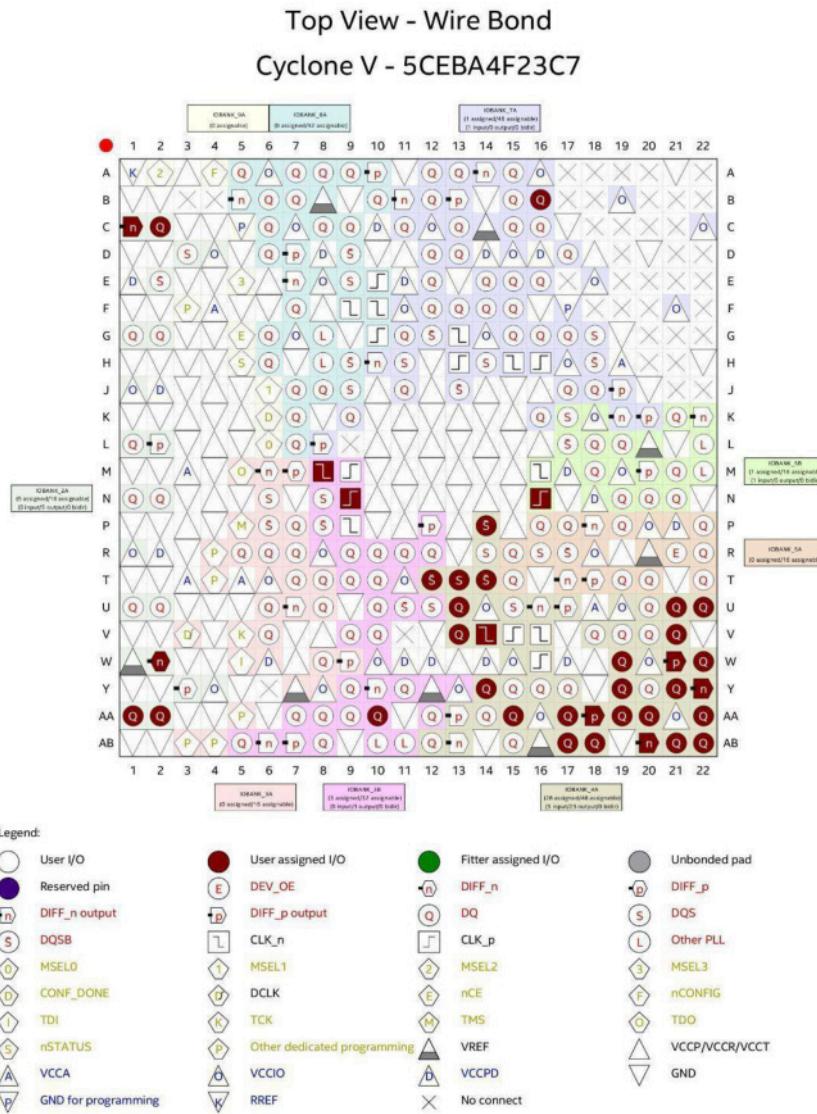
DB_MEMORIA(3)	display HEX13	PIN_AA18	-
DB_MEMORIA(4)	display HEX14	PIN_AB18	-
DB_MEMORIA(5)	display HEX15	PIN_AA17	-
DB_MEMORIA(6)	display HEX16	PIN_U22	-
DB_REGISTRAR	led LEDR9	PIN_L1	-
DB_ZERAC	led LEDR4	PIN_N2	-
DB_ZERAR	led LEDR8	PIN_L2	-
INICIAR	chave SW0	PIN_U13	-
PRONTO	led LEDR0	PIN_AA2	-
RESET	GPIO_0_D1	PIN_B16	StaticIO - Button 0/1

Figura 5 – RTLViewer



A seguir, consta a *Top View* da placa FPGA, que permite a visão superior da pinagem após a definição da localização de cada sinal.

Figura 6 – Top view da placa FPGA



5.2 ESTRATÉGIA DE MONTAGEM

Para o início da montagem, os alunos chegarão na bancada com o “pin planner” já configurado com as entradas e saídas corretamente designadas aos seus respectivos pinos na FPGA. O próximo passo é verificar as conexões na placa DE0-CV, tanto com a fonte de alimentação, quanto com o computador.

Após verificar a conexão de todos os equipamentos, os alunos inicializarão o Intel Quartus Prime no computador da bancada B1 do Laboratório Digital, importarão o projeto via arquivo .qar, compilarão novamente e iniciarão a programação na FPGA.

Serão monitorados os sinais de depuração “db_contagem”, “db_memoria”, “db_estado” e “db_chaves” pelos displays de 7 segmentos HEX0, HEX1, HEX2 e HEX5, além de outros sinais de depuração de um bit, que serão monitorados por meio dos LEDs.

5.3 ESTRATÉGIA DE DEPURAÇÃO

A depuração do sistema digital sequencial será realizada por meio de diversas etapas que assegurem a verificação das conexões entre os componentes (placa, computador e fonte de alimentação) e a correta operação do circuito. Durante os testes, os sinais de depuração fornecem informações cruciais para a validação do funcionamento do sistema. Os sinais principais a serem monitorados incluem: "db_contagem", "db_igual", "db_estado", "db_memoria", "db_chaves", "db_fimC", entre outros. O processo de depuração envolve as seguintes etapas:

- Verificação do número do contador e o respectivo dado na memória: Durante os testes, será verificada a lógica entre o endereço sendo lido e o valor esperado. Caso haja inversão ou erro lógico, o código Verilog será revisado, juntamente com a pinagem definida no "Pin Planner", buscando possíveis erros de inversão de bits ou conexões incorretas.
- Verificação das comparações: A comparação entre as chaves de entrada e os dados armazenados na memória será monitorada. Caso o resultado da comparação seja inesperado, será realizada uma análise sistemática alterando apenas os valores das chaves ou resetando o contador, com foco na identificação de erros lógicos, inversões de bits ou falhas no fluxo de dados.
- Verificação do sinal de fim e de pronto: O sinal "db_fimC" e "pronto" indicam a conclusão da operação do contador. É esperado que caso o LED não acenda, seja verificado se o contador foi para o início ou se há alguma lógica mal implementada.

Após a detecção de qualquer erro, será feita a correção necessária, seguida pela repetição dos testes, com foco no módulo afetado e nos testes gerais do sistema. Será também utilizado o *Analog Discovery* e, caso seja encontrado algum erro na utilização do software *WaveForms*, os sinais serão cautelosamente depurados para entender se o problema aconteceu devido ao uso incorreto, já que será o primeiro contato dos alunos com esse tipo de atividade na bancada.

5.4 EXECUÇÃO PRÁTICA DO CENÁRIO DE TESTE 1 - TESTES NA FPGA DO CIRCUITO COMPLETO

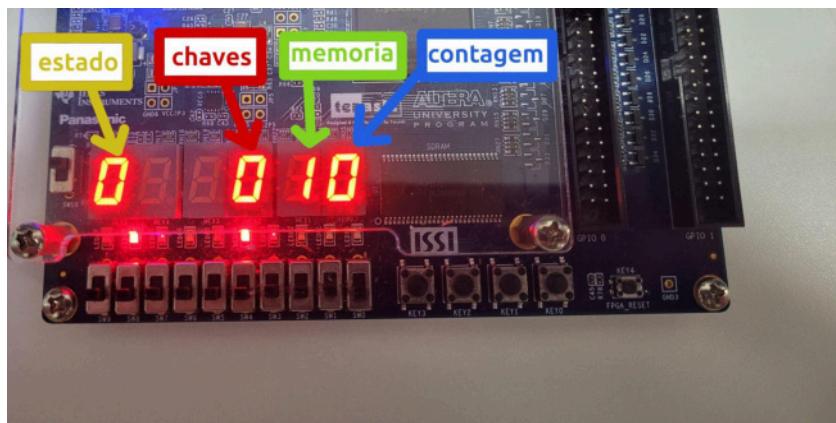
Testes na FPGA do funcionamento do módulo circuito_exp3, descrito na linguagem Verilog, utilizando a *TestBench* circuito_exp3_tb.

Tabela 5 – Descrição e Resultados Práticos do Cenário de Teste 1

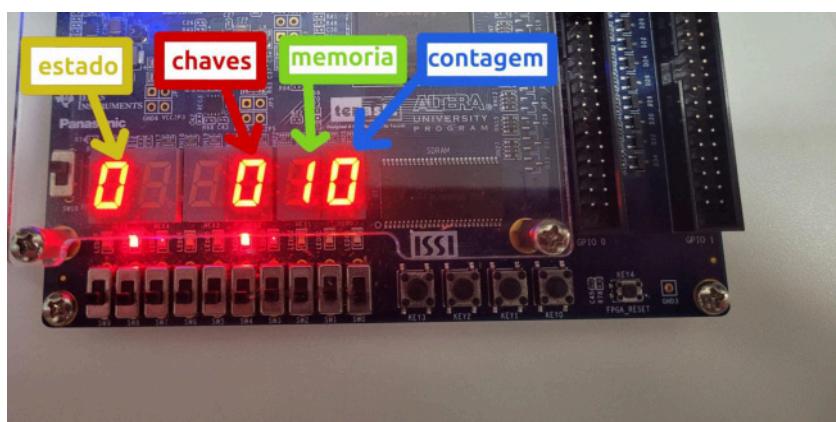
#	Operação	Sinais de Controle	Resultado Esperado	do simulador
c.i.	Condições iniciais	clock=0, reset=0, iniciar=0, chaves=0000	pronto=0, db_igual=0, db_contagem=0000, db_memoria=0000, db_chaves=0000, db_estado=0000	Sim
1	Resetar circuito e observar a saída da memória	reset=1, clock↑	pronto=0, db_igual=0, db_contagem=0000, db_memoria=0001, db_chaves=0000, db_estado=0000	Sim
2	Acionar sinal de clock 5 vezes com iniciar=0	reset=0, iniciar=0, clock↑ (5x)	pronto=0, db_igual=0, db_contagem=0000, db_memoria=0001, db_chaves=0000, db_estado=0000	Sim
3	Ajustar chaves para 0100, ativar iniciar=1 e acionar clock 1x	chaves=0100, iniciar=1, clock↑	pronto=0, db_igual=0, db_contagem=0000, db_memoria=0001, db_chaves=0000, db_estado=0001	Sim
4	Mantém chaves em 0100 e acionar clock 1x	chaves=0100, iniciar=0, clock↑	pronto=0, db_igual=0, db_contagem=0000, db_memoria=0001, db_chaves=0000, db_estado=0100	Sim
5	Mantém chaves em 0100 e acionar clock 1x	chaves=0100, iniciar=0, clock↑	pronto=0, db_igual=0, db_contagem=0000, db_memoria=0001, db_chaves=0100, db_estado=0101	Sim
6	Mantém chaves em 0100 e acionar clock 1x	chaves=0100, clock↑	pronto=0, db_igual=0, db_contagem=0000, db_memoria=0001, db_chaves=0100, db_estado=0110	Sim
7	Mantém chaves em 0100 e acionar clock 3x	chaves=0100, clock↑ (3x)	pronto=0, db_igual=0, db_contagem=0001, db_memoria=0010, db_chaves=0100, db_estado passa por 4, 5 e 6	Sim
8	Mantém chaves em 0100 e acionar clock 3x	chaves=0100, clock↑ (3x)	pronto=0, db_igual=1, db_contagem=0010, db_memoria=0100, db_chaves=0100, db_estado passa por 4, 5 e 6	Sim
9	Mantém chaves em 0100 e acionar clock 9x	chaves=0100, clock↑ (9x)	pronto=0, db_igual=0, db_contagem=(3, 4, 5), db_memoria=(8, 4, 2), db_chaves=0001, db_estado passa 3x por 4, 5 e 6	Sim
10	Ajustar chaves para 0001 e acionar clock 6x	chaves=0001, clock↑ (6x)	pronto=0, db_igual=1, db_contagem=(6, 7), db_memoria=0001, db_chaves=0001, db_estado passa 2x por 4, 5 e 6	Sim
11	Ajustar chaves para 0010 e acionar clock 6x	chaves=0010, clock↑ (6x)	pronto=0, db_igual=1, db_contagem=(8, 9), db_memoria=0010, db_chaves=0010, db_estado passa 2x por 4, 5 e 6	Sim
12	Ajustar chaves para 0100 e acionar clock 6x	chaves=0100, clock↑ (6x)	pronto=0, db_igual=1, db_contagem=(10, 11), db_memoria=0100, db_chaves=0100, db_estado passa 2x por 4, 5 e 6	Sim

13	Ajustar chaves para 1000 e acionar clock 6x	chaves=1000, clock↑ (6x)	pronto=0, db_igual=1, db_contagem=(12, 13), db_memoria=1000, db_chaves=1000, db_estado passa 2x por 4, 5 e 6	Sim
14	Ajustar chaves para 0000 e acionar clock 3x	chaves=0000, clock↑ (3x)	pronto=0, db_igual=0, db_contagem=1110, db_memoria=0001, db_chaves=0000, db_estado passa por 4, 5 e 6	Sim
15	Mantém chaves em 0000 e acionar clock 3x	chaves=0000, clock↑ (3x)	pronto=1, db_igual=0, db_contagem=1111, db_memoria=0100, db_chaves=0000, db_estado passa por 4, 5 e F	Sim
16	Mantém chaves em 0000 e acionar clock	chaves=0000, clock↑	pronto=0, db_igual=0, db_contagem=1111, db_memoria=0100, db_chaves=0000, db_estado=0000	Sim

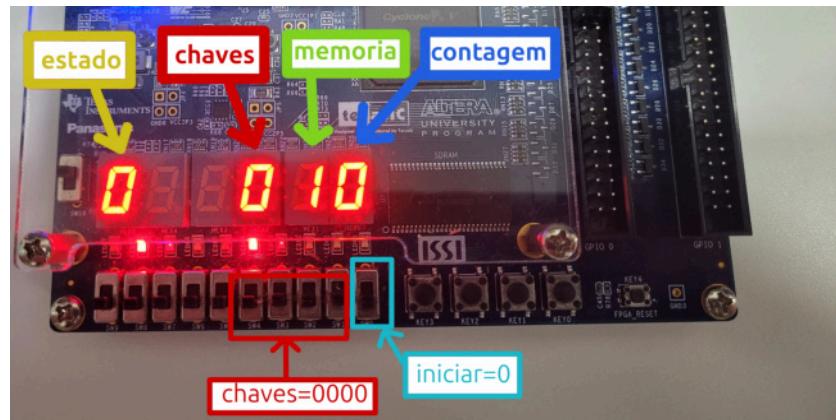
- Condições iniciais



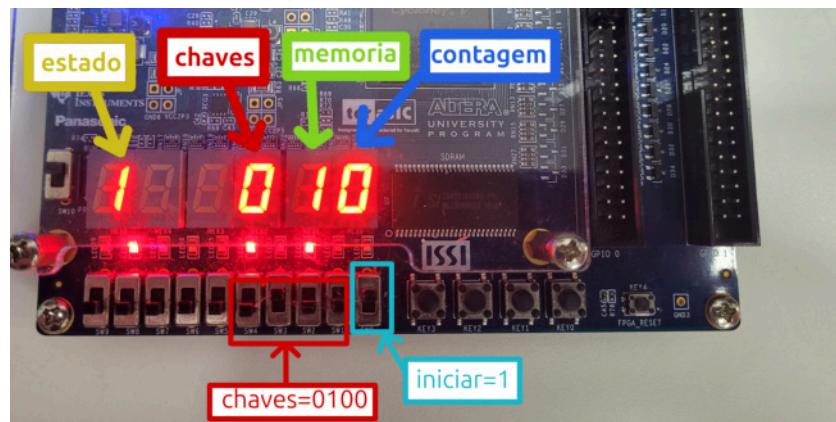
- Teste 1 - Resetar circuito e observar a saída da memória



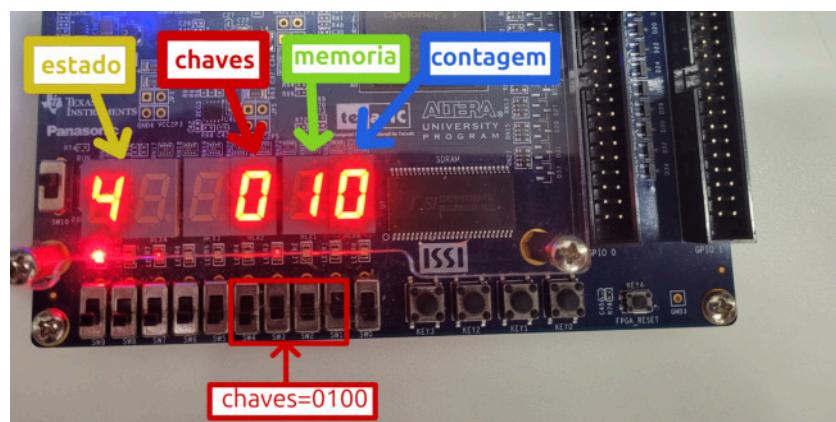
- Teste 2 - Acionar sinal de clock 5 vezes com iniciar=0



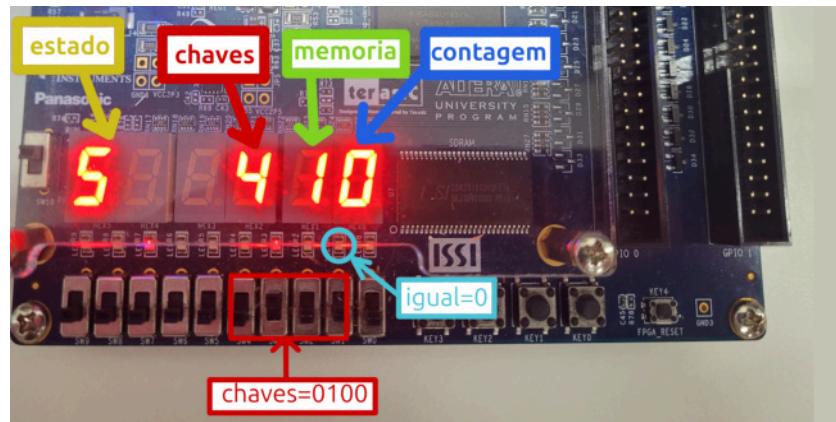
- Teste 3 - Ajustar chaves para 0100, ativar iniciar=1 e acionar clock 1x



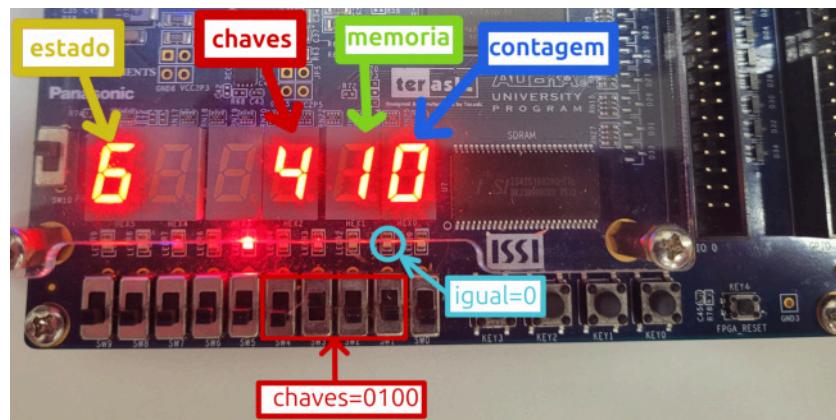
- Teste 4 - Mantém chaves em 0100 e acionar clock 1x



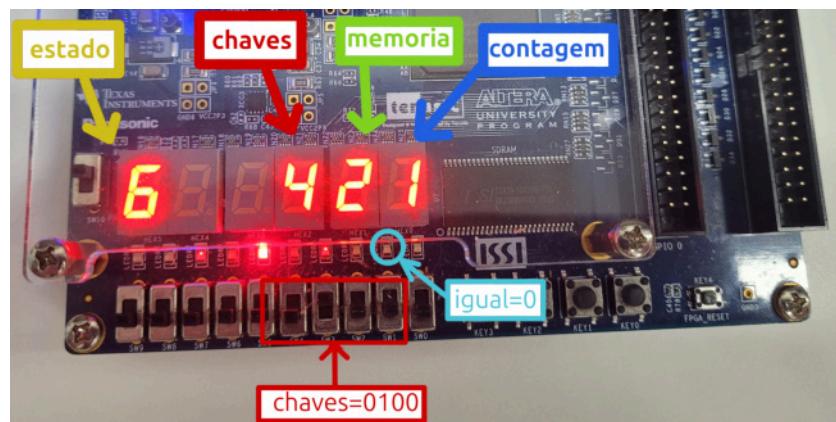
- Teste 5 - Mantém chaves em 0100 e acionar clock 1x



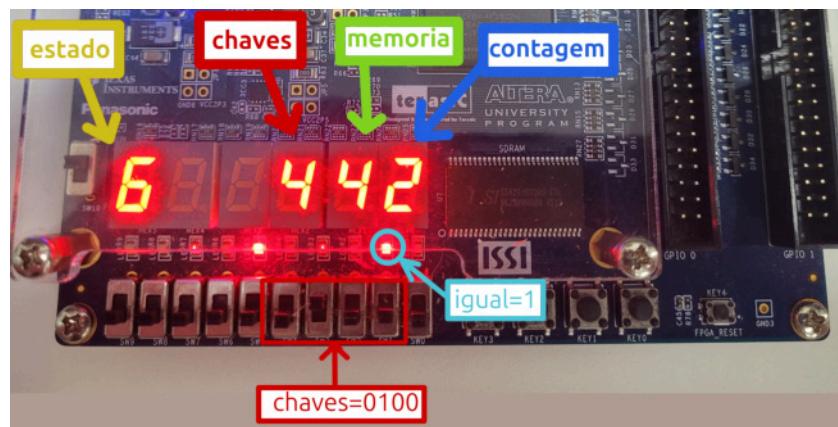
- Teste 6 - Mantém chaves em 0100 e acionar clock 1x



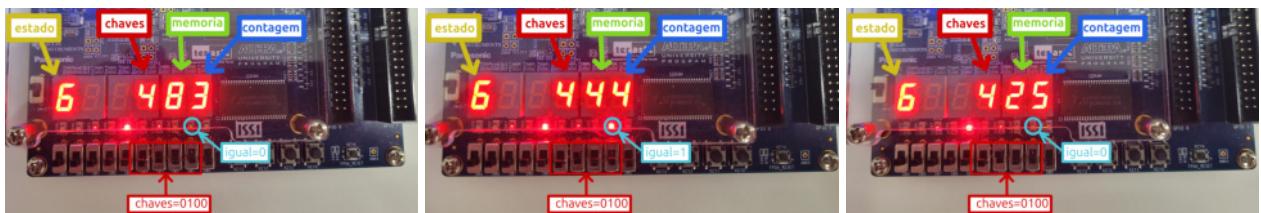
- Teste 7 - Mantém chaves em 0100 e acionar clock 3x



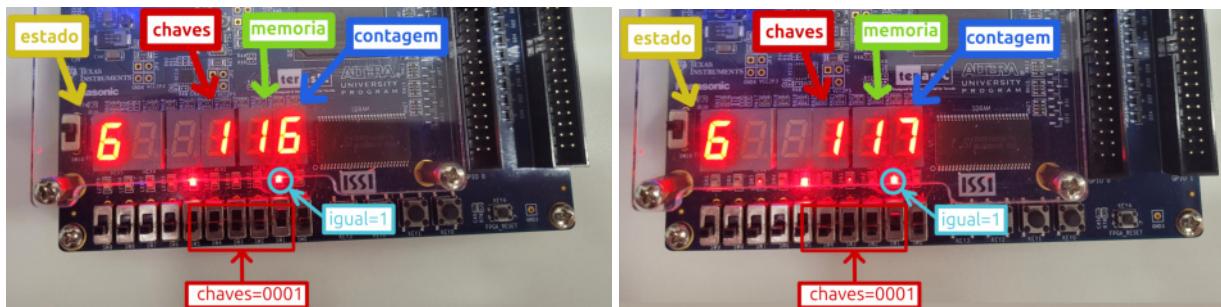
- Teste 8 - Mantém chaves em 0100 e acionar clock 3x



- Teste 9 - Mantém chaves em 0100 e acionar clock 9x



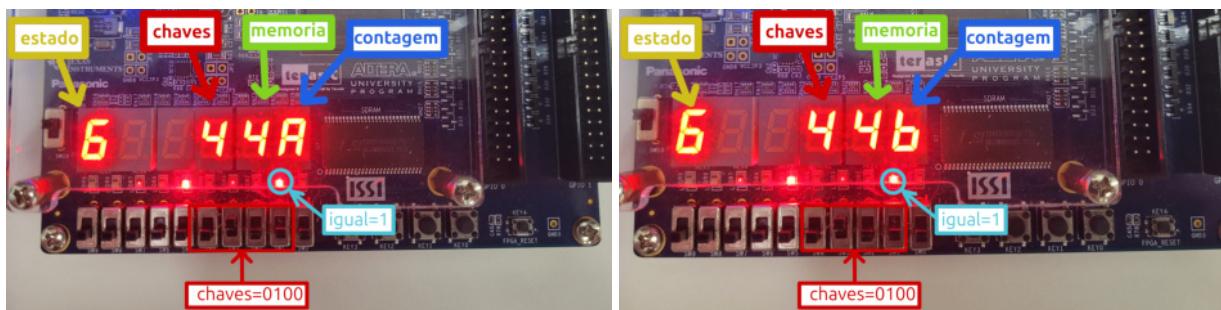
- Teste 10 - Ajustar chaves para 0001 e acionar clock 6x



- Teste 11 - Ajustar chaves para 0010 e acionar clock 6x



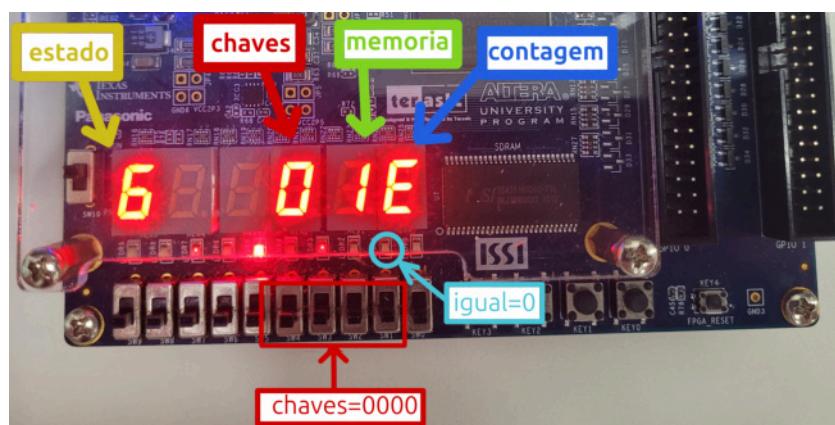
- Teste 12 - Ajustar chaves para 0100 e acionar clock 6x



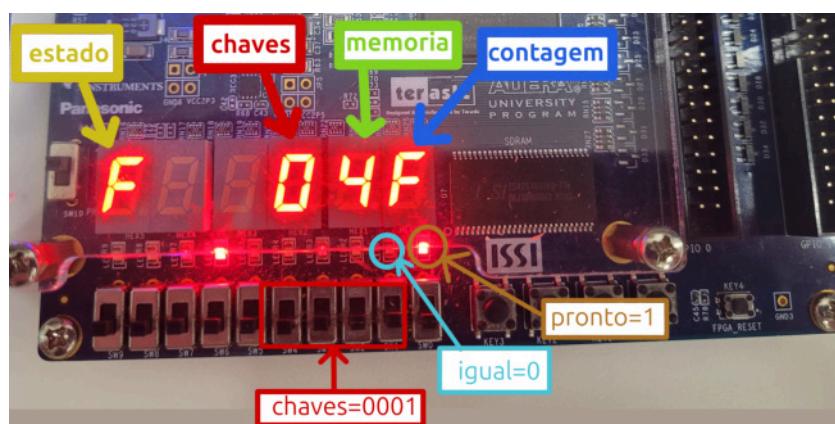
- Teste 13 - Ajustar chaves para 1000 e acionar clock 6x



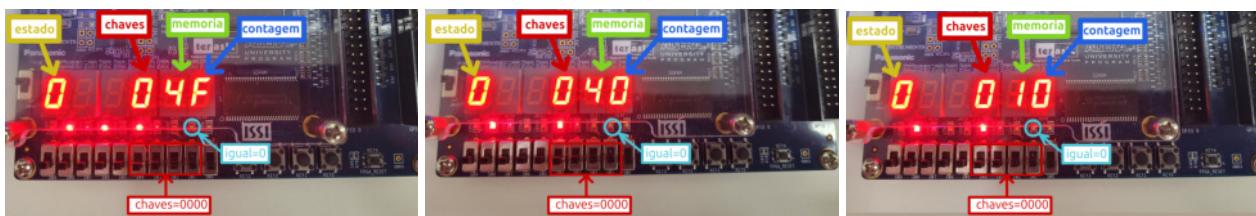
- Teste 14 - Ajustar chaves para 0000 e acionar clock 3x



- Teste 15 - Mantém chaves em 0000 e acionar clock 3x



- Teste 16 - Mantém chaves em 0000 e acionar clock



Durante a execução da experiência, os alunos confirmaram que, até o teste 15, o circuito funcionava como esperado, conforme monitorado pelos sinais de depuração. No entanto, no teste 16, foi observada uma inconsistência entre o comportamento descrito no planejamento e o comportamento efetivamente registrado. Segundo o planejamento, esperava-se que, assim que ocorresse a transição para o estado "iniciar", o conteúdo da memória e do contador (sinais de depuração "db_memoria" e "db_contagem") fossem atualizados para 1 e 0, respectivamente, indicando a leitura do conteúdo da posição 0 da memória. Contudo, o que ocorreu foi a transição para o estado 0, enquanto os sinais "db_contagem" e "db_memoria" permaneceram com os valores F e 4, respectivamente.

Após análise detalhada dos sinais de depuração, os alunos compreenderam que a unidade de controle operava de forma assíncrona, realizando a transição para o estado "iniciar" e ativando os sinais "zeraC" e "zeraR". Porém, o "contador_163" e o "registrar_4" são componentes síncronos e, portanto, dependem do clock, assim como o "sync_rom_16x4". Isso significa que, embora os sinais "zeraR" e "zeraC" tenham sido ativados, o contador e o registrador só seriam resetados no próximo ciclo de clock. O mesmo ocorre com o acesso à memória: mesmo após o incremento do contador, seria necessário mais um ciclo de clock para acessar o novo dado da memória.

Concluiu-se, então, que ao acionar mais um ciclo de clock, o valor do contador seria resetado para 0, enquanto o conteúdo de "db_memoria" permaneceria em 4. Apenas após outro ciclo de clock, o valor de "db_memoria" seria atualizado para 1, correspondente à posição 0 da memória ROM.

Esse processo ajudou os alunos a compreenderem, de forma mais profunda, o funcionamento dos estados da máquina, as razões pelas quais os sinais de depuração mudaram nos momentos observados e a necessidade dos estados 4, 5 e 6. A análise cuidadosa durante os testes do desafio foi fundamental para a experiência e contribuiu significativamente para o aprendizado.

6 PROJETO DO DESAFIO DA EXPERIÊNCIA

6.1 DESCRIÇÃO DO DESAFIO

O desafio consiste em implementar dois novos sinais de depuração no sistema digital: "acertou" e "errou". O sinal "acertou" é ativado ao final da contagem quando o operador da placa consegue inserir corretamente, nas chaves, todos os valores correspondentes aos dados armazenados na memória ROM previamente programada, à medida que essa memória é percorrida.

Por outro lado, o sinal "errou" é acionado quando o valor inserido nas chaves não corresponde ao valor armazenado na memória para a posição específica durante a contagem. Nesse caso, o ciclo de execução do circuito é interrompido, e o sinal "errou" é ativado, alertando o operador sobre o erro no valor inserido.

6.2 DESCRIÇÃO DO PROJETO LÓGICO

Para o projeto lógico do desafio a UC foi alterada para controlar o fim do ciclo caso o sinal "errou" precisasse ser ativado e o sinal "acertou" no fim do ciclo.

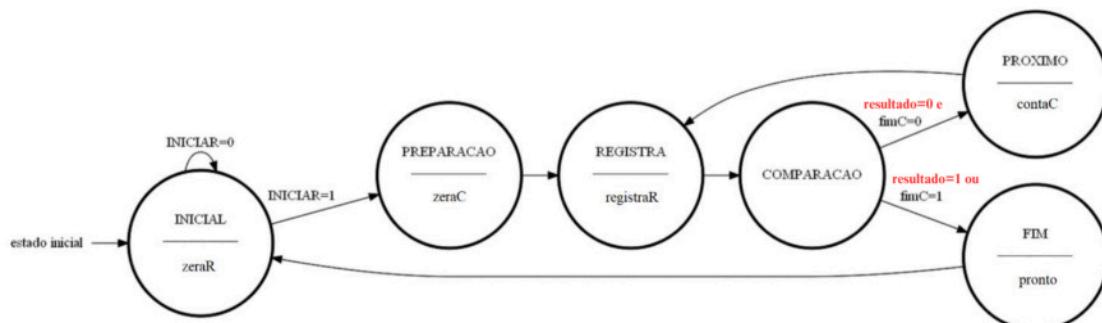
6.2.1 Alterações do Fluxo de Dados

O Fluxo de Dados não precisou ser alterado, como já foi explicado na descrição.

6.2.2 Alterações da Unidade de Controle

Foram adicionados um sinal de entrada que indicava o resultado da comparação ("s_resultado"), quanto dois sinais de saída "acertou" e "errou". Para a entrada utilizou-se o sinal " $\sim s_igual$ ", saída do FD "chavesIgualMemoria", para ser o sinal de condição no acionamento dos sinais desejados. A seguir uma nova foto do diagrama de transição de estados com as modificações feitas na lógica:

Figura 7 – Diagrama de Estados do Desafio



6.2.3 Alterações do Sistema Digital

A pinagem da placa e o RTLViewer foram alterados para a inclusão dos novos sinais.

Tabela 6 – Pinagem na placa FPGA do Desafio

Sinal	Pino na Placa DE0-CV	Pino no FPGA	Analog Discovery
ACERTOU	led LEDR8	PIN_L2	-
CHAVES(0)	chave SW1	PIN_V13	-
CHAVES(1)	chave SW2	PIN_T13	-
CHAVES(2)	chave SW3	PIN_T12	-
CHAVES(3)	chave SW4	PIN_AA15	-
CLOCK	GPIO_0_D0	PIN_N16	StaticIO - Button 0/1
DB_CHAVES(0)	display HEX20	PIN_Y19	-
DB_CHAVES(1)	display HEX21	PIN_AB17	-
DB_CHAVES(2)	display HEX22	PIN_AA10	-
DB_CHAVES(3)	display HEX23	PIN_Y14	-
DB_CHAVES(4)	display HEX24	PIN_V14	-
DB_CHAVES(5)	display HEX25	PIN_AB22	-
DB_CHAVES(6)	display HEX26	PIN_AB21	-
DB_CONTAC	led LEDR5	PIN_N1	-
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_FIMC	led LEDR6	PIN_U2	-
DB_IGUAL	led LEDR1	PIN_AA1	-
DB_INICIAR	led LEDR2	PIN_W2	-

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_REGISTRAR	led LEDR7	PIN_U1	-
DB_ZERAC	led LEDR4	PIN_N2	-
DB_ZERAR	led LEDR3	PIN_Y3	-
ERROU	led LEDR9	PIN_L1	-
INICIAR	chave SW0	PIN_U13	-
PRONTO	led LEDR0	PIN_AA2	-
RESET	GPIO_0_D1	PIN_B16	StaticIO - Button 0/1

A seguir, também consta a RTL View do circuito, importante para conferir o funcionamento do circuito:

Figura 8 – RTL View do circuito do desafio

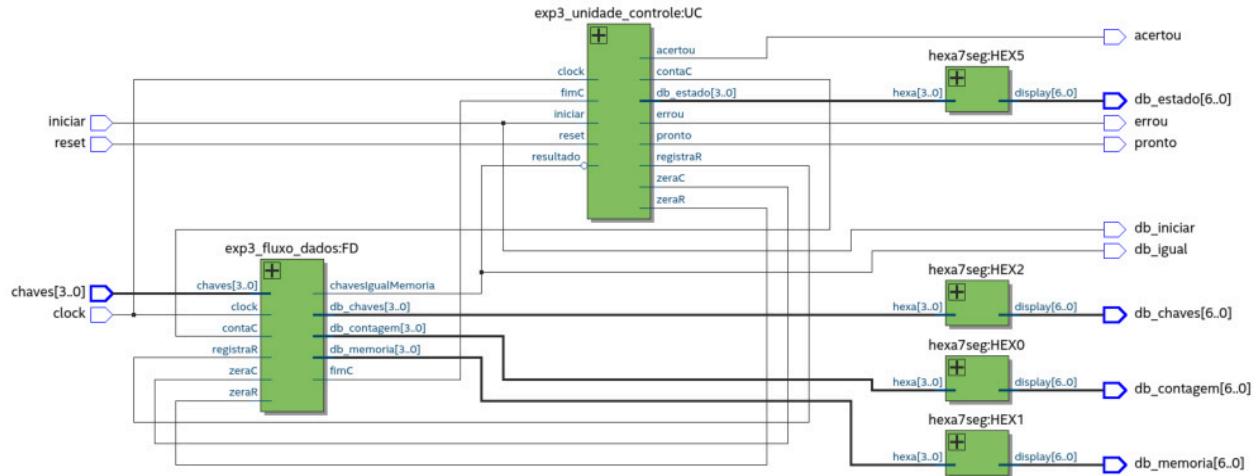
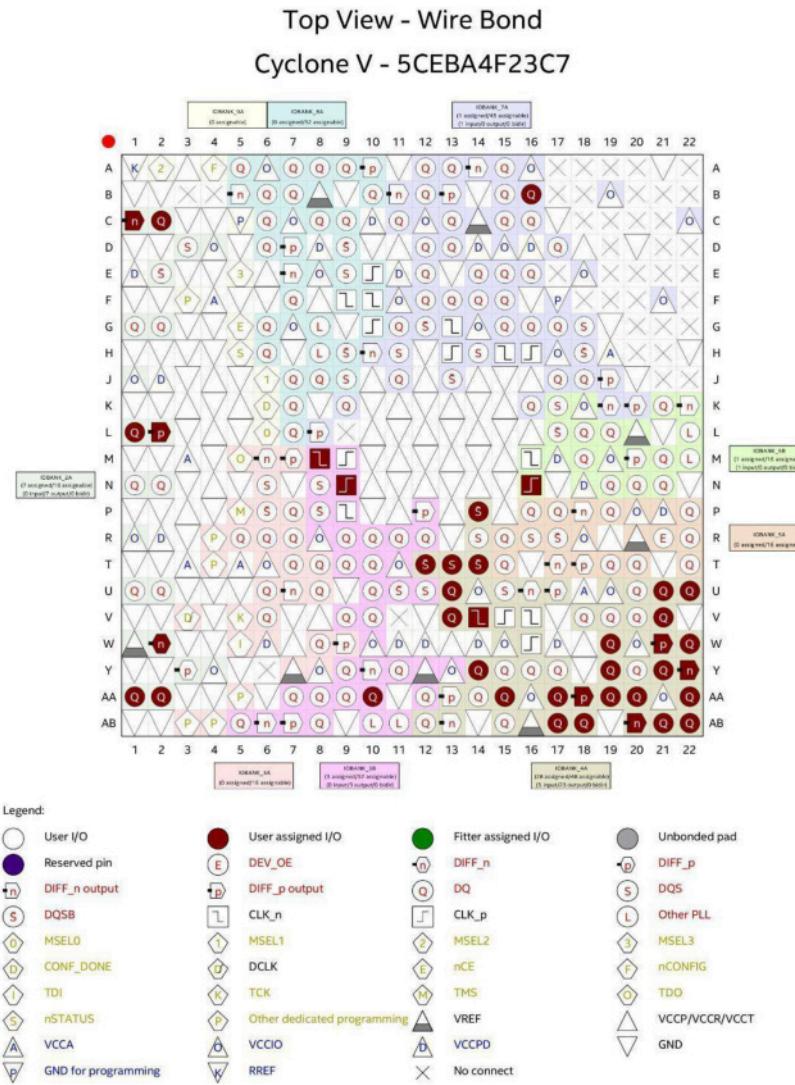


Figura 9 – Imagem da visão do topo da placa



6.3 VERIFICAÇÃO E VALIDAÇÃO DO DESAFIO

Para os novos testes, serão monitorados atentamente os sinais de “pronto”, “acertou” e “errou” para os dois casos de Teste descritos, indicando tanto o caso em que houver acerto de todos os casos, quanto no caso em que houver erro no quarto dado.

Para isso foram feitas duas *TestBench*s com os dois casos a serem descritos nos cenários de Teste. Os (*circuito_exp3_desafio1_tb* e *circuito_exp3_desafio2_tb*) foram feitos cada um para focar na observação do comportamento de “acertou” e “errou”.

6.3.1 Cenário de Teste 1 – Simulação do Desafio no *ModelSim* - caso 1

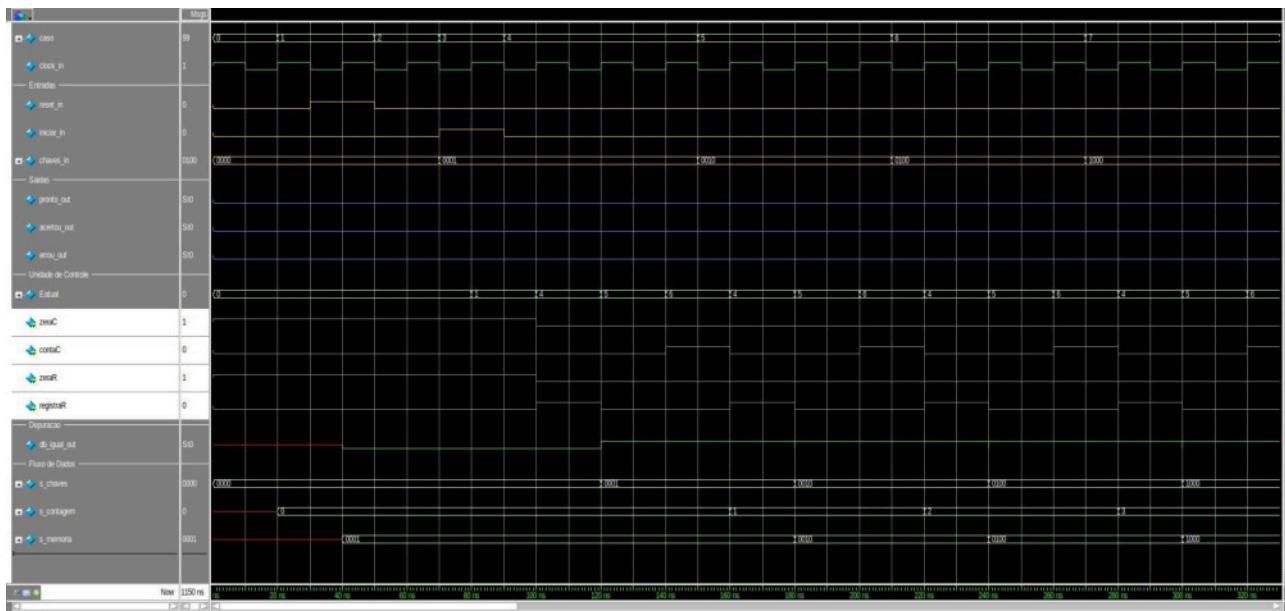
Simulação no *Modelsim* do funcionamento do módulo *circuito_exp3_desafio*, descrito na linguagem Verilog, utilizando a *TestBench* *circuito_exp3_desafio2_tb*.

Tabela 7 – Descrição e Resultados do Cenário de Teste 1 para o Desafio

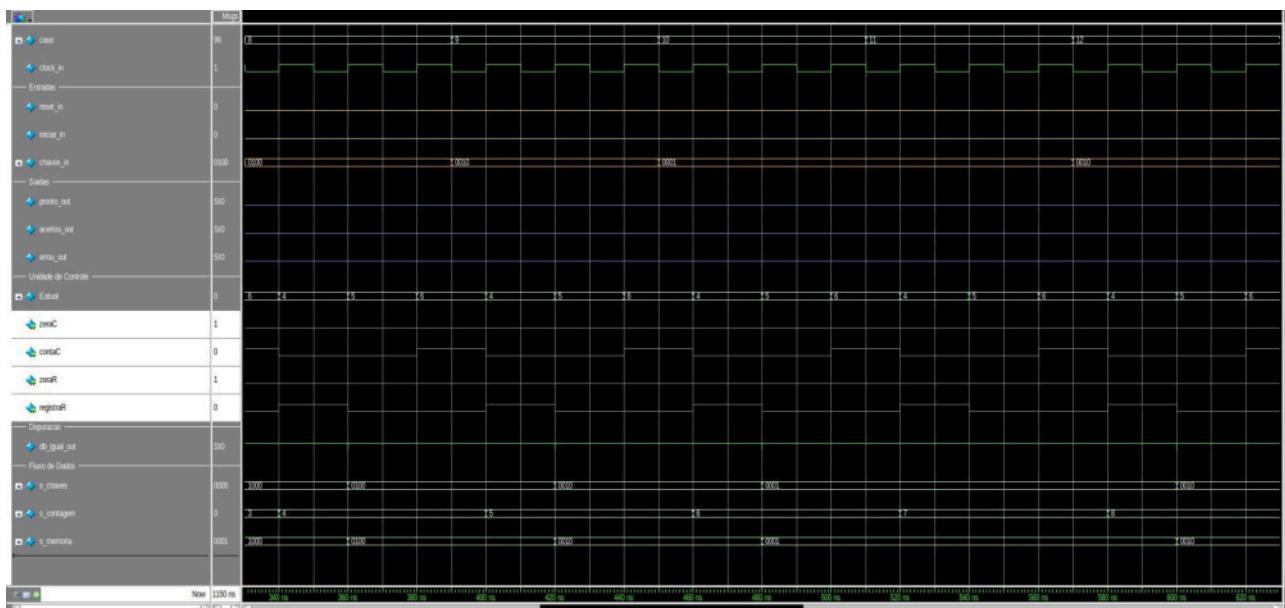
#	Operação	Sinais de Controle	Resultado Esperado	Resultado simulado ok?
c.i.	Condições iniciais (todas as entradas desativadas)	clock=0, reset=0, iniciar=0, chaves=0000	pronto=0, db_igual=0, db_contagem=0000, db_memoria=0000, db_chaves=0000, db_estado=0000, acertou=0, errou=0	Sim
1	“Resetar” circuito e observar a saída da memória	reset=1, clock ↑	pronto=0, db_igual=0, db_contagem=0000, db_memoria=0001, db_chaves=0000, db_estado=0000	Sim
2	Acionar sinal de clock 5 vezes com iniciar=0	reset=0, iniciar=0, ↑ (5x)	pronto=0, db_igual=0, db_contagem=0000, db_memoria=0001, db_chaves=0000, db_estado=0000	Sim
3	Ajustar chaves para 0001, ativar iniciar=1 e acionar clock 1x	chaves=0001, iniciar=1, clock ↑	pronto=0, db_igual=0, db_contagem=0000, db_memoria=0001, db_chaves=0000, db_estado=0001	Sim
4	Mantém chaves em 0001 e acionar clock 3x	chaves=0001, iniciar=0, clock ↑(3x)	(passa por estados 4, 5 e 6), pronto=0, db_igual=1, db_contagem=0000, db_memoria=0001, db_chaves=0001, acertou=0, errou=0	Sim
5	Ajustar chaves em 0010 e acionar clock 3x	chaves=0010, clock ↑(3x)	(passa por estados 4, 5 e 6), pronto=0, db_igual=1, db_contagem=0001, db_memoria=0010, db_chaves=0010, acertou=0, errou=0	Sim
6	Ajustar chaves em 0100 e acionar clock 3x	chaves=0100, clock ↑(3x)	(passa por estados 4, 5 e 6), pronto=0, db_igual=1, db_contagem=0010, db_memoria=0100, db_chaves=0100, acertou=0, errou=0	Sim
7	Ajustar chaves em 1000 e acionar clock 3x	chaves=1000, clock ↑(3x)	(passa por estados 4, 5 e 6), pronto=0, db_igual=1, db_contagem=0011, db_memoria=1000, db_chaves=1000, acertou=0, errou=0	Sim
8	Ajustar chaves em 0100 e acionar clock 3x	chaves=0100, clock ↑(3x)	(passa por estados 4, 5 e 6), pronto=0, db_igual=1, db_contagem=0100, db_memoria=0100, db_chaves=0100, acertou=0, errou=0	Sim
9	Ajustar chaves em 0010 e acionar clock 3x	chaves=0010, clock ↑(3x)	(passa por estados 4, 5 e 6), pronto=0, db_igual=1, db_contagem=0101, db_memoria=0010, db_chaves=0010, acertou=0, errou=0	Sim
10	Ajustar chaves em 0001 e	chaves=0001, clock ↑(3x)	(passa por estados 4, 5 e 6), pronto=0, db_igual=1, db_contagem=0110,	Sim

	acionar clock 3x		db_memoria=0001, db_chaves=0001, acertou=0, errou=0	
11	Mantém chaves em 0001 e acionar clock 3x	chaves=0001, clock ↑(3x)	(passa por estados 4, 5 e 6), pronto=0, db_igual=1, db_contagem=0111, db_memoria=0001, db_chaves=0001, acertou=0, errou=0	Sim
12	Ajustar chaves em 0010 e acionar clock 3x	chaves=0010, clock ↑(3x)	(passa por estados 4, 5 e 6), pronto=0, db_igual=1, db_contagem=1000, db_memoria=0010, db_chaves=0010, acertou=0, errou=0	Sim
13	Mantém chaves em 0010 e acionar clock 3x	chaves=0010, clock ↑(3x)	(passa por estados 4, 5 e 6), pronto=0, db_igual=1, db_contagem=1001, db_memoria=0010, db_chaves=0010, acertou=0, errou=0	Sim
14	Ajustar chaves em 0100 e acionar clock 3x	chaves=0100, clock ↑(3x)	(passa por estados 4, 5 e 6), pronto=0, db_igual=1, db_contagem=1010, db_memoria=0100, db_chaves=0100, acertou=0, errou=0	Sim
15	Mantém chaves em 0100 e acionar clock 3x	chaves=0100, clock ↑(3x)	(passa por estados 4, 5 e 6), pronto=0, db_igual=1, db_contagem=1011, db_memoria=0100, db_chaves=0100, acertou=0, errou=0	Sim
16	Ajustar chaves em 1000 e acionar clock 3x	chaves=1000, clock ↑(3x)	(passa por estados 4, 5 e 6), pronto=0, db_igual=1, db_contagem=1100, db_memoria=1000, db_chaves=1000, acertou=0, errou=0	Sim
17	Mantém chaves em 1000 e acionar clock 3x	chaves=1000, clock ↑(3x)	(passa por estados 4, 5 e 6), pronto=0, db_igual=1, db_contagem=1101, db_memoria=1000, db_chaves=1000, acertou=0, errou=0	Sim
18	Ajustar chaves em 0001 e acionar clock 3x	chaves=0001, clock ↑(3x)	(passa por estados 4, 5 e 6), pronto=0, db_igual=1, db_contagem=1110, db_memoria=0001, db_chaves=0001, acertou=0, errou=0	Sim
19	Ajustar chaves em 0100 e acionar clock 3x	chaves=0100, clock ↑(3x)	(passa por estados 4, 5 e F), pronto=1, db_igual=1, db_contagem=1111, db_memoria=0100, db_chaves=0100, acertou=1, errou=0	Sim

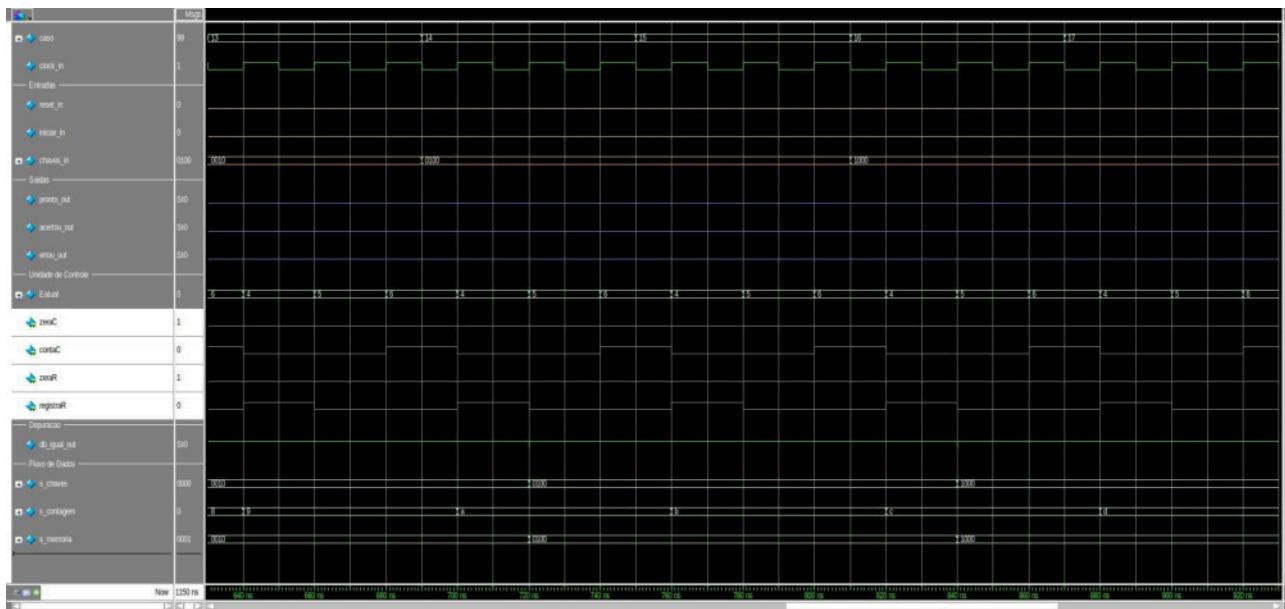
- Testes de 0 a 7



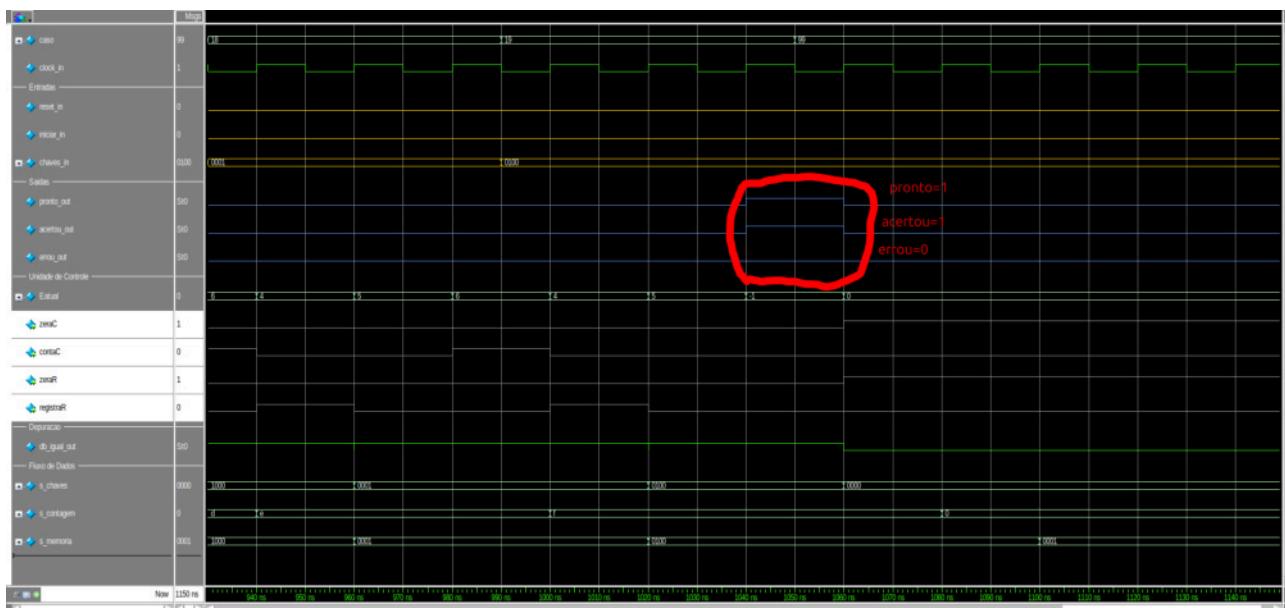
- Testes de 8 a 12



- Testes de 13 a 17



- Teste 18 e finalização



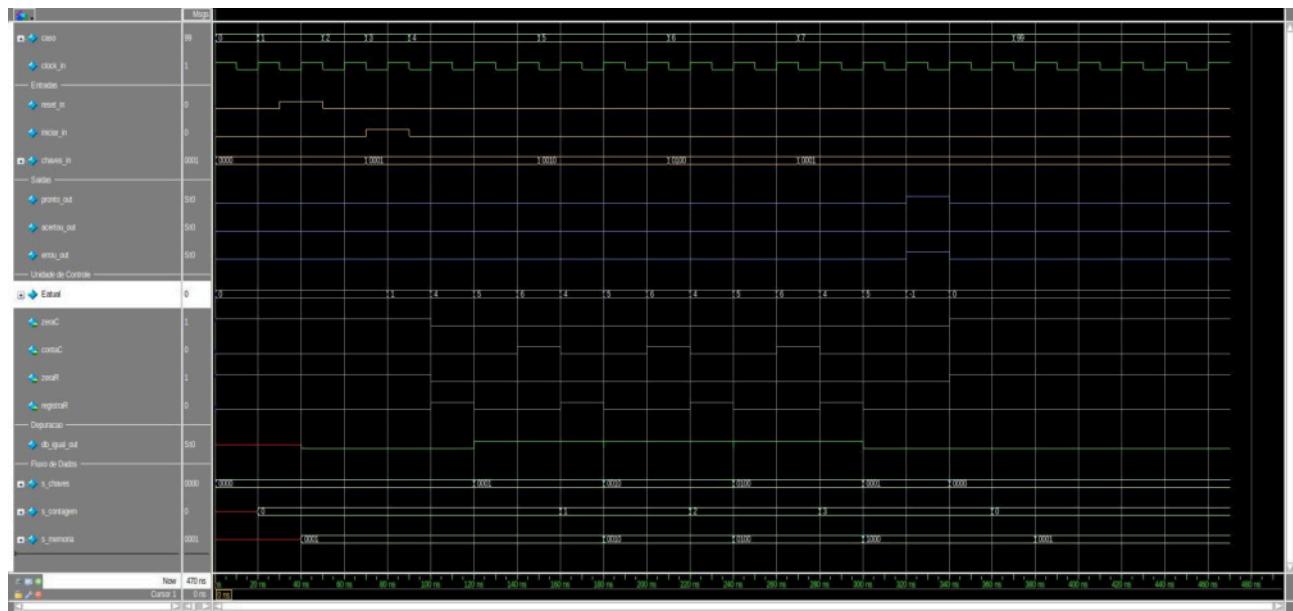
6.3.2 Cenário de Teste 2 – Simulação do Desafio no ModelSim - caso 2

Simulação no *Modelsim* do funcionamento do módulo *circuito_exp3_desafio*, descrito na linguagem Verilog, utilizando a *TestBench* *circuito_exp3_desafio2_tb*.

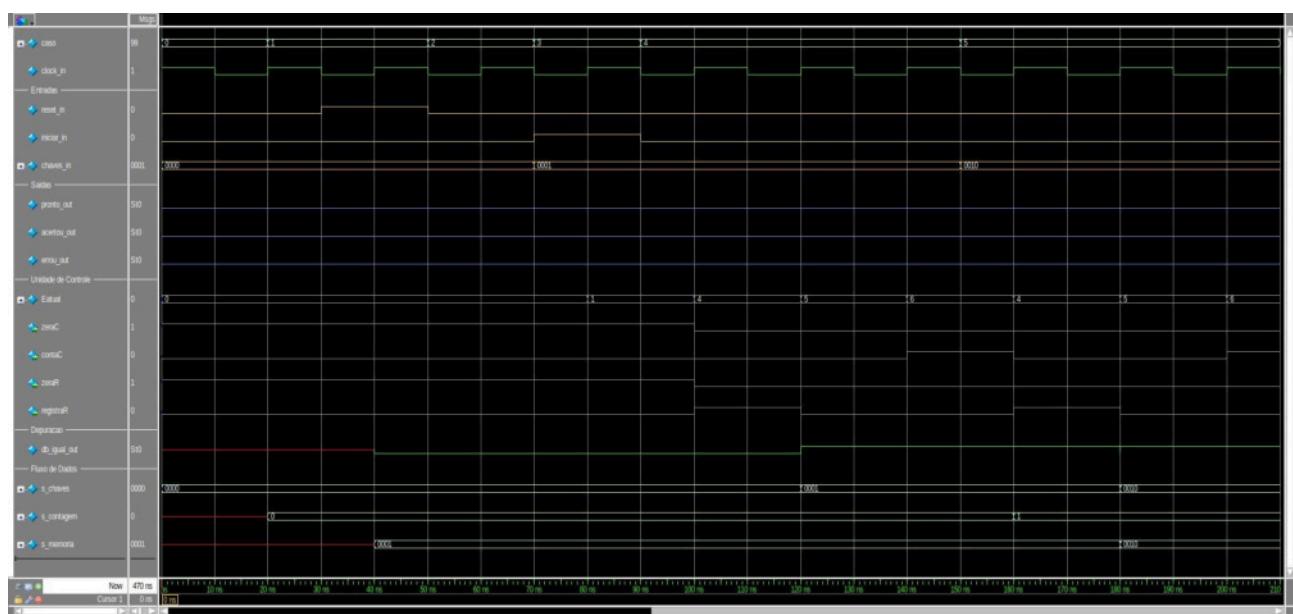
Tabela 8 – Descrição e Resultados do Cenário de Teste 2 para o Desafio

#	Operação	Sinais de Controle	Resultado Esperado	Resultado simulado ok?
c.i.	Condições iniciais (todas as entradas desativadas)	clock=0, reset=0, iniciar=0, chaves=0000	pronto=0, db_igual=0, db_contagem=0000, db_memoria=0001, db_chaves=0000, db_estado=0000, acertou=0, errou=0	Sim
1	“Resetar” circuito e observar a saída da memória	reset=1, clock ↑	pronto=0, db_igual=0, db_contagem=0000, db_memoria=0001, db_chaves=0000, db_estado=0000	Sim
2	Acionar sinal de clock 5 vezes com iniciar=0	reset=0, iniciar=0, ↑ (5x)	pronto=0, db_igual=0, db_contagem=0000, db_memoria=0001, db_chaves=0000, db_estado=0000	Sim
3	Ajustar chaves para 0001, ativar iniciar=1 e acionar clock 1x	chaves=0001, iniciar=1, clock ↑	pronto=0, db_igual=0, db_contagem=0000, db_memoria=0001, db_chaves=0001, db_estado=0001	Sim
4	Mantém chaves em 0001 e acionar clock 3x	chaves=0001, iniciar=0, clock ↑(3x)	(passa por estados 4, 5 e 6), pronto=0, db_igual=1, db_contagem=0000, db_memoria=0001, db_chaves=0001, acertou=0, errou=0	Sim
5	Ajustar chaves em 0010 e acionar clock 3x	chaves=0010, clock ↑(3x)	(passa por estados 4, 5 e 6), pronto=0, db_igual=1, db_contagem=0001, db_memoria=0010, db_chaves=0010, acertou=0, errou=0	Sim
6	Ajustar chaves em 0100 e acionar clock 3x	chaves=0100, clock ↑(3x)	(passa por estados 4, 5 e 6), pronto=0, db_igual=1, db_contagem=0010, db_memoria=0100, db_chaves=0100, acertou=0, errou=0	Sim
7	Ajustar chaves em 0001 e acionar clock 3x	chaves=0001, clock ↑(3x)	(passa por estados 4, 5 e F), pronto=1, db_igual=0, db_contagem=0011, db_memoria=1000, db_chaves=0001, acertou=0, errou=1	Sim

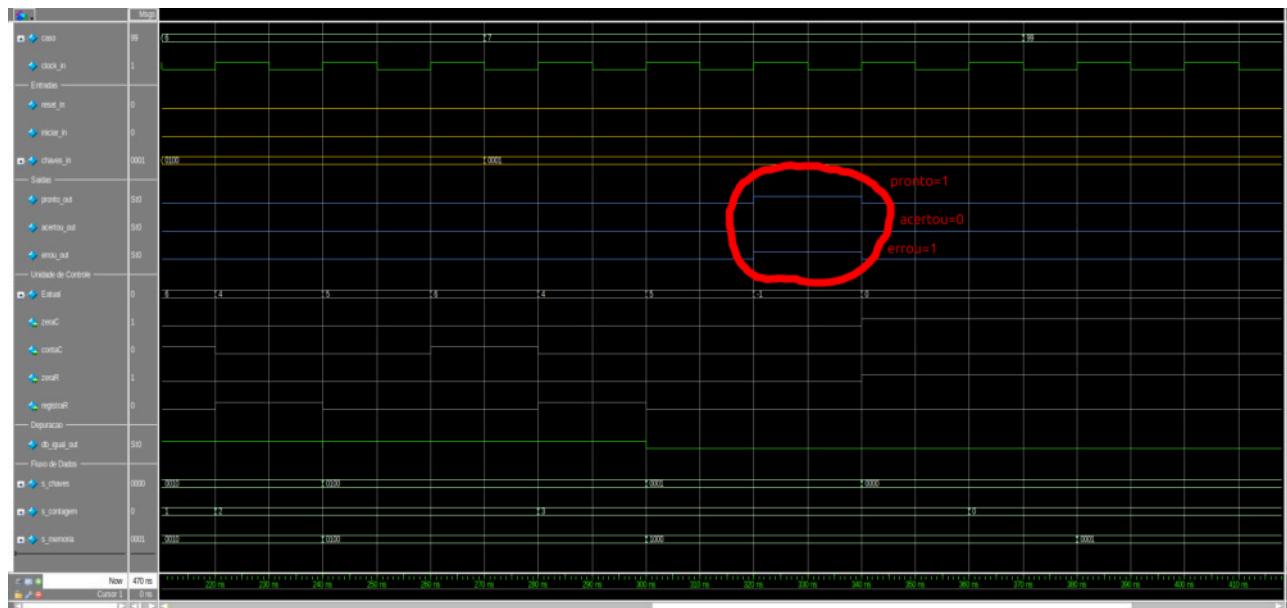
- Visão Geral dos Testes



- Testes de 1 a 5



- Testes 6, 7 e finalização



6.3.3 EXECUÇÃO PRÁTICA DO CENÁRIO DE TESTE 1 – TESTES NA FPGA DO DESAFIO - CASO 1

Testes na FPGA do funcionamento do módulo circuito_exp3_desafio, descrito na linguagem Verilog, utilizando a *TestBench* circuito_exp3_desafio1_tb.

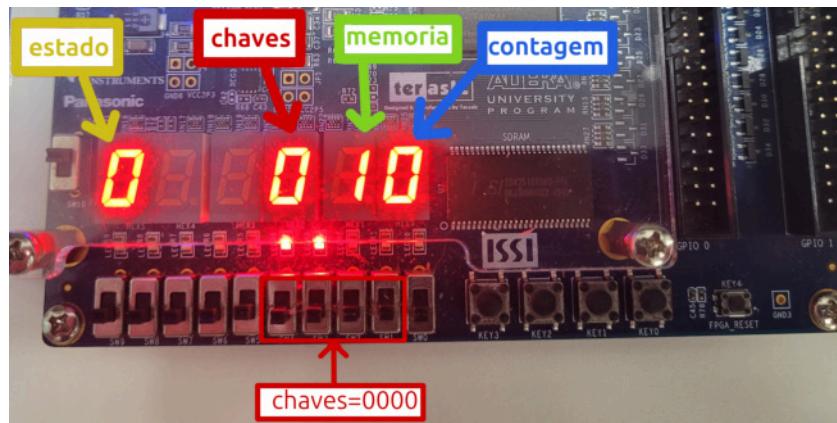
Tabela 9 – Descrição e Resultados Práticos do Cenário de Teste 1 do desafio

#	Operação	Sinais de Controle	Resultado Esperado	Resultado prático ok?
c.i.	Condições iniciais (todas as entradas desativadas)	clock=0, reset=0, iniciar=0, chaves=0000	pronto=0, db_igual=0, db_contagem=0000, db_memoria=0001, db_chaves=0000, db_estado=0000, acertou=0, errou=0	Sim
1	“Resetar” circuito e observar a saída da memória	reset=1, clock ↑	pronto=0, db_igual=0, db_contagem=0000, db_memoria=0001, db_chaves=0000, db_estado=0000	Sim
2	Acionar sinal de clock 5 vezes com iniciar=0	reset=0, iniciar=0, ↑ (5x)	pronto=0, db_igual=0, db_contagem=0000, db_memoria=0001, db_chaves=0000, db_estado=0000	Sim
3	Ajustar chaves para 0001, ativar iniciar=1 e acionar clock 1x	chaves=0001, iniciar=1, clock ↑	pronto=0, db_igual=0, db_contagem=0000, db_memoria=0001, db_chaves=0000, db_estado=0001	Sim
4	Mantém chaves em 0001 e acionar clock 3x	chaves=0001, iniciar=0, clock ↑(3x)	(passa por estados 4, 5 e 6), pronto=0, db_igual=1, db_contagem=0000, db_memoria=0001, db_chaves=0001, acertou=0, errou=0	Sim
5	Ajustar chaves em 0010 e acionar	chaves=0010, clock ↑(3x)	(passa por estados 4, 5 e 6), pronto=0, db_igual=1, db_contagem=0001,	Sim

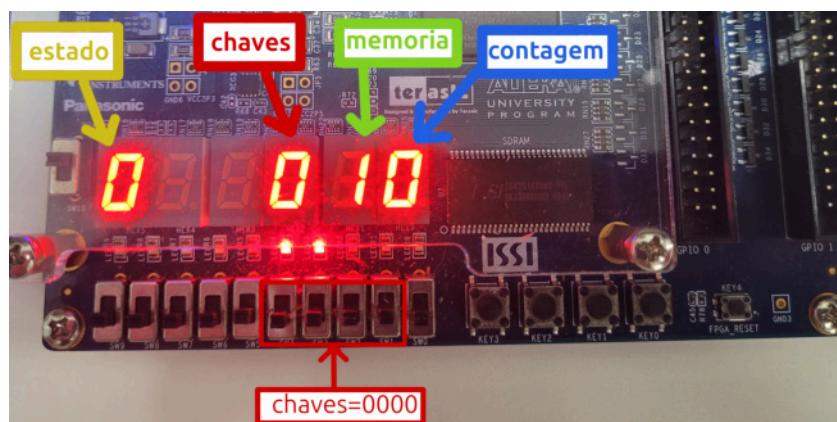
	clock 3x		db_memoria=0010, db_chaves=0010, acertou=0, errou=0	
6	Ajustar chaves em 0100 e acionar clock 3x	chaves=0100, clock ↑(3x)	(passa por estados 4, 5 e 6), pronto=0, db_igual=1, db_contagem=0010, db_memoria=0100, db_chaves=0100, acertou=0, errou=0	Sim
7	Ajustar chaves em 1000 e acionar clock 3x	chaves=1000, clock ↑(3x)	(passa por estados 4, 5 e 6), pronto=0, db_igual=1, db_contagem=0011, db_memoria=1000, db_chaves=1000, acertou=0, errou=0	Sim
8	Ajustar chaves em 0100 e acionar clock 3x	chaves=0100, clock ↑(3x)	(passa por estados 4, 5 e 6), pronto=0, db_igual=1, db_contagem=0100, db_memoria=0100, db_chaves=0100, acertou=0, errou=0	Sim
9	Ajustar chaves em 0010 e acionar clock 3x	chaves=0010, clock ↑(3x)	(passa por estados 4, 5 e 6), pronto=0, db_igual=1, db_contagem=0101, db_memoria=0010, db_chaves=0010, acertou=0, errou=0	Sim
10	Ajustar chaves em 0001 e acionar clock 3x	chaves=0001, clock ↑(3x)	(passa por estados 4, 5 e 6), pronto=0, db_igual=1, db_contagem=0110, db_memoria=0001, db_chaves=0001, acertou=0, errou=0	Sim
11	Mantém chaves em 0001 e acionar clock 3x	chaves=0001, clock ↑(3x)	(passa por estados 4, 5 e 6), pronto=0, db_igual=1, db_contagem=0111, db_memoria=0001, db_chaves=0001, acertou=0, errou=0	Sim
12	Ajustar chaves em 0010 e acionar clock 3x	chaves=0010, clock ↑(3x)	(passa por estados 4, 5 e 6), pronto=0, db_igual=1, db_contagem=1000, db_memoria=0010, db_chaves=0010, acertou=0, errou=0	Sim
13	Mantém chaves em 0010 e acionar clock 3x	chaves=0010, clock ↑(3x)	(passa por estados 4, 5 e 6), pronto=0, db_igual=1, db_contagem=1001, db_memoria=0010, db_chaves=0010, acertou=0, errou=0	Sim
14	Ajustar chaves em 0100 e acionar clock 3x	chaves=0100, clock ↑(3x)	(passa por estados 4, 5 e 6), pronto=0, db_igual=1, db_contagem=1010, db_memoria=0100, db_chaves=0100, acertou=0, errou=0	Sim
15	Mantém chaves em 0100 e acionar clock 3x	chaves=0100, clock ↑(3x)	(passa por estados 4, 5 e 6), pronto=0, db_igual=1, db_contagem=1011, db_memoria=0100, db_chaves=0100, acertou=0, errou=0	Sim
16	Ajustar chaves em 1000 e acionar	chaves=1000, clock ↑(3x)	(passa por estados 4, 5 e 6), pronto=0, db_igual=1, db_contagem=1100,	Sim

	clock 3x		db_memoria=1000, db_chaves=1000, acertou=0, errou=0	
17	Mantém chaves em 1000 e acionar clock 3x	chaves=1000, clock ↑(3x)	(passa por estados 4, 5 e 6), pronto=0, db_igual=1, db_contagem=1101, db_memoria=1000, db_chaves=1000, acertou=0, errou=0	Sim
18	Ajustar chaves em 0001 e acionar clock 3x	chaves=0001, clock ↑(3x)	(passa por estados 4, 5 e 6), pronto=0, db_igual=1, db_contagem=1110, db_memoria=0001, db_chaves=0001, acertou=0, errou=0	Sim
19	Ajustar chaves em 0100 e acionar clock 3x	chaves=0100, clock ↑(3x)	(passa por estados 4, 5 e F), pronto=1, db_igual=1, db_contagem=1111, db_memoria=0100, db_chaves=0100, acertou=1, errou=0	Sim

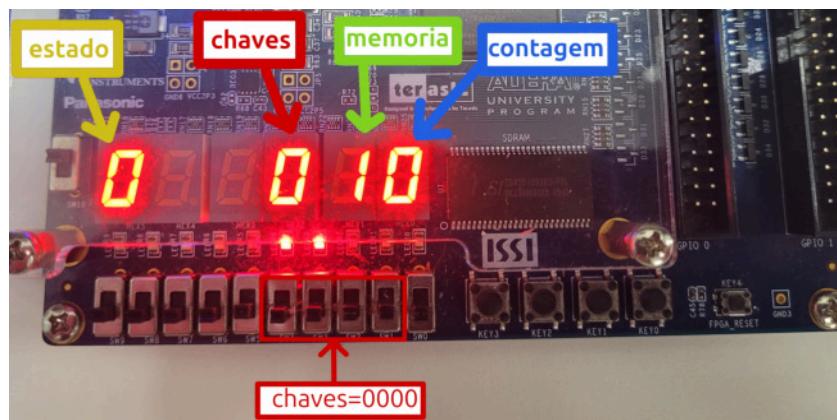
- Condições iniciais



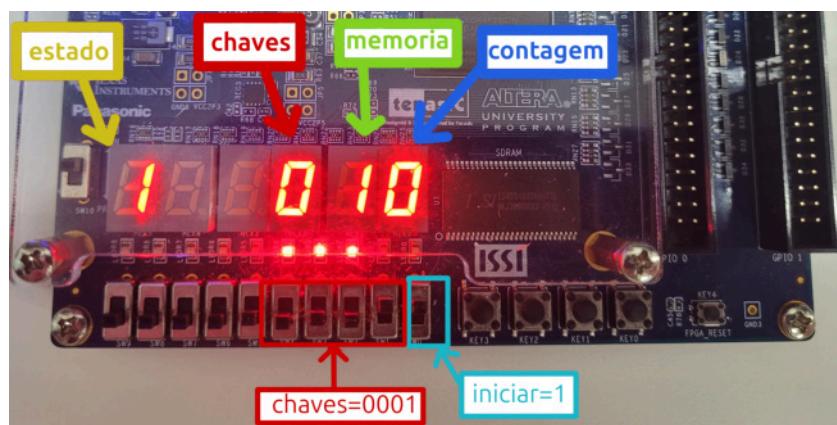
- Teste 1 - “Resetar” circuito e observar a saída da memória



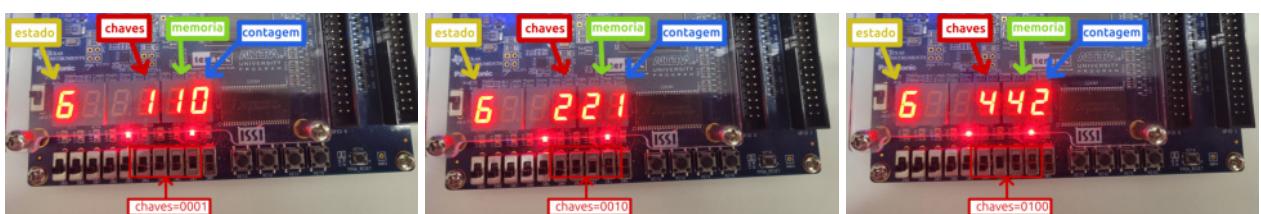
- Teste 2 - Acionar sinal de clock 5 vezes com iniciar=0



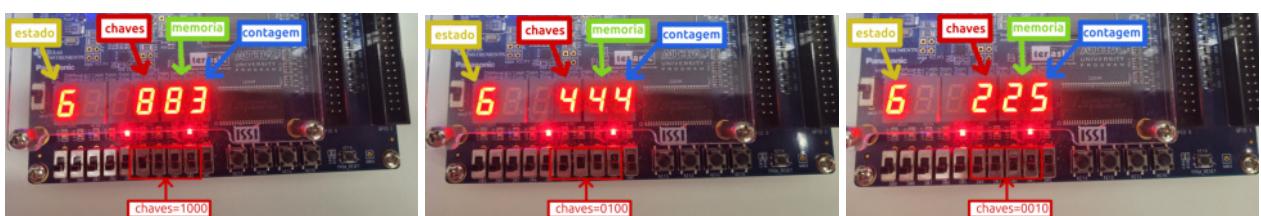
- Teste 3 - Ajustar chaves para 0001, ativar iniciar=1 e acionar clock 1x



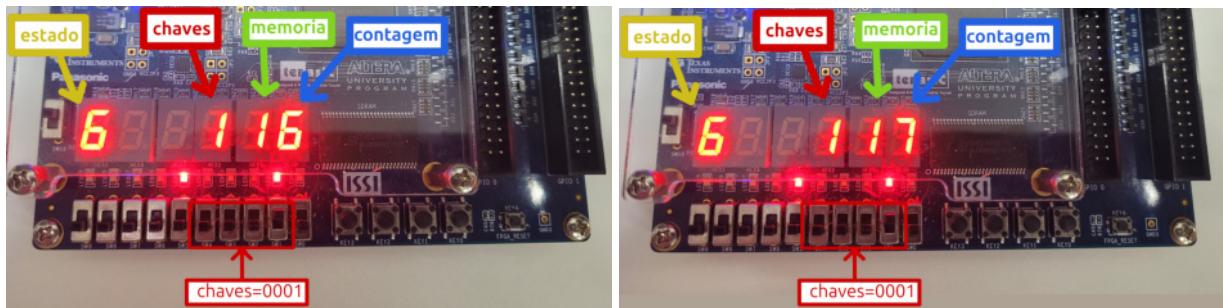
- Teste 4, 5 e 6 - Acionar clock 3x nos cenários de chaves em 0001, 0010 e 0100



- Testes 7, 8 e 9 - Acionar clock 3x nos cenários de chaves em 1000, 0100 e 0010



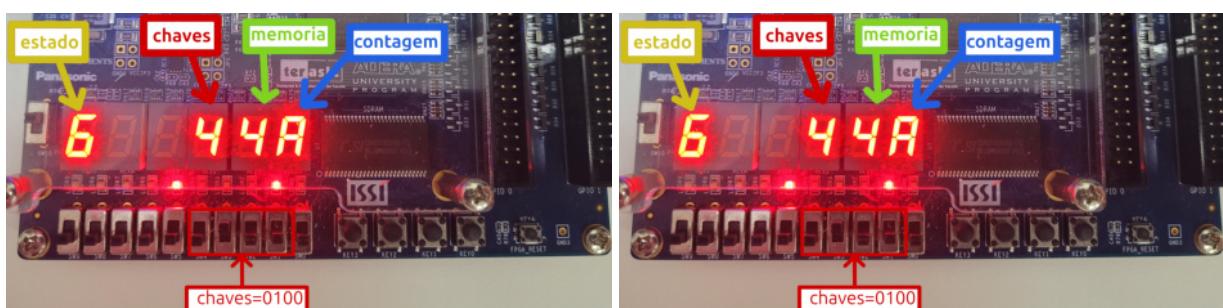
- Testes 10 e 11 - Manter chaves em 0001 e acionar clock 6x



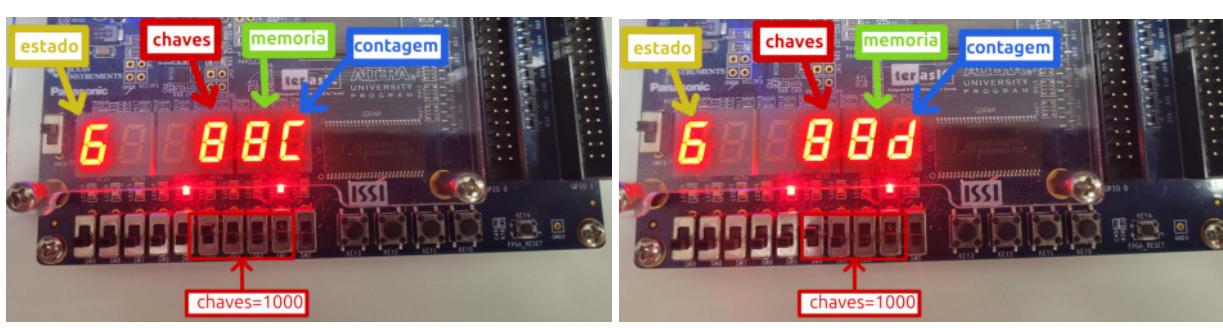
- Teste 12 e 13 - Mantém chaves em 0010 e acionar clock 6x



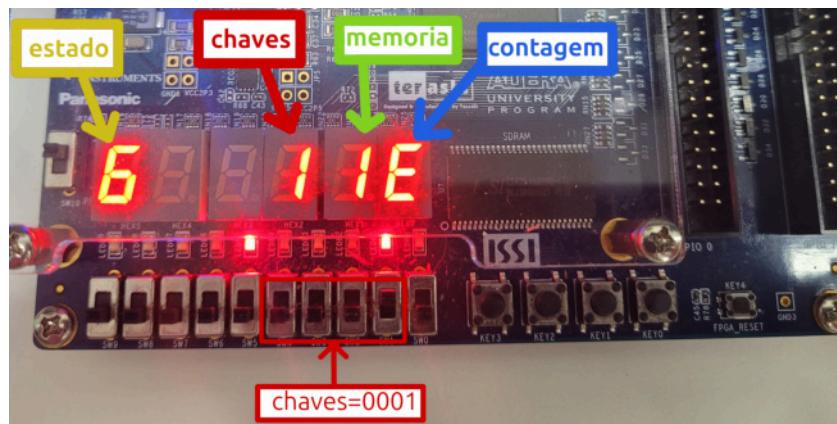
- Teste 14 e 15 - Mantém chaves em 0100 e acionar clock 6x



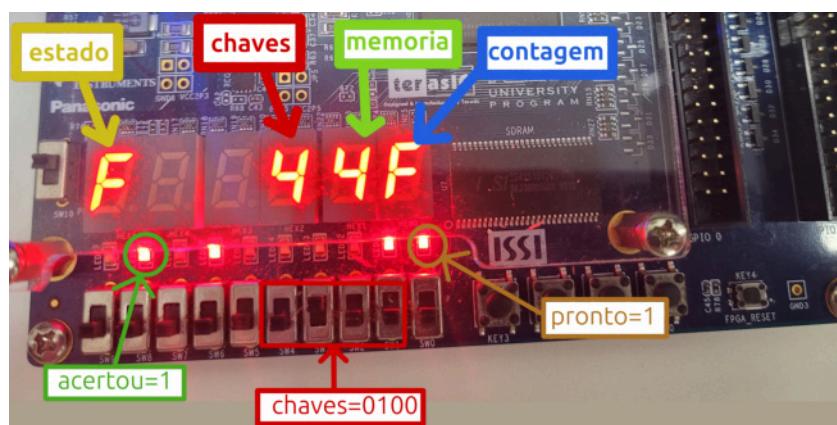
- Teste 16 e 17 - Mantém chaves em 0100 e acionar clock 6x



- Teste 18 - Ajustar chaves em 0001 e acionar clock 3x



- Teste 19 - Ajustar chaves em 0100 e acionar clock 3x



6.3.4 EXECUÇÃO PRÁTICA DO CENÁRIO DE TESTE 2 – TESTES NA FPGA DO DESAFIO – CASO 2

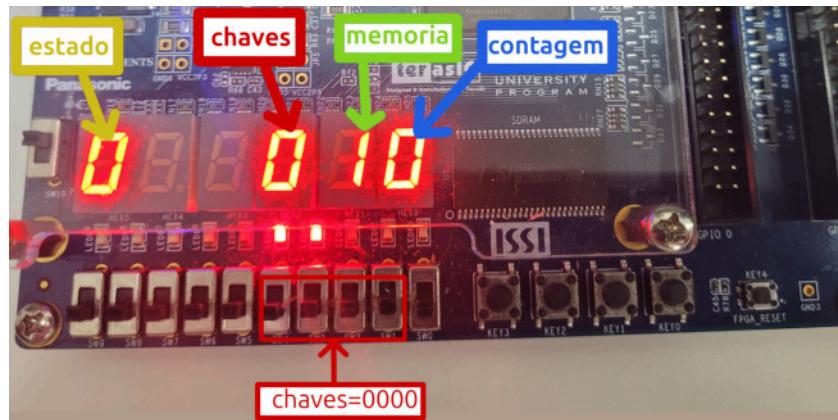
Testes na FPGA do funcionamento do módulo circuito_exp3_desafio, descrito na linguagem Verilog, utilizando a *TestBench* circuito_exp3_desafio2_tb.

Tabela 10 – Descrição e Resultados Práticos do Cenário de Teste 2 do desafio

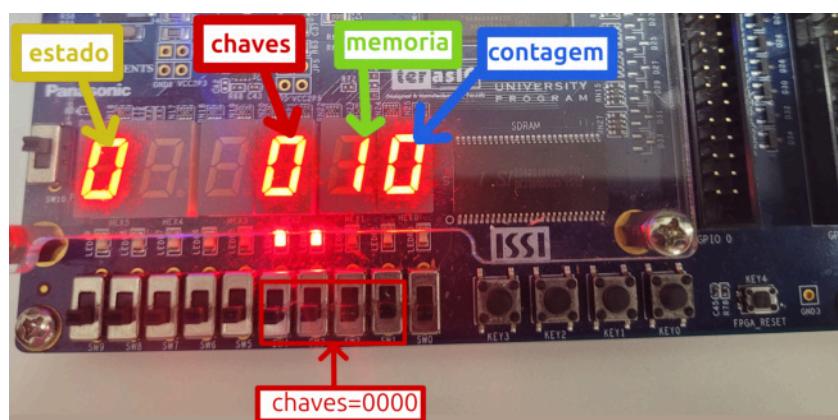
#	Operação	Sinais de Controle	Resultado Esperado	Resultado prático ok?
c.i.	Condições iniciais (todas as entradas desativadas)	clock=0, reset=0, iniciar=0, chaves=0000	pronto=0, db_igual=0, db_contagem=0000, db_memoria=0001, db_chaves=0000, db_estado=0000, acertou=0, errou=0	Sim
1	“Resetar” circuito e observar a saída da memória	reset=1, clock ↑	pronto=0, db_igual=0, db_contagem=0000, db_memoria=0001, db_chaves=0000, db_estado=0000	Sim
2	Acionar sinal de clock 5 vezes com iniciar=0	reset=0, iniciar=0, ↑ (5x)	pronto=0, db_igual=0, db_contagem=0000, db_memoria=0001, db_chaves=0000, db_estado=0000	Sim

3	Ajustar chaves para 0001, ativar iniciar=1 e acionar clock 1x	chaves=0001, iniciar=1, clock ↑	pronto=0, db_igual=0, db_contagem=0000, db_memoria=0001, db_chaves=0000, db_estado=0001	Sim
4	Mantém chaves em 0001 e acionar clock 3x	chaves=0001, iniciar=0, clock ↑(3x)	(passa por estados 4, 5 e 6), pronto=0, db_igual=1, db_contagem=0000, db_memoria=0001, db_chaves=0001, acertou=0, errou=0	Sim
5	Ajustar chaves em 0010 e acionar clock 3x	chaves=0010, clock ↑(3x)	(passa por estados 4, 5 e 6), pronto=0, db_igual=1, db_contagem=0001, db_memoria=0010, db_chaves=0010, acertou=0, errou=0	Sim
6	Ajustar chaves em 0100 e acionar clock 3x	chaves=0100, clock ↑(3x)	(passa por estados 4, 5 e 6), pronto=0, db_igual=1, db_contagem=0010, db_memoria=0100, db_chaves=0100, acertou=0, errou=0	Sim
7	Ajustar chaves em 0001 e acionar clock 3x	chaves=0001, clock ↑(3x)	(passa por estados 4, 5 e F), pronto=1, db_igual=0, db_contagem=0011, db_memoria=1000, db_chaves=0001, acertou=0, errou=1	Sim

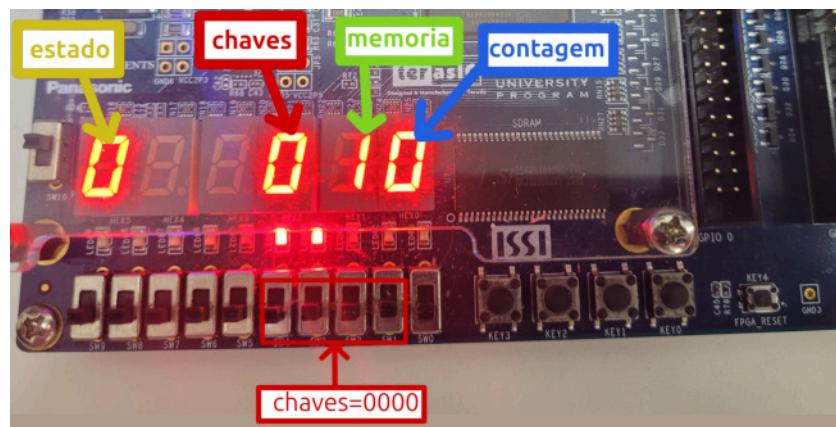
- Condição inicial



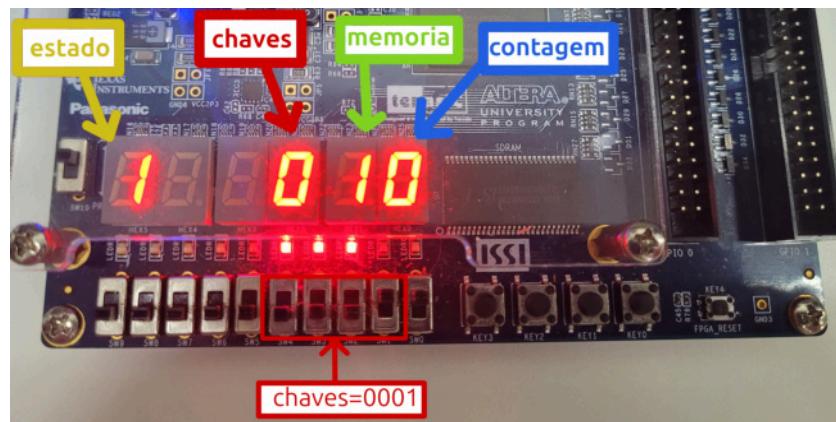
- Teste 1 - “Resetar” circuito e observar a saída da memória



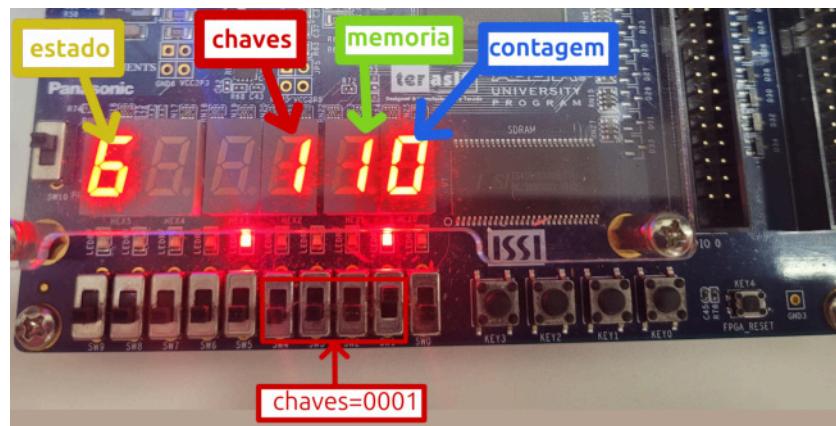
- Teste 2 - Acionar sinal de clock 5 vezes com iniciar=0



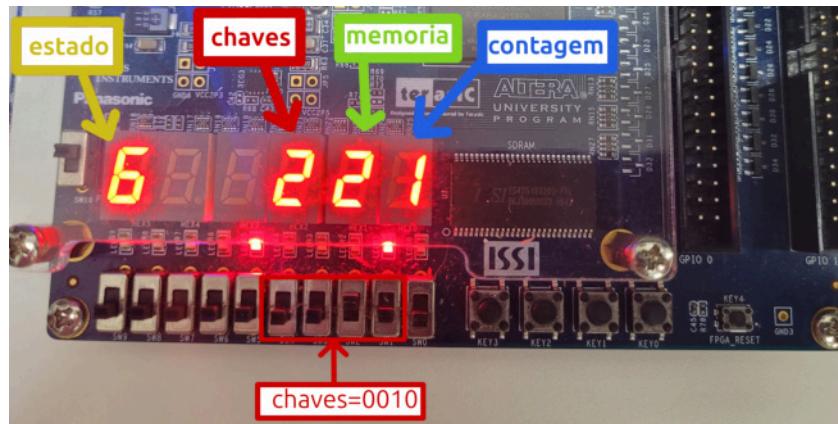
- Teste 3 - Ajustar chaves para 0001, ativar iniciar=1 e acionar clock 1x



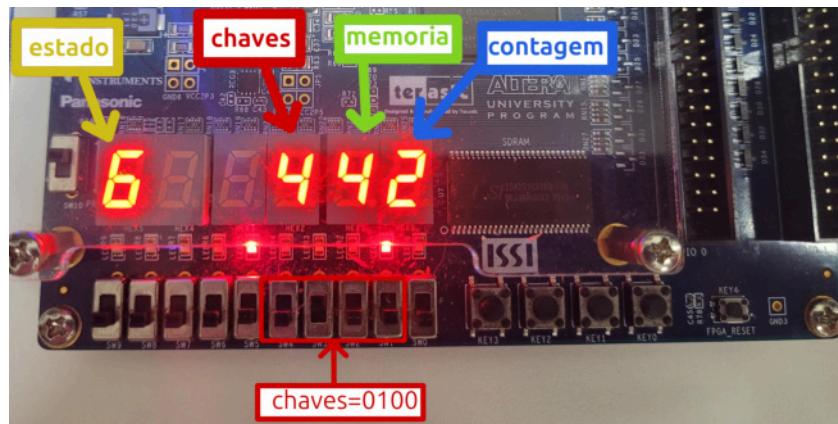
- Teste 4 - Mantém chaves em 0001 e acionar clock 3x



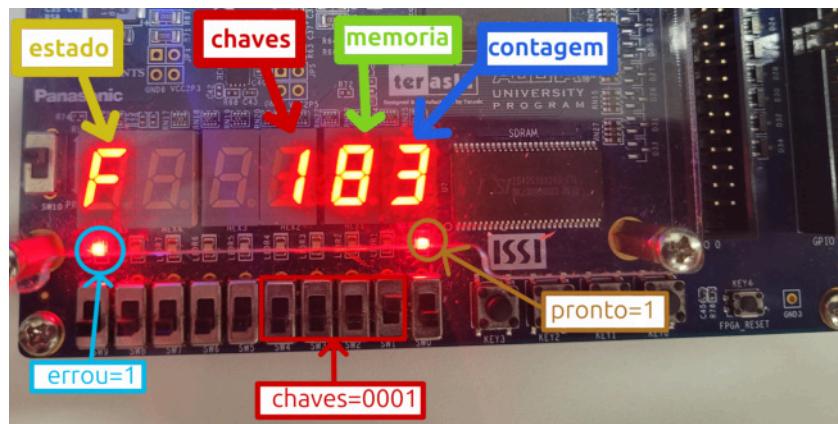
- Teste 5 - Ajustar chaves em 0010 e acionar clock 3x



- Teste 6 - Ajustar chaves em 0100 e acionar clock 3x



- Teste 7 - Ajustar chaves em 0001 e acionar clock 3x



7 CONCLUSÕES

Até o momento do planejamento, todos os testes cumpriram os objetivos planejados para a experiência. Foram dirigidos dois testes: um primeiro acerca do funcionamento do módulo do fluxo de dados, e o segundo acerca do funcionamento geral do circuito.

Sobre o desafio, foi implementado um sistema que confere se a sequência indicada nas chaves condiz com a sequência indicada na memória, se todos os resultados estiverem corretos, um sinal de “acertou” será levantado, se algum (apenas um, no mínimo) estiver errado, um sinal de “errou” será levantado. Os testes foram realizados e bem sucedidos.

Uma única intercorrência no desenvolvimento do projeto se demonstrou ser a pinagem da placa, na qual alguns sinais não aparecem na tabela gerada pelo *Intel Quartus Prime*. O problema foi discutido entre os integrantes do grupo e foi solucionado através de uma solução proposta pela apostila de colocar os sinais de depuração adicionais no código em Verilog do circuito.

No dia da experiência, a preparação foi realizada com sucesso. Tanto a experiência na forma original quanto o desafio foram compilados e processados no *Intel Quartus Prime* e transferidos para a placa FPGA, na qual foram testados em conjunto com o uso do software *WaveForms* para o controle dos sinais de reset e clock, através da conexão com o circuito utilizando o *Analog Discovery*. Todos os testes executados funcionaram corretamente. A única dúvida em relação ao funcionamento do circuito foi esclarecida com o professor; ela estava relacionada aos elementos síncronos e assíncronos do circuito, devido ao comportamento do sinal de relógio.

No geral, a experiência mostrou-se extremamente construtiva para o aprendizado dos alunos, proporcionando uma visão mais clara de como o circuito se tornará o jogo proposto pelos orientadores da disciplina.